UNIVERSITY OF TWENTE.

STRATEGICAL CAPACITY PLANNING FOR TWO OF SENSATA'S AUTOMOTIVE DEPARTMENTS

EDITH – IOANA STANCA

INDUSTRIAL ENGINEERING AND MANAGEMENT, 2019

SUPERVISORS:

University of Twente:
MARCO SCHUTTEN
ERWIN HANS

Sensata Technologies:
GERBEN VAN DER VELDE
CHRIS WILDERMUTH

Management Summary

This research takes place in collaboration with Sensata Technologies. Sensata Technologies is one of the world's leader suppliers in terms of sensing, control, power management, and electrical protection solutions. The highest revenue industry in which Sensata competes is the automotive industry. This research is carried out at the initiative of two capacity managers from Sensata's automotive departments and focuses on the departments' strategic capacity planning.

Problem Description

This research starts with the aim of redesigning their current strategic capacity planning model. This need arose due to their forecasts showing a rapid increase in demand[1] over the upcoming years. Besides the increase in the actual number of units, a demand increase generates an increase in number of equipment and types of products. Considering the high investment incurred by purchasing new equipment, the company wants to postpone such an investment for as long as possible. Another reason for initiating this research consists in the simplicity of their current model. The managers use an Excel model from which they exclude a variety of relevant factors. Moreover, the factors they do take into account are assigned with average values. Given all these simplifications the model is not capable of reflecting aspects such as performance differences between individual machines of the same type and is hence not able to reflect the real-life situation in an accurate manner.

Objective

We aim to redesign Sensata's strategic capacity planning model. First, we want to reflect the differences between individual products or machines, hence disaggregating the average values. Next, we want to model the real-life situation in a more accurate manner by including relevant factors such as: machine capabilities[2], machine releases[3] and Overall Equipment Effectiveness components[4]. The outcome of our research represents a proof of concept of a strategic capacity planning model.

Approach

We start by analyzing the current situation of the two departments. During this analysis we take a close look at their current model and aim to better understand their manufacturing processes. Together with the capacity managers we decide on the additional factors which we should consider to reflect the real-life situation in a more accurate manner. Further on, we identify the main actions the managers can perform to cope with the increase in demand. The actions refer to: building inventory in prior months, releasing existing machines or purchasing new equipment. Since each action has an associated cost, our aim is to reduce the total costs across a certain planning horizon.

[1] The number of units of each individual product the company has to produce in a certain period of time

[2] Given the multitude of sensors the company manufactures, not all the equipment is physically capable of producing each single product. These capabilities reflect which sensors can each machine process.

[3] Similar to the capabilities, the machine releases also reflect whether a machine is allowed to process a certain product. Unlike the machine capabilities, the releases are not related to the machine's specifications but to Sensata's customers. According to the capacity managers, in the automotive industry only machines and raw materials approved by the customer can be used in the manufacturing of each product. To obtain the customer's approval the company produces sample sensors, using a sequence of machines selected by their manufacturing facilities, which are sent to and tested by each customer to ensure their quality. If the quality reaches the desired standards, the customer allows Sensata to further produce his sensors, however, by using only the exact equipment used for building the samples. Hence, through this approval the customer releases the machines.

[4] Metric of the efficiency and effectiveness of a process which can be divided into three different components: Availability, Quality and Performance

We develop a heuristic to show how the managers can cope with the increase in demand by incurring one or more of the three actions. The heuristic starts by creating an initial schedule[5] which we then aim to improve through local search optimization. Next, if the capacity does not suffice, the model attempts to build inventory in prior months. In a similar manner, if the demand is still not covered after inventory buildup we attempt to release existing machines. As a last resort the heuristic decides in favor of purchasing a new machine. We approach the releasing of either existing or newly purchased equipment through local search metaheuristics. To choose the best local search algorithm we compare between Simulated Annealing and Tabu Search.

To benchmark the results of our heuristic, we formulate a mixed integer linear programming model. In this model we focus on reducing the total costs incurred for a given planning horizon. We use the CPLEX solver to solve this model.

Conclusions and Recommendations

For each of the three optimization steps we perform a comparison between Simulated Annealing and Tabu Search. Starting with the optimization of the initial schedule we compare them in terms of monthly overall loading[6] and computational efforts. Although the former outperforms the latter in terms of running times and, sometimes in terms of the objective function value, our final choice is Tabu Search. We base our choice on two reasons. First, Tabu Search outperforms Simulated Annealing in 13 out of 24 months. Second, Simulated Annealing is not able to improve the monthly overall loading for a total of 9 months, while Tabu Search finds an improvement in 22 out of 24 months.

Next, for releasing existing machines we also choose in favor of Tabu Search. Yet again, Simulated Annealing outperforms our choice in terms of computational efforts, however, given the random selection process of Simulated Annealing, Tabu Search is always better at releasing the preferred machines for each product. By preferred machines we refer to the machines having smaller processing times. Considering the third optimization stage, we yet again choose in detriment of Simulated Annealing. We base this choice on the objective function values we obtain from each algorithm.

In the end we provide the company with recommendations regarding the implementation of our algorithm as Sensata's new strategic capacity planning model. A major part of these recommendations relate to the improvement of their input data. Finally, we introduce our recommendations in terms of future work and the limitations of this research. Such limitations refer to our algorithm representing a proof of concept and not a fully functional tool combined with the fact that the company can include more factors to reflect the real-life situation even further.

[5] Allocation of products to machines showing the number of units of each product which should be processed by each machine

[6] Percentage of the total available capacity which is used for production on a monthly basis

Acknowledgments

This master thesis represents the last milestone of my study in Industrial Engineering and Management. This thesis is the result of my research in strategic capacity planning and has turned into a proof of concept of a capacity planning tool for Sensata Technologies.

I would like to express my gratitude to the two capacity managers, Gerben van der Velde and Chris Wildermuth for initiating this project and giving me the chance to be part of this organization. I appreciate their valuable feedback and the time invested in helping me during my research.

I am extremely grateful to Gerben for answering all my questions and discussing all my concerns, supporting me, and helping me overpass all the problems I have encountered along the way. I enjoyed working with you for the past nine months and I will surely miss it.

Furthermore, I want to extend my gratitude to Marco Schutten and Erwin Hans, my thesis coordinators from the University of Twente. Their helpful advice and relentless support gave me important insights on how can I improve my research.

In addition, I want to thank my family for always being there for me and for all their support during my study years. Finally, there are my friends, who were of great moral support, as well as providing happy distractions to rest my mind outside of my research.

I hope you will enjoy reading this report.

Hengelo, August 2019

Edith - Ioana Stanca

Table of Contents

List of Abbreviations

HVAC	Heating, ventilation, and air conditioning
MEMS	Micro-Electro-Mechanical Systems
MSG	Metal String Gages
SEA	Sense Element Assembly
PCB	Printed Circuit Board
EMA	Electronic Module Assembly
RTV	Room Temperature Vulcanization
CT	Cycle Time
OEE	Overall Equipment Performance
APP	Aggregate Production Planning
HPP	Hierarchical Production Planning
ICM	Infinite Capacity Machine
SA	Simulated Annealing
TS	Tabu Search

List of Figures

List of Tables

Chapter 1 Introduction

Capacity management is of critical importance for any company given that limitations in terms of capacity or resource availability can represent a major bottleneck. When considering capacity planning from a production point of view, restrictions in the number of equipment available or poor planning techniques can result in delaying orders' completion. Therefore, an appropriate planning for the assignment of products to machines is required. In this research, we develop an optimization algorithm for the strategic capacity planning for two of Sensata's automotive departments.

This first chapter explains the context of our project. Section 1.1 covers general details about the company's background. In the end of the section we give a small overview of the departments' current situation. Section 1.2 covers our research plan, while in Section 1.3 we provide an outline of this report.

1.1 Company Introduction

Sensata Technologies is one of the world's leading suppliers in matters such as: sensing, electrical protection, control, and power management solutions. The company is an early innovator in terms of electrical protection and mission-critical sensors. Hence, they design their products with the goal of improving the safety, efficiency and comfort of their customers.

The company started in 1916, under the name of "General Plate Company", by providing gold plates for the jewelry industry. In 1931 they merged with Spencer Thermostat while in 1959 they merged with Texas Instruments. In 2006 they were bought by Bain Capital and, from that moment onwards, continued under the name of Sensata Technologies. Currently Sensata owns 12 other brand names and ships around 1 billion products per year.

At the moment the company has manufacturing facilities and business plants in North and South America, Europe and Asia. Figure 1 shows an overview of Sensata's manufacturing facilities and business sites, while Figure 2 displays their revenue percentages over the three big regions.

Figure 1 Sensata's manufacturing facilities and business sites

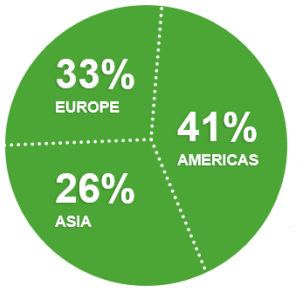

Figure 2 Highest Revenue Percentages per Region

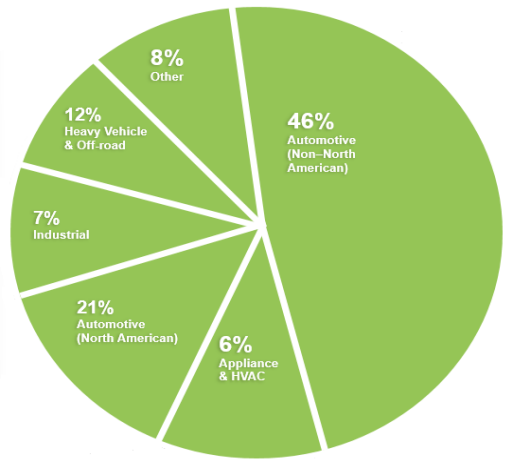

Figure 3 Highest Revenue Percentages per Application

The industries in which they compete vary from industries such as aerospace, automotive, construction, or energy, to fields such as agriculture and medical industry. Their sensors can be found in commercial jets, large boats, automobiles, construction vehicles, but also in people's homes. At the moment, the biggest percentage revenue of the company comes from the automotive industry, as we display in Figure 3[7].

In Sensata, the departments are divided based on the technology handled by each of them. This research takes place in collaboration with two of the departments specialized in the automotive industry, precisely the MEMS (Micro-Electro- Mechanical Systems) and MSG (Metal String Gages) departments.

The MEMS department focuses on low pressure sensors, but also on other various intake and exhaust components. For this specific group the production plants are located in Malaysia and China, while the MSG group handles manufacturing in Bulgaria as well.

The production sites are organized by using the farm concept. This is based on the existence of multiple processes (each process with a certain number of machines) and on different types of sensors having different paths through the production plant. Due to their similarities we associate the farm concept with the job-shop concept. Since no universal flow applicable for all products exists, the complexity in assigning products to machines is rather high, compared to, for example, traditional flow shops.

At the moment, the managers handle the strategic capacity planning through an Excel model. According to the company, due to its simplicity, this model does not consider a variety of factors they deem to be important. For example, regarding the farm concept, approximately 20 different routes are considered in their model, although more actually exist. Averages are used for similar routes, rather than creating a different category for each route. Various other differences are currently not considered individually, but used as average values in the model.

Although the capacity managers wish to add incorporate additional factors, the Excel model is close to its maximum computational limits, making it very complicated for any modification to be implemented.

1.2 Research Plan

In this section we describe the plan for this research. Section 1.2.1 states the motivation and the objective of this research while in Section 1.2.2 we describe the problem encountered by the company. Section 1.2.3 contains our research questions and Section 1.2.4 connects the research approach to the framework of activities we introduce in Section 1.2.3. Section 1.2.5 mentions the limitations of this research by describing the project scope.

[7] HVAC - Heating, ventilation, and air conditioning

1.2.1 Research Motivation and Objective

Considering the MEMS department, the demand forecasts show that the request for their sensors is going to double in approximately one and a half years. The longer horizon forecasts show that in the next four years the demand will continue to grow rapidly. Given these forecasts, the company needs to be adequately prepared to handle this growth. Therefore, it becomes crucial to know precisely how much and when they need to order new equipment.

Considering that the model Sensata currently uses for capacity planning is very simplistic and makes use of a variety of averages, the reasons for initiating this assignment start with the aim of redesigning this model. Hence, the desired output for the project is a new model for the strategic capacity planning of the two automotive departments. A new model represents a way of predicting the future requirements, in terms of equipment, in a more adequate and precise manner.

At the moment their model simply checks whether the demand fits the available capacity. However, for the outcome of this research we are looking for an algorithm that also considers the allocation of tasks to machines. We wish to implement an optimization technique so that the new algorithm can predict the loading percentage of a machine in a more accurate manner. By loading percentage of a machine, we refer to the ratio between the time needed to produce the forecasted demand and the available time.

1.2.2 Problem Description

As we state in Section 1.2.1, the company's marketing department has predicted a very rapid increase in demand for the next few years. Because of what they call a steep ramp-up curve was forecasted for their demand, the capacity managers state that they have to add new production equipment each quarter.

The automotive industry is based on the just-in-time concept, which the company explains as having to deliver the components requested just before they are needed in the manufacturing line of their customers. The need to deliver in time arises from the fact that, if some components are not available, the production of the specific car will be stopped (it is either too costly or nearly impossible to continue producing a car in spite of lacking few components). Hence, if a late delivery occurs, the company has to pay fines for each day past the deadline.

Considering the increase in demand that the company is facing and the strict deadlines they have to meet to produce all the demand, it becomes more and more important to know the precise moment when new equipment is required. Moreover, knowing that ordering new equipment involves long lead times (around 1.5 years), the expansions need to be planned accurately and in advance.

It is obvious how ordering a new machine too late can affect the chances of meeting the demand requirements in time. However, ordering additional equipment too early is not a feasible option either. The managers state that because of Sensata's nature of a stock market listed company, postponing investments for as long as possible is the most desired option.

From the company's perspective, one of their biggest challenges is to accurately determine the amount of additional equipment needed, accompanied by the times when the orders need to be placed. Hence, this becomes one of the main focuses of this research.

Based on the details we obtain from the company, we describe the problem bundle in the diagram from Figure 4. The capacity managers believe that their current model can function according to expectations in a stable/static environment. By stable/static environment we refer to a constant level of demand, which also implies a constant amount of equipment and unique products. If both the equipment and the unique products are kept to a constant amount, this leads to a constant level of process flows. By process flow we refer to the route a product takes through the production facility, precisely the combination of machines needed for its manufacturing.

Since the demand has greatly increased in the past couple of years, we cannot consider this environment static. Thus, the combination of the very simplistic model and the nature of the environment leads to inefficiencies/ non-reliable results in the current model. An example of such unreliable results is highlighting certain machines/process as overloaded without this being actually the case.

Considering the upcoming years, the demand forecasts show an even bigger increase. This implies that an increase in the number of equipment, unique products and process flow will follow. As a consequence, the current environment will become less stable and will further accentuate the inefficiencies already present in the model.

When analysing the context in more detail, some of the simplifications that lead to unreliable results are: only using the average machine performance per process and considering fewer routes than sensors actually take through the production plant.

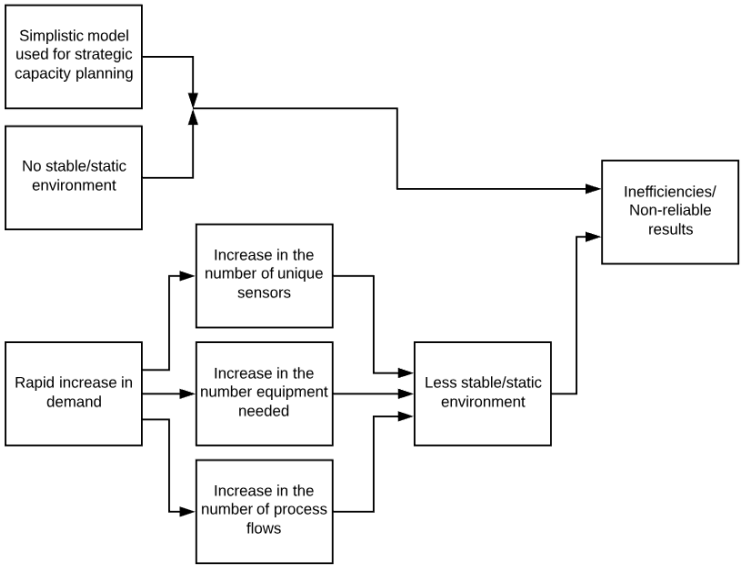

Figure 4 Problem Bundle

Another such example is based on the fact that in the automotive industry customer approval is required for any product – machine combination. A device can only be produced on the machines already approved by the customer. If the company wishes to produce a certain product on a new machine, a so-called machine release is necessary. Through machine release, we mean that, although not allowed before, a new product – machine combination can now be used. By releasing new equipment, a certain product can now be manufactured on an increased number of machines, without the need of further investment in additional equipment. A quality testing procedure is required for the customer to approve this release; this procedure takes around 3 months.

The customer releases are not taken into consideration in the current model, although the capacity managers believe that being aware of the available capacity that can be used for a certain product through the release of machines can avoid further additional investment.

1.2.3 Research (sub) questions

We define the research questions having in mind the concepts of action and knowledge problems, and trying to follow the timeline of activities we set for this assignment. We base the division of the research questions on the stages illustrated in Figure 5.

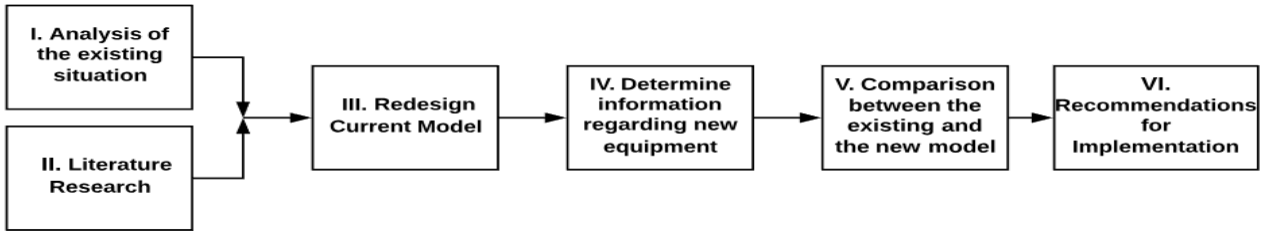

Figure 5 Framework of the assignment

I. Analysis of the existing situation

The first step of this research is to better understand the existing situation, precisely to analyse the existing capacity planning model. Next, we look into the additional factors that have the potential of bringing an improvement. For this stage we define the first two research questions.

The new model should be used to determine information about the amounts of new equipment required and the times when the orders have to be placed. However, before reaching this stage, research needs to be done on the most efficient way to do this. The third question serves this purpose. We answer these questions in Chapter 2.

1. *How is the capacity planning currently handled within the MEMS department?*
    a. *What data and factors are currently included in the model?*
    b. *What simplifications are being used in the model?*
2. *What factors are currently excluded from the model?*
    a. *Which one of these factors does the company consider crucial for the new model?*
3. *What factors/data need to be considered for determining the number of machines the company has to order and the time the orders need to be placed in order to keep up with the upcoming increase of demand?*

II. Literature Research

Our next step is to conduct a literature review. The most relevant concepts we consider for this assignment are: capacity planning, production planning and resource allocation. Therefore, the goal of this stage is to find models related to such tasks. We cover these aspects in Chapter 3. We choose the following research questions for this stage:

4. *What models are covered in literature that could serve as a basis for redesigning Sensata's capacity planning algorithm?*
5. *What optimization techniques can be used as a basis for the strategical decisions regarding capacity planning?*

III. Redesign Current Model

During the first three stages we show in Figure 5 we collect the information we need for designing a new model. Therefore, in this stage we focus on this new model and answer the sixth research question.

Instead of buying new equipment every time the demand of a certain product increases, the company has the option of releasing other machines. According to the company, they prefer to not always order new machines because of the high investment required. For this purpose, we define the seventh research question. Chapter 4 covers the methods used for redesigning the existing model, while Chapter 5, where we introduce our results, covers the answer for question 7.

6. *How can the existing algorithm used for planning the machine loading be redesigned?*
7. *What machines can be released for a certain product to avoid ordering a too high amount of new equipment every time the demand is increased?*

IV. Determine information regarding new equipment

In the end, after performing research, designing a new model and also checking what machine releases are possible, we answer question 8 in Chapter 5.

8. *How many machines does the company need to order and when should these orders be placed to properly handle the upcoming increase in demand?*

V. Comparison between the existing and the new model

Once we redesign Sensata's capacity planning model, we need to perform a comparison between the old and the new model. First, we look into the criteria for checking the differences between the two, especially in terms of results. Further on, we check these differences and the results obtained from both models. For this stage we define the following research questions:

9. *What are the criteria on which we compare the 2 models?*
10. *What are the differences between Sensata's model and its redesigned version?*
11. *What results are obtained from the new model and how accurate are these results compared to the outputs of Sensata's model?*

VI. Recommendations for Implementation

As we describe in Section 1.2.6, one of the limitations of this assignment is the fact that providing the company with a piece of software ready to use right away is not feasible. Therefore, the outcome of this research represents a proof of concept for an optimization algorithm to serve their purposes. Since the actual implementation is not part of our scope, Chapter 6 gives a list of recommendations on how to approach the implementation. Hence, the contents of this chapter include the answers to the following question:

12. *What steps/guidelines should Sensata follow for the complete implementation of the new algorithm?*

1.2.4 Research Approach

This research consists of multiple stages as we illustrate in Figure 5. The first three stages of the project involve performing research. This research refers both to Sensata's case, covered by the first two research questions, and to a literature research. Through this literature review we aim to find models that can be used for allocating and scheduling products to machines. In this way, we address the strategic decision of how much new equipment is needed and when do the orders need to be placed, in a more accurate manner. The information collection techniques for these first stages vary from literature research and analysing the current Excel algorithm to interviews with the capacity managers of the MEMS and MSG departments.

Once we complete the research part, in stage 4, we aim to redesign the current model by making use of the information we previously gathered. The answer for the sixth research question represent the methods through which we design a new capacity planning model.

In order to determine the total number of new equipment needed and the time when the orders should be placed, when redesigning this model, we consider releasing both existing and new machines to certain products, and aim to answer the seventh research question at this point.

The fourth stage represents determining the details (quantity, time) regarding additional equipment and hence answering question 8. Next, in the fifth stage, we perform a comparison between the new model and the initial one. The final stage of the project involves introducing recommendations both for the future research and strictly for the implementation of the algorithm within the company.

1.2.5 Project Scope

Based on the details we present so far, this assignment covers a great variety of factors to consider and issues to become aware of and aim to solve. This being the case, one of the limitations of this research is the fact that the outcome of this project does not consist in a finished capacity planning tool.

The ideal result that we can obtain is creating a proof of concept, in the form of an algorithm, which takes into consideration the important aspects we discover during this research. Due to the big amount of data available, we decide to test the model on a smaller subset of the existing data. We base the comparison between the existing and the new model on the results obtained from both models while using the same data subset.

1.3 Outline of the Report

We organize the remainder of this research as follows. In Chapter 2, we describe the current situation by giving more details about the production process, the capacity planning model and issues Sensata is currently facing. We identify the factors that the capacity managers consider crucial for the new model. We explain the literature available on our topics of interest and give an overview of the available models that could fit our purpose in Chapter 3. Chapter 4 presents the way in which we construct the new model while in Chapter 5 we describe our experimental design and results. Chapter 6 highlights the conclusions and recommendations of this research.

Chapter 2 Analysis of the Current Situation

This chapter focuses on describing the current situation the two departments are dealing with. Section 2.1 contains a process and system description, which includes various details about the production plants, the sensors produced, and the machines used for manufacturing. Further on, Section 2.2 first describes the existing capacity model in detail, followed by an overview of issues we encounter in this model. These issues refer to the simplifications the managers' use in their model and the results that follow. Section 2.3 gives a list of factors that the capacity managers believe should be considered for the redesigned model.

2.1 Process and System Description

When it comes to the automotive industry, Sensata produces custom-made sensors used in various parts of a car. These parts vary from the engine, air conditioning and transmission to parts such as the exhaust system and tires. According to the company, more than 50 sensors produced by Sensata can be used in an automobile.

Considering the MEMS group, they handle the production of around 500 different sensors, which are divided in 20 different categories. The sensors produced by this department are mainly pressure sensors, custom-made based on their clients' requirements. For the year of 2017, the forecasts showed that the MEMS group should produce 13 million sensors. The demand shows a rapid growth, 30 million being the prediction for 2019. This trend continues over the next couple of years, being forecasted to reach 54 million in 2022.

Considering that the sensors are custom made, no two sensors are identical, so a high level of variation between the sensors can be found. The sensors are divided based on their output: either analogue or digital, based on the type of connector required, etc.

As mentioned in Chapter 1, the MEMS group manufactures their sensors in two productions plants located in China and Malaysia. Both facilities are designed based on what the company refers to as the "farm concept". This implies that no traditional production lines are present in these manufacturing sites. For each device, a combination of various individual machines/processes is required, hence, not all sensors follow the same processing flow. One could state that each sensor has its own routing through the manufacturing facility; some products skip a processing step, while others have additional steps required by the customers. After a machine processes a device, the device is loaded on a trolley and transported to a waiting area. The product remains there until it can move on to the next processing step.

Besides the multiple stages and processes through which the sensors must go, for each processing step there are also multiple machines available. Each device must go through only one machine per process; however there are also differences between the machines within the same process. They consist in differences in machine performance, the equipment varying in age and generation.

We associate the farm concept with a job shop. Just as in a job shop, the individual machines are grouped based on their functionality and different products follow different routings within the facility. We illustrate this concept in Figure 6.

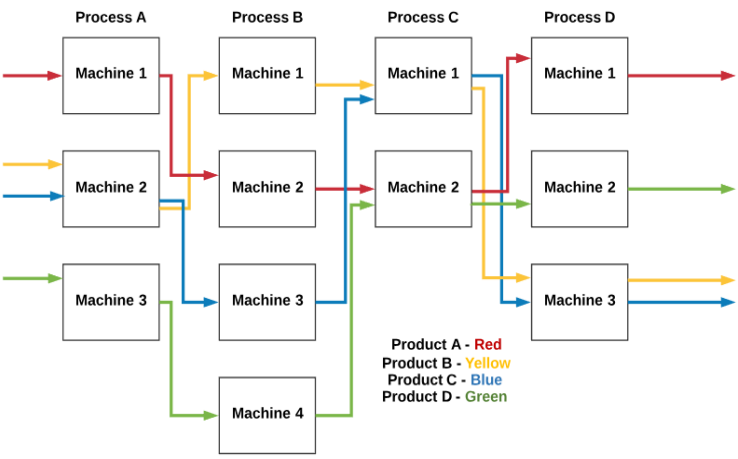

Figure 6 Farm Concept - Job Shop Concept

As we state above, each device has to go through multiple stages, precisely 3 stages referred to as front-end, calibration, and back-end. Apart from calibration, the other two stages are composed of multiple processing steps. In the front-end there are roughly 5 different steps, while in the back-end there are around 10. The variation is increased here even more, since, for example, there are several types of back-end. This means that depending on the type of sensor produced only some of the steps are necessary.

Figures 7 and 8 display an example of such a sensor, however as we mention before, almost each sensor differs from the others. Figure 7 shows the parts of the sensor assembled in the front end, also referred to as the clean room. For this sensor the basic idea is that on the top two components an adhesive pattern is applied and then they are pressed down on the black plastic carrier. To harden the adhesive, the sub-assembly is oven cured, and, in the end, this sub-assembly is being visually inspected. Figure 9 shows these steps in more detail.[89]

Figure 7 Part of a sensor assembled in the front end (cleanroom) of the production process

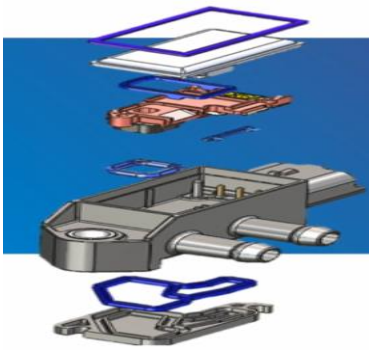

Figure 8 Part of a sensor assembled in the back-end of the production process

[8] SEA – Sense Element Assembly
[9] PCB – Printed Circuit Board

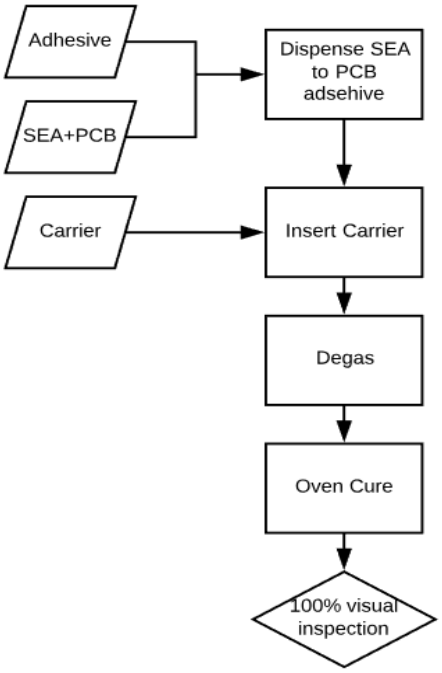

Figure 9 Steps that the sub-assembly from Figure 2 undergoes in the front end (cleanroom) of the production process

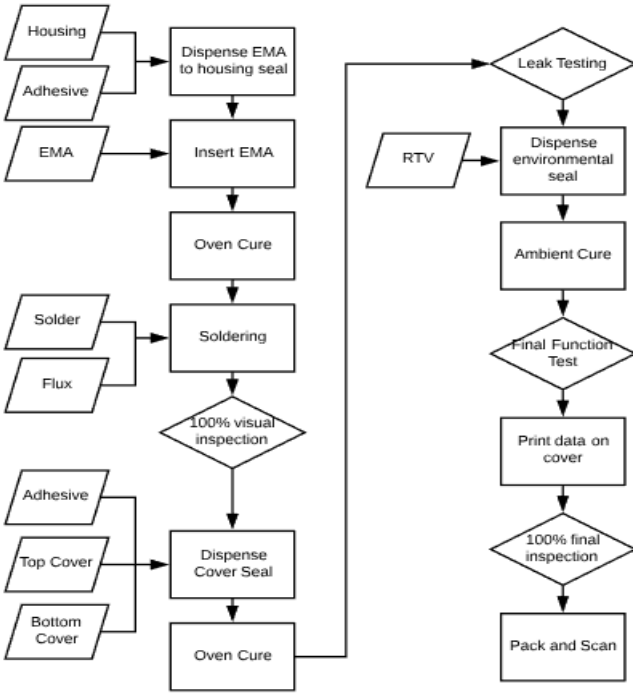

Figure 10 Steps that the sub-assembly from Figure 4 undergoes in the back-end of the production process

Once the part from Figure 7 is assembled in the front end, the next stage it goes through is calibration. The calibration stage consists of only one processing step; therefore each device will only go through one single machine. Once the calibration part is done, the component arrives in the back-end of the production process.

Figure 8 shows the back-end assembly for the same device as in Figure 7. Figure 10 shows the steps this device undergoes in the back-end stage of the production process.[10] [11] The part we previously display in Figure 7 is now shown in Figure 8, the fourth level from the top. This part of the sensor is fixed in the casing (sixth level) with adhesive layers. Further on, the top and the bottom cover are also attached with adhesive. Oven curing is performed with the same purpose as before. Besides the visual test, in this case a leak test is also performed by applying pressure on the sensor and a function test is done to ensure the proper functioning of the final device. Details about the product are engraved on one of the covers.

Figure 11 shows an overview of the manufacturing facility in Malaysia.

[10] EMA – Electronic Module Assembly
[11] RTV – Room Temperature Vulcanization

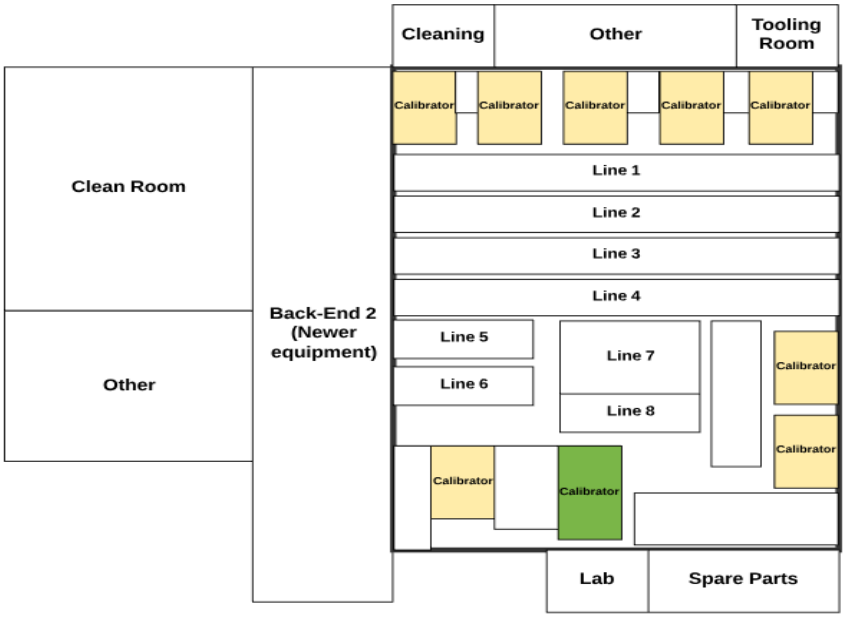

Figure 11 Manufacturing Plant - Malaysia

The front end of the production process is located in the left of the plant and referred to as the clean room. This bolded rectangle represents the initial location of the back-end of the production process. This is the area where the production lines were firstly located and as the need for new equipment increased, additional machines were placed in what we refer to as "Back End 2". We highlight the calibrators as the beige rectangles on the sides of the bolded rectangle.

In the bolded rectangle, we find both calibrators and equipment belonging to the various production lines. Considering the calibrators, Figure 11 displays differences in size between the green one and the rest. These differences represent technical differences, between different generations of equipment. Considering the production lines, we see total of 8 lines in the bolded area. These lines vary in the type of equipment they contain and in the type of sensors they produce. Again, just by looking at the diagram in Figure 11, we notice the differences in size between the lines. The first four lines are the oldest ones. Each production line contains a variety of machines used for manufacturing and one or two ovens. The ovens are used for stabilizing the adhesive layers applied on the sensors. Looking at the other four lines, the size reduction is the result of a new machine design. Most of the processing steps performed by the individual machines from the oldest lines were combined by Sensata's engineers in one single machine.

The difference in size between the newer four lines, lines 5-8 from Figure 11, comes from the addition of different types of equipment in each of them. The old lines and the newly designed machine perform the basic tasks to manufacture a device, however if the client desires the addition of a certain element, new equipment is required.

According to the company supervisor, they try to keep the arrangement of machines to resemble a line for simplicity and better organization. However, the processing flows through the facility are based on what we describe above as the farm concept or the job shop concept.

2.2 Planning and control description

As we mention in Chapter 1, the current planning of the machine loading is performed with the help of a very simplistic Excel model. The managers refer to this model as a high-level capacity planning in which they simply check whether the forecasted demand can fit the available capacity. We associate this with the strategic level of capacity planning, which we describe in the upcoming chapter.

The inputs used in the Excel model are the demands forecasted for each of the 20 sensors types. For the equipment of the 3 production stages (front-end, calibration and back-end) parameters such as CT (Cycle Time) and OEE (Overall Equipment Performance) are set. The cycle time refers to the time it takes for a machine to process a certain device, while the overall equipment effectiveness is used to evaluate how effective each resource is being utilized. The algorithm is then used to calculate the loading for each process in a month.

We describe this algorithm in more detail in Section 2.2.1. We provide an overview of the issues in Section 2.2.2 and a list of relevant factors, excluded at the moment, in Section 2.2.3.

2.2.1 Algorithm Description

In order to answer our first research question ("*How is the capacity planning currently handled within the MEMS department?*"), this section contains a more detailed description of the current model. We describe the contents of the most relevant Excel sheets. For each sheet, we mention the data contents and, where applicable, the formulas used for various calculations. For some of the sheets we provide various figures to better visualize their contents. These figures do not show the actual sheets, and the numbers they display represent random values we choose. Moreover, we do not show these sheets in their full size, so more products, processing flows and time periods (months, years) exist in the actual model.

Model Description

1. **Demand Sheet**

Figure 12 shows the contents of the monthly demand sheet. The most important elements are a list of all product IDs together with their demand for each month of the upcoming 5 years.

		Year	2018				
		Month	January	February	March	April	May
Already on the Volume Sheet?	ProductID						
Yes	1			600		400	
Yes	2		300			200	500
Yes	3		500	1000	600		
No	4		200	400	1300		800
No	5		600	900			700

Figure 12 Monthly Demand Sheet

The capacity manager obtains this information from the marketing department of the company. This demand data can be split in the following 2 categories:

- approved orders: the orders for which contracts have already been arranged with the customers and for which they know for sure that they need to produce the requested volume
- unapproved orders: for these orders the bidding on winning the contract is ongoing between Sensata and its competitors, therefore the capacity manager does not know for sure whether or not they will have to accommodate such volumes. For each bidding, a winning probability is set by the marketing department.

According to the capacity manager there are two ways of dealing with this uncertainty. The first one is to multiply the volume of each unapproved order with its probability and consider the resulting amount as the volume that they need to produce. The second option is to set all the probabilities to 100%, therefore

consider the unapproved orders as approved and accommodate the full volumes of each such order in their model. The capacity manager chooses to apply the second option. The reasoning behind this is that he prefers to have enough equipment to produce more than needed rather than have shortages if more orders are accepted. Moreover, another reason for choosing the second option is related to his past experience. He state that in general more orders are approved than rejected.

The yearly demand is calculated for each sensor by adding up the monthly demand per product ID. Except the demand for each individual product, this sheet contains one more demand row which is used to include the production of samples in the monthly planning. A product ID, chosen by the capacity manager is associated, with these samples. As we describe before, for any new product or any change in the production process the company performs a testing procedure, creating sample sensors, which are then checked by their customer. To not overlook the production of such sensors the capacity manager includes a constant demand value for monthly samples.

A simple check is performed here to see if any new products have been added to the list. This check involves comparing the new list of product IDs with the old one available in the "Volume" sheet (see "Already on the Volume Sheet?" column). Whenever a new product is found ("No" appears on the first column), it's ID, demand and the process flow needed for its manufacturing are manually added to the "Volume" sheet.

2. Volume Sheet

Figure 13 shows the template of the volume sheet. The sheet shows to which process flow each product belongs. A process flow is defined as the route (combination of machines) a product follows through the manufacturing facility. The products are considered over the rows and the flows over the columns and, at first, a value of 1 is assigned to the correct product-route combination. As we mention in Chapter 1, around 20 different routes, the mostly used ones, are considered in the current model. For the sensors having a different route, the capacity manager select the most similar one from the 20 available flows.

				Year	2018									
				Month	January					February				
	Processing Flow				Make Site		Processing Flow			Make Site		Processing Flow		
ProductID	1	2	3		China	Malaysia	1	2	3	China	Malaysia	1	2	3
1	0	1	0		1	0		200		1	0		600	
2	0	0	1		0	1			300	0	1			
3	1	0	0		0	1	500			0	1	1000		
4	0	1	0		1	0		200		1	0		400	
5	0	0	1		0	1			600	0	1			900
				Total										
				China			0	400	0			0	1000	0
				Malaysia			500	0	900			1000	0	900

Figure 13 Volume Sheet

The same sheet provides information about the location where each device is currently being produced. The locations available are the two production sites from China and Malaysia. Just as before, the capacity manager assigns a value of 1 to the correct product – facility combination.

Further on, for each month of the upcoming 5 years the total volume (bottom values under "Total") that needs to be produced by a certain processing flow is being calculated. This is done by assigning each sensor's demand to its corresponding processing flow and summing up the values over each flow.

Another sum, which we do not include in Figure 13, shows the total amount to be produced in each of the 2 manufacturing sites.

3. Cycle Time and Overall Equipment Effectiveness Sheet

The different types of equipment used in the 3 stages (calibration, front end and back-end) of the manufacturing process are considered here. For each equipment type the CT (Cycle Time) and OEE (Overall Equipment Effectiveness) values are given. These values represent an average of the individual values for each machine belonging to an equipment type.

In some cases, the capacity manager considers differences in equipment performance, but not within the machines of the same equipment type. Such differences arise from certain products requiring different settings, which leads to considerable differences within equipment performance. For example, the calibration stage differs between sensors. Therefore, three equipment sub-types are used to highlight this difference and for each of them the CT and OEE values are given. The OEE value takes into consideration aspects related to both the machines themselves and the products that need to be manufactured on each machine. Such aspects are: yield loss, which is dependent on the product-machine combination, maintenance and downtime which are strictly related to the machine itself. No division between machine and product related factors are available.

Additional information such as the total output per day that can be produced using one equipment type and the output that can be produced by a single machine are also shown. Figure 14 shows the contents of this sheet. We consider only one equipment type, the calibrators.

						Year	2018				2019			...	2018			2019		
		Current Measures				Month	January	February	March	...	January	February	March	...	January	February	March	...	January	February	March	...
General	Equipment Type	CT	OEE	Output per day (all machines)	Output per day (1 machine)		CT				CT				OEE				OEE			
Calibration	1	12	70%	26000	5200		12	12	12		12	11.5	11.5		70%	70%	70%		70%	70%	70%	
	2	7	78%	48000	9600		7	7	7		6.5	6.5	6.2		78%	78%	78%		78%	78%	78%	
	3	14	75%	28000	4000		14	14	14		14	14	13		75%	75%	75%		75%	75%	75%	

Figure 14 Cycle Time and Overall Equipment Effectiveness Sheet

4. Equipment Quantity Sheet

In this sheet the capacity manager uses the same template as in the previous one. Figure 15 illustrates the equipment quantity sheet. Instead of the CT per month of each year, we see the quantity of each equipment type. In this same sheet we also notice the increase in the number of machines over time. For example, the number of calibrators available is predicted to increase from 6, in 2018, to 10 by the beginning of 2020. The capacity managers manually increases these amounts if an equipment type seems overloaded.

The company keeps a 15% buffer to be able to cope with unexpected production, so the capacity manager considers a machine type to be overloaded when the loading percentage (calculated by using the monthly loading formula from below) is above 85%. However, from the management's point of view, if a request for additional equipment is placed at 85%, it normally gets denied. Such a request usually gets approved when the loading is forecasted to reach 100%.

						Year	2018			2019			...			
		Current Measures				Month	January	February	March ...	January	February	March ...		Process Flow		
General	Equipment Type	CT	OEE	Output per day (all machines)	Output per day (1 machine)		Quantity			Quantity				1	2	3
Calibration	1	12	70%	26000	5200		5	5	5	5	6	6		1	0	1
	2	7	78%	48000	9600		5	5	5	6	6	7		0	1	1
	3	14	75%	28000	4000		7	7	7	7	7	8		1	1	0

Figure 15 Equipment Quantity Sheet

In this same sheet, the equipment types are assigned to the process flows. The equipment types are listed over the rows and the process flows over the columns and a value of 1 is given to the routes the equipment belongs to.

5. Equipment Loading Quantity Sheet

In this sheet the total volume that needs to be produced in a month is calculated for each equipment type. This is achieved by summing up the monthly demand forecasted for the process flow to which a specific equipment type belongs. These calculations are also performed for all the months of the current year and a couple of upcoming years.

6. Equipment Loading Percentage Sheet

This sheet is the one mostly used in deciding the quantity and the right time of ordering additional machines. It contains the monthly loading percentages, which are calculated by using the following formula:

$$Monthly\ Loading\ \% = \frac{Equipment\ Load\ (Quantity)}{Monthly\ Capacity}$$

The equipment load (quantity) represents the volume (total number of units) that each equipment type needs to produce. All the values in this formula are considered per equipment type.

2.2.2 Summary of issues

Considering the first research question, the second point listed was: "*What simplifications are being used in the model?*". In this section we offer details about these simplifications. Furthermore, we give a short overview of the results/consequences obtained by using this model.

As we state in Chapter 1, the managers make use of various simplifications to keep the size of the current capacity planning model under control. This is due to the current need of being able to manually control and update the model. This need is a result of the lack of an optimization algorithm.

By simplifications we mainly refer to the averages used in the model instead of the actual values. Other simplifications consist of not using any of the factors we introduce in Section 2.3.

Part of the simplifications we find, when analysing the Excel model, are introduced in the following list:

- the sensors are grouped into 22 types and any other individual difference between the 500 sensors is ignored.
- for each processing step the differences between machines are not individually stated, the model only using the average performance of all the machines from a process. In this way, all the machines belonging to a processing step are, in fact, considered as one.
- the multitude of routings present through the manufacturing facility is also simplified by only considering the most common ones (approximately 20 different routes).

Apart from these simplifications, the current capacity planning model does not consider a variety of factors. At first, these factors were not included because the demand was not so high, so a simple check of whether it fits in the available capacity was enough from the company's perspective. However, considering the demand increase they are currently experiencing, this model becomes more and more limited, since no algorithm or optimization technique is actually included. Furthermore, at the moment, more factors are not added also because of Excel's limitations. The model's computational speed has already become an impediment, so adding more conditions will just decrease it further on.

To highlight the results obtained by making use of simplifications in the current model, we mention some of the issues encountered by the MEMS department. These issues are:

- because the machine performance is considered as an average of all machines from the same process, some machines with lower performance are expected to produce more than they can, while others are scheduled to produce less than they actually could
- because the process flows of the sensors are not individually considered, for some processes the model states that the machines are overloaded when that it is not the case. Furthermore, at times, there are more sensors using the same processing step than the model takes into consideration, so in reality the process is overloaded while the model does not show that
- by not considering possible machine releases in the current model, the company is guided to order additional equipment, although it might not be needed.

2.3 Factors

This section gives an overview of the factors the managers exclude from the current model, as an answer for the second research question ("*What factors are currently excluded from the model?*").

- **Location Accessibility**: At the moment, the Excel model contains information regarding the current manufacturing location of a device. However, considering the example of the MEMS department, some of the sensors can only be produced in China, some in Malaysia and only a few flexible ones can be produced in both places. The capacity manager believes that it is important to track where each product can be manufactured since this information could facilitate the exchange of products from one facility to another. By this we mean that if the capacity does no suffice at one of the plants and a device is allowed to be produced in multiple locations, the model could plan ahead to switch the production from one site to another.
- **Customer Releases**: According to the company supervisor, in the automotive industry only machines and raw materials approved by the customer can be used to manufacture the products requested.

In order to gain the customer's approval, samples of the sensors are built, and they undergo an extensive lab testing lasting approximately 3 months. Once the customer has given his approval no change can be made to the manufacturing process without the customer's consent and another round of lab testing. For example, no other machine can be used for this device and not even changing the location of the machine through the production facility is permitted.

- **Correct Machine Performances:** The performance of the individual machines is averaged into one value per equipment type. It is possible for machines to belong to the same process and have different performance levels, due to age and machine generation. However, the problem becomes more complicated when the difference in performance levels are so high that a 100% difference is found. At times this is the case when the latest machine generation has a new design which leads to double the output of the older ones. Hence, the capacity managers wish to have more accurate values highlighting the differences between machines.
- **Machine Capabilities:** Different machines can produce different types of sensors. Some of the equipment can produce only analogue sensors, while other can only produce digital. According to the capacity manager of the MEMS departmen most of the volume that needs to be produced consist of digital sensors, however the company also owns a considerable amount of equipment to manufacture analogue sensors. The company wants to include this differentiation in the new model to have a better overview of capacity limitations.
- **Tooling Set Availability:** To be able to manufacture a product on a certain machine, tooling sets are required to attach the sensors. The availability of these tooling sets is limited, but no details about this are included in the current model.
- **Machine Lead Time:** The process of ordering additional equipment and testing the sample products can last approximately 1.5 years.

  The steps a department has to go through to obtain the needed capital start with providing internal justification for the new investment. This stage lasts around 3 months and once the order is placed the department must wait for 1 year, which is the lead time from the equipment supplier. Once the equipment is in their manufacturing sites the process of getting the customers' approval takes approximately 3 more months. Based on the very long lead time the department has to plan their orders very accurately, by not only considering their need of additional equipment, but also trying to postpone an investment by as much as possible.

Chapter 3 Literature Review

We summarize Sensata's problem as not having a proper capacity planning algorithm. The current model does not include a variety of factors and does not use any optimization techniques. We describe the factors that are important for Sensata's managers in Chapter 2, and in this chapter, we focus on our literature review.

The general terms we discuss in this chapter relate to the main concepts we mention in Chapter 1. Such concepts include capacity planning, machine loading and allocation of resources.

Section 3.1 gives a detailed classification of manufacturing systems. Section 3.2 contains details regarding capacity planning and its relevance while Section 3.3 and 3.4 describe various models related to capacity planning. Section 3.5 focuses on the machine loading problem and Section 3.6 focuses on scheduling problems. In Sections 3.7 and 3.8 we describe Simulated Annealing and Tabu Search individually, while in Section 3.9 we discuss a comparison between them. In the last section we further describe the concept of Overall Equipment Effectiveness.

3.1 Classification of manufacturing systems

In our literature research we encounter various articles with applications strictly focused on certain types of manufacturing systems. In this section we aim to associate Sensata's case with the most suitable types of manufacturing systems. Given the various terms we encounter in literature, we search for details regarding each term and the category to which it belongs. The terms we most often find in papers are: discrete manufacturing, single machine problem, flow shop, job shop and make - to - stock manufacturing system. We enlist the most relevant categories below.

Classification based on volume and flexibility (Caramia & DellOlmo, 2006)**:**

- **Continuous production:** involves product processing without interruption and makes use of a clearly defined sequence of steps through which each product has to pass (Shah, 2016). According to Chatterjee (2012) the definition of continuous manufacturing, from an engineering point of view, implies the simultaneous material charging and discharging from the processing stages. Examples of its applications vary from the pharmaceutical industry to the processing of food and natural gas (Chaudhary, Pazhayattil, & Spes, 2017).
- **Mass production:** is used for the manufacturing of similar products for which the production process is standardized. This method is used when the desired output represents large volumes of such products. Due to the level of standardization required for this type of manufacturing system, mass production is also referred to (repetitive) flow production (Kenton, 2018).
- **Batch manufacturing:** refers to the simultaneous production of multiple units. The products are grouped in batches and are moved from one stage to another. Furthermore, all the units in one batch are simultaneously processed in each of the stages.
- **Project manufacturing:** implies the manufacturing of one-of-a-kind products which require a very complex production process, therefore being thought of as a whole project (Shah, 2016).
- **Discrete manufacturing:** unlike mass production, it handles the manufacturing of noticeably different units and unlike continuous production, each processing stage can be individually performed at different times. The automobile industry is one example of discrete manufacturing

(Shah, 2016) and the application we are interested in; therefore, we enlist a further division of this category below.

According to Caramia and Dell'Olmo (2006), discrete manufacturing can be further divided in the following sub-categories:

- **Single machine:** in this model only one machine performing all manufacturing stages is available. The decomposition into separate operations is therefore not necessary.
- **Identical processors:** also referred to as the "parallel machines" model by Maccarthy and Liu (1993), implies the usage of a certain number of identical machines and the fact each job can only be processed by a single machine.
- **Flow shop:** different machines are available for this manufacturing system, the common aspect of all products being the sequence of processing stages. As MacCarthy and Liu (1993) state: "each job has an identical flow pattern".
- **Job shop:** as we previously explained in Chapter 2, a job shop contains different machines, and, unlike a flow shop, each product manufactured can have a different processing flow.
- **Fixed position shop:** in a fixed position shop the product is not the one moving from one processing stage to another, but the processing stages are being performed around the units.

Another classification we are interested in, due to its various mentions in literature, is the one based on operational objectives as McCarthy (1995) presents it. Similar categories were mentioned as production control systems by Shah (2016). These categories are:

- **Make - to - stock:** This type of systems involves manufacturing units ahead of time, based on very accurate demand forecasts. The production is therefore planned based on what is believed to sell in the future and the units manufactured are stored in inventory. Once an order is received, this order will be fulfilled from the units available in inventory.
- **Make - to - order:** For this type of system, every product is manufactured after a customer order was received and based on each customer's requirements. Therefore, such units are custom-made and increase the variety existent in such a production facility.

3.1.2 Sensata's Case

Taking into consideration all the categories and details we provide above, we first classify Sensata' case as a discrete manufacturing system. Their devices are noticeably different since they are designed based on the client's requirements. From the sub-categories we define for the discrete manufacturing we believe that our case fits in the job shop category, as we previously explain in Chapter 2.

Regarding the last classification we describe, we believe that the make - to - order production system is the one that most accurately fits Sensata's case. Once a customer order is approved they start the actual production of units.

3.2 Capacity Planning

This section introduces the concept of capacity planning. We further discuss three different levels of capacity planning in Sections 3.2.2, 3.3.3 and 3.2.4. Section 3.2.5 relates Sensata's case to what we believe is the most appropriate level.

Martinez-Costa et al. (2014) state that one of the definitions for the capacity of a production system relates to "the volume of products that it can generate in a given period". However, the authors claim that the volume of products manufactured by a production system is highly dependent on the product-mix. Hence, they define the capacity of a production system as "the availability of various types of production resources" rather than the volume it can manufacture in a certain period of time.

Capacity planning represents an important aspect for any company, considering that inadequate management of capacity can affect the performance and financial prospects of any business (Bloomenthal, 2019). As mentioned in the review paper by Martinez-Costa et al. (2014) "in the long term, capacity planning supports strategic business plans of new process technology and new products".

Considering manufacturing companies, Zijm and Buitenhek (1996) state that capacity planning is important because the available capacity (resources) of a company can restrict its production capabilities in terms of volume (Zijm & Buitenhek, 1996). The automotive industry represents one of the most relevant manufacturing industries, since almost no other industry has such a high economic relevance (Volling, Matzke, Grunewald, & Spengler, 2013) .

From the literature review, we can distinguish 3 different levels for capacity planning: strategic, tactical, and operational. These are the same as the ones found for operations management and they should be clearly noticeable within any organization.

3.2.2 Strategic Level

From a general point of view, the strategic planning is the way through which a company defines the strategy/direction it must choose in order to continue growing and stay ahead of competitors. These plans are related to the business as a whole and do not consider individual parts/departments. Decisions considered at this level refer to the market they compete in, the level of investment they can afford for the future and collaborations they can set up to achieve their goals. The time horizon considered by strategic planning is long term, normally varying between 3 to 5 years (Riskope, 2014). According to Martinez-Costa et al. (2014), strategic planning is concerned with "changes in the facilities along the long and medium term, typically several years".

In the review paper of Martínez-Costa et al. (2014), the authors enlist various decisions related to strategic capacity planning. Few examples of such decisions relate to capacity size, allocation, inventory, backlogging, workforce and financial planning.

3.2.3 Tactical Level

Unlike for the strategic level, the time horizon for which tactical plans are being developed is medium-term, typically one year. At this level, managers decide the tasks for each part of an organization and aim to align these tasks to ensure a proper fit with the strategic goals (Riskope, 2014).

The important actions for this level refer to order acceptance and due date quotation for each accepted order. Furthermore, managers dealing with the midterm planning should also be responsible for checking the loading of the available resources. This action is useful to determine the feasibility of the plan whenever a new order has been accepted.

3.2.4 Operational Level

The operational planning shows in which way are the tactical decisions going to be carried out. Operational decisions are considered on the short term, sometimes being considered on a daily basis (Riskope, 2014). Thus, the job scheduling, day-to-day planning and resource allocation (equipment, employees, and materials), fall under the operational capacity planning (Kant, 2014).

3.2.5 Sensata's Case

As we previously mention in Chapter 1, the goal of this research consists in redesigning the strategic capacity planning model currently used within Sensata. We fit this model into the strategic level because of the very high-level of detail considered by the capacity managers. Furthermore, the decisions resulting from this model are the amount of new equipment that needs to be ordered, and the time when the orders need to be placed. Considering that these decisions are related to expansion and investment, we associate them with the decisions previously described for the strategic level.

3.3 Capacity Planning Models

This section introduces literature algorithms focused on the concept of capacity planning. At this point we are interested in the types of objective functions and constraints, related to capacity planning, available in literature.

The first model we consider is based on the one developed by Chen and Fan (2015). They aim to solve a capacity allocation problem in a make-to-order environment with limited resources. This research was developed as a case study for a factory handling semiconductor packaging and testing. The manufacturing system is composed of different production lines, each divided in three production stages. For each stage multiple machines are available in each production line. Each product has to go through all the three production stages and no constraints regarding the allocation of products to machines are imposed. Each machine is able to produce any type of product.

The main decisions composing their problem refer to resource migration and capacity allocation. Resource migration is defined as machines or tools being re-allocated between production lines. In order to solve this capacity allocation problem, the authors chose maximizing the net profits as the objective function. In this case the net profits were calculated based on the following formula:

$$Net\ Profits = Revenue - Production\ Cost - Migration\ Cost$$

To model their problem, the authors formulate a two-stage stochastic programming model with various types of constraints. The five types of constraints we are mostly interested in are:

- Capacity constraints: in this case they limit the workload to the total available resource limits. The workload is defined as the amount to be produced (for one product type) multiplied by the time (work hours) that specific product type needs to use the resource for.

- Material constraints: are defined in a similar manner to the capacity constraints, being strictly related to the (raw) materials rather than the machines or workforce available.
- Production capability constraints: mention whether or not a certain product is allowed to use a certain resource.
- Demand fulfilment constraints: show that the demand/volumes forecasted to be sold should equal the amount produced for each product type.
- Service level constraints: sets the ratio between volume produced and customer demand equal to the customer service level.

Considering the results of this algorithm, besides the maximum value of the net profit, the number of units produced for each customer in each time period is one important overview.

In the article of Siddharth et al. (2011) the authors describe a model in which aspects of scheduling and order acceptance are being used. The research takes place at the request of a gear manufacturing company from the US. The manufacturing system under consideration resembles a job shop in which recirculation is allowed. Recirculation is defined as jobs being allowed to be processed by the same resource multiple times. Each job, in this case customer order, is composed of multiple operations for which precedence constraints are present. Furthermore, for each operation the processing times are known and for each job a fixed deadline, sale price and process route are given.

The authors aim to develop a model for deciding whether an incoming order should be accepted and if accepted, when its processing should be planned. They define a mixed-integer linear programming model having as objective function the maximization of the overall net profit. The net profit is defined as the difference between revenue and manufacturing costs.

Since data such as due dates and precedence constraints between operations are taken into consideration, we tend to associate this research with a scheduling problem. The most relevant constraints from this model are:

- Capacity constraints: make sure that the available capacity, in hours, of a resource is not exceed in any period of time. Such constraints can also include overtime when allowed by the problem description.
- Resource allocation constraints: make sure that the total hours of a resource allocated for producing any operations equals the operation's processing time.
- Shift duration constraints: the processing of each job should be less than the duration of a shift, in this case 8 hours, in any time period.
- Precedence constraints: processing of operation o is not allowed to start before the completion of operation o-1.
- Order Feasibility: before accepting any order, check whether the completion time of its last operation exceeds the job's due date. If so, it is impossible for the job to be completed in time.

In their results, the authors show that the computation time of the model increases drastically with the increase in the number of operations and therefore the number of jobs. They state that for an instance of 5 jobs and 8 operations, CPLEX could not achieve the optimal value even after 16 hours. Furthermore, not

only the running time increases with the increase in the number of operations, but also the deviation from optimal results, referred to as optimality gap.

Kim and Kim, (2001) define an extended linear programming model based on what they refer to as the "traditional production planning model". They define a new version of this model because, according to them, the classical model uses capacity constraints that might not reflect the actual production behaviour in an accurate manner.

The classical linear programming model they refer to is the model of Byrne and Bakir (1999), which aims to minimize the sum of various costs while considering capacity and inventory constraints. The modifications brought to this model by Kim and Kim (2001) start by redefining one of its decision variables. Instead of using the amount of product i which was produced in period t as a decision variable, the authors change it to the amount of product i which needs to start in period t. Further on they introduce a new parameter referred to as the "effective loading ratio". This parameter represents the proportion of the amount of product i that needs to start in period t which is part of the workload assigned to machine k in that same period. By introducing this parameter, they are able to define a formula computing the total workload per machine per period of time.

Another parameter they introduce is the "effective utilization" per machine per period of time. This parameter is defined as the "proportion of the total capacity of the machine available to process the start quantities during the period". This parameter is used to calculate the adjusted capacity of each machine in each period. The adjusted capacity equals the effective utilization multiplied by the capacity of each machine in each period.

The authors claim that their version of the model represents a more realistic vision of the production process. Furthermore, they combine this linear programming model with a simulation algorithm. By using such a combination, they obtain a production plan as output of the LP model and check it through the simulation model. If the values obtained for the quantities to be produced in every period are similar, then the model stops. Otherwise, they re-run the linear programming model with the values for the effective loading ratio and utilization obtained from the simulation model.

3.4 Aggregate Production Planning

Weinstein and Chung (1999) introduce an evaluation model of maintenance policies. Their approach starts with producing aggregate production plans by using a linear programming (LP) model. Next, a master production schedule is used to minimize the deviations between a selected goal and the aggregate plans. In the last step they use rough-cut capacity planning to determine the loading requirements per work centre. Rough-cut capacity planning techniques are used to help the managers in creating "a trade-off between the expected delivery performance and the expected costs of exploiting flexibility by using nonregular capacity" (Hans, Herroelen, Leus, & Wullink, 2007). Besides, the authors also state that during the order acceptance stage, RCCP techniques can be useful for weighing the consequences each decision can have on the production stage.

We are mainly interested in the models they develop for the first two stages, therefore we do not describe the third stage further on.

The first phase, namely the aggregate planning model is performed with the aim of minimizing certain costs by considering capacity and, in their case, maintenance restrictions. Weinstein and Chung (1999) state that "the aggregate production planning model described in this article uses a mixed-integer linear programming (LP) formulation to determine the aggregate policies for the rate of production, inventory, regular and overtime workforce, and workforce smoothing activities."

The aggregation starts with grouping the end products in product families and considering the product families in the actual LP model. Furthermore, the authors make two assumptions for simplicity. First, the resources (equipment) function at maximum efficiency, therefore no idle time is considered. Moreover, no yield loss is present either. The second assumption relates to raw materials and states that all raw materials are available whenever needed, therefore no time waste can result from lacking components.

The first set of constraints they consider for this model are the capacity constraints, which state that the regular working time and the overtime available is enough to complete all production. The list of constraints continues with an inventory balance equation and a similar work balance restriction. The last constraint states that the amount of overtime required for one period should not exceed a certain percentage of the regular working time from that period.

In the second phase, the master production scheduling, the authors formulate another LP model. This stage disaggregates the aggregated plan obtained in the first stage. Disaggregation results in specific timing and sizing requirements for each product individually. The output of this stage is a master production schedule that does not violate capacity constraints.

The objective function remains focused on minimizing a sum of various types of costs. The constraints in this case are:

- Inventory balance equation: similar to the inventory constraints we previously describe. The difference is that, in this research, backordering demand is a possibility. Therefore, the inventory balance equation also takes into consideration the number of backorders associated with each type of product.
- Workforce capacity: limits the total labour capacity to the summation between standard working hours and overtime. Such constraint can be formulated for individual manufacturing facilities or for all the facilities together.
- Overtime capacity: limits the number of hours which can be used in overtime to the value specified in the previous step of the model, the aggregate plan.
- Production volume: ensures that the summation of the quantities that need to be produced in each disaggregated period of time is equivalent to the total value stated in the aggregated plan. They compute the value stated in the aggregate plan for each product family, therefore the summation of the disaggregated information is also computed accordingly.
- Inventory volume: similar to the production volume restriction, this restriction relates the disaggregated inventory values to the aggregated total.

Another research performed in a make-to-order environment is the research of Neureuther et al. (2004). This paper introduces a three-tiered hierarchical production plan designed for a make-to-order steel fabrication site. The production facility contains 14 machines used to produce thousands of different items.

The first stage of the hierarchical production model consists of a linear programming model the authors use to obtain an aggregate production plan. The aggregation takes place at item level, the thousands of items being grouped in 12 product archetypes. The inputs they use at this stage are monthly aggregated demand forecasts.

The objective function used is related to the minimization of relevant costs. Similar to the ones Weinstein and Chung (1999) introduce the constraints for this model are: inventory balance equations and capacity constraints. However, this time, the capacity constraints are divide in multiple types:

- Facility Storage Limit: size of the products under production plus the ones in inventory should not exceed the available capacity of the entire plant.
- Capacity of Equipment: limit the number of items produced to the total available monthly limit.
- Capacity of Workforce: similar to the one related to equipment, such constraint ensures that the limit of total hours available, both regular and in overtime, is not exceeded.

The output from this stage, the aggregated plan, together with the monthly forecasts for each individual product are inputs for the second stage. According to Neureuther et al (2004), "the key to any disaggregation model is to ensure that the production quantities determined in this model agree with those dictated by the aggregate model". The second stage is a non-linear model that assigns each product to an available process plan. Its objective function aims to minimize the total set-up costs. The most relevant constraint is the one disaggregating the information from the first stage. It does so by ensuring that the number of units to be produced for each product group (sum of the number of units to be produced for each individual product belonging to that product group) does not exceed the value from the aggregate plan. The output they obtain from this stage is a weekly disaggregated production plan.

The third stage represents a master production schedule for which the processing of the 12 product types is sequenced on machines available in the factory. For each week they create such a schedule with the aim of minimizing the makespan. They solve the scheduling algorithm in C++ through enumeration. One of the reasons for them choosing enumeration is that even though there is a big number of possible combinations, the computers available could easily handle the task. Moreover, they state that enumeration makes it possible to identify the worst combination.

The authors claim that their model can be applied to any make-to-order manufacturing system, no matter the number of products or types of processes present in the factory. They state that the most important factor for this model is the proper determination of the archetypes in which all the individual products are grouped.

3.5 Machine Loading Problem

In this section we provide information about the machine loading problem. We describe this problem because we believe it has certain common characteristics with Sensata's case.

As stated by Stecke (1983), the main production problems are: part type selection problem, machine grouping problem, production ratio problem, resource allocation problem, and loading problem. The one we focus on in this section is the loading problem, which Stecke (1983) defines as follows: "Allocate the operations and required tools of the selected part types among the machine groups subject to technological and capacity constraints".

Grieco et al. (2001) provides similar definitions for both the loading concept and the loading problem. Singh, Singh, and Khan (2015) give another definition which states that the loading problem refers to "tooling individual/group of machine(s) to collectively accomplish all manufacturing operations concurrently for all part type in a batch".

As Singh et al. (2015) state a solution to the loading problem should output the processing flow for each job. In this case, by processing flow we refer to the sequence of machines a certain job has to follow. Furthermore, the output should include the tooling associated with each machine. The restrictions of such a problem are related to capacity constraints and technological constraints, such as individual machine capabilities.

Taking into consideration the factors and conditions we describe in Chapter 2 as being important for the new version of the capacity model, the machine loading problem is similar in terms of capacity restrictions and machines' capabilities. Furthermore, we associate the machine loading problem with a sequencing of jobs model.

3.6 Scheduling Problems

We start this section by defining the scheduling concept. Further on, Sections 3.6.1 describes the basic concepts behind scheduling problems.

Scheduling is considered to be one of the difficult tasks in industry. According to Mckay et al. (1998) there are several reasons behind its difficulty. The first reason the authors state in this paper refers to the non-static behaviour of set-up and processing times (which we refer to as cycle time). The authors mention that differences can appear even between the same unit being processed on the same piece of equipment and making use of the same human resources.

The second reason refers to allowing pre-emption and changing the schedules when an urgent task needs to be prioritized as other factors which increase the scheduling complexity. Furthermore, another aspect they state is the miscommunication between top management and the production managers. The authors refer to cases such as senior managers accepting customer orders without consulting with the production managers regarding the available resource/capabilities.

3.6.1 Job Shop Scheduling Problem

According to the literature, the job shop scheduling problem is one of the most important and difficult to solve combinatorial optimization problem. This problem is classified as an NP-hard problem and various articles mention various attempts of solving it being available in literature (Çaliş & Bulkan, 2013).

The same authors describe the context of the job shop scheduling problem as consisting of two finite sets, one of jobs and another one of machines. Each job consists of multiple operations and needs to be processed by certain machines, according to its processing flow. Each machine has the capability of processing a single operation at a time. Normally the goal is to assign a sequence of operations to each machine which minimizes a certain criterion (Pham & Klinkert, 2008). The criterion we found as the one most mentioned in papers refers to the minimization of the makespan. By makespan, the majority of articles refer to the maximum value of the completion times of all operations (Dell'Amico & Trubian, 1993).

For a more complex context explanation, we find various assumptions and restrictions regarding this problem in literature. Based on Dell'Amico and Trubian (1993), the assumptions for the traditional job shop model are:

- processing times (which we refer to as cycle times) are known and fixed
- all jobs are available for processing in the beginning of the time bucket considered

Furthermore, the most common assumption, which we encounter in the majority of articles, is related to pre-emption and states that no pre-emption is allowed. By pre-emption we refer to the "temporary interruption of a task with the intention to resume this task later" (Techopedia, n.d.).

The constraints we find in literature are divided in 2 categories. The first one refers to the machines capabilities of only processing one job at a time, therefore they are non-interference restrictions as Manne (1960) describes. The second category Manne (1960) mentions are precedence relations between operations of the same job, referred to as sequencing restrictions.

3.6.1.2 Job Shop Scheduling Techniques

Due to the multitude of techniques available for approaching the job shop scheduling technique we decide to make a comparison to better review them. The factors we mainly focus on refer to the size of the instances for which each method can be used and the respective computational time. Moreover, we are searching for an algorithm with low computational time that given our problem size, can achieve a decent solution in a low time frame. We do not necessarily focus in finding a method that can reach the optimal solution because from on our research we conclude that these methods require both high computer capabilities and computational time.

In Table 1 we present various techniques available in literature for solving the job shop scheduling problem. We base the information presented in this table on the paper of Morshed, Meeran and Jain (2017).

Optimization Procedures		
Name	**Description**	**Remarks**
Mathematical formulations	The most well know method for this case is using mixed integer linear programming. The review paper states that for obtaining good results, a relaxation of the problem is required (Lagrangian relaxation).	The authors call such methods “inadequate” for solving job shop scheduling problems. Even by applying the Lagrangian relaxation the procedures seem to require high computational effort and the results present large deviations from the optimal value.
Branch and bound	This method represents the best one out of the enumerative procedures. This method is based on the dynamic construction of a solution space resembling a tree structure. The method starts with the root node (the highest location in the tree) and ends once all the solutions have been analyzed. This implies that the algorithm has reached the lowest node in the tree and has fathomed the worst solutions.	According to this review paper the most common way of solving a job shop scheduling problem through branch and bound is to decompose the set of operations into one machine problems. The algorithm solves each of these instances and chooses the bottleneck machine as a bound for the next iteration. The articles reviewed in this paper shows that even the method considered to be the best for branch and bound requires high computing time and cannot be used for solving large instances.
Approximation Procedures		
Name	**Description**	**Remarks**
Priority dispatch rules	For this algorithm, every iteration the operations to be scheduled are assigned a priority weight and the operation with highest priority is selected for scheduling. The weights are assigned based on a certain criterion, such as: shortest or longest processing times. This method is an approximation method since it involves choosing one operation to add to the already existent sequence, instead of evaluating all possible combinations as an enumerative method would do. Beam search represents a middle ground between approximation and enumerative methods, where couple of solutions are analyzed every iteration, as opposed to just one or all.	The authors state that algorithms making use of a single rule based on which to prioritize the operations have limited capabilities. Therefore, various extensions showing different levels of priority classes used by the same method have emerged. Although we could consider such a technique for our case, the article mentions that such techniques are “more suitable as an initial solution technique rather than being considered as a complete system”
Bottleneck based heuristics	The Shifting Bottleneck Procedure is the most well-known method from this category. It is similar to branch and bound in terms of decomposing the whole problem into one machine instances and solving each individual one iteratively. In this model each such instance is ranked based on its result and later	Just as for branch and bound, the best algorithm from this category designed by Balas and Vazacopoulos requires high computing efforts.

	on compared to the others. The chosen one for every iteration is the bottleneck machine, the one having the largest value. The chosen machine is then included in the existent schedule and the other problems are again solved in the next iteration.	
Local Search & Meta-heuristics		
Name	**Description**	**Remarks**
Simulated Annealing	Represents a search method having its roots in the physical concept of annealing ("heating a material above its recrystallization temperature, maintain a suitable temperature for a suitable amount of time and then cooling"). This method is based on creating a neighborhood structure, by for example swapping two consecutive job. The result obtained from this swap is computed and compared to the best-known value until that point. If the new result is better than the best value, then the new result becomes the best value. The innovative aspect of Simulated Annealing is that with a certain probability it also accepts neighborhood structures with worse results, to avoid getting stuck in a locally optimum point. This probability decreases, so towards the end the method avoids accepting worse solutions.	The trickiest part when it comes to Simulated Annealing is defining the right neighborhood structure. Furthermore, the authors of the review paper state that in order to reach good solutions, excessive computational times are required for this method as well.
Tabu Search	It is a search procedure which stores part of the search history in its memory. A number of most recent solutions are stored in what is called a tabu list, which is used to avoid repeating or similar solutions. Just as Simulated Annealing it also uses a neighborhood structure and the most recent moves are marked as forbidden and stored in the tabu list. Each solution from the tabu list receives an aspiration criterion, which allows it to be selected as feasible if it reaches a certain level of quality.	Just as for Simulated Annealing, one of the biggest challenges when it comes to this method is defining an appropriate neighborhood structure. The conclusion given by the review paper when it comes to Tabu Search is that is outputs the best results out of all techniques and is able to so with decent computational efforts.

Table 1 Job Shop Scheduling Techniques

3.7 Simulated Annealing

Simulated Annealing represents an optimization method originating from the metallurgic industry. A material undergoing the annealing process is first heated until its fusion point. Once it is liquefied, its temperature is gradually reduced until it is brought back to a solid state. This method was first proposed in 1983 and the authors state that this process starts by melting the system in question, followed by slowly reducing its temperature until it freezes and therefore no further change can be performed (Kirkpatrick, Gelatt, & Vecchi, 1983).

Starting from an initial solution, in each iteration, Simulated Annealing creates a new solution from the neighborhood of the current one. If the change in the objective function improves the current solution value then the transition from the current solution to the neighboring one is accepted with a probability of 1. In a minimization problem, accepting a solution with a smaller objective function than the current one refers to performing a downhill move. However, to prevent getting stuck in local minimum, Simulated Annealing also accepts uphill with a certain probability. An uphill move represent an increase in the current objective function. The acceptance probability of uphill moves is calculated from the $\exp(-\frac{S'-S}{T_k})$ formula (Park & Kim, 1998). In this formula S' represents the objective function of the neighboring solution while S represents the objective function of the current solution. Next T_k represent the temperature at iteration k. At the beginning, the model will accept almost all transitions, uphill or downhill. However, as the temperature decreases, the acceptance probability for uphill moves decreases as well. Hence, towards the end, the model will only accept improving solutions which in the case of a minimization problem are the downhill transitions.

3.8 Tabu Search

Tabu Search represents an optimization procedure used for solving combinatorial optimization problems.

According to Glover (1986) its applications vary from graph theory to solving mixed integer programming models. This method resembles a hill-climbing heuristic which improves the solution unidirectional up to a local optimum. The limitation of a hill-climbing heuristic is that the local optimum found by the algorithm might not coincide with the best solution available, the global optimum. Because this heuristic only accepts solutions improving the objective function, once no improvement is available the method stops, leaving part of the solution space unexplored. To overcome this limitation, Tabu Search guides the search past the local optimum.

In a similar manner as Simulated Annealing, starting from an initial solution Tabu Search generates a list of neighbor solution of the current one. A neighboring solution is generated from the current solution by performing a single move. Once a solution is accepted, the move leading from the current solution to its accepted neighbor is stored in a tabu list. This list has the role of preventing the algorithm from moving back to previously explored areas. Once a move was added in the tabu list it is considered forbidden. However, the moves in the tabu list do not necessarily remain forbidden until the end of the algorithm. A move only remains forbidden for a number of iterations equal to the tabu tenure. The tabu tenure is defined as "the time, measured in terms of iterations that must elapse for a node to be removed from the tabu list" (Rolland, Schilling, & Current, 1997).

Besides waiting for a number of iterations equal to the tabu tenure, one can also overwrite the tabu status of a certain move. This can be done by using an aspiration criteria which determines whether a move can be performed although being in the tabu list. According to Rolland, Schilling and Current (1997) the most common aspiration criteria is overwriting the tabu status of a move if by performing it the value of the objective function is better than the best known solution.

3.9 Simulated Annealing versus Tabu Search

In the paper of Morshed, Meeran and Jain (2017), Tabu Search is presented as being able to achieve the best results. However, since our problem does not entirely match the job shop scheduling problem we perform a literature research comparing the two algorithms for different problems.

St-Hilaire and Liu (2011) compare the performance of the two metaheuristics, Tabu Search and Simulated Annealing, for the network topology planning problem. According to the authors, this problem is classified as NP-hard. Besides the two metaheuristics, they also includes the performance of a genetic algorithm in the comparison. They compares the results of the three heuristics with the results they obtain when using CPLEX for a total of 42 different problem sizes. Furthermore, for each problem size three instances are randomly generated.

Their results show that Tabu Search is the best algorithm in terms of finding the optimal solutions. Tabu Search is able to find the optimal solution in 57 out of 126 problems. Simulated Annealing found the optimal solution in 9 cases while the genetic algorithm could only find an optimal solution for 2 of the problems. In terms of running time Tabu Search and the genetic algorithm show a similar performance, while Simulated Annealing requires much higher execution time.

Another research comparing the performance of Simulated Annealing and Tabu Search is the research of Paul (2010). The author compares the two algorithms when solving a quadratic assignment problem, an NP-hard combinatorial optimization problem. His conclusions show that Simulated Annealing can achieve the lowest value for the objective function in 5 out of 6 cases, while Tabu Search can only do so for 3 instances. Furthermore, he also states that that Simulated Annealing can be better for some instances while Tabu Search can be better for others.

Semba and Mujuni (2019) reach similar conclusions. In this research the authors perform a comparison between the 2 metaheuristics and an Ant Colony Optimization algorithm with or without time restrictions. The problem they consider in this research is the school bus routing problem. In the time restricted case the algorithms are allowed to run for a maximum of 1000s. Considering time restrictions, Tabu Search seems to perform worst out of the 3 algorithms while the Ant Colony Optimization algorithm outperforms Simulated Annealing after 300s. However, when no time restrictions are set Tabu Search performs best for 2 out of 3 problems, while Simulated Annealing performs best for the third one.

Based on the findings of these research papers we conclude that each of the two metaheuristics have a good performance for a variety of problems. Furthermore, in terms of which metaheuristic is best we are inclined to believe that it is highly dependent on the problem to solve.

3.10 Overall Equipment Effectiveness

In this section we present a short description of the OEE components. As we introduce in Chapter 2, OEE is one of the factors currently considered by Sensata. Overall Equipment Effectiveness represents a measure of the efficiency and effectiveness of a process. This measure is usually defined through the following formula:

$$OEE = \frac{Number\ of\ good\ units * Processing\ Time}{Total\ Time\ Planned\ for\ Production}$$

By good units we refer to the ones that meet the quality standards and do not require any sort of rework. The processing time used in this formula is the theoretical minimum amount of time required to manufacture a single unit. Although this formula represents an accurate way of calculating the OEE of a process, it does not give any further details related to the three OEE components. These components are: availability, performance and quality. Each of the three component highlights a certain kind of loss. Availability considers any expected or unexpected event that prevents the system from producing. Performance is related to the various aspects that can prevent a system from operating at its optimal speed, while quality relates to the yield loss. The yield loss refers to the amount of parts which do not meet the quality requirements. This category also includes any part that requires rework. (Calculate OEE - Definitions, formulas and examples | OEE, n.d.)

Chapter 4 Model Description

This chapter introduces the heuristic we design to fit Sensata's manufacturing process and the mixed integer linear programming model we use to benchmark our heuristic's results. Section 4.1 provides a detailed description of the problem statement and further details on the decisions covered by our model. Next, in Section 4.2 we introduce the mixed integer linear programming model designed to match our problem description. Furthermore, the same section also contains details regarding the model size and the solver we use. Section 4.3 describes the heuristic we design for solving the strategic capacity planning issues Sensata is currently facing. In the same manner as in Section 4.2, in Section 4.3 we also describe the heuristic's components and the software we use. Finally, in Section 4.4 we state the conclusions of this chapter.

4.1 Problem Description

We start this section by providing a recapitulation of the project's goal. Furthermore we introduce the possible decisions the capacity managers can take in order to keep up with the continuous increase in demand and a detailed description of the existent guidelines regarding the actual production.

The desired output for the project is a new model for the strategic capacity planning within Sensata. The outcome we are looking for is an algorithm that performs a high-level allocation of tasks to machines, by taking into consideration all the restrictions available. We wish to use optimization techniques, so that the new algorithm can predict the loading percentage of a machine in a more accurate manner.

As we mention in Chapter 2, when describing the current situation, the company has to meet a monthly demand for each product. These values fluctuate depending on their customers' requests. Apart from having to produce the demand of each product, the capacity managers also need to consider the yield loss. As we describe in Chapter 3, we associate the yield loss with the quality component from OEE.

Each sensor produced by Sensata goes through three main stages referred to as: front end (cleanroom), calibration and back end. In the front and back end stages, each sensor goes through multiple processing steps (is being processed by multiple machines), while for the calibration only one processing step (machine) is needed for each sensor. Although having to go through the same main stages, when it comes to the processing steps, each sensor can have its own routing. This routing is defined as the sequence of machines required for the production of each sensor. Considering these product-machine combinations two different guidelines have to be considered:

- whether a machine has the physical capability of producing a certain sensor (for example: there are some machines used strictly for either analogue or digital sensors);
- whether a machine is released by the customer to produce his sensors. (A customer release represents the customer's approval of having a certain set of machines used in the production of his sensors. To obtain this approval, Sensata produces sample sensors that are further tested by the customer themselves).

Another aspect we mention in Chapter 2 is the difference in performance between the machines. Such differences mainly appear due to different generations of the same machine. This being the case, the processing time, referred to as cycle time, of the same product on different machines can have different

values. Because of these differences a certain allocation of products to machines can result in insufficient capacity to meet the whole demand, while another allocation can highlight leftover capacity available.

Another OEE component that is relevant for our problem is the availability. Besides differences in cycle time, having various machine generations can lead to differences in the maintenance levels required. Such aspects are part of the OEE and are covered by the availability component. The availability values are used to reflect the percentage of the total capacity during which a machine is actually used for production.

Regarding the capacity aspect, three values have to be taken into consideration. The first one is the number of working days in each month. Sensata's manufacturing facilities have a schedule of 7 working days a week, excluding national holidays. Since their manufacturing facilities are located in various countries (the MEMS department has facilities in Malaysia and China, while the MSG department also produces in Bulgaria), the number of national holidays in a month varies. Hence, so does the number of working days in a month.

The second value to look at is the number of hours available for production in a day. Normally, the manufacturing facility functions 24 hours a day, however approximately 1.5 hours are used for breaks and shift changes.

The last relevant value in terms of capacity is referred to as the buffer level. We compare this buffer with the safety stock concept, however without having any physical stock. This buffer represents a percentage of the total monthly capacity that should not be used and is introduced so that the company can deal with sudden fluctuations in demand. Although the capacity managers would like to keep the buffer level set to 15%, this is not always possible due to differences in opinion between them and the higher management. Therefore, in some of the months this threshold is exceeded and the capacity managers consider the process as overloaded.

In Section 4.2 we show how we include all the guidelines we mention above in the mixed integer linear programming model, while in Section 4.3 we show the same for the heuristic we design.

4.1.1 Decisions Considered

In order to help Sensata cope with the increase in demand, the capacity managers have three different decisions available. These decisions are referred to as: building inventory, releasing existing machines and ordering a new machine. Each decision is associated with one of the following costs:

- costs of holding one unit in inventory
- costs of releasing an existing machine for one product: releasing an existing machine for a certain sensor implies building sample units on that machine and testing them to make sure they meet the specifications (the sample validation costs are usually 20.000-30.000€)
- costs of buying a new piece of equipment: Apart from the costs of the equipment itself (buying a new calibrator will cost 1million €) when a new machine is purchased, the releasing costs are also implied since this machine has to be released for production.

Besides the costs, another difference between releasing an existing machine and purchasing a new one is the lead time incurred for each of the two actions. As we mention in Chapter 2, releasing an existing machine has a lead time of approximately 3 months, while ordering a new machine has a lead time of approximately 15 months, excluding the 3 months required for releasing it.

4.2 Mathematical Model

Based on the literature review, the guidelines we introduce in Section 4.1, and the three main decisions available for the capacity managers, we build a mixed integer linear programming model for strategic capacity planning. This model is able to create a capacity planning overview and highlight which decision(s) should be chosen in which month.

The project's scope involves creating a proof of concept for a strategic capacity planning algorithm to be used by Sensata's capacity managers. We decide to create this proof of concept in the form of a heuristic, which we describe in Section 4.3. To benchmark the results we obtain from our heuristic we decide to set up this mixed integer linear programming model and use a commercially available optimization software to solve it to optimality.

In this section we provide a detailed explanation of the components of this model, followed by details regarding its size and the solver we use.

4.2.1 Model Description

The MILP serves as validation for the heuristic we design. This model highlights which actions should the capacity managers consider during a certain planning horizon. The actions available refer to the three decisions: building inventory, releasing existing machines and purchasing a new piece of equipment, which the company can take to cope with their forecasted increase in demand. Each of these actions has an associated cost therefore whenever one action is chosen, its cost is added to the total cost. The model's objective is to minimize the total spend for the chosen planning horizon.

By aiming to minimize the total spend, the model actually tries to prevent any kind of investment that is not needed. Moreover, since ordering a new machine can require a big investment for the company, the model tries to postpone such an investment by first trying to build inventory or release the existing machines.

The model's objective is subjected to various restrictions, which vary from meeting the demand of each product and not exceeding the capacity of any machine to restrictions stating the machines on which each product is allowed to run. Regarding the decisions variables, unless they require binary values, we allow most of them to be continuous.

4.2.2 Modelling Choices

Before starting to describe the mathematical model, we discuss the motivation behind some choices we make when designing the mixed integer linear programming model.

The number of decision variables of a model has a direct effect on the computational difficulty of an integer programming model. Later in this section we further discuss about the number of decision variables, and offer an example showing the exact values. Furthermore, considering that the goal of the project is related to the strategic level of planning, we do not consider constraints modelling aspects such as precedence relationships between products on the same machine or the machines required for the manufacturing of one product to be relevant. Based on these arguments, we make the decision of solving the MILP model for each machine type at once.

Once the samples are approved by the customers and the mass production process starts, Sensata has to produce different volumes of these sensors for a couple of years. For some customers they have to deliver monthly or weekly volumes, while for others they can be requested to deliver 2-3 times a week or even on a daily basis. Because these times can vary from customer to customer, the capacity managers in charge of the high-level overview, take into consideration the monthly demand for each sensor. In this model we choose to do the same, after having various interviews and being explained about these delivery requests, the moment when the real data becomes available (replacing the forecasts) and the fluctuations in demand which, although not allowed, can still occur on very short term.

The last modelling choice we address is the existence of so called "dummy" machines in our model. These machines represent the machines that can be added by the model, whenever a new machine has to be purchased. For these machines we duplicate the inputs: cycle times, availability and capabilities, of some of the existing machines and set all their releases equal to 0 (before being purchased they should not be released for any of the products). Another aspect we find out during the interviews is that Sensata normally uses the same supplier for the same machine type and that they do not have multiple options of the same equipment to pick from. This being the case, we make all the "dummy" machines identical.

4.2.2 Sets and Indices

We define the following sets in our model. The first set, I, contains the list of all sensors under production starting from month 1 to month T. The second set, K, contains the list of all machines, belonging to the machine type for which we solve the model. Just as for products, the list contains all the machines existing in the manufacturing facility from month 1 to month T. However, besides these machines, set K also contains the "dummy" machines we previously mention. The last set, M, contains all the months belonging to the planning horizon (e.g. if the input data is available for 2 years then the months will take values from 1 to 24).

- I = Products: {1,..i…,I}
- K = Machines: {1,..k…,K}
- M = Months {1,..t…,T}

4.2.3 Parameters

Below, we list all the parameters we choose for our model. These parameters serve as inputs for the mixed integer linear programming model. We define these parameters considering the data relevant for the company. The only parameter we define strictly for modelling purposes is the "big-M" which facilitates the formulation of the so called "big-M constraints".

Costs:

Considering that the company's goal is to postpone a big investment, such as ordering a new machine, for as long as it is feasible, we define all three cost components as time dependent. The following holds for all three cost components: performing any action in month t is more expensive than performing the same action in month t+1. At the moment, we consider the costs for keeping an item in inventory in month t, regardless of the product, to be equal. The same applies for releasing a machine in month t, no matter for

which product. Moreover, purchasing a new machine has the same costs, regardless of which machine is chosen from the fictive machines we create.

- **InventoryCost(t):** inventory costs associated with month t
- **ReleasingCost(t)**: releasing costs associated with month t
- **OrderingCost(t):** ordering costs associated with month t

Products:

- **D(i,t):** demand of product i in month t
- **InitialInventory(i):** initial inventory of product i. We define this parameter to represent the inventory that exists for each product before running the model.
- **TotalDemand(i,t)** : total demand of product i in month t. This parameter represents the sum of the demand for product i over a certain number of months. We use this parameters to reflect the shelf life of a product and therefore to set a limit on the inventory allowed for product i. The shelf life of a product represents the period of time, following its manufacturing moment, during which a product can be kept as inventory and remain within its approved product specifications. Example: If the TotalDemand(i,t) is the sum of demands for the next year, then the inventory value in each month is not allowed to be more than that. This being the case, no units produced in month t will stay under inventory for more than one year.

Machines:

- **InitialExistingMachines(k)**=
  $$\begin{cases} 1, if\ machine\ k\ is\ available\ at\ the\ start\ of\ the\ planning\ horizon \\ 0, otherwise \end{cases}$$
  This parameter shows which machines exist in the manufacturing facility at the beginning of the planning horizon or have been ordered by the company and their delivery is expected in a certain month. If index k refers to one of the machines that is in the manufacturing facility or already ordered this parameter has a value of 1. If index k refers to one of the fictive machines then this parameter has a value of 0.
- **AvailabilityOEE(k,t)**: availability (OEE component) of machine k in month t. A value is given for all the machines, whether they exist in the manufacturing facility or not. When a new machine is purchased, its availability is taken into consideration for capacity calculations. In some cases, some machines have already been ordered and are known to become ready for production in a certain month. Until month t when they are ready, we set their availability to 0, therefore preventing any products to be allocated on such a machine.

Capacity:

- **HoursAvailable**: number of hours available for production in each day. This parameter represents a constant value for all days in all months.
- **Days(t):** number of days available for production in month t.
- **Buffer:** constant value that we use to model the spare capacity which, in Sensata's case, is kept to cope with fluctuations in demand. This parameter gets values between 0 and 1. Normally the capacity managers aim to have a 15% buffer on capacity.

Product-Machine:

- **CT(i,k,t):** cycle time for product i on machine k in month t. This value represents the processing time of machine k for product i in month t. This value is also time dependent because the company performs improvements on their equipment to decrease the processing times, therefore, the processing time might not be the same in all months.

- **MachineCapabilities(i,k)** = $\begin{cases} 1, machine\ k\ is\ capable\ of\ processing\ product\ i \\ 0, otherwise \end{cases}$
  This parameter is used to show which machine has the physical capability of producing a sensor. This parameter is not time dependent because the physical capabilities of machine cannot be modified (it cannot start processing a product that it could not process before).
- **InitialMachineReleases(i,k,t) =** $\begin{cases} 1, machine\ k\ is\ released\ for\ product\ i \\ 0, otherwise \end{cases}$
  In a similar manner as the initial existing machines parameter, this parameter shows the releases already decided by the company at the beginning of month t. The difference between the two parameters is the fact that these initial machine releases also depend on the month index. We consider the month index here because releases could have already been planned to occur in a certain month t, before running the model.
- **Quality(i,k):** quality (OEE Component associated with yield loss) for product i on machine k. We use this value to determine how many more units of product i should be produced to compensate for the yield loss. Same as the availability parameter, we set a value for all the machines, whether they are available or not.

Others:

- **bigM:** as we previously mention, we define this parameter to be able to model "big-M" constraints. This parameter has a constant value. In order to ensure the proper function of the model, its value should be at least equal to the maximum demand of a product found across the entire planning horizon.

4.2.4 Decision Variables

In this section we introduce the decision variables of our MILP model. They represent the output of our model and can be divided into main decision variables and auxiliary ones. We define the auxiliary variables to facilitate the modelling of our constraints.

Three of our main decision variables relate to the three available decisions: building inventory, releasing existing machines and ordering a new machine. The fourth one is showing the allocation of products to machines, together with the associated volumes.

- **InventoryLevel(i,t)**: variable showing how many units (if any) are kept as inventory for each product in each month t. We use this variable to allow the possibility of, for example, producing part of the demand of month 2 in month 1 and keeping it as inventory.
- **New Release (i,k,t)** = $\begin{cases} 1, if\ machine\ k\ gets\ released\ for\ product\ i\ in\ month\ t \\ 0, otherwise \end{cases}$
  We define this variable to keep track of the machine releases (if any) occurring every month.

- **NewMachine(k,t)** = $\begin{cases} 1, if\ machine\ k\ is\ becoming\ available\ in\ month\ t \\ 0, otherwise \end{cases}$
  We define this variable to keep track of the new machines purchased (if any) every month.
- **Units Produced (i,k,t)**: this decision variable shows the number of units of product I being produced by machine k in month t.

The auxiliary decision variables are:

- **TotalInventory(t)**: variable that we use to calculate the total number of units, for all the products, kept in inventory in month t.
- **ExistingReleases(i,k,t)** = $\begin{cases} 1, if\ machine\ k\ is\ released\ to\ process\ product\ i\ in\ month\ t \\ 0,\ otherwise \end{cases}$ We use this variable to update the releases for each product - machine combination in each month t.
- **TotalMachinesReleased(t):** variable that serves for calculating the total number of new releases (if any) in each month t.
- **ExistingMachines(k,t)** = $\begin{cases} 1, if\ machine\ k\ is\ available\ in\ month\ t \\ 0, otherwise \end{cases}$
  This variable helps to update the status for each machine k in each month t.
- **TotalMachinesOrdered(t):** variable that we use to calculate the total number of machines ordered in each month t.

4.2.5 Objective Function

The objective function for our model is to minimize the costs incurred by performing any of the three possible actions: building inventory, releasing existing machines and ordering new machines, summed over the entire planning horizon.

MINIMIZE

$$\sum_{t=1}^{T}(Inventory\ Costs(t) * TotalInventory(t) + ReleasingCost(t) * TotalMachinesReleased(t) + OrderingCost(t) * TotalMachinesOrdered(t))$$

4.2.6 Constraints

1. Inventory Balance Constraints: we set these constraints to ensure that the demand of each product will be met in time, whether this demand is produced in month t or prior to month t. Since the units produced in the months prior to month t can also be used for month t, in a similar manner, units produced in month t can be used in month t and in the upcoming months.
   We include the quality component of OEE in these constraints in order to make sure that the company produces enough units to both cover the full demand and make up for the yield loss.
   a. Month t=1

   $InventoryLevel(i,1) = InitialInventory(i) + \sum_{k=1}^{K} UnitsProduced(i,k,1) * QualityOEE(i,k) - D(t,1), \forall\ i \in I$

b. Month t>1

$$InventoryLevel(i,t) = InventoryLevel(i,t-1) + \sum_{k=1}^{K} UnitsProduced(i,k,t) * QualityOEE(i,k) - D(i,t), \forall\, i \in I, t > 1$$

2. Defining the variable TotalInventory(t):

$$TotalInventory(t) = \sum_{i=1}^{I} InventoryLevel(i,t), \forall\, t \in M$$

3. Shelf Life Constraint: this constraint limits the amount of inventory the company can hold for each product i to the shelf life of the each product. We consider this shelf life to be 12 months.

$$InventoryLevel(i,t) \leq TotalDemand(i,t), \forall\, i \in I, t \in M$$

4. Releasing Constraint1: this constraint ensures that the model can only release machines that are capable of processing a product i (such machines should have the machine capabilities parameter equal to 1).

$$NewReleases(i,k,t) \leq bigM * MachineCapabilities(i,k), \forall\, i \in I, k \in K, t \in M$$

5. Releasing Constraint2: this constraints limits the machines the model can release in each month only to the machines that are not already released in that month. This applies for each product I and releasing a machine only becomes available from month 4.

a. Month t<=3

$$NewReleases(i,k,t) \leq 1 - InitialMachineReleases(i,k), \forall\, i \in I, k \in K, 1 \leq t \leq 3$$

b. Months t>3

$$NewReleases(i,k,t) \leq 1 - ExistingReleases(i,k,t-1), \forall\, i \in I, k \in K, t > 3$$

6. Updating the Existing Releases variable:

a. Month t<=3

$$ExistingReleases(i,k,t) = InitialMachineReleases(i,k) + NewReleases(i,k,1) + NewReleases(i,k,2) + NewReleases(i,k,3), \forall\, i \in I, k \in K, 1 \leq t \leq 3$$

b. Months t>3

$$ExistingReleases(i,k,t) = InitialMachineReleases(i,k) + \sum_{\pi=1}^{t} NewReleases(i,k,t), \forall\, i \in I, k \in K, t > 3$$

7. Production Constraint1: this constraint helps to make sure that the production of product i is only allowed on machines released for that product. This applies for every month t.

$$UnitsProduced(i,k,t) \leq bigM * ExistingReleases(i,k,t), \forall\, i \in I, k \in K, t \in M$$

8. Defining the variable TotalMachineReleased(t):

$$TotalMachineReleased(t) = \sum_{i=1}^{I} \sum_{k=1}^{K} NewReleases(i,k,t)\,, \forall\, i \in I, k \in K, t \in M$$

9. Purchasing Constraint 1: this constraint helps to make sure that the model can only purchase machines that are not already in the manufacturing facility. This applies for every month t.

    a. Month t<=15

$$NewMachines(k,t) = 0, \forall k \in K, 1 \leq t \leq 15$$

    b. Months t>15

$$NewMachines(k,t) \leq 1 - ExistingMachines(k,t-1), \forall k \in K, t > 15$$

10. Updating the Existing Machines variable:

    a. Month t<=15

$$ExistingMachines(k,t) = InitialExistingMachines(k) + NewMachines(k,1), \forall k \in K, 1 \leq t \leq 15$$

    b. Months t>15

$$ExistingMachines(k,t) = InitialExistingMachines(k) + NewMachines(k,t) + NewMachines(k,t-1) + \ldots + NewMachines(k,1), \forall k \in K, t > 15$$

11. Production Constraints2: this constraint ensures that the production of product i is only allowed on machines that exist in the manufacturing facility. This applies for every month t.

$$UnitsProduced(i,k,t) \leq bigM * ExistingMachines(k,t), \forall\, i \in I, k \in K, t \in M$$

12. Defining the variable TotalMachinesOrdered(t):

$$TotalMachinesOrdered(t) = \sum_{k=1}^{K} NewMachines(k,t), \forall\, t \in M$$

13. Capacity constraints: through this constraint we make sure that the capacity of each individual machine is not exceeded. The total capacity of each machine is equal to the product between the hours available for production in a day, converted to seconds, the number of days available for production and the availability component of OEE of the corresponding machine. From this value, the aim is to keep a certain percentage as buffer, hence the multiplication with (1-Buffer). This applies for every month t.

$$\sum_{i=1}^{I} UnitsProduced(i,k,t) * CT(i,k,t)$$
$$\leq (1 - Buffer) * AvailabilityOEE(k,t) * DaysAvailable(t) * HoursAvailable * 3600, \forall\, i \in I, k \in K, t \in M$$

14. Range Constraints:

$$UnitsProduced(i,k,t) \geq 0, \forall\, i \in I, k \in K, t \in M$$
$$InventoryLevel(i,t) \geq 0, \forall\, i \in I, t \in M$$
$$TotalInventory(t) \geq 0, \forall\, t \in M$$
$$NewReleases(i,k,t) \in \{0,1\}, \forall\, i \in I, k \in K, t \in M$$
$$ExistingReleases(i,k,t) \in \{0,1\}, \forall\, i \in I, k \in K, t \in M$$
$$TotalMachinesReleased(t) \geq 0, \forall\, t \in M$$
$$ExistingMachines(k,t) \in \{0,1\}, \forall\, k \in K, t \in M$$
$$NewMachines(k,t) \in \{0,1\}, \forall\, k \in K, t \in M$$
$$TotalMachinesOrdered(t) \geq 0, \forall\, t \in M$$

Although ILP models are very often used for formulating real-life situations, they are also much harder to solve than LP models. The LP-relaxation of an IP is defined as the LP resulting when ignoring the integrality constraints of an IP. From the feasible region of the LP-relaxation, the feasible solution set for the IP is restricted to those solutions in which some or all of the decision variables take integer values. Therefore, the set of all feasible solutions of the IP model represents a subset of the set of feasible solutions of the corresponding LP-relaxation. (Winston & Goldberg, 2004)

Considering that the feasible region for an IP model is contained in the feasible region of its corresponding LP-relaxation, the optimal objective value of the relaxation is at least as good as the one of the IP. Since having continuous variables results in a less constrained model which might lead to better solutions, we decide that besides the variables which are bound to have binary values we allow all the other variables to be continuous. (Winston & Goldberg, 2004)

4.2.8 Model Size

As we previously mention, we run the MILP model for one machine type at once. The planning horizon varies in our tests from 1 to 2 years. During this time the model decides on the right values of the decision variables for every single month. To illustrate the number of decision variables and constraints composing our model we consider the following data as our input: 260 products running on 9 different calibrators for a period of 24 months. Besides the 9 calibrators we add 6 more as "dummy" machines which can become available starting from month 16. Considering these numbers, the progress tab from AIMMS shows the total number of variables and constraints we use in the model. Figure 16 displays this overview.

AIMMS	: ReadFromFile.ams
Math.Program	: MinCosts
# Constraints	: 482017
# Variables	: 288217 (94320 integer)
# Nonzeros	: 2352143
Model Type	: MIP
Direction	: minimize

Figure 16 Model Size - AIMMS

To demonstrate how such numbers are obtained, we perform the calculations considering the decision variables. In Table 2 we display each main group of decision variables together with their indices and the number of individual variables belonging to each group. In this table, T represents the number of months in the planning horizon, in our case 24, I represents the total number of products, 260, and K represents the total number of machines, 15.

Group of variables	Formula	Number of individual variables
Number Of Units(t,i,k)	$T * I * K$	260 * 15 * 24 = 93600
Inventory Level(t,i)	$T * I$	260 * 24 = 6240
New Releases(t,i,k)	$T * I * K$	260 * 15 * 24 = 93600
New Machines(t,k)	$T * K$	15 * 24 = 360
Total		193800

Table 2 Number of variables for the MILP model – 24 months

Besides the main groups of decision variables, we create auxiliary ones which we use for updating, for example, the possible releases in every month or calculating values such as the total number of releases taking place in month t. We calculate the individual number of decision variables from these groups in a similar manner to the one in Table 2 and the total, in this case, is: 94417. Therefore, the total number of decision variables from our example is 288517, as we display in the overview from Figure 16.

4.2.9 Software and Solver

We use the solver provided in the AIMMS Developer version 4.65 from 2017 to solve our model. This developer represents an extended series of embedded solvers such as CPLEX, CONOPT, GUROBI and Knitro. While, CONOPT and Knitro are highly focused on solving large-scale nonlinear programming models, CPLEX and Gurobi are capable of handling both linear and mixed integer programming models. From the four solvers we mention, the solver we use for our model is CPLEX (version 12.9). Considering our model formulation, the default solver automatically chosen by AIMMS is MIP which refers to a Mixed-Integer Linear Programming model.

4.3 Heuristic

As we previously mention, we decide to create a heuristic as a proof of concept for the goal of our project. We create this heuristic to be able to solve Sensata's capacity issues in a similar fashion as the mixed integer linear programming we introduce in Section 4.2. We design it to follow the same objective and restrictions as in the MILP model. The choices we make, while designing this heuristic are based both on the literature study we conduct, and on the results we obtain from various tests. We introduce these tests in Chapter 5. In this section we provide a detailed description of the entire heuristic.

4.3.1 Model Description

The capacity planning heuristic we design follows the same basic idea introduced for our MILP model. The capacity managers have three different decisions to pick from in order to deal with the demand fluctuations and their goal is to postpone any kind of investment for as long as possible.

Our heuristic tries to achieve this by starting with an initial monthly schedule and applying different methods to ensure a feasible capacity fit. We compare two local search algorithms in an attempt of optimizing the initial schedule.

In our case, the initial schedule refers to an allocation of products to machines for each month of the planning horizon. For this schedule we assign the monthly demand of each product to one or more machines while respecting all the constraints relevant at this point. The constraints we consider here are the ones stating that a product is not allowed to run on a machine which is not released, the capacity restrictions for each machine and the demand restrictions for each product. Since it is possible to not have enough capacity in one month to cover the entire demand, we allocate the amounts which do not fit to an infinite capacity machine. We create this infinite capacity machine in order to keep track of the volumes that could not fit on the existing machines.

Once the algorithm creates an initial schedule we use a local search metaheuristic with the goal of decreasing the overall loading across all existing machines. Whether or not the capacity is sufficient in month t, we perform this optimization step in an attempt of freeing some capacity in case inventory build-up turns out to be needed in future steps. If the algorithm assign products on the infinite capacity machine in month t, once we obtain a new schedule as a result of the local search algorithm we perform a check to see if more volume can now fit in the existing machines.

In case the loading on the infinite capacity machine is still above 0, we move on to our next step, trying to decrease this loading by building inventory. We only apply this step starting from the second month of the planning horizon since we cannot create any inventory before month 1. In this step we simply check if we can allocate any of the volumes currently assigned on the infinite capacity machine to any machine in the prior months. The capacity constraints and the ones stating that a product is not allowed to run on a machine which is not released still hold, followed by the shelf life constraint. This means that we only try to build inventory for a maximum of 12 months prior to month t.

If the loading on the infinite capacity machine is still greater than 0, we perform the next step, trying to release existing machines. At this point there are two reasons for which products might still be assigned on the infinite capacity machine. The first one is the fact that capacity might not be enough to handle the entire demand, while the second one points to the fact that a new product might have been introduced in month t, and no machine releases were arranged for this product. No matter the reason, for this step we perform another local search algorithm in an attempt to decrease the loading from the infinite capacity machine even further.

For the releasing step we make sure that we consider the constraint stating that a machine which does not have the physical capability of producing a certain product cannot be released for that product. Furthermore, we also consider the lead time of performing such a release, therefore this step only becomes available starting from month 4.

As a last resort, if the current capacity turns out to be insufficient, we add another machine. Just as for the MILP, we create some "dummy" machines which the model can make available when needed. In this step we perform one last local search algorithm to check for which products, currently assigned on the infinite capacity machine, should we release the newly bought piece of equipment. We take into account all the constraints related to new machines and also the lead time associated with ordering a new machine. This step becomes available starting from month 16.

4.3.2 Description of Individual Steps

Obtaining an Initial Schedule

We create the following five different ways of obtaining the initial schedule:

1. Assign the products to machines in increasing value of their IDs.
2. Sort the products on decreasing demand value: in this case we assign the product with the highest demand first. This schedule involves no sorting for the machines.
3. Sort the machines on increasing cycle time values: in this case we assign the first product on the machine with the best cycle time. If this machine is also having the best cycle time for another product, we also assign the second product here, without violating capacity restrictions. So, if the whole volume of this product does not fit, we assign the units that do fit on this machine and the rest we assign on the second best machine. This schedule involves no sorting for the products.
4. Sorting the products on decreasing demand value and sorting the machines on increasing cycle time value: this case represents a combination of cases 2 and 3. For this case, we assign the product with the highest demand on its preferred machine (machine with the smallest cycle time).
5. Sort the products on increasing number of releases: for this case we count the number of machines released for each product and we start by assigning the product(s) with least releases.

Since some products might require enough volume to cover the capacity of an entire machine, we prefer to first find space for such big volumes and leave the smaller volumes at the end. Hence, we sort the products on decreasing demand values. We believe that if machines are not completely full when we try to decrease the loading from the infinite capacity machine, it might be easier to find space for a smaller volume than for a larger one. We associate this rule with the Longest Processing Time rule, which is used for finding the minimum makespan of a schedule. By assigning the products with longer processing times first, it makes sure that at the end of the schedule no job with a very large processing time has to be assigned.

When sorting the machines on increasing cycle time, we aim to fit the products on their preferred machine (smallest cycle time) if capacity allows. The combination we choose for case 4 is an attempt of assigning the most important products, which in this case are the ones with higher demand values, to the machines with lower cycle times with the aim of decreasing the loading across all the machines.

For the last case, we choose to sort the products in increasing number of releases in an attempt to minimize the number of releases later on. We test each of the 5 cases for multiple months with different input data and we show the results in Chapter 5.

In Figure 17 we show the logic behind creating an initial schedule, taking case number 4 into consideration.

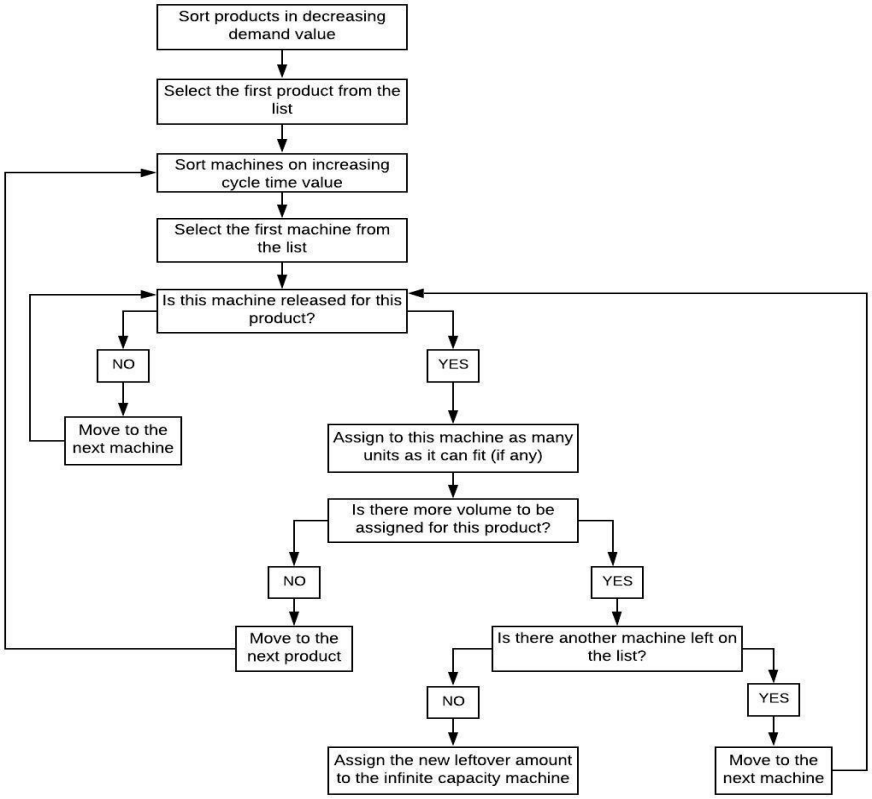

Figure 17 Steps taken for creating an initial schedule

We start by trying to assign the full demand of each product on their preferred machine, if capacity allows. If the full demand fits we move to the next product, otherwise we allocate as much volume as possible to the first preferred machine and then move to the next machine. If, for example, one product can only be produced by a single machine then we try to assign as much volume as we can fit to that machine and allocate its remaining volume on the infinite capacity machine. The outcome of this step represents a production overview showing to which machines, including infinite capacity machine, do we allocate which products, together with the corresponding volumes.

Optimization of the Initial Schedule

Once we have an initial schedule we try to optimize the total loading across the existing machines, whether or not some products are allocated to the infinite capacity machine. We do this optimization through a local search algorithm. Through our literature research we conclude that the two best such algorithms are Simulated Annealing and Tabu Search. As we state in Chapter 3, some authors believe Simulated Annealing to be better, while others believe Tabu Search outperforms the former both in terms of the solution and the running time. Furthermore, the authors claim that the answer regarding which algorithm performs best depends on the problem itself.

Therefore, we decide to implement both local search algorithms and compare the results. We implement both algorithms using a swap neighborhood structure, in which two products assigned on two different

machines change places. Moreover, in both cases the objective is to minimize the total loading across the existing machines. If we manage to obtain a better schedule then we try to reduce the loading on the infinite capacity machine (if any) by checking if we can now fit more units on the existing machines.

Building Inventory

The steps incurred for building inventory are similar to the ones we discuss for creating the initial schedule. We display the exact steps in Figure 18. Once we explore all the options for one month, the algorithm moves to the next month (if any). We allow the algorithm to explore only a total of 12 months prior to current month t.

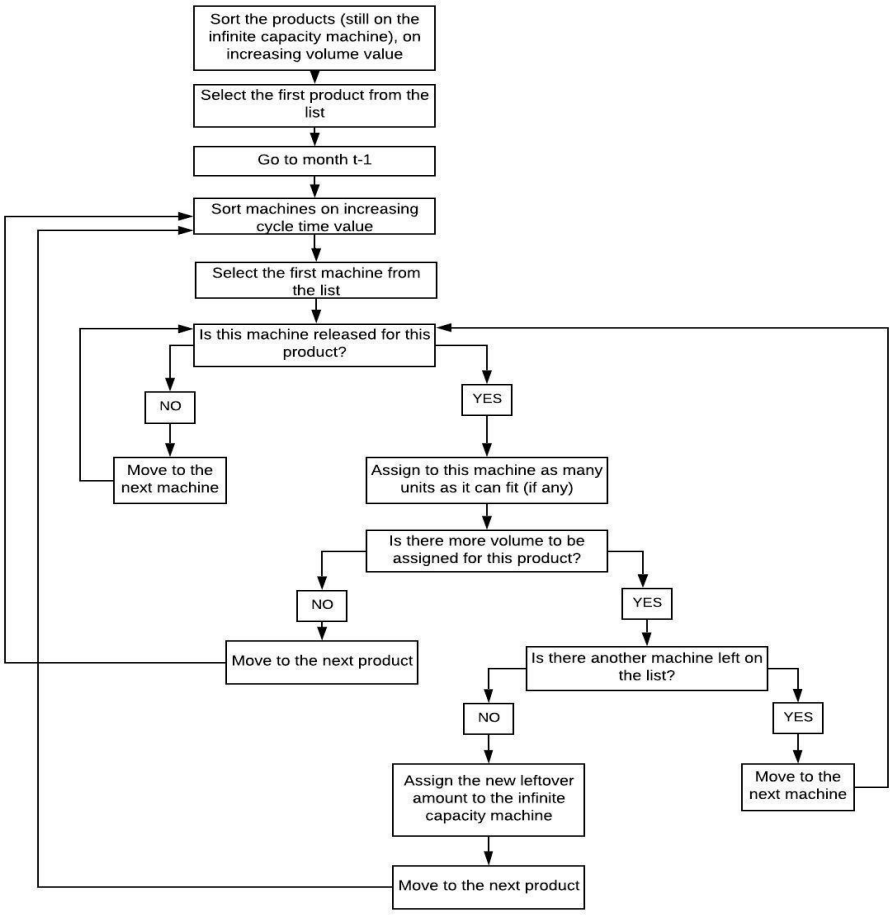

Figure 18 Steps taken for building inventory in prior months

Releasing Existing Machines

If the capacity of the existing machines turns out to be insufficient after trying to build inventory or if new products have been added and they are not yet released for production, the algorithm reaches this stage. Also, as we mention before, this stage can only occur starting from month 4.

In order to find the best combination between a product and a machine to release, we again test with both Simulated Annealing and Tabu Search. Since we aim to keep the number of releases to a minimum, we pick a neighborhood structure which releases one machine at a time for a single product. This structure resembles the 1-flip neighborhood in which a binary variable changes value. In both cases, the objective function is a combination between minimizing the loading from the infinite capacity machine and the existing machines and keeping the number of releases to a minimum. Considering the combination between the loading of the infinite capacity machine and the existing machines, we include the former since the goal is to try and reduce the loading of the ICM through releases. Regarding the loading on the existing machines, we choose to add it as part of the objective function so that the algorithm attempts to allocate products on their preferred machines. If we do not include this loading as part of the objective function then all the solutions in which a certain amount can be assigned on couple of different machines would have the same objective function value. This would occur since all these solutions would decrease the loading on the infinite capacity machine by the exact same amount.

Apart from the loading, we also want to ensure that the algorithm does not perform any unnecessary releases. Without such a check, the algorithm would be free to release all the capable machines for each of the products from the infinite capacity machine.

Ordering a New Machine

As a last resort the algorithm decides in favour of one or more new machines, depending on the number of units still allocated on the infinite capacity machine. Once it takes this action, this machine needs to be released for any of the products that it has to produce. Since it might be the case that we need more than one machine, we implement another local search algorithm to decide on the best products to allocate on each new machine. Just as before, we implement both Simulated Annealing and Tabu Search, having as objective minimizing the loading of the ICM. In the case of Simulated Annealing, one product for which the new machine should be released is randomly selected at once, while for Tabu Search a list of candidate solutions showing the products available for selection with the corresponding objective functions is generated for every iteration.

4.4 Conclusion

In this chapter we describe how we combine the knowledge we obtain through the literature research we introduce in Chapter 3 to create one MILP model, and a heuristic algorithm to serve for solving the strategic capacity planning problems which Sensata is currently facing. We design both the MILP model and the heuristic to run for one machine type at once, considering that precedence relationships are not in our main interest and the number of variables and constraints increases as the number of months in the planning horizon increases.

Through these models, we provide the answer for our sixth research question: "*How can the existing algorithm used for planning the machine loading be redesigned?".* Furthermore, we use these models to obtain the results we require for some of our next research questions.

Chapter 5: Computational Experiments

In this chapter, we present the results of our research. We start by describing the experimental setup of our tests in Section 5.1. Next, in Section 5.2 we highlight the actions we take for the verification of the MILP model. Moreover, for validation purposes, we provide a comparison between the outcomes of our MILP model and Sensata's current model. In Section 5.3 we show the results we obtain when experimenting with different methods of generating an initial solution required by the local search algorithms.

As we previously mention, for our heuristic we decide to use different local search algorithms to perform the following actions: optimize the initial schedule in terms of total machine loading, check if releasing the existing machines can improve the total loading, and check the best products for which a newly ordered machine should be released. As a result of our literature research, for each of the three actions we decide to perform a comparison between Simulated Annealing and Tabu Search. In Section 5.4 we highlight the outcomes from each local search method and conclude on the best algorithm for each step. Sections 5.5 presents the results we obtain when running both the MILP model and the full version of our heuristic.

5.1 Experimental Design

As we explain in Chapter 4, both the MILP model and the heuristic we design are meant to be used for one machine type at once. For our tests, we choose to focus on the calibrators since they represent one of the current bottlenecks in Sensata's manufacturing process. Although both the MEMS and MSG departments have production facilities in more than one country, we decide to focus on the calibrators available for MEMS in Malaysia. Table 3 shows more details with regard to the data set we use for our initial tests.

Length of planning horizon	*12/24 months*
Number of products	*260*
Number of existing machines	*9*
Number of fictive machines	*6*
Demand	*Sensata's demand forecast*
Cycle Times	*cycle times currently used in their model*
Machine Capabilities	*all set to 1*
Machine Releases	*either 0 or 1*
Availability OEE	*all set to 0.75*
Quality OEE	*all set to 1*
Buffer	*0*
Working Days	*Number of working days in each month (currently used in their model)*
Working Hours Per Day	*22.5*
Lead Time Releases	*3 months*
Lead Time Ordering a New Machine	*15 months*

Table 3 Characteristic of the initial data set

Most of the data is provided by the capacity manager of the MEMS department. The demand values we use are based on the company's demand forecasts. These values are available on a monthly basis for all products. The cycle times we choose for the initial data set represent the average values the company is currently using in their capacity planning model. Similarly, the number of working days in a month are the values currently used in their planning.

For some of the factors we fill in the missing data. An example of such missing data are the values for the physical capabilities of a machine. Since no flexibility matrix, showing the machine capabilities, is available, we decide to set them all to 1. In this case each machine is capable of producing any of the sensors. Another example of missing data refers to the OEE components. Until now Sensata only used a single value to reflect both components. We decide to set the availability component to the current value used by the company and set the other component to 1. Thus, the values we use for these parameters are 0.75 and 1. Further on, a parameter that we set to 0 is the buffer on capacity the company aims to have.

When selecting the values for the missing parameters we aim to minimize their impact on our results. In this manner, we try to keep our initial data set as close as possible to the one used in Sensata's current model.

Another factor which is not included in the company's model is related to the machine releases. Hence, such a flexibility matrix is not available. However, when checking the list of cycle times, we notice that for some product-machine combinations no values are available. According to the capacity manager, if no cycle time is found for a product-machine combination, that machine is therefore not released to produce that sensor. This being the case, we set the machine releases to either 0 or 1 by checking if a cycle time value is available for each product-machine combination.

For the initial tests, in order to validate our MILP model, we compare its results with the ones from Sensata's current model. Once we run the initial tests, we make various adjustments to the initial data set, depending on the type of test we wish to perform. Figure 19 shows the modifications we make to fit each of our tests.

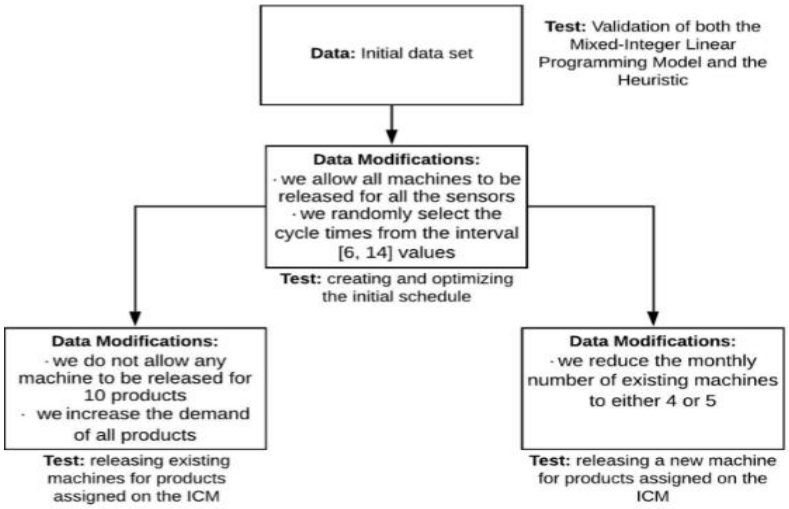

Figure 19 Input data sets used for each test

Depending on the test, we choose to perform the modifications from Figure 19 to strictly focus on one aspect at once. Since we compare different local search algorithms for three stages, we adjust the initial data set to serve as inputs for each individual stage. First, for creating and optimizing the initial schedule we set the releases of all machines to 1, regardless of the product. At this stage we are interested in knowing which of the two metaheuristics performs best when simply trying to reduce the total loading of an initial schedule. Hence, by setting all releases to 1 we allow the algorithm to strictly focus on minimizing the overall loading, given the demand of each product and the processing times associated with each product machine combination. A second modification we make for this test

refers to the processing times. As we mention when describing the initial data set, the cycle times of each product across all machines have the same value. In order for both metaheuristics to improve the initial schedule we need more variation within the cycle times. Therefore, we decide to vary these values. We choose the [6, 14] interval by checking for the minimum and maximum values from the company's data.

Next, in order to test the metaheuristics for releasing existing machines, we make two modifications to the inputs we use when optimizing the initial schedule. We choose to adjust this data set, rather than the initial one, because we believe that variation within cycle times is important for this step as well. During this stage we not only aim to release existing machines to fit the products allocated on the infinite capacity machine (ICM), but we also wish to assign these products where the lowest processing time is, if capacity allows. We provide more details regarding the objective function we choose for this stage in Figure 20.

The first modification we perform for this test is to remove the machine releases for a total of 10 products. We select the first 10 products, with demand greater than 0, which we find in the first month. Although all these products have demand in the first month of the planning horizon, we notice that they do not have a positive demand across the entire horizon. Therefore, the number of releases we expect the algorithm to perform will not always be equal to minimum 10. We perform two different tests for this stage. For the first test we only use the first modification, while for the second one, we apply a combination of both. The second modification refers to increasing the demand of each individual product. We do this to model the situation when the capacity does not suffice to fit the volumes of all products assigned on the ICM and force the metaheuristics to select the best products to relocate.

In the last stage we allow the algorithm to order a new machine. Once a newly purchased machine becomes available for production the company needs to select the best products for which this machine can be released. Hence, this stage is related to releasing the newly available machine in an attempt of reducing the loading of the ICM as much as possible. For this test we decrease the number of existing machines to either 4 or 5 and analyse the results after the algorithm decides in favour of ordering a new machine.

Besides different data sets, depending on the stage, we also consider different objective functions and sets of constraints. We choose this approach in order to focus only on one aspect at a time. For example, when optimizing for the releases of the existing machines we do not consider constraints related to building inventory or purchasing a machine. In Figure 20 we provide an overview of the objective functions and constraints which we consider in each of the stages of our heuristic.

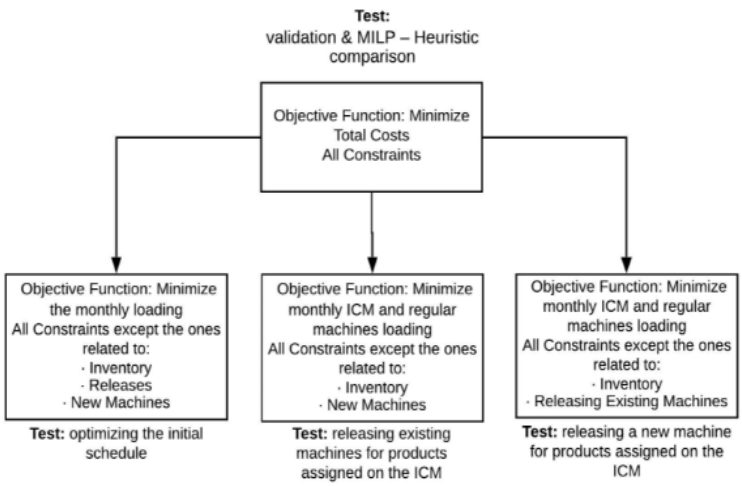

Figure 20 Objective functions and constraints considered for each test

5.2 Verification and Validation

According to Sargent (n.d.) model verification is defined as ensuring that a model is "transformed from one form into another, as intended, with sufficient accuracy". Furthermore, the authors state that in model verification, one checks the accuracy with which a model formulation is transformed into an executable program. Regarding model validation, Sargent (n.d.) mentions that it involves building the correct model associated with the real-life situation. Moreover, as validation, one should ensure that the programmed model reaches a satisfactory level of accuracy.

In both verification and validation, model testing can highlight any errors existing in the model. As stated in Sargent (n.d.), "some tests are devised to evaluate the behavioural accuracy (i.e., validity) of the model, and some tests are intended to judge the accuracy of model transformation from one form into another (verification)".

In our case, when formulating the MILP model, we try to reflect the real-life situation as accurately as possible. We start with identifying the factors already present in the Sensata's current capacity planning model, and, as a next step we check the ones that were excluded in favor of simplicity. The aspects that they could not consider in their planning model vary from machine capabilities and releases to different OEE components. We address all these factors in Chapter 2.

For the objective of our model we analyse the main decisions the capacity managers can make. These decisions refer to building inventory, releasing existing machines or ordering a new machine. Since each decision has an associated cost, ordering a new machine being the most expensive, the goal for the company is to postpone a big investment for as long as possible. Hence, the objective we define for our MILP model refers to cost minimization.

In order to validate the mixed-integer linear programming model, we use the initial data set which we describe in Section 5.1. We base this data set on the same inputs used in the company's model. We run the MILP model for a planning horizon of 24 months and compare the overall loading percentage in each month t. The formula we use for calculating the overall loading percentage is:

$$\frac{\sum_{i=1}^{I}\sum_{k=1}^{K} NumberOfUnits(i,k,t) * CycleTime(i,k,t)}{Total\ available\ capacity\ in\ month\ t} * 100\%,$$

where i and k refer to the products and machines sets.

In order to compare the overall loading percentages, we calculate the averages, per month, of the loadings for each individual machine. We display the individual loadings in Figure A.1 from Appendix A. Figure 21 displays these averages, while Figure 22 displays the results from Sensata's current capacity planning model

Month	Overall Loading Percentange
1	85.39
2	86.12
3	86.07
4	85.04
5	100.00
6	82.26
7	95.04
8	68.38
9	93.18
10	82.62
11	77.09
12	64.98
13	94.50
14	99.02
15	98.40
16	100.00
17	100.00
18	100.00
19	100.00
20	94.58
21	100.00
22	100.00
23	99.36
24	92.24

Figure 21 Monthly Loading Percentages per Machine type - 24 months - AIMMS

Month	Overall Loading Percentange
1	85.38
2	86.12
3	86.07
4	77.77
5	107.52
6	82.34
7	95.13
8	68.45
9	93.22
10	82.69
11	77.13
12	65.02
13	94.50
14	99.02
15	87.25
16	97.27
17	102.41
18	105.02
19	105.81
20	82.90
21	100.85
22	108.99
23	98.54
24	91.42

Figure 22 Monthly Loading Percentages per Machine type – 24 months – Sensata's Model

In Figure 21, the cells highlighted in orange show the months in which our model builds inventory in order to cope with the increase in demand. The cells highlighted in red from Figure 22 show the months in which the overall loading exceeds 100%. Since the managers do not include any capacity restrictions in this model, these percentages can exceed 100% if the process is completely overloaded. The two cells highlighted in blue show the months in which machines 8 and 9 become ready for production.

Comparing the percentages from the two figures we can see that in the months in which our model does not build inventory, the values are very similar. We notice bigger differences when the model decides to produce ahead and keep items in inventory. This is a result of including capacity constraints based on which loading is not allowed to exceed 100%, no matter the demand. Considering the cost differences between each month (any action is cheaper in month t than in month t-1), the results show that inventory is being produced right before it is needed. For example, since in month 5 there is a capacity shortage, our results show inventory build-up in the prior month. For the exact number of units kept in inventory in each of the months, we include more details in Appendix A.

Another action that our model takes is to allow certain machine releases. After checking the reasons behind these machine releases, we conclude that in the input data there was no machine released for a total of 7 products. We present the outputs related to these releases in Appendix A.

Overall, by checking the results from Figures 21 and 22 and analysing aspects such as: why did the model decide for inventory build-up in a certain month and why were certain releases performed, we believe that the MILP model reflects the real situation in an accurate manner.

5.3 Initial Schedules

Considering that a local search algorithm starts from a feasible initial solution and, in our case, searches in its neighborhood for possible swaps to perform, we decide to test various methods for obtaining such a solution. In our case, an initial solution represents a schedule (allocation of products to machines) which does not violate any of the constraints applicable for this step. These constraints relate to demand, capacity and machine releases.

As we previously mention in Chapter 4, we implement the following 5 alternatives for generating an initial solution:

1. Assign the products to machines in increasing value of their IDs.
2. Sort the products on decreasing demand value: in this case we assign the product with the highest demand first. This schedule involves no sorting for the machines.
3. Sort the machines on increasing cycle time values: in this case we assign the first product on the machine with the best cycle time. If this machine is also having the best cycle time for another product, we also assign that other product here, without violating capacity restrictions. If the entire volume of this product does not fit, we assign the units that do fit on this machine and the rest we assign on the second best machine. This schedule involves no sorting for the products.
4. Sorting the products on decreasing demand value and sorting the machines on increasing cycle time value: this case represents a combination of cases 2 and 3. For this case, we assign the product with the highest demand on its preferred machine (machine with the smallest cycle time).
5. Sort the products on increasing number of releases: for this case we count the number of machines released for each product and we start by assigning the product(s) with least releases.

To decide which of the 5 options performs best we generate a monthly initial schedule for a planning horizon of 24 months. We previously describe the data set we use in this test in Section 5.1.

We compare the results of the 5 options with the overall loading percentage we obtain after solving the MILP model. At this step, we choose the overall loading percentage as the comparison factor. By generating an initial schedule and optimizing it through local search we aim to fit as much of the demand as possible within the existing machines. Next, we want to reduce the overall loading to fit the volumes assigned on the ICM, if any, or facilitate inventory build-up, if needed. To ensure a proper comparison between the results obtained from AIMMS and our options for generating an initial schedule, we design another MILP model, which focuses on minimizing the loading. This model, although similar to one in Chapter 4, only contains the constraints relevant for this step. We show the entire formulation of this model in Appendix B and display the results in Appendix C. In Figure 23 we show the average loading across the entire planning horizon for the optimal allocation and the five methods for generating an initial schedule. Furthermore we also look at the average difference and average relative difference when comparing each of the schedules with the optimal allocation. We calculate the average differences by subtracting the optimal overall loading percentage from the results associated with each schedule. To calculate the relative differences, we divide the individual differences between the results of schedule and the optimal value by our reference value, the optimal overall loading percentage.

	AIMMS	No Sorting	Sorting Demand	Sorting Cycle Times	Sorting Both	Sorting Releases
Average	68.381	100.694	102.112	69.764	69.445	93.431
Average Difference		32.313	33.731	1.383	1.064	25.050
Average Relative Difference		47.575	49.834	1.928	1.501	36.401

Figure 23 Comparison between the two best schedules

In Figure 23 we notice which schedules lead to best results. Sorting the machines based on cycle time and sorting based on both demand and cycle time leads to an average relative difference of 1.9% and 1.5% respectively. Considering the number of units allocated on the ICM, the same two schedules lead to the best results. In Appendix C we show the monthly number of units assigned on the ICM, for each schedule, and the percentage they represent from the monthly volume. When sorting based on cycle times and both on cycles times and demand the average percentage of units assigned on the ICM is 0. This means that no units are allocated on the ICM during the entire planning horizon. Considering the other 3 schedules the percentages showing the number of units, considered from the total volume across the 24 months, assigned on the ICM are 4.9 (No Sorting), 4.6 (Sorting Demand) and 2.1 (Sorting Releases).

Because each of the two best methods finds better results in exactly 12 out of the 24 months we perform a more detailed comparison between the two. We calculate the average difference between the two, followed by maximum and minimum difference. We display these results in Figure 24.

Best Schedule	Average Difference	Maximum Difference	Minimum Difference
Sorting Cycle Times	0.647	2.511	0.036
Sorting Both	1.285	3.448	0.176

Figure 24 Best two schedules - Average, Maximum and Minimum difference

From this comparison, we notice that when sorting based on cycle times leads to better results all the three differences are lower than when sorting both based on demand and cycle times. Based on the aforementioned results we reach the conclusion that when the former performs better, the latter is closer to these values than when the opposite occurs. So, at this point, we state that the best method for generating the initial schedule is to first sort the products on decreasing demand and then, for each product, sort the machines on increasing cycle times. In the next section we show which of the local search algorithms can better improve our best initial schedule.

5.4 Local Search Algorithms

As we previously state, from the literature review we introduce in Chapter 3, we conclude that Simulated Annealing and Tabu Search are the two algorithms best matching our problem statement. Furthermore, also through the literature research we find out that the opinions on which algorithm performs best are quite divided and that some authors even state it is highly dependent on the problem at hand. Hence, we decide to test which algorithm fits best for each of our three different goals: optimizing the initial schedule, releasing existing machines and releasing a newly purchased machine.

The behaviour of each of the two metaheuristics we compare depends on the values of certain parameters. For each algorithm we perform various tests, for which we introduce the results in our Appendix D, F and H, to choose the right parameters.

First, considering Simulated Annealing, four parameters are relevant for defining an appropriate cooling schedule. As we describe in the literature review the algorithm starts with a high initial temperature and slowly reduces it until a stopping criterion is achieved. The initial temperature for a Simulated Annealing algorithm should be set to a value large enough to make the acceptance probability as close to 1 as possible. The idea of having an acceptance probability close to 1 is to ensure that, in the beginning, the algorithm accepts almost all transitions. Hence, to choose an appropriate value for the starting temperature we calculate the ratio between the accepted and proposed number of transitions at the start of the algorithm. In general, we choose the value of the starting temperature for which the acceptance probability turns out to be close or above 0.95.

Regarding the stopping criterion for a Simulated Annealing algorithm, the two most common ones are: letting the temperature gradually decrease until it reaches a very small value or stopping when no improvements are found for a certain number of consecutive iterations/temperature values. We use the first one for our experiments, hence for all Simulated Annealing algorithms we set the stopping temperature to 0.5.

The next relevant parameter for a Simulated Annealing algorithm is the Markov chain length. This parameter shows the number of iterations that are performed at each temperature level. From literature we find that the Markov chain length should be proportional to the number of neighbouring solutions that can be generated from a given current solution. In every iteration our Tabu Search algorithm creates a list of all the possible candidates, hence the neighbouring solutions that can be reached from the current state. We start our experimentation by setting the length of the Markov chain equal to the number of candidates and, depending on the results, we double it. Furthermore, we also test the feasibility of smaller values. The last parameter we require for defining a cooling schedule is the decreasing factor. We choose the appropriate values for this parameter by varying its values and comparing the results in terms of objective function values and running times.

Regarding the second metaheuristic we use in our research, there are two parameters for us to consider. These parameters are the maximum number of iterations and the length of the tabu list. According to Osman (1993) appropriate estimates for the two parameters depend on the problem characteristics. We check multiple papers presenting different guidelines for choosing the right values for these two parameters.

In the paper of Romero-Conrado et al. (2019), the tabu tenure is set to represent a certain percentage (25%, 50% 75%) from the number of iterations. Schweiger and Sahamie (2013) state that if the length of the tabu list is too small then the algorithm will be cycling around a local optimum solution, while, if the length is too high possible solutions can be excluded, hence limiting the solution space. Furthermore they mention that if the length of the tabu list becomes equal to the number of iterations then no further improvement can be achieved and the value of the objective function stagnates. To select the right value for the tabu list length they keep the number of iterations constant and analyse the solution in term of objective function values for different lengths of the tabu list.

We follow a similar approach as Schweiger and Sahamie (2013). We start by setting the number of iterations in a similar manner as the length of the Markov chain, proportional to the number of candidate solutions. Next, we vary the length of the tabu size to represent a certain percentage from the number of iterations.

In this section we show the results, in terms of objective function and running time, we obtain when using the six algorithms. Moreover, in Appendix D, F and H we show how we select the right parameters for each algorithm.

5.4.1 Optimizing the Initial Schedule

Once we create an initial schedule, the next step in our heuristic has the goal of minimizing the total loading of the existing machines. To check if any of the two local search algorithms can improve our best initial schedule, we run both models using the inputs we describe in Section 5.1. We display the parameters we choose for each method in Figure 25, while Figure 26 shows a comparison between the results.

Simulated Annealing		Tabu Search	
Starting Temperature (T0)	300	Number Of Iterations	2500
Stopping Temperatiure (T1)	0.5	Tabu List Length	750
Decreasing Factor (α)	0.97		
Length of Markov Chain (L)	5000		

Figure 25 Chosen parameters

	AIMMS	Simulated Annealing	Tabu Search
Average	68.381	69.062	69.095
Average Difference		0.681	0.714
Average Relative Difference		0.950	0.995

Figure 26 Overview of Simulated Annealing and Tabu Search in comparison with the optimal values

We calculate the average differences by subtracting the overall loading percentage of the initial schedule from the results we obtain from the metaheuristics. For the relative differences we divide the individual differences between the results of the metaheuristics and the initial schedule loading by our reference value, the overall loading of the initial schedule.

As we notice in Figure 26 the average difference when using Simulated Annealing is closer to the optimal schedule and the difference between the two metaheuristics is of 0.033%. In Appendix E we display the overall loading percentages on a monthly basis. We compare the results both in terms of which algorithm manages to improve the initial schedule the most and in terms of the difference from the optimal objective function value. In Appendix E we see that in two months no algorithm manages to obtain a product-machine allocation that results in a smaller overall loading. However, when comparing the other months we conclude that in 13 out of the 24 months Tabu Search improves the initial schedule more than Simulated Annealing. Considering the months in which the algorithms cannot improve the initial schedule, we see that Tabu Search does not find a better allocation for only 2 months, while for Simulated Annealing there are 9 months in which the initial schedule is not improved at all.

Although Tabu Search finds a better solution for a higher number of months, when checking the average improvement over the 24 months we notice a higher value for Simulated Annealing. We notice a similar behaviour in terms of the maximum improvement across the 24 months. Hence, at this point, we conclude that although Simulated Annealing does not find the best solution as often as Tabu Search, in some of the months where improvements are found the reduction in objective function value is much greater than the

one of Tabu Search. For example, in month 21 Tabu Search can only improve the overall loading by a total of 0.1 % while Simulated Annealing performs a reduction of 1.2%.

Both algorithms start from the best initial schedule we choose in Section 5.3. Considering the number of units assigned on the ICM, given that we find a 0% of the total volume assigned on the ICM in the best initial schedule, the same remains true after running the local search algorithms.

Given the results, we conclude that for finding as many improved solutions as possible one should choose Tabu Search. However, if reducing the overall loading is critical one should cross check the results of Tabu Search by also running the Simulated Annealing algorithm.

Month	Simulated Annealing	Tabu Search
1	832.38	336.66
2	1645.17	251.91
3	988.59	344.06
4	686.81	353.42
5	722.33	321.08
6	560.58	757.83
7	602.03	783.91
8	494.22	783.77
9	492.31	980.19
10	501.44	838.58
11	449.99	808.24
12	431.05	910.98
13	434.70	746.11
14	402.92	907.36
15	401.92	1198.48
16	394.81	938.39
17	464.89	1143.53
18	442.11	1154.50
19	394.50	936.64
20	371.17	1039.17
21	425.97	907.67
22	442.06	770.16
23	407.31	1094.55
24	389.61	986.34
Average	557.453	803.897
Max	1645.17	1198.48
Min	371.17	251.91

Figure 27 Monthly Running times – Simulated Annealing, Tabu Search

To compare the running times, in seconds, of the local search methods we calculate the average, maximum, and minimum running time over 24 months. We show these results in the last three rows of Figure 27. Based on these results we state that, on average, Simulated Annealing outperforms Tabu Search in terms of execution time, even though the maximum value for running times of Simulated Annealing is 446 seconds greater than the one of Tabu Search.

Even though the average difference between the execution times of the two algorithms is 246 seconds we still choose Tabu Search, given the more months in which it finds improved solutions. However since such a difference in running times might appear for each month in the planning horizon, depending on the length of this horizon, we might incline in favour of Simulated Annealing in order to reduce the total running time.

5.4.2 Releasing Existing Machines

After optimizing the initial schedule, and building inventory if needed, we reach the next optimization step in our heuristic. At this step we address our following research question: "*What machines can be released for a certain product to avoid ordering a too high amount of new equipment every time the demand is increased?*".

Our heuristic performs this step if the loading on the infinite capacity machine is greater than 0, meaning some volumes were previously allocated there. In this step the algorithm checks if any of the existing machines can be released for the products allocated on the ICM. Thus, at this point, our aim is to decrease the loading of the infinite capacity machine by allocating more volumes within the existing machines, if possible.

Even though we try to reduce the loading of the infinite capacity machine, we cannot neglect the overall loading of the existing machines. If the objective function of the local search method would strictly refer to the minimization of the ICM loading, allocating certain volumes to any of the existing machines would lead to the same improvement in our objective function. However, if we try to allocate the volumes to the machines with smaller cycle time, we also keep the overall loading of the existing machines under control.

We consider both loadings to be equally important, therefore, we combine the two into one objective function focused on minimizing the sum of both loadings.

To ensure that no decrease in the ICM loading creates a higher increase in the other loading, we set the cycle times on the ICM to have a higher value than the maximum cycle time on the existing machines.

To prevent the algorithm from releasing all machines for all products assigned on the ICM we perform a single release at a time, followed by refreshing the initial schedule to calculate the new objective function. If all the volume of a product from the ICM gets assigned within the regular machines, the algorithm will not perform another release for that product.

For this test we consider a planning horizon of 12 months. Regarding the input data, we previously describe it in Section 5.1. Just as before, in Figure 28 we display the parameters we choose for each local search method. In Appendix F we discuss how we choose these parameters.

Simulated Annealing		Tabu Search	
Starting Temperature (T0)	20	Number Of Iterations	10
Stopping Temperatiure (T1)	0.5	Tabu List Length	3
Decreasing Factor (α)	0.1		
Length of Markov Chain (L)	17		

Figure 28 Local Search Parameters – Releasing existing machines

In Figure 29 we display the results of the two metaheuristics compared to the initial schedule we obtain when the releases are cancelled. We perform this comparison to check which algorithm manages to improve this schedule the most, by releasing the preferred machines for the products allocated on the ICM. Figure 30 displays similar results, however, this time we compare the result with the best initial schedule we choose in Section 5.3. Because the only modification we make in this data set is cancelling the machine releases for 10 products we expect the algorithms to be able to return results similar to the ones from this schedule.

	Current Initial Schedule	Simulated Annealing	Tabu Search
Average Loading	75.810	67.949	65.203
Average Difference		-7.861	-10.607
Average Relative Difference		-10.355	-14.103

Figure 29 Overview of Simulated Annealing and Tabu Search in comparison with the current initial schedule

	Target Initial Schedule	Simulated Annealing	Tabu Search
Average Loading	65.241	67.949	65.203
Average Difference		2.708	-0.038
Average Relative Difference		4.329	-0.053

Figure 30 Overview of Simulated Annealing and Tabu Search in comparison with the target initial schedule

We calculate the average differences by subtracting the overall loading percentage of the two initial schedules from the results we obtain from the metaheuristics. For the relative differences we divide the individual differences between the results of the metaheuristics and the two initial schedules by our reference values, the overall loading from the initial schedules. As we show in Figure 29, Tabu Search is the one best decreasing the initial overall loading. In Figure 29 we notice negative values for both differences. When analysing the monthly loading percentages we notice that for a total of 6 months Tabu Search manages to obtain results improving the target initial schedule.

As we show in Appendix G Tabu Search finds better results for all the 12 months. Besides, in some months the results we obtain from Tabu Search are improving the loading of the target initial schedule, hence making them even closer to the optimal values. The reason behind the poor performance of Simulated Annealing is based on its random generation of moves. When performing either local search heuristic for releasing existing machines, a product – machine combination has its releasing value flipped from 0 to 1. Once the algorithm updates this release, it generates the initial schedule again. As the monthly loadings from Appendix G show there is plenty of capacity available in each month. Hence, once the algorithm refreshes the initial schedule the product for which an existing machine was released will be allocated within existing machines and removed from the ICM. One of the main differences between the two metaheuristics is that while Tabu Search generates a list of candidates and aims to select the best one, Simulated Annealing generates a move based on the random selection of an existing machine. This being the case, Tabu Search will always aim to fit each product on its preferred machine (in terms of cycle times) while Simulated Annealing will fit it on a randomly selected machine.

Regarding the number of units remaining on the ICM, when using our chosen parameters the loading of the ICM is 0 in all 12 months.

Besides the overall loading resulting from this step, we also check the number of releases the algorithms decide to perform for each of the 12 months. In Figure 31 we display the number of releases that the two metaheuristics perform. As we previously mention in Section 5.1, for this test we cancel the machine releases associated with the first 10 products with positive demand in month 1. Because these products do not have a positive demand across the entire planning horizon and since the algorithm only releases a machine if the demand is positive, we expect the monthly number of releases to vary. Moreover, we expect the number of monthly releases to be equal to the minimum number of products having positive demand in each of the 12 months.

Month	1	2	3	4	5	6	7	8	9	10	11	12
Number of Products with Demand > 0	10	8	8	8	8	7	6	6	6	6	7	7
Number Of Releases SA	10	8	8	9	8	7	6	6	6	6	7	7
Number Of Releases TS	10	8	8	8	8	7	6	6	6	6	7	7

Figure 31 Monthly Number of Releases – Simulated Annealing, Tabu Search

The only month in which the number of releases is different than the number of products with positive demand is month 4, when Simulated Annealing performs one more release than Tabu Search. The number of releases is highly related to the total costs, since for each release, depending on the month, a certain cost is incurred. Hence, considering the total costs, in month 4 the outputs of Tabu Search result into a lower cost that the company has to cover.

In terms of running times, we perform a similar analysis to the one we introduce in the previous section. This time, the average running times are: 1.4 seconds for Simulated Annealing and 44.3 seconds for Tabu

Search. Even though, Simulated Annealing seems to outperform Tabu Search in terms of running times, we decide in favour of Tabu Search given the differences in loading.

Given that, in this test, the existing machines have enough leftover capacity to fit the products allocated on the ICM, we want to check the results if the contrary occurs. Hence, we perform a similar test in which we increase the demand of each product by a constant amount with the aim of increasing the overall loading. We present the results, in terms of loading, number of releases and running times, in Appendix G.

5.4.3 Releasing a New Machine

In this section we address the following research question: "*How many machines does the company need to order and when should these orders be placed to properly handle the upcoming increase in demand?"*

We only include a new machine if, after performing all the previous steps, the loading on the ICM is still above 0. When we do add a new machine, the algorithm selects the first fictive machine, and starts considering it as ready for production. Just as for the MILP model, all the fictive machines are identical in terms of capabilities, cycle times and OEE components. Moreover, no releases are set for any of the fictive machines.

Since we design identical fictive machines, we no dot require any optimization to be able to select the best one to purchase. However, because no releases are set for these machines, we again use local search algorithms to pick the best products for which one or each (if more machines have to be purchased) machine should be released.

As we state in the beginning of this chapter, we again perform a comparison between Simulated Annealing and Tabu Search. At this point in the model, we include all the constraints we introduce in the formulation of our MILP model.

Regarding the input data, we previously describe it in Section 5.1. Instead of the 7, 8 or 9 machines available in each month, we only use 5. We consider the other machines as fictive and check how many machines the company has to purchase to fit the entire demand. We perform this test for a period of 12 months. Figure 32 shows the parameters we select for each local search algorithm.

Simulated Annealing		*Tabu Search*	
Starting Temperature (T0)	*20*	*Number of Iterations*	*100*
Stopping Temperature (T1)	*0.5*	*Tabu List Size*	*5*
Decreasing Factor (α)	*0.4*		
Length of Markov Chain (L)	*50*		

Figure 32 Local Search Parameters – Releasing a new machine

As we display in Appendix I, in three of the twelve months (4,6,8) the overall loading resulting after generating the initial schedule, turns out to be below 100%. Hence, we do not attempt to order a new machine in these months. Considering the other months, we notice that in all of them Tabu Search reaches a loading value better than or at least equal to the one from Simulated Annealing. Moreover, in months 5, 9, 10 and 11 Tabu Search outperforms Simulated Annealing by a minimum percentage of 0.7 and a maximum of 11.

In terms of running times we notice a big difference between the two algorithms, the difference between the two maximum values being of 999.2 seconds (16.65 minutes). In order to test multiple configurations and provide a hint of the robustness of our model we repeat the same test when only 4 machines are available. Just as before, we allow the algorithm to order one new machine and introduce the results in Appendix I.

For this step, we decide to, yet again, decide in favor of Tabu Search. We make this decision considering the 11.029% we find in one of the months and we believe this option to be used as a last resort. As we previously state, ordering a new machine will only be used if all the other steps are not capable of reducing the loading of the ICM to 0. However, if running times are more relevant we advise for selecting Simulated Annealing as an alternative. We introduce more recommendations in Chapter 6.

5.5 Heuristic vs MILP Model

In this section, we analyse the results of our heuristic and compare them with the optimal ones that we obtain from solving our MILP model. Furthermore, we also show Sensata's results when using the same input data.

We perform this final test using the initial data set we describe in Section 5.1. This set contains the data currently used in Sensata's capacity planning model. We aim to obtain a product – machine allocation for a total of 260 products and 9 machines. Furthermore, we run this test for a planning horizon of 24 months.

We define the criteria on which we perform the comparison between our results to address the following research question: "*What are the criteria on which we compare the 2 models?".* Even though the objective function we choose for the MILP model aims to minimize the total incurred costs, we first compare the two models when attempting to minimize the overall monthly loading. We set up the three different types of costs associated with the three decisions available for capacity managers, in order to reflect the preferred order for these decisions. We attempt to minimize the total cost in the MILP model to ensure that it selects the decisions in this preferred order. If we try to minimize the total loading, the model will have the liberty of, for example, purchasing as many machines as possible (maximum number of machines it could purchase is equal to the number of fictive machines we decide to use in our inputs). Hence, we choose to compare the two in terms of overall loading, since our main interest lies in ensuring that Sensata has enough capacity to meet their increasing demand.

Figure 33 shows the overall loading we obtain from the MILP model and the heuristic, plus the loading from Sensata's current model. Because we do not notice a difference in terms of cycles times between existing machines (each product requires an equal amount of processing time on all the machines) we do not use include the optimization of the initial schedule.

Month	Overall Loading Percentange (MILP)	Overall Loading Percentage (Heuristics)	Overall Loading Percentange (Sensata's Model)
1	85.39	85.39	85.38
2	86.12	86.12	86.12
3	86.07	86.07	86.07
4	85.04	85.94	77.77
5	100.00	100.17	107.52
6	82.26	82.19	82.34
7	95.04	95.03	95.13
8	68.38	72.00	68.45
9	93.18	92.36	93.22
10	82.62	81.46	82.69
11	77.09	76.78	77.13
12	64.98	79.62	65.02
13	94.50	93.37	94.50
14	99.02	92.10	99.02
15	98.40	88.75	87.25
16	100.00	95.45	97.27
17	100.00	98.41	102.41
18	100.00	99.89	105.02
19	100.00	99.79	105.81
20	94.58	91.52	82.90
21	100.00	99.25	100.85
22	100.00	100.00	108.99
23	99.36	97.96	98.54
24	92.24	88.94	91.42
Average	91.01	90.36	90.87

Figure 33 Overall Loading Percentages – MILP, Heuristics, Sensata's Model

By highlighting certain cells, we display the months in which either the MILP model or the heuristic builds units for inventory. As we highlight in the second column, opposite to the MILP model, our heuristic builds inventory in the majority of months. When closely analysing the product – machine allocation, we notice that for items having machines released in the prior months the algorithms builds the associated amounts as inventory in those previous months. For example, consider that product p is a new product introduced by the company. This means that no machines are released to handle its manufacturing. Furthermore, assume that this product start having demand planned from month 4 onwards. Our algorithm will release one or more machines, depending on how high the demand is, to cover the newly added item. This release will occur in month 4. Next, in month 5, the first decision our algorithm attempts to make is building inventory. Because, for month 5, product p still has its volume assigned on the ICM the algorithm builds the associated number of units as inventory in month 4. This happens because we do not revisit the decision we priorly take. In our recommendations section we further address this issue.

Considering the overall loading for month 5, we notice that the values are above 100%. This happens because the capacity of the entire machine type is fully booked. When the capacity is fully booked and there are couple of products for which existing machines should be released, our algorithm is not able to proceed with the releases since none of the existing machines can fit these volumes. Hence, in this case, two products remain assigned on the infinite capacity machine.

When looking at the average values from the three models we notice that the values are very close to each other. Although the average value we obtain from the heuristic seem to be the best, we do not fully agree with this. A reason behind this is the fact that when comparing the demand summed up across all products in the entire planning horizon, we find a difference of approximately 598 units. The input data we use for

our heuristic is 598 units short compared to the one used for the MILP model. This occurs because we priorly round these values for simplicity.

To validate the results we obtain from building inventory, we only allow the algorithm to build inventory and analyse the months in which it does so. Figure 34 shows the monthly inventory levels for the entire planning horizon.

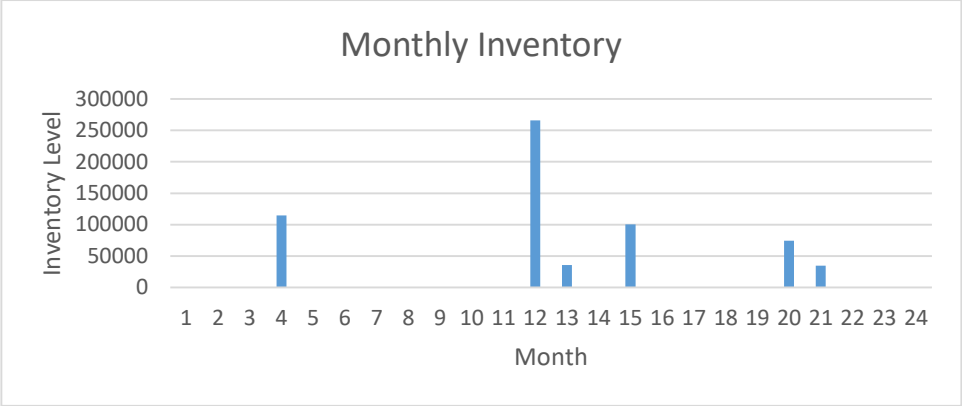

Figure 34 Monthly Inventory Levels - Heuristic - 24 months

This time we notice that the heuristic builds inventory in fewer months than before. While the MILP model decides for building inventory in 7 out of 24 months, our heuristic does so in only 6. We notice the same pattern of building inventory in months prior to having an overloaded process (above 100% in Sensata's model). However, because we do not allow any machine releases, some products stay allocated on the infinite capacity machine, and less inventory is being built. We observe that some products are still allocated on the ICM through the percentages above 100 which we can see in Figure 35.

Month	Overall Loading
1	85.39
2	86.12
3	86.07
4	85.85
5	100.18
6	82.34
7	95.12
8	68.47
9	93.68
10	83.35
11	77.77
12	81.34
13	92.88
14	95.17
15	91.53
16	94.70
17	100.36
18	104.34
19	104.99
20	89.61
21	106.24
22	105.66
23	101.12
24	94.00

Figure 35 Overall Loading Percentages – Building Inventory – 24 months

Considering the total costs, we obtain different results. First, since when allowed to perform releases, the heuristics decides on building more inventory than the MILP model, the costs suffer an increase. Regarding the number of releases, for each of the 7 products for which no machine is released, both models require a single release. Finally, since none of them decides for purchasing a new machine, these costs are not incurred. Based on these results, we conclude that the MILP model performs better in terms of costs and that the difference comes from the number of units built as inventory.

Chapter 6 Conclusions and Recommendations

In this chapter we conclude our research and provide recommendations both in terms of implementation of our final deliverable and in terms of future work that could improve our results. The first section discusses our concluding remarks related to the steps performed during this research and the outcomes we obtained. Next, in Section 6.2 we state the recommendations we deem important for implementing our algorithm as the new capacity planning model for the MEMS and MSG departments. Section 6.3 provides our suggestions related to the future work or research that can be done to further ensure the feasibility of our heuristic. Finally, in Section 6.4 we introduce the limitations of our research.

6.1 Conclusions

As we state throughout this research, our main goal is to develop a new strategic capacity planning model for two of Sensata's automotive departments. The need for such a model arose due to the simplicity of their current model and the limitations imposed by Excel. To structure our research, we define multiple research questions, which we previously introduce in Chapter 1. Throughout this research, we provide the answers for these questions and we conclude by summarizing the most relevant remarks here.

To achieve our goal, first we perform a detailed analysis of the current situation, specifically, the current capacity planning model used by the two departments. Through this analysis we identify the factors considered in their model, the simplifications incurred by the capacity managers and the relevant factors which were excluded due to such simplifications. In terms of existing factors, we identify the following: demand, capacity related aspects (number of working days, number of hours available for production, buffer and OEE), and processing times.

Considering the simplifications incurred by the capacity managers, these vary from using average values for parameters such as processing times and OEE to excluding relevant factors such as machine capabilities or releases. As a result of using averages for processing times and OEE, their model is not capable of reflecting the differences between multiple machines of the same type. Such an example is the difference on processing times, for the same product, present between two machines of two different generations.

Moreover, when analyzing the current situation, we also investigate what decisions can the capacity managers make in order to keep up with the constant increase in demand. As we state on multiple occasions, the three main decisions refer to building inventory, releasing existing machines or purchasing a new machine.

Besides obtaining a better understanding of the current situation, we perform a literature study focused on two main aspects. On one hand, we focus on defining a linear programming model to match Sensata's manufacturing process, hence looking into various types of objective functions and constraints. As a result, we formulate the mixed-integer linear programming model we previously describe in Chapter 4 and use it for benchmarking the results we obtain from our heuristic. On the other hand, we look into the best algorithms, known in literature, which can be used for designing such a heuristic. We choose to have this heuristic serve as the main deliverable of this research and the basis of a capacity planning tool that Sensata can use in the future. As a result of the literature review, we perform comparisons between

Simulated Annealing and Tabu Search for three different stages of our heuristic. These stages refer to optimizing the initial schedule, selecting the best machines to release for the products assigned on our ICM, and selecting the best products from this ICM for which a newly arriving machine should be released.

To ensure that the MILP model reflects the real-life situation of Sensata's manufacturing process, we use the inputs from their current model in our testing. Because aspects such as building inventory or releasing existing machines are not considered in the company's model, we could only compare the two models in terms of the monthly overall loading.

When building our heuristic, we pick an iterative approach where we implement one step at a time. At each of the three optimization stages, we perform a comparison between two different local search algorithms. We select the parameters for each local search algorithm by combining the guidelines available in literature and the experimentation with different values. Based on the results, our final choices in terms of optimization methods are performing Tabu Search for all the three optimization stages.

In this research we create a heuristic to replace the current strategic capacity model used within Sensata's automotive departments. We compare the results with the optimal values, which we obtain by defining a MILP formulation matching their manufacturing environment.

Our method distinguishes itself from the ones we encounter during our literature research due to the approach we choose. Most of the models for strategic capacity planning rely on (MI)LP models modelling the constraints of the various manufacturing environment. Some authors choose a combination of such models and simulation techniques, while others combine it with combinatorial optimization techniques, such as Branch and Price. Given the resemblance of Sensata's manufacturing process with a job shop, we opt for local search algorithms, such as Simulated Annealing and Tabu Search, in the design of our heuristic. According to our literature study, these methods are the best at solving scheduling models in a job shop manufacturing system.

Besides our approach, another difference comes from the factors we decide to consider in our model. Most of the strategic capacity planning models strictly consider aspects such as inventory build-up, capacity requirements and the number of resources required for production. In our research we choose to extend on the general models we encounter in literature and include additional constraints related to, for example, machine capabilities and releases, OEE components and lead times for the three main actions available to the capacity managers. We do this in the attempt of reflecting the real-life situation, as accurately as possible, in our model.

Considering the applicability of our algorithm, we believe that it can be adjusted to fit similar manufacturing systems in the make-to-order environment.

6.2 Recommendations for Implementation

In this section we present our recommendations for ensuring a smooth implementation of our heuristic as the new strategic capacity planning model for the two automotive departments. We start by suggesting various improvements in terms of the input data. Next, we address multiple recommendations in terms of code optimization and thresholds that the company needs to set to ensure an accurate reflection of the real-life situation. In the end, we draw attention to the various guidelines, regarding local search parameters, which the managers should be aware of when using our algorithm.

Input Data

In their current model, the capacity managers are using average values for parameters such as processing times and OEE. This approach fails to reflect the differences between individual machines of the same type. Furthermore, as we describe throughout this research, the company can reflect the real-life situation in a more accurate manner by considering parameters such as machine capabilities and releases. Hence, we suggest the following:

- processing times: For newer generation machines, the company makes use of automatic data collection. We suggest implementing automatic data collection also for the older machines, if possible. Once this data is available, the company can create a matrix showing the real value of processing times for each product – machine combination.
- OEE components: at the moment, the company is using a single value to compensate for the effect of this parameter. This value serves as the availability component. We suggest the separation of this value into quality, reflecting the yield loss, and availability, reflecting the percentage of time each machine is up for production. We suggest this separation since the availability parameter is strictly dependent on the machines themselves, while, the yield loss also depends on the products manufactured.
- machine capabilities and releases: because these factors were not included in their current model, no matrix showing, for example, which machines are released or capable to produce which product, is available. We recommend the managers to obtain such information from the manufacturing facility or create such overview if none is already available.

Cost related thresholds

Each of the three main decisions available for the capacity managers has an associated cost. In the MILP model we aim to minimize the total cost incurred for the planning horizon. This is done by inspecting all the available combinations, and searching for the combination which results in the minimum value of these costs. However, in our heuristic we do not examine all these possible combinations. Instead, our algorithm decides to use one of the three actions if any volume is allocated on the infinite capacity machine. The algorithm choses these actions in a certain order, from the least to the most expensive one, which means the algorithm will first try to reduce the loading on the ICM by building inventory. If the loading on the ICM is still above 0 the algorithm tries to perform releases within existing machines, and, as a last option the algorithm decides that a new machine is needed.

In our heuristic we do not include any predefined limits for the number of units that can be held in inventory or for the number of machines that can be released. However, in the MILP model the solver decides, for example, at which point is more profitable to release existing machines instead of building

inventory. This being the case, we recommend the implementation of two thresholds reflecting the number of units to be kept in inventory and the number of existing machines to release. Such limits should be set considering certain cost thresholds that each step should not exceed.

Code Optimization

First, since the goal of this research was to provide a proof of concept of a working strategic capacity planning algorithm and due to time limitations, we could not provide the company with a fully optimized capacity planning tool. We recommend further optimization of the code, which in our opinion has the potential of further reducing the running time. For example, one can keep track of the fully loaded machines on a monthly basis. This becomes useful when the algorithm generates the initial schedule or attempts to build inventory. By keeping track of the fully loaded machines one avoids checking whether a machine has any available capacity every time a product needs to be assigned.

Guidelines in terms of local search parameters

- Optimization of the initial schedule: based on some discussions we had with the capacity managers, we reach the conclusion that sometimes they might be interested in performing a quick capacity check to ensure they can fit additional last-minute demand. At this point, they are not interested in a result as close to optimal as possible. Hence, we suggest adjusting the parameters to provide the managers with an output in less time than if a better result is desired. In this way, quicker running times are preferred in detriment of the objective function value. However, if, for example, they are looking into the possibility of postponing the purchasing of a new machine by one month, then they should strive to have a better objective function value in detriment of running times.
- Releasing existing machines: the number of iterations for Tabu Search should be set either to the total number of releases they want to allow in each month or to the total number of products for which no machine is released. The first option applies for the case when the entire demand of some products did not fit within existing released machines. The second applies in case new products are added in the demand forecasts. When new products are added, no machines are released for these products, hence the capacity managers can just use our algorithm to determine the best machines to release.

Given the results we describe in Chapter 5 we recommend the company to proceed with the implementation of our Delphi algorithm. We do not recommend the purchasing of AIMMS, or any similar commercially available software, unless the optimal results are desired. The most important reason for which we choose in favor of Delphi is the flexibility our algorithm offers. We program various procedures, which the managers can use either sequentially or separate. Using the procedures sequentially means running all the steps of the algorithm to provide a full overview of the decisions they can make. However, our algorithm offers the freedom of performing quick checks for different purposes. For example, if the managers are aware of leftover capacity in a certain month, but they wish to check an approximate amount they can simply generate an initial schedule. Considering that the results of the initial schedules are already close to the optimal ones, this quick check is capable of providing a reliable overview.

In the case of new products being added to the production list, no machine releases being in place and knowing from previous results that the available capacity will suffice, the managers can generate an initial schedule and only run the releasing existing machines optimization step. Such flexibility is hard to obtain

when using AIMMS, since that involves removing certain constraints and, at times, adjust the objective function.

6.3 Future Work

In this section we mention our recommendations for the future work which can be done either if an improvement or further validation of the model is desired. First, we refer to the future work concerning our local search algorithms. Next we mention recommendations related to the revision of the decisions the algorithm priorly takes. In the end we draw attention to the analysis the company should perform if they later decide in favour of buying AIMMS followed by describing certain extensions which they can implement by further disaggregating their input data.

- **Local Search algorithms**:
    - for the local search algorithms related to the optimization of the initial schedule, we recommend experimenting with different neighboring structures. For this stage, we believe that the solution space can be searched more extensively. We suggest using a neighborhood structure that allows changing the number of products assigned on a machine. For example, one could try swapping the volumes of 2 products assigned on one machine with 1 product assigned on another machine. However, since we are not aware of how easy it can be to find 3 such products, we recommend combining the one we mention above with our neighboring structure (swap one with one). At this point the algorithm can make use of a random number generator based on which it can decide what kind of swap to perform in each iteration.
    - if the company wishes to bring the results even closer to the optimal values, then we suggest performing more experimentation with the parameters associated with each method. However, the company should be aware of the fact that if better results are desired this can lead to a substantial increase in terms of running times.
- **Revision of previous decisions:** as we state when describing our heuristic in Chapter 4, we choose to build our algorithm by using a hierarchical/iterative approach. We implement one step at a time and test each step separately. Only after ensuring the well-functioning of each individual step we start by testing combinations of two stages and end with testing all the stages together. Because of this approach, the decisions that the algorithm makes at the first steps are not revisited at later stages. For example, when deciding to build a certain number of units as inventory, the algorithm does not revisit this decision if it needs to check for releasing existing machines. When releasing existing machines, we flip the release value for one product - machine combination, and refresh the initial schedule. However, when refreshing the initial schedule, the demand of the products for which we build inventory is lowered by the number of units kept as inventory.
  To check if lower costs can be attained, we recommend that once a new decision must be made, the previous ones should be revisited. To achieve this, one has to refresh the algorithm for every new decision (applicable just for releasing existing machines and newly purchased ones). For example, if the loading of the ICM is above 0, in month t, after inventory build-up, all the inventory built for month t should be cancelled. Next, one should run the optimization method for releasing existing machines and if the loading of the ICM could not be reduced to 0 then one should try building inventory again.

- **AIMMS:** if the company decides for purchasing an AIMMS license, then we recommend performing a sensitivity analysis regarding the cost parameters. Due to time limitations we could not perform a detailed analysis ourselves, however we do see differences in results when the ratio between the three costs varies.
- **Extensions:** the MILP model we design, can be extended to also consider differences in costs when performing the same actions. For example, releasing an older generation machine can either be less or more expensive than releasing a newer one.
  Besides implementing differences in the cost parameters, one could also do the same in terms of the two lead times. Depending on the product – machine combination, performing a release in month t can require different lead times. Based on our discussion with the capacity managers, we believe that this extension would reflect the real-life situation in a more accurate manner.

6.4 Limitations

In this section, we describe the limitations of our research. First, our goal is to show the company how their model can be redesigned. Hence, developing a fully functional capacity planning tool is out of our scope. In our experiments we focus on fine-tuning the local search parameters and picking the best method for each stage. We choose to test our algorithm with the data from one of their bottleneck machine types (calibrators). However, we do not perform an extensive testing covering all the machine types existing in both automotive departments.

Furthermore, we program our algorithm in Delphi to run for one machine type at once. This means that, with the code in its current state, the company is not able to simply input their data concerning all machines types and obtain a result covering their entire manufacturing process. Therefore, to overcome this limitation, we suggest them to further optimize our code to accommodate all machine types at once.

Another important limitation of our research is that we do not fully model the real-life situation. Within our recommendations we mention elements such as variable costs and lead times which can be added to better reflect the real-life situation. However, because we develop a strategic capacity planning model a variety of aspects such as maintenance, precedence relationships or set-up times are not included.

References

Bang, J., & Kim, Y. (2010). Hierarchical Production Planning for Semiconductor Wafer Fabrication Based on Linear Programming and Discrete-Event Simulation. *IEEE Transactions on Automation Science and Engineering*, 326 - 336.

Bloomenthal, A. (2019, May 22). *Investopedia*. Retrieved from Investopedia: https://www.investopedia.com/terms/c/capacity-management.asp

BusinessDictionary. (n.d.). Retrieved from BusinessDictionary: http://www.businessdictionary.com/definition/manufacturing-resource-planning-MRP-II.html

Byrne M., D., & Bakir M., A. (1999). Production planning using a hybrid simulation-analytical approach. *International Journal of Production Economics*, 305-311.

Calculate OEE - Definitions, formulas and examples | OEE. (n.d.). Retrieved from OEE: https://www.oee.com/calculating-oee.html?fbclid=IwAR3GBkW6zPX6DwuxLEY6034E36iNRaf5ZWdPMDkaRAcAQiJc7BQpRePY2ws

Çaliş, B., & Bulkan, S. (2013). A research survey: review of AI solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing*, Banu Çaliş Uslu.

Caramia, M., & DellOlmo, P. (2006). Manufacturing Systems: Trends, Classification, and. In M. C. Dell'Olmo, *Effective Resource Management in Manufacturing Systems* (pp. 1-34). Roma: Springer-Verlag London Limited.

Chatterjee, S. (2012, January). FDA Perspective on Continuous Manufacturing. Baltimore, Maryland, U.S.

Chaudhary, R. S., Pazhayattil, A., & Spes, J. (2017, May 30). *PharmTech*. Retrieved from PharmTech: http://www.pharmtech.com/continuous-manufacturing-generic-industry-perspective

Chen, Y., & Fan, H. (2015). An Application of Stochastic Programming in Solving Capacity Allocation and Migration Planning Problem under Uncertainty. *Mathematical Problems in Engineering*.

Dell'Amico, M., & Trubian, M. (1993). Applying tabu search to the job shop scheduling problem. *Annals of Operations Research*, 231-252.

Glover, F. (1989). Tabu Search - Part 1. *Journal of Computing*, 190-206.

Grieco, A., Semeraro, Q., & Tolio, T. (2001). A review of different approaches to the FMS Loading Problem. *The International Journal of Flexible Manufacturing Systems*, 361-384.

Hans, E., Herroelen, W., Leus, R., & Wullink, G. (2007). A hierarchical approach to multi-project. *International Journal Of Management Science*, 563-577.

Kant, G. (2014, April 17). *Supplychain247*. Retrieved from Supplychain247: https://www.supplychain247.com/article/advanced_planning_optimization_in_transportation

Kenton, W. (2018, April 19). *Investopedia*. Retrieved from Investopedia: https://www.investopedia.com/terms/m/mass-production.asp

Kim, B., & Kim, S. (2001). Extended model for a hybrid production planning approach. *International Journal of Production Economics*, 165-173.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by Simulated Annealing. *Science*, 671-680.

MacCarthy, B. L., & Liu, J. (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 59-79.

Manne, A. S. (1960). On the job shop scheduling problem. *Operations Research*, 219-223.

Martínez-Costa, C., Mas-Machuca, M., Benedito, E., & Corominas, A. (2014). A review of mathematical programming models for strategic capacity planning in manufacturing. *International Journal of Production Economics*, 66-85.

McCarthy, I. (1995). Manufacturing classification: Lessons from organizational systematics and biological taxonomy. *Integrated Manufacturing Systems*, 37-48.

Mckay, K. N., Safayeni, F. R., & Buzacott, J. A. (1988). Job-shop scheduling theory: What is relevant? *Interfaces*, 84-90.

Morshed, M., Meeran, S., & Jain, A. (2017). *A State-of-the-art Review of Job-Shop Scheduling Techniques.*

Neureuther, B. D., P. G., & Sanders, N. R. (2004). A hierarchical production plan for a make-to-order steel fabrication plant. *Production Planning and Control*, 324-335.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 421-451.

Park, M., & Kim, Y. (1998). A systematic procedure for setting parameters in simulated annealing algorithms. *Computers & Operations Research*, 1197-1212.

Paul, G. (2010). Comparative performance of tabu search and simulated annealing heuristics for the quadratic assignment problem. . *Operations Research Letters*, 577-581.

Pham, D., & Klinkert, A. (2008). Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*, 1011-1025.

Riskope. (2014, April 3). Retrieved from Riskope: https://www.riskope.com/2014/04/03/lets-define-strategic-tactical-and-operational-planning/

Rolland, E., Schilling, D. A., & Current, J. R. (1997). An efficient tabu search procedure for the p-Median Problem. *European Journal of Operational Research*, 329-342.

Romero-Conrado, A., Coronado-Hernandez, J., Rius-Sorolla, G., & García-Sabater, J. (2019). A Tabu List-Based Algorithm for Capacitated Multilevel Lot-Sizing with Alternate Bills of Materials and Co-Production Environments. *Applied Sciences*.

Sargent, R. (n.d.). Verification, validation and accreditation of simulation models. *2000 Winter Simulation Conference Proceedings*.

Schweiger, K., & Sahamie, R. (2013). A hybrid Tabu Search approach for the design of a paper recycling network. *Transportation Research Part E: Logistics and Transportation Review*, 98-119.

Semba, S., & Mujuni , E. (2019). An Empirical Performance Comparison of Metaheuristic Algorithms for School Bus Routing Problem. *Tanzania Journal of Science*, 81-92.

Shah, P. (2016, August 182). *Linkedin*. Retrieved from Linkedin: https://www.linkedin.com/pulse/classification-production-control-systems-beginners-purvesh-shah/

Siddharth Mestry, P. D.-S. (2011). A branch and price solution approach for order acceptance and capacity planning in make-to-order operations. *European Journal of Operational Research*, 480-495.

Singh, R., Singh, R., & Khan, B. K. (2015). A critical review of machine loading problem in the flexible manufacturing system. *World Journal of Engineering and Technology*, 271-290.

Stecke, K. E. (1983). Formulation and solution of nonlinear integer production planning problems for flexible manufacturing. *Management Science*, 273-393.

St-Hilaire, M., & Liu, S. (2011). Comparison of different meta-heuristics to solve the global planning problem of UMTS networks. *Computer Networks*, 2705-2716.

Techopedia. (n.d.). Retrieved from Techopedia: https://www.techopedia.com/definition/27086/pre-emption

Volling, T., Matzke, A., Grunewald, M., & Spengler, T. S. (2013). Planning of capacities and orders in build-to-order automobile production: A review. *European Journal of Operational Research*, 240-260.

Weinstein, L., & Chung, C. (1999). Integrating maintenance and production decisions in a hierarchical production planning environment. *Computers & Operations Research*, 1059-1074.

Winston, W. L., & Goldberg, J. B. (2004). *Operations research: Applications and algorithms.* Toronto: Thomson.

Zijm, W., & Buitenhek, R. (1996). Capacity Planning and Lead Time Management. *International Journal of Production Economics*, 165-179.

Appendix A: Validating the MILP model in comparison with Sensata's model

In here we provide more details regarding the comparison between our MILP model and Sensata's capacity planning model. As we mention in Chapter 5, we perform this comparison by using the initial data set we describe in Section 5.1.

Figure A.1 shows the monthly loading percentages of each individual machine. Figure A.2 shows the inventory levels of each month, which we obtain as an output of the MILP model. Next, Figure A.3 shows the number of machine releases the model chooses to perform in each month.

[Data Page] LoadingPercentage

S... Level

m / t	1	2	3	4	5	6	7	8	9
1	88.86820694	56.47901235	100	100	65.82968254	86.52813051	100		
2	100	100	100	40.59544160	62.26323520	100	100		
3	100	100	47.61393867	90.86851454	100	71.13181999	92.87048984		
4	88.10942167	64.25483028	100	100	100	100	42.93720165		
5	100	100	100	100	100	100	100		
6	45.37577633	100	100	100	100	100	30.42083185		
7	100	78.16922135	100	100	87.12005583	100	100		
8	69.95620826	49.12758946	20.56906735	100	90.94842971	24.35446936	100	92.07230321	
9	100	100	84.25688847	68.31393911	100	92.84400824	100	100	
10	22.90880343	73.72147343	100	100	84.55326119	75.19612924	87.21559923	100	100
11	91.99887611	70.23321933	78.13148681	9.284854526	100	100	100	100	44.19084067
12	25.41170349	33.66868575	79.01605850	100	60.63510364	73.01777177	100	13.05269117	100
13	100	100	100	50.50059647	100	100	100	100	100
14	100	100	100	100	100	91.19189938	100	100	100
15	100	100	100	85.62207932	100	100	100	100	100
16	100	100	100	100	100	100	100	100	100
17	100	100	100	100	100	100	100	100	100
18	100	100	100	100	100	100	100	100	100
19	100	100	100	100	100	100	100	100	100
20	100	100	100	98.27867462	100	90.45442296	62.48417464	100	100
21	100	100	100	100	100	100	100	100	100
22	100	100	100	100	100	100	100	100	100
23	100	100	100	94.26249619	100	100	100	100	100
24	30.14597040	100	100	100	100	100	100	100	100

Figure A.1 Monthly loading percentages of each individual machine

In Figure A.1 we display the loading percentages per machine, in each of the 24 months, we obtain as output of the MILP model. In the first 7 months, only 7 machines are available, therefore no units are allocated on machines 8 and 9. These machines are the ones the company has already ordered and are expected to arrive. As we display in this figure machine 8 can be used starting from month 8, while machine 9 becomes available in month 10.

[Data Page] TotalInventory

Suffix t	Level	basic
1		NonBasic
2		NonBasic
3		NonBasic
4	69931.31999	Basic
5		NonBasic
6		NonBasic
7		NonBasic
8		NonBasic
9		NonBasic
10		NonBasic
11		NonBasic
12		NonBasic
13		NonBasic
14		NonBasic
15	138672.7257	Basic
16	177805.2085	Basic
17	151047.8766	Basic
18	77081.09121	Basic
19		NonBasic
20	132410.8661	Basic
21	115295.8728	Basic
22		NonBasic
23		NonBasic
24		NonBasic

Figure A.2 Total number of units kept in inventory in each of the 24 months

[Data Page] NumberOfReleases

Suffix t	Level	basic
1		NonBasic
2		NonBasic
3		NonBasic
4	1	Basic
5	2	Basic
6		NonBasic
7		NonBasic
8		NonBasic
9		NonBasic
10		NonBasic
11		NonBasic
12	1	Basic
13	1	Basic
14	1	Basic
15		NonBasic
16		NonBasic
17		NonBasic
18	1	Basic
19		NonBasic
20		NonBasic
21		NonBasic
22		NonBasic
23		NonBasic
24		NonBasic

Figure A.3 Number of releases performed in each of the 24 months

The algorithm performs machine releases to facilitate the production of 7 sensors for which no machine releases are available. Figure A.4 shows the product indices associated with each release, while in Figure A.5 we display the demand associated with products 213 and 216. Moreover, in Figure A.6 we show the initial releases of all machines. In here we can notice that no machines are released for products 213 and 216. Hence, since machines are released for products 213 and 216 right when their demand became greater than 0, we can conclude that the model will only perform a certain action when needed or if it leads to smaller total costs.

Month	Product
4	36
5	38
	39
12	30
13	213
14	57
18	216

Figure A.4 Product IDs with zero machine releases

213	214	215	216
132			
132			
132			
132			
132			
132			9000
132			9000
132			9000
132			9000
132			9000
132			9000
132			9000

Figure A.5 Demand for products 213 and 216

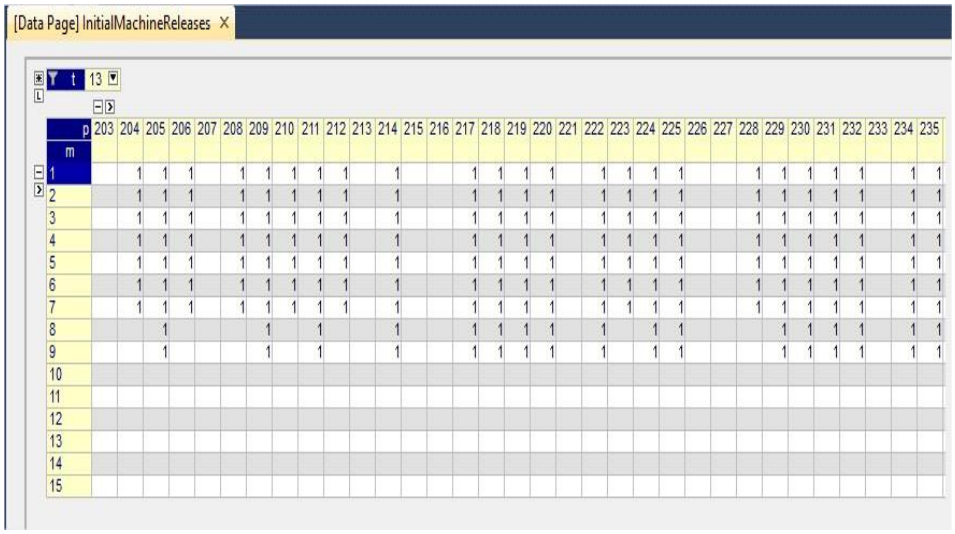

Figure A.6 Initial machine releases for products 213 and 216

Appendix B: LP formulation – Initial Schedule

MINIMIZE

$$\sum_{t=1}^{T} Inventory\ Costs(t) * TotalInventory(t) + ReleasingCost(t) * TotalMachinesReleased(t) + OrderingCost(t) * TotalMachinesOrdered(t)$$

subjected to:

1. $\sum_{k=1}^{K} UnitsProduced(i,k,t) * QualityOEE(i,k) = D(i,t), \forall\ i \in I, t \in M$

2. $UnitsProduced(i,k,t) \leq bigM * InitialMachineReleases(i,k,t), \forall\ i \in I, k \in K, t \in M$

3. $\sum_{i=1}^{I} UnitsProduced(i,k,t) * CT(i,k,t) \leq (1 - Buffer) * AvailabilityOEE(k,t) * DaysAvailable(t) * HoursAvailable * 3600, \forall\ i \in I, k \in K, t \in M$

4. $UnitsProduced(i,k,t) \geq 0, \forall\ i \in I, k \in K, t \in M$

Appendix C:Initial Schedules Comparison

As we describe in Section 5.3 we compare 5 different ways of generating the initial schedule. In Figure C.1 we display the optimal monthly overall loading and the loadings associated with each of the 5 schedules. We obtain the optimal loading by solving the model we introduce in Appendix B and use its results to select the best schedule.

Month	AIMMS	No Sorting	Sorting Demand	Sorting Cycle Times	Sorting Both	Sorting Releases
1	65.169	100.475	100.031	65.564	65.374	85.386
2	70.659	89.854	102.198	70.673	72.313	86.123
3	66.834	91.900	103.859	67.098	67.460	86.069
4	58.400	83.196	89.424	58.400	58.726	77.773
5	82.714	117.959	116.849	87.167	83.719	119.932
6	63.843	100.042	99.218	64.544	64.176	82.340
7	75.644	115.246	101.873	77.151	76.822	95.121
8	50.631	76.052	80.918	50.685	50.985	68.475
9	71.306	108.098	99.253	72.851	72.950	93.679
10	62.778	91.295	98.225	63.034	63.316	83.354
11	57.104	86.729	88.593	58.292	57.661	77.774
12	48.806	78.891	71.029	50.979	49.390	65.978
13	70.960	100.704	110.693	71.610	74.121	96.126
14	76.952	118.209	106.019	79.389	77.861	100.828
15	65.171	101.890	101.230	66.063	66.099	88.766
16	71.984	108.111	97.956	74.420	72.658	98.837
17	76.893	109.741	110.894	77.716	78.637	104.030
18	78.458	107.924	116.986	81.641	79.950	111.633
19	75.282	116.925	118.449	76.015	76.638	113.676
20	59.221	86.946	97.728	59.616	59.735	85.560
21	75.580	112.491	115.371	79.030	77.867	103.608
22	79.321	115.552	120.241	83.514	80.969	122.150
23	72.142	101.656	104.139	72.822	73.364	101.122
24	65.297	96.765	99.513	66.062	65.886	93.998
Average	68.381	100.694	102.112	69.764	69.445	93.431

Figure C.1 Monthly overall loading obtained from AIMMS and the 5 initial schedules

Through the highlighted cells in Figure C.1 we want to show which schedule is closest to the results of the LP model. For three of the options (no sorting, sorting on decreasing demand and sorting on the number of releases) the results are much worse than the optimal solution we display in column 2. However, for the other two options (sorting on increasing cycle times and sorting both on demand and cycle times), the results are very close to the optimal values. Besides the monthly loading, we also compare the schedules in terms of the number of units allocated on the ICM and the percentage these units represent from the monthly volumes. We display these results in Figures C.2 and C.3.

Month	Total Volume	No Sorting	Sorting Demand	Sorting Cycle Times	Sorting Both	Sorting Releases
1	1158464	5655	365	0	0	0
2	1115900	0	24300	0	0	0
3	1334737	0	50870	0	0	0
4	1168322	0	0	0	0	40
5	1539014	221480	207775	0	0	122907
6	1176632	526	0	0	0	1421
7	1291102	181539	22300	0	0	1315
8	1120045	0	0	0	0	1937
9	1461696	114132	0	0	0	9608
10	1437851	0	0	0	0	15195
11	1427521	0	0	0	0	15131
12	1208720	0	0	0	0	20944
13	1650463	10781	163684	0	0	33681
14	1605147	258839	85550	0	0	34317
15	1689872	32028	20842	0	0	34317
16	1776955	133035	0	0	0	34317
17	1804044	154449	172715	0	0	34317
18	1847956	125634	269308	0	0	92232
19	1813421	259110	282420	0	0	104682
20	1554702	0	0	0	0	43317
21	1784223	198050	243702	0	0	43317
22	1861374	238089	309855	0	0	169553
23	1820962	27171	67872	0	0	41517
24	1711010	0	0	0	0	41517

Figure C.2 Number of units assigned on the ICM in each of the 5 initial schedules

Month	No Sorting	Sorting Demand	Sorting Cycle Times	Sorting Both	Sorting Releases
1	0.488	0.032	0	0	0
2	0	2.178	0	0	0
3	0	3.811	0	0	0
4	0	0	0	0	0.003
5	14.391	13.501	0	0	7.986
6	0.045	0	0	0	0.121
7	14.061	1.727	0	0	0.102
8	0	0	0	0	0.173
9	7.808	0	0	0	0.657
10	0	0	0	0	1.057
11	0	0	0	0	1.060
12	0	0	0	0	1.733
13	0.653	9.917	0	0	2.041
14	16.126	5.330	0	0	2.138
15	1.895	1.233	0	0	2.031
16	7.487	0	0	0	1.931
17	8.561	9.574	0	0	1.902
18	6.799	14.573	0	0	4.991
19	14.288	15.574	0	0	5.773
20	0	0	0	0	2.786
21	11.100	13.659	0	0	2.428
22	12.791	16.647	0	0	9.109
23	1.492	3.727	0	0	2.280
24	0	0	0	0	2.426
Average	4.916	4.645	0	0	2.197

Figure C.3 Percentage of the monthly volume assigned on the ICM in each of the5 initial schedules

From these figures, we notice that for the two schedules performing best in terms of monthly loading, the number of units allocated on the ICM is 0. Considering the other schedules, the average percentage of volume allocated on the ICM varies between 4.9% and 2.1%.

Since two of the schedules, sorting based on cycle times and sorting on both cycle times and demand, perform best in exactly 12 out of 24 months, we create a more detailed comparison between them. Figure C.4 shows this comparison.

Month	AIMMS	No Sorting	Sorting Demand	Sorting Cycle Times	Sorting Both	Sorting Releases
1	65.169	100.475	100.031	65.564	65.374	85.386
2	70.659	89.854	102.198	70.673	72.313	86.123
3	66.834	91.900	103.859	67.098	67.460	86.069
4	58.400	83.196	89.424	58.400	58.726	77.773
5	82.714	117.959	116.849	87.167	83.719	119.932
6	63.843	100.042	99.218	64.544	64.176	82.340
7	75.644	115.246	101.873	77.151	76.822	95.121
8	50.631	76.052	80.918	50.685	50.985	68.475
9	71.306	108.098	99.253	72.851	72.950	93.679
10	62.778	91.295	98.225	63.034	63.316	83.354
11	57.104	86.729	88.593	58.292	57.661	77.774
12	48.806	78.891	71.029	50.979	49.390	65.978
13	70.960	100.704	110.693	71.610	74.121	96.126
14	76.952	118.209	106.019	79.389	77.861	100.828
15	65.171	101.890	101.230	66.063	66.099	88.766
16	71.984	108.111	97.956	74.420	72.658	98.837
17	76.893	109.741	110.894	77.716	78.637	104.030
18	78.458	107.924	116.986	81.641	79.950	111.633
19	75.282	116.925	118.449	76.015	76.638	113.676
20	59.221	86.946	97.728	59.616	59.735	85.560
21	75.580	112.491	115.371	79.030	77.867	103.608
22	79.321	115.552	120.241	83.514	80.969	122.150
23	72.142	101.656	104.139	72.822	73.364	101.122
24	65.297	96.765	99.513	66.062	65.886	93.998
Average	68.381	100.694	102.112	69.764	69.445	93.431

Figure C.4 Best two schedules – Difference from optimal

In columns 4 and 6 we display the differences between the results from the two schedules and the optimal values. Again, we use the highlighting to show which method performs best on a monthly basis.

Appendix D:Local Search Parameters for Optimizing the Initial Schedule

In this section we show the results on which we base our choices for the local search parameters. This section focuses on the Simulated Annealing and Tabu Search algorithms for optimizing the initial schedule.

Appendix D.1: Simulated Annealing

In here we show how we select the four parameters for Simulated Annealing. These parameters refer to the initial and final temperature, the cooling factor and the Markov chain length.

We start by selecting the Markov chain length. This value has to be proportional to the number of neighboring solutions, which can be generated from a given current solution. As we mention in Chapter 5, we check the number of candidates from tabu search in order to find our initial value for the Markov chain length. Next, to check the difference in terms of objective function value (monthly overall loading) and running times we also run tests with double this value. Hence, the 2 values we use for this parameter are 5000 and 10000.

Next, we select the initial temperature. We select this temperature such that the acceptance ratio at the beginning of the algorithm is close to 1. We calculate the acceptance ratio from the following formula: $\frac{Number\ of\ Accepted\ Transitions}{Number\ of\ Proposed\ Transitions}$. We display the results when using both values for the Markov chain length in Figures D.1 and D.2.

Starting Temperature	AT	PT	Ratio
50	4048	5000	0.8096
80	4325	5000	0.865
100	4427	5000	0.8854
150	4667	5000	0.9334
200	4725	5000	0.945
300	4835	5000	0.967
400	4839	5000	0.9678
600	4913	5000	0.9826
800	4941	5000	0.9882
1000	4944	5000	0.9888
1200	4953	5000	0.9906

Figure D.1 Acceptance probability for different starting temperatures and a Markov chain length of 5000

Starting Temperature	AT	PT	Ratio
50	8003	10000	0.8003
80	8686	10000	0.8686
100	8860	10000	0.886
150	9287	10000	0.9287
200	9432	10000	0.9432
300	9629	10000	0.9629
400	9752	10000	0.9752
600	9851	10000	0.9851
800	9872	10000	0.9872
1000	9885	10000	0.9885
1200	9923	10000	0.9923

Figure D.2 Acceptance probability for different starting temperatures and a Markov chain length of 10000

We notice that, in both cases, starting from an initial temperature of 150 the acceptance probability is greater than 0.9. We choose 300 as the value of our starting temperature since, in both cases, the acceptance probability was greater than 0.95. Furthermore, knowing that the final temperature has to have a really small value we decide to set it at 0.5.

Next, we select the cooling factor. We do this by experimenting with multiple values and checking the results in terms of objective function value and computational efforts. We vary the cooling factor while keeping all the other factors constant. Moreover, we perform two tests, one for each Markov chain length.

In Figure D.3 we show the results when the Markov chain length is 5000. For each value of alpha we run the algorithm 5 times and display the average values for both monthly loadings and running times.

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Optimal Loading	Running Time (1 month) (s)	Runing Time (1 month) (min)
300	0.5	0.1	5000	74.121	74.120	70.960	8.081	0.135
300	0.5	0.2	5000	74.121	74.120	70.960	10.169	0.169
300	0.5	0.3	5000	74.121	74.120	70.960	13.300	0.222
300	0.5	0.4	5000	74.121	73.202	70.960	15.641	0.261
300	0.5	0.5	5000	74.121	73.598	70.960	22.791	0.380
300	0.5	0.6	5000	74.121	74.120	70.960	29.157	0.486
300	0.5	0.7	5000	74.121	74.120	70.960	37.900	0.632
300	0.5	0.8	5000	74.121	74.120	70.960	66.569	1.109
300	0.5	0.85	5000	74.121	73.594	70.960	88.525	1.475
300	0.5	0.9	5000	74.121	74.120	70.960	126.078	2.101
300	0.5	0.95	5000	74.121	72.018	70.960	260.544	4.342
300	0.5	0.97	5000	74.121	71.491	70.960	451.478	7.525
300	0.5	0.99	5000	74.121	71.584	70.960	1323.681	22.061

Figure D.3 Overall loading and running times for different cooling factors (Starting Temperature 300, Stopping Temperature 0.5, Length of Markov Chain 5000)

We notice that we obtain the minimum value for the objective function for an alpha equal to 0.97. In Figures D.4 and D.5 we show how the objective function values and the running times change with different values of alpha.

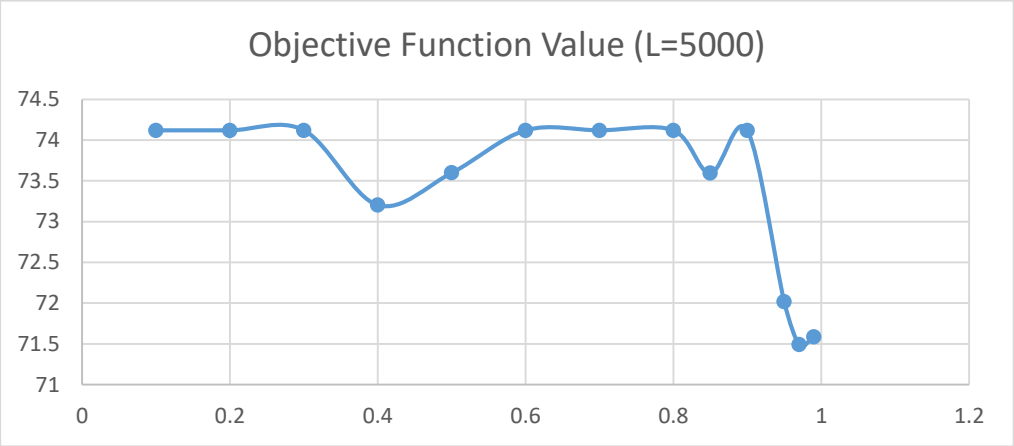

Figure D.4 Objective function values for different cooling factors (Starting Temperature 300, Stopping Temperature0.5, Length of Markov Chain 5000)

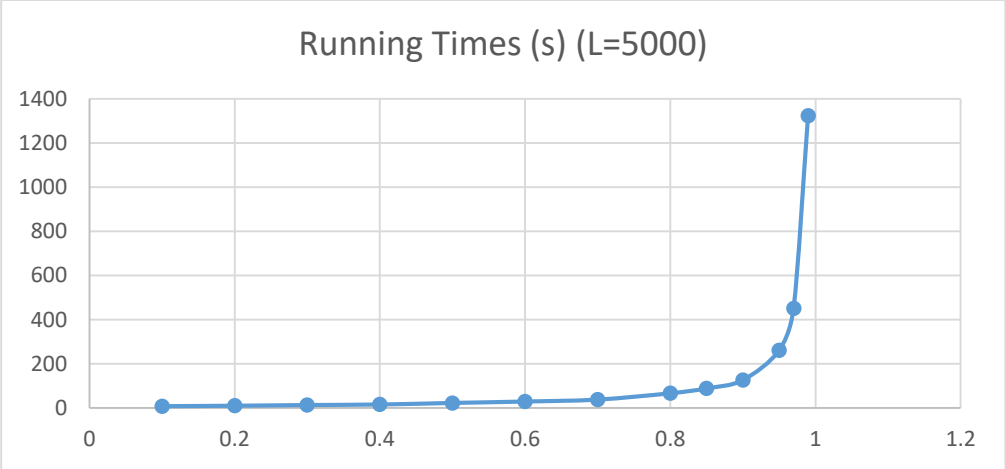

Figure D.5 Running time values for different cooling factors (Starting Temperature 300, Stopping Temperature 0.5, Length of Markov Chain 5000)

Considering the objective function value, we see that we obtain better results when alpha is greater or equal to 0.95. Furthermore, we also notice that on an interval from 0.1 to 0.8 , excluding alpha being equal to 0.4 and 0.5, we can find no difference in terms of the monthly overall loading.

In terms of computational efforts, we see that the overall pattern of the running times shows an exponential increase. However, we also see that on an interval from 0.1 to 0.7, the running times display a very small increase and that starting from an alpha of 0.8 the values start increasing, reaching a total of 22 minutes when alpha equals 0.99. These values show the running times for a single month in our model and we have to consider that the company wants to run our model for a planning horizon of multiple years.

In Figures D.6, D.7, and D.8 we display the same results for a Markov chain length of 10000.

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Optimal Loading	Running Time (1 month) (s)	Runing Time (1 month) (min)
300	0.5	0.1	10000	74.121	74.120	70.960	15.262	0.254
300	0.5	0.2	10000	74.121	74.120	70.960	19.622	0.327
300	0.5	0.3	10000	74.121	73.319	70.960	27.610	0.460
300	0.5	0.4	10000	74.121	73.613	70.960	32.081	0.535
300	0.5	0.5	10000	74.121	73.084	70.960	46.284	0.771
300	0.5	0.6	10000	74.121	73.598	70.960	62.444	1.041
300	0.5	0.7	10000	74.121	73.597	70.960	84.378	1.406
300	0.5	0.8	10000	74.121	72.546	70.960	139.169	2.319
300	0.5	0.85	10000	74.121	72.684	70.960	185.466	3.091
300	0.5	0.9	10000	74.121	72.018	70.960	277.850	4.631
300	0.5	0.95	10000	74.121	72.015	70.960	543.159	9.053
300	0.5	0.97	10000	74.121	71.484	70.960	919.137	15.319
300	0.5	0.99	10000	74.121	71.590	70.960	2781.973	46.366

Figure D.6 Overall loading and running times for different cooling factors (Starting Temperature 300, Stopping Temperature 0.5, Length of Markov Chain 10000)

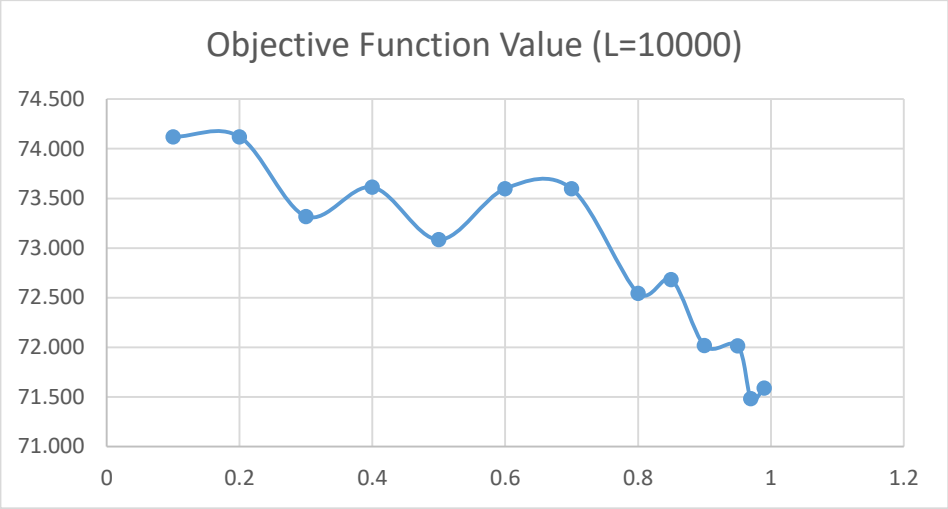

Figure D.7 Objective function values for different cooling factors (Starting Temperature 300, Stopping Temperature 0.5, Length of Markov Chain 10000)

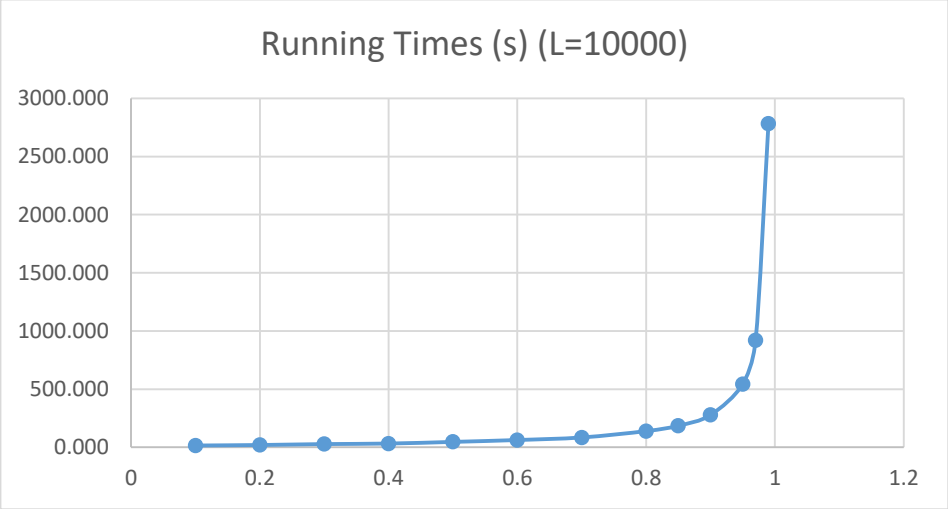

Figure D.8 Running time values for different cooling factors (Starting Temperature 300, Stopping Temperature 0.5, Length of Markov Chain 10000)

For a Markov chain length of 10000 we find an improvement 0f 0.007% in terms of monthly overall loading. In this case, the same value of alpha, 0.97, brings the best results. Although, when using these values for our parameters, the algorithm can find an even better solution the difference in running times between the two is of 7.794 minutes.

In an attempt of reducing the running times we experiment with some other values for the Markov chain length. The following three figures display the results when using 2500, 1250 and 625 as our Markov chain length. Furthermore for the last two tests we set alpha to 0.99. In figure D.9 we display the values as averages of 5 runs, while in figures D.10 and D.11 we display the values of each individual run.

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Optimal Loading	Running Time (1 month) (s)	Runing Time (1 month) (min)
300	0.5	0.95	2500	74.121	73.594	70.960	133.353	2.223
300	0.5	0.97	2500	74.121	72.542	70.960	230.246	3.837
300	0.5	0.99	2500	74.121	71.588	70.960	745.594	12.427

Figure D.9 Overall loading and running times for different cooling factors (Starting Temperature 300, Stopping Temperature 0.5, Length of Markov Chain 2500)

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Optimal Loading	Running Time (1 month) (s)	Runing Time (1 month) (min)
300	0.5	0.99	1250	74.121	74.120	70.960	350.578	5.843
300	0.5	0.99	1250	74.121	71.485	70.960	356.922	5.949
300	0.5	0.99	1250	74.121	71.515	70.960	377.375	6.290
300	0.5	0.99	1250	74.121	71.487	70.960	345.937	5.766
300	0.5	0.99	1250	74.121	71.491	70.960	344.016	5.734

Figure D.10 Overall loading and running times for α=0.99 (Starting Temperature 300, Stopping Temperature 0.5, Length of Markov Chain 1250)

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Optimal Loading	Running Time (1 month) (s)	Runing Time (1 month) (min)
300	0.5	0.99	625	74.121	74.120	70.960	177.031	2.951
300	0.5	0.99	625	74.121	74.120	70.960	176.281	2.938
300	0.5	0.99	625	74.121	71.501	70.960	185.812	3.097
300	0.5	0.99	625	74.121	71.501	70.960	182.438	3.041
300	0.5	0.99	625	74.121	71.494	70.960	183.719	3.062

Figure D.11 Overall loading and running times for α=0.99 (Starting Temperature 300, Stopping Temperature 0.5, Length of Markov Chain 625)

From Figure D.11 we notice that the only configuration which could find an objective function value close to the previous values is for alpha 0.99. We also find similar values for a Markov chain length of 1250 or 625. However, such values do not appear in all five runs. When selecting our parameters for this algorithm we look for the values which lead to good results for all five runs.

Appendix D.2: Tabu Search

In this section we show how we select the two parameters for Tabu search. These parameters are: the number of iterations and the tabu list length. We use different values for the number of iterations and, in each test, we keep this value constant and vary the tabu list length. We vary the tabu list length starting from a value of 5 till we reach the value we use for the number of iterations. We aim to find an interval for

the tabu list length, chosen as percentage from the total number of iterations, in which the objective function value results in the best solution.

In Figure D.12 we show the results when setting the number of iterations to 100.

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Optimal Loading	Running Time (1 month) (s)	Running Time (1 month) (min)
100	5	5.0	74.121	73.896	70.960	49.047	0.817
100	10	10.0	74.121	73.887	70.960	52.266	0.871
100	20	20.0	74.121	73.887	70.960	49.437	0.824
100	30	30.0	74.121	73.886	70.960	51.656	0.861
100	40	40.0	74.121	73.886	70.960	55.422	0.924
100	50	50.0	74.121	73.886	70.960	41.125	0.685
100	60	60.0	74.121	73.886	70.960	52.047	0.867
100	70	70.0	74.121	73.886	70.960	52.266	0.871
100	80	80.0	74.121	73.886	70.960	39.75	0.663
100	90	90.0	74.121	73.887	70.960	40.859	0.681
100	100	100.0	74.121	73.887	70.960	40.406	0.673

Figure D.12 Overall loading and running times for different lengths of the tabu list (100 iterations)

The best values, in terms of monthly overall loading, occur when the tabu list length varies from 30% to 80% of the total number of iterations. However, since, for this test, the difference in objective function values is very small, we are not able to draw relevant conclusions. In Figure D.13 we show how the objective function values varies for different values of the tabu list length. We see that, excluding the tabu list length of 5, the values are very similar.

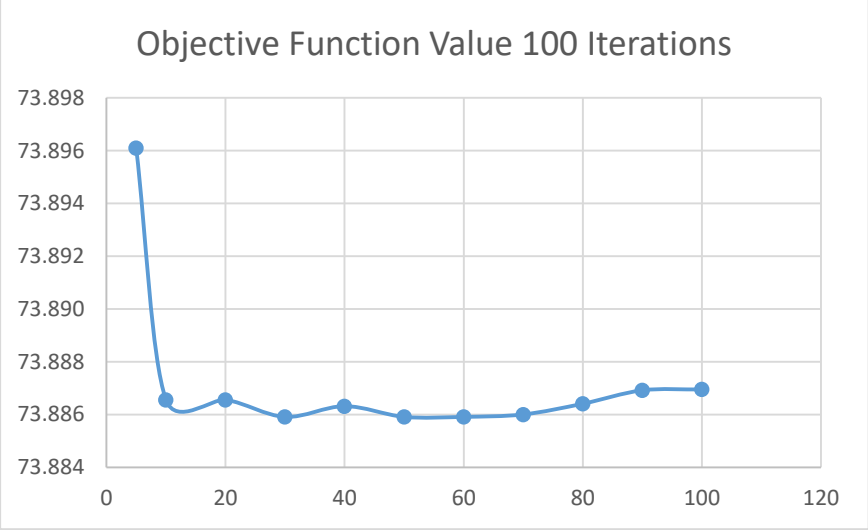

Figure D.13 Objective function values for different lengths of the tabu list (100 iterations)

Next, in Figures D.14 and D.15 we show the same results when using 300 as the number of iterations.

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Optimal Loading	Running Time (1 month) (s)	Running Time (1 month) (min)
300	5	1.7	74.121	73.896	70.960	150.172	2.503
300	10	3.3	74.121	73.887	70.960	120.406	2.007
300	20	6.7	74.121	73.887	70.960	125.141	2.086
300	30	10.0	74.121	73.886	70.960	119.25	1.988
300	40	13.3	74.121	73.886	70.960	121.468	2.024
300	50	16.7	74.121	73.886	70.960	116.36	1.939
300	60	20.0	74.121	73.886	70.960	132.953	2.216
300	70	23.3	74.121	73.886	70.960	128.453	2.141
300	80	26.7	74.121	73.885	70.960	127.594	2.127
300	90	30.0	74.121	73.886	70.960	127.688	2.128
300	100	33.3	74.121	73.885	70.960	124.609	2.077
300	110	36.7	74.121	73.886	70.960	128.75	2.146
300	120	40.0	74.121	73.886	70.960	119.157	1.986
300	130	43.3	74.121	73.886	70.960	125.407	2.090
300	150	50.0	74.121	73.886	70.960	130.735	2.179
300	170	56.7	74.121	73.887	70.960	135.953	2.266
300	200	66.7	74.121	73.887	70.960	146.14	2.436
300	230	76.7	74.121	73.887	70.960	118.562	1.976
300	250	83.3	74.121	73.886	70.960	117.328	1.955
300	270	90.0	74.121	73.886	70.960	131.016	2.184
300	300	100.0	74.121	73.887	70.960	139.187	2.320

Figure D.14 Overall loading and running times for different lengths of the tabu list (300 iterations)

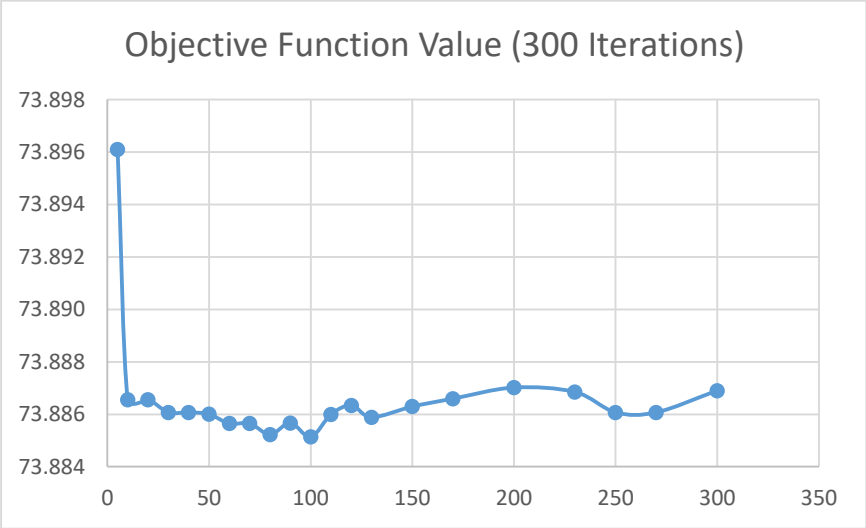

Figure D.15 Objective function values for different lengths of the tabu list (300 iterations)

We obtain similar results, small variation in terms of objective function values, for 300 iterations. However, because in this case the best 2 values we find are for tabu list length equal to 26.7% and 33.3% of the number of iterations, we decide to continue our tests by varying the tabu list length on an interval from 20% to 35%, 40% or 45%.

In Figure D.16 we show our results when using 1000 iterations and a tabu list length varying from 20% to 40%. No difference in terms of monthly overall loading is present.

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Optimal Loading	Running Time (1 month) (s)	Running Time (1 month) (min)
1000	200	20	74.121	73.887	70.960	397.266	6.621
1000	250	25	74.121	73.887	70.960	395.875	6.598
1000	300	30	74.121	73.887	70.960	450.203	7.503
1000	350	35	74.121	73.887	70.960	375.219	6.254
1000	400	40	74.121	73.887	70.960	391.484	6.525

Figure D.16 Overall loading and running times for different lengths of the tabu list (1000 iterations)

Next, we increase the number of iterations to 2500, 5000 and 10000 and vary the tabu list length accordingly.

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Optimal Loading	Running Time (1 month) (s)	Running Time (1 month) (min)
2500	500	20	74.121	73.890	70.960	954.578	15.910
2500	625	25	74.121	73.887	70.960	957.188	15.953
2500	750	30	74.121	71.822	70.960	813.125	13.552
2500	875	35	74.121	73.891	70.960	928.797	15.480
2500	1000	40	74.121	73.889	70.960	995.954	16.599
2500	1125	45	74.121	73.886	70.960	906.532	15.109

Figure D.17 Overall loading and running times for different lengths of the tabu list (2500 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Optimal Loading	Running Time (1 month) (s)	Running Time (1 month) (min)
5000	1000	20	74.121	71.821	70.960	1811.891	30.1982
5000	1250	25	74.121	73.890	70.960	1995.218	33.2536
5000	1500	30	74.121	71.827	70.960	1632.062	27.2010
5000	1750	35	74.121	71.830	70.960	1520.359	25.3393
5000	2000	40	74.121	73.896	70.960	1964.125	32.7354
5000	2250	45	74.121	73.892	70.960	2053.797	34.2300

Figure D.18 Overall loading and running times for different lengths of the tabu list (5000 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Optimal Loading	Running Time (1 month) (s)	Running Time (1 month) (min)
10000	2000	20	74.121	71.824	70.960	3957.558	65.959
10000	3000	30	74.121	71.828	70.960	3939.743	65.662
10000	3500	35	74.121	71.829	70.960	3459.26	57.654

Figure D.19 Overall loading and running times for different lengths of the tabu list (10000 iterations)

The best value for the objective function value we find is 71.821 and it occurs when running 5000 iterations with a tabu list length of 1000. We obtain a similar monthly loading when using 2500 iterations and 750 as tabu list length.

In Figure D.20 we show the average running times for different number of iterations. Unlike Simulated Annealing, Tabu Search shows a linear increase in terms of computational efforts.

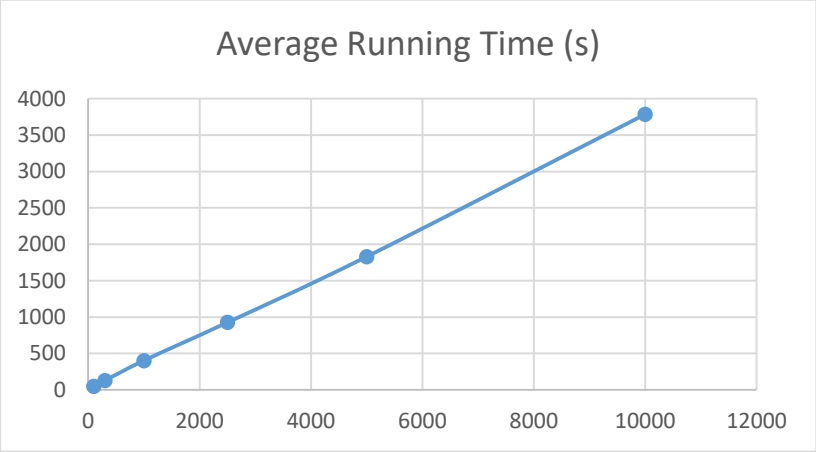

Figure D.20 Average running time values for different number of iterations

Appendix E:Additional Results - Optimizing the Initial Schedule

As we state in Section 5.4.1 we compare the results we obtain when using either Simulated Annealing or Tabu Search in terms of monthly loading. We first compare these results with the loading of the initial schedule in order to determine the level of improvement that each local search method could bring. Next, we compare the results with the optimal values.

Month	Initial Schedule	Simulated Annealing	Difference(SA-Init)	Tabu Search	Difference(Init-TS)
1	65.374	65.374	0.000	65.374	0.000
2	72.313	71.543	-0.770	71.634	-0.680
3	67.460	67.460	0.000	67.460	0.000
4	58.726	58.438	-0.288	58.449	-0.277
5	83.717	83.717	0.000	83.477	-0.240
6	64.175	64.175	0.000	64.156	-0.019
7	76.821	76.404	-0.417	76.436	-0.385
8	50.983	50.692	-0.291	50.686	-0.297
9	72.949	72.279	-0.670	72.229	-0.719
10	63.314	63.314	0.000	63.246	-0.068
11	57.659	57.330	-0.329	57.585	-0.075
12	49.389	49.162	-0.227	49.088	-0.301
13	74.120	71.491	-2.629	71.818	-2.302
14	77.860	77.615	-0.245	77.654	-0.205
15	66.098	65.381	-0.717	65.742	-0.356
16	72.657	72.657	0.000	72.429	-0.228
17	78.636	78.597	-0.039	78.487	-0.148
18	79.949	79.949	0.000	79.922	-0.027
19	76.636	76.379	-0.257	75.918	-0.718
20	59.734	59.734	0.000	59.376	-0.358
21	77.866	76.619	-1.247	77.707	-0.159
22	80.968	79.961	-1.007	80.217	-0.751
23	73.363	73.329	-0.033	73.317	-0.046
24	65.885	65.885	0.000	65.881	-0.004
Maximum Difference			0.000		0.000
Average Difference			-0.382		-0.348
Minimum Difference			-2.629		-2.302

Figure E.1 Monthly overall loading – Comparison between SA and TS related to their initial solution

Just as before, in columns 4 and 6 we display the differences (in terms of overall loading) between the initial schedule and the outputs of the local search algorithms.

Month	AIMMS	Simulated Annealing	Difference(SA-Optimal)	Tabu Search	Difference(TS-Optimal)
1	65.169	65.374	0.205	65.374	0.205
2	70.659	71.543	0.884	71.634	0.975
3	66.834	67.460	0.626	67.460	0.626
4	58.400	58.438	0.038	58.449	0.049
5	82.714	83.717	1.003	83.477	0.762
6	63.843	64.175	0.332	64.156	0.313
7	75.644	76.404	0.760	76.436	0.792
8	50.631	50.692	0.061	50.686	0.055
9	71.306	72.279	0.973	72.229	0.924
10	62.778	63.314	0.536	63.246	0.469
11	57.104	57.330	0.226	57.585	0.480
12	48.806	49.162	0.356	49.088	0.282
13	70.960	71.491	0.531	71.818	0.858
14	76.952	77.615	0.663	77.654	0.703
15	65.171	65.381	0.211	65.742	0.572
16	71.984	72.657	0.674	72.429	0.446
17	76.893	78.597	1.703	78.487	1.594
18	78.458	79.949	1.491	79.922	1.464
19	75.282	76.379	1.097	75.918	0.636
20	59.221	59.734	0.513	59.376	0.155
21	75.580	76.619	1.039	77.707	2.127
22	79.321	79.961	0.639	80.217	0.896
23	72.142	73.329	1.188	73.317	1.175
24	65.297	65.885	0.588	65.881	0.584
Maximum Difference			1.703		2.127
Average Difference			0.681		0.714
Minimum Difference			0.038		0.049

Figure E.2 Monthly overall loading – Comparison between SA, TS related to the optimal values

Appendix F:Local Search Parameters for Releasing Existing Machines

In this section we show the results on which we base our choices for local search parameters. This section focuses on the Simulated Annealing and Tabu Search algorithms for releasing existing machines.

Appendix F.1: Simulated Annealing

In a similar manner to the one we describe in Appendix D, we select the parameters for this Simulated Annealing algorithm. This time, we set the Markov chain length to a high number and stop the algorithm when no more products are assigned on the ICM. Varying the starting temperature, we check how many transitions does is actually propose and the resulting acceptance probability. We pick a starting temperature of 20. Figure F.1 shows these results.

Starting Temperature	Accepted Transitions	Proposed Transitions	Ratio
1	15	16	0.938
5	13	14	0.929
10	14	15	0.933
15	14	15	0.933
20	18	19	0.947
25	15	16	0.938
30	15	16	0.938
40	13	14	0.929
50	20	21	0.952

Figure F.1 Acceptance probability for different starting temperatures

Given the results, we set the initial temperature to 20. Moreover, we keep the final temperature to its previous value of 0.5.

From the number of proposed transitions in Figure F.1, we check the minimum, maximum, and average values and use them as three different Markov chain lengths to experiment with. We show the results we obtain when using these three values and an alpha of 0.1 in Figures F.2, F.3, and F.4. In each Figure we display the values we obtain when running the algorithm five times with the same settings. Knowing our input data, we know that the model has to release machines for ten products, hence, we started our tests by setting the cooling factor to 0.1. Given that when using 0.1 for the cooling factor, the model manages to release machines for all ten products we did not vary its values further.

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Optimal Loading	Running Time (1 month) (s)	Runing Time (1 month) (min)
20	0.5	0.1	17	65.374	68.561	65.169	2.344	0.039
20	0.5	0.1	17	65.374	69.064	65.169	2.859	0.048
20	0.5	0.1	17	65.374	67.895	65.169	2.141	0.036
20	0.5	0.1	17	65.374	67.878	65.169	1.984	0.033
20	0.5	0.1	17	65.374	67.934	65.169	1.984	0.033

Figure F.2 Overall loading and running times for α=0.1 (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 17)

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Optimal Loading	Running Time (1 month) (s)	Runing Time (1 month) (min)
20	0.5	0.1	14	65.374	68.970	65.169	2.812	0.047
20	0.5	0.1	14	65.374	68.962	65.169	2.703	0.045
20	0.5	0.1	14	65.374	66.286	65.169	2.297	0.038
20	0.5	0.1	14	65.374	70.313	65.169	2.797	0.047
20	0.5	0.1	14	65.374	68.341	65.169	3.094	0.052

Figure F.3 Overall loading and running times for α=0.1 (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 14)

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Optimal Loading	Running Time (1 month) (s)	Runing Time (1 month) (min)
20	1	0.1	21	65.374	69.310	65.169	1.594	0.027
20	1	0.1	21	65.374	67.241	65.169	2.250	0.038
20	1	0.1	21	65.374	68.505	65.169	2.421	0.040
20	1	0.1	21	65.374	68.705	65.169	1.844	0.031
20	1	0.1	21	65.374	69.359	65.169	2.329	0.039

Figure F.4 Overall loading and running times for α=0.1 (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 15)

In the previous test there is enough available capacity within existing machines to fit the volumes of all products allocated on the infinite capacity machine. To further analyse the behaviour of the local search algorithms, we adjust our input data to increase the loading of the existing machines. We do this by increasing the demand of each product by a constant value. By increasing the loading of the existing machines, we expect the model to not be able to fit the number of units associated with all the products assigned on the ICM. Furthermore, when performing a new release to fit one of these products, some of the products previously assigned within existing machines might be reallocated to the ICM. At that point, the model is allowed to pick the best release to perform from an increased list of neighbours. We use the same values for the starting and end temperature, and the length of Markov chain, and we vary the cooling factor. In Figure F.5 we display the overall loading we obtain when counting the number of units assigned on the ICM. We calculate this loading through the following formula:

$$\frac{Loading\ Existing\ Machines + Loading\ ICM}{Available\ Capacity}$$

Figure F.6 shows the loading when we exclude the volumes allocated on the ICM. In both figures we display the running times associated with each cooling factor. Just as before, the values we display in both figures represent the averages from the 5 repetitions we perform for each cooling factor.

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Running Time (1 month) (s)	Running Time (1 month) (min)
20	0.5	0.1	17	114.349	111.243	10.328	0.172
20	0.5	0.2	17	114.349	110.973	17.410	0.290
20	0.5	0.3	17	114.349	111.162	22.234	0.371
20	0.5	0.4	17	114.349	110.040	31.240	0.521
20	0.5	0.5	17	114.349	110.040	38.656	0.644
20	0.5	0.6	17	114.349	109.238	42.825	0.714
20	0.5	0.7	17	114.349	109.974	60.910	1.015
20	0.5	0.8	17	114.349	110.232	109.722	1.829
20	0.5	0.9	17	114.349	109.083	217.909	3.632
20	0.5	0.99	17	114.349	109.332	2180.331	36.339

Figure F.5 Overall loading (both existing machines and ICM) and running times for different cooling factors (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 17)

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Running Time (1 month) (s)	Running Time (1 month) (min)
20	0.5	0.1	17	93.957	100	10.328	0.172
20	0.5	0.2	17	93.957	100	17.410	0.290
20	0.5	0.3	17	93.957	100	22.234	0.371
20	0.5	0.4	17	93.957	100	31.240	0.521
20	0.5	0.5	17	93.957	100	38.656	0.644
20	0.5	0.6	17	93.957	100	42.825	0.714
20	0.5	0.7	17	93.957	100	60.910	1.015
20	0.5	0.8	17	93.957	100	109.722	1.829
20	0.5	0.9	17	93.957	100	217.909	3.632
20	0.5	0.99	17	93.957	100	2180.331	36.339

Figure F.6 Overall loading (existing machines) and running times for different cooling factors (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 17)

As we notice in Figure F.5, when using a cooling factor of 0.9 we obtain the best results in terms of loading. In Figure F.6 we see that in all cases the existing machines are fully loaded, which means that the algorithm performs as many releases as possible until no further units can fit. Unlike the previous test, this time the results show a positive loading on the ICM. Therefore, for a clearer overview, in Figure F.7 we display the number of units allocated on both the existing machines and the ICM. Using a cooling factor of 0.9 results in the smallest number of units assigned on the ICM.

T0	T1	Alpha	L	Volume IS (Existing Machines)	Volume SA (Existing Machines)	Volume IS (ICM)	Volume SA (ICM)
20	0.5	0.1	17	1609058	1663527	121406	66937
20	0.5	0.2	17	1609058	1665137	121406	65327
20	0.5	0.3	17	1609058	1664009	121406	66455
20	0.5	0.4	17	1609058	1670689	121406	59775
20	0.5	0.5	17	1609058	1670690	121406	59774
20	0.5	0.6	17	1609058	1675463	121406	55001
20	0.5	0.7	17	1609058	1671083	121406	59381
20	0.5	0.8	17	1609058	1669546	121406	60918
20	0.5	0.9	17	1609058	1676389	121406	54075
20	0.5	0.99	17	1609058	1674906	121406	55558

Figure F.7 Number of units (both existing machines and ICM) for different cooling factors (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 17)

Appendix F.2:Tabu Search

Knowing that the algorithm needs to release machines for ten products, we only vary the number of iterations between 5 and 20 and keep the tabu list length to a constant value. Since the capacity allows it, we expect the algorithm to perform a machine release for one product in each iteration. Hence, we expect the number of iterations it needs to release machines for all products allocated on the ICM to be 10. As we expect, the algorithm requires ten iterations to reduce the loading from the ICM to 0.

Number Of Iterations	Tabu List Length	Loading Initial Schedule	Loading TS	Optimal Loading	Running Time (1 month) (s)	Running Time (1 month) (min)
5	5	65.374	66.125	65.169	54.25	0.904
7	5	65.374	65.460	65.169	65.735	1.096
10	5	65.374	65.374	65.169	69.609	1.160
15	5	65.374	65.374	65.169	69.657	1.161
20	5	65.374	65.374	65.169	62.938	1.049

Figure F.8 Overall loading and running times for different number of iterations (tabu list length 5)

Given that the capacity allows to fit all the products, without having to decide which product(s) should fit to lead to a better objective function, we believe that the value of the tabu list length should have no impact for this particular case. We keep the number of iterations constant to a value of 10 and vary the tabu list length from 1 to 10 to check if any change occur in terms of monthly overall loading. As we expect, the monthly overall loadings are equal for all ten different values of the tabu list length.

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Optimal Loading	Running Time (1 month) (s)	Running Time (1 month) (min)
10	1	10	65.374	65.374	65.169	72.265	1.204
10	2	20	65.374	65.374	65.169	69.375	1.156
10	3	30	65.374	65.374	65.169	69.797	1.163
10	4	40	65.374	65.374	65.169	68.75	1.146
10	5	50	65.374	65.374	65.169	69.609	1.160
10	6	60	65.374	65.374	65.169	69.266	1.154
10	7	70	65.374	65.374	65.169	70.969	1.183
10	8	80	65.374	65.374	65.169	69.407	1.157
10	9	90	65.374	65.374	65.169	70.125	1.169
10	10	100	65.374	65.374	65.169	69.516	1.159

Figure F.9 Overall loading and running times for different lengths of the tabu list (10 iterations)

Just as for Simulated Annealing, we check the results after increasing the demand of each product. We start by using the same number of iterations and analyze the results when varying the length of the tabu list. We display the loadings when considering both the existing machines and the ICM, and only the existing machines in Figures F.10 and F.11. Next, in Figure F.12 we display the number of units assigned on both the existing machines and the ICM.

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
10	1	10	114.349	105.598	113.032	1.884
10	2	20	114.349	105.598	106.969	1.783
10	3	30	114.349	105.598	122.203	2.037
10	4	40	114.349	105.598	113.579	1.893
10	5	50	114.349	105.598	120.953	2.016
10	6	60	114.349	105.598	141.703	2.362
10	7	70	114.349	105.598	123.125	2.052
10	8	80	114.349	105.598	118.594	1.977
10	9	90	114.349	105.598	101.922	1.699
10	10	100	114.349	105.598	114.312	1.905

Figure F.10 Overall loading (both existing machines and ICM) and running times for different lengths of the tabu list (10 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
10	1	10	93.957	100	113.032	1.884
10	2	20	93.957	100	106.969	1.783
10	3	30	93.957	100	122.203	2.037
10	4	40	93.957	100	113.579	1.893
10	5	50	93.957	100	120.953	2.016
10	6	60	93.957	100	141.703	2.362
10	7	70	93.957	100	123.125	2.052
10	8	80	93.957	100	118.594	1.977
10	9	90	93.957	100	101.922	1.699
10	10	100	93.957	100	114.312	1.905

Figure F.11 Overall loading (existing machines) and running times for different lengths of the tabu list (10 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Volume Initial Schedule (Existing Machines)	Volume TS (Existing Machines)	Volume Initial Schedule (ICM)	Volume TS (ICM)
10	1	10	1609058	1697138	121406	33326
10	2	20	1609058	1697138	121406	33326
10	3	30	1609058	1697138	121406	33326
10	4	40	1609058	1697138	121406	33326
10	5	50	1609058	1697138	121406	33326
10	6	60	1609058	1697138	121406	33326
10	7	70	1609058	1697138	121406	33326
10	8	80	1609058	1697138	121406	33326
10	9	90	1609058	1697138	121406	33326
10	10	100	1609058	1697138	121406	33326

Figure F.12 Number of units (both existing machines and ICM) for different lengths of the tabu list (10 iterations)

We notice that no matter the tabu list length the results in terms of loading and number of units are the same. Furthermore, we also notice that, just as Simulated Annealing, the algorithm manages to fully load the existing machines. To check if we can find a better set of products to assign on the ICM we increase the number of iterations to 30 and 50 and repeat the same test. We display the results in the following figures.

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule (Current)	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
30	3	10	114.349	105.598	155.828	2.597
30	6	20	114.349	105.598	144.188	2.403
30	9	30	114.349	105.598	166.610	2.777
30	12	40	114.349	105.598	163.813	2.730
30	15	50	114.349	105.598	155.235	2.587
30	18	60	114.349	105.598	158.046	2.634
30	21	70	114.349	105.598	150.187	2.503
30	24	80	114.349	105.598	143.579	2.393
30	27	90	114.349	105.598	142.578	2.376
30	30	100	114.349	105.598	141.750	2.363

Figure F.13 Overall loading (both existing machines and ICM) and running times for different lengths of the tabu list (30 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule (Current)	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
30	3	10	93.957	100	155.828	2.597
30	6	20	93.957	100	144.188	2.403
30	9	30	93.957	100	166.610	2.777
30	12	40	93.957	100	163.813	2.730
30	15	50	93.957	100	155.235	2.587
30	18	60	93.957	100	158.046	2.634
30	21	70	93.957	100	150.187	2.503
30	24	80	93.957	100	143.579	2.393
30	27	90	93.957	100	142.578	2.376
30	30	100	93.957	100	141.750	2.363

Figure F.14 Overall loading (existing machines) and running times for different lengths of the tabu list (30 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Volume Initial Schedule (Existing Machines)	Volume TS (Existing Machines)	Volume Initial Schedule (ICM)	Volume TS (ICM)
30	3	10	1609058	1697138	121406	33326
30	6	20	1609058	1697138	121406	33326
30	9	30	1609058	1697138	121406	33326
30	12	40	1609058	1697138	121406	33326
30	15	50	1609058	1697138	121406	33326
30	18	60	1609058	1697138	121406	33326
30	21	70	1609058	1697138	121406	33326
30	24	80	1609058	1697138	121406	33326
30	27	90	1609058	1697138	121406	33326
30	30	100	1609058	1697138	121406	33326

Figure F.15 Number of units (both existing machines and ICM) for different lengths of the tabu list (30 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule (Current)	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
50	5	10	114.349	105.598	139.782	2.330
50	10	20	114.349	105.598	140.938	2.349
50	15	30	114.349	105.598	140.406	2.340
50	20	40	114.349	105.598	140.250	2.338
50	25	50	114.349	105.598	141.093	2.352
50	30	60	114.349	105.598	141.703	2.362
50	35	70	114.349	105.598	142.344	2.372
50	40	80	114.349	105.598	140.766	2.346
50	45	90	114.349	105.598	140.984	2.350
50	50	100	114.349	105.598	120.938	2.016

Figure F.16 Overall loading (both existing machines and ICM) and running times for different lengths of the tabu list (50 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule (Current)	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
50	5	10	93.957	100	139.782	2.330
50	10	20	93.957	100	140.938	2.349
50	15	30	93.957	100	140.406	2.340
50	20	40	93.957	100	140.250	2.338
50	25	50	93.957	100	141.093	2.352
50	30	60	93.957	100	141.703	2.362
50	35	70	93.957	100	142.344	2.372
50	40	80	93.957	100	140.766	2.346
50	45	90	93.957	100	140.984	2.350
50	50	100	93.957	100	120.938	2.016

Figure F.17 Overall loading (existing machines) and running times for different lengths of the tabu list (50 iterations)

Iterations	Tabu List Length	Percentage from Iterations	Volume Initial Schedule (Existing Machines)	Volume TS (Existing Machines)	Volume Initial Schedule (ICM)	Volume TS (ICM)
50	5	10	1609058	1697138	121406	33326
50	10	20	1609058	1697138	121406	33326
50	15	30	1609058	1697138	121406	33326
50	20	40	1609058	1697138	121406	33326
50	25	50	1609058	1697138	121406	33326
50	30	60	1609058	1697138	121406	33326
50	35	70	1609058	1697138	121406	33326
50	40	80	1609058	1697138	121406	33326
50	45	90	1609058	1697138	121406	33326
50	50	100	1609058	1697138	121406	33326

Figure F.18 Number of units (both existing machines and ICM) times for different lengths of the tabu list (50 iterations)

By increasing the number of iterations we do not notice any change in the results. The algorithm always fully loads the machines and, based on the loadings and volumes we observe, we can conclude that the products it chooses are always the same or they have equal volumes allocated on the ICM and equal cycle times on existing machines.

Appendix G:Additional Results – Releasing Existing Machines

As we mention in Section 5.4.2, we compare the monthly loading we obtain from both local search algorithms with the current and target initial schedule. The current initial schedule shows the loading, including the ICM, we obtain when generating the initial schedule for this test. The target initial schedule shows the loading values we obtain before cancelling the machine releases of the 10 products. Hence, by comparing the results with the current initial schedule we see which algorithm is able to improve it the most. By comparing the results with the target initial schedule we aim to observe if the metaheuristic can reach similar results by only performing machine releases. In Figures G.1 and G.2 we show these comparisons.

Month	Current Initial Schedule	Simulated Annealing	Difference(SA - Init)	Tabu Search	Difference(TS - Init)
1	76.689	67.776	-8.912	65.374	-11.314
2	82.887	74.891	-7.996	72.313	-10.573
3	79.817	71.074	-8.743	67.460	-12.357
4	71.241	62.778	-8.463	58.726	-12.515
5	95.614	85.310	-10.304	83.533	-12.081
6	73.395	67.829	-5.566	64.167	-9.228
7	89.137	78.815	-10.322	76.504	-12.633
8	60.769	54.410	-6.359	50.983	-9.786
9	82.732	74.171	-8.561	73.069	-9.663
10	72.505	66.956	-5.549	63.314	-9.191
11	66.831	59.834	-6.998	57.731	-9.101
12	58.102	51.543	-6.559	49.262	-8.840
Maximum Difference			-5.549		-8.840
Average Difference			-7.861		-10.607
Minimum Difference			-10.322		-12.633

Figure G.1 Monthly overall loading – Comparison between SA, TS related to the current initial solution

Month	Target Initial Schedule	Simulated Annealing	Difference(SA - Init)	Tabu Search	Difference(TS - Init)
1	65.374	67.776	2.402	65.374	0.000
2	72.313	74.891	2.578	72.313	0.000
3	67.460	71.074	3.614	67.460	0.000
4	58.726	62.778	4.052	58.726	0.000
5	83.719	85.310	1.591	83.533	-0.186
6	64.176	67.829	3.653	64.167	-0.010
7	76.822	78.815	1.993	76.504	-0.318
8	50.985	54.410	3.426	50.983	-0.001
9	72.950	74.171	1.221	73.069	0.120
10	63.316	66.956	3.640	63.314	-0.001
11	57.661	59.834	2.173	57.731	0.070
12	49.390	51.543	2.152	49.262	-0.128
Maximum Difference			4.052		0.120
Average Difference			2.708		-0.038
Minimum Difference			1.221		-0.318

Figure G.2 Monthly overall loading – Comparison between SA, TS related to the target solution

In Figure G.3 we display a comparison of the running times incurred by the two local search algorithms.

Month	Simulated Annealing	Tabu Search
1	1.578	74.047
2	1.719	47.156
3	1.672	52.672
4	2.14	38.766
5	3.157	117.281
6	1.641	43.093
7	2.609	69.562
8	0.343	9.985
9	1.329	28.547
10	1.172	32.765
11	0.109	5.86
12	0.516	12.875
Max	3.157	117.281
Average	1.499	44.384
Min	0.109	5.86

Figure G.3 Running times – Comparison between SA and TS

Next, we show a similar comparison of the two algorithms when we increase the demand of each product to further load the existing machines. In Figure G.4 we compare the results when considering the loading of both existing machines and ICM, while in Figure G.5 we show the results related only to the existing machines. Furthermore, in Figure G.6 we show the number of units allocated on both the existing machines and the ICM.

Month	Initial Schedule	Simulated Annealing	Difference (Init- SA)	Tabu Search	Difference (Init- TS)
1	114.349	107.970	6.380	105.5975393	8.752
2	129.287	127.937	1.350	127.1658662	2.121
3	113.944	111.775	2.169	100.7467789	13.197
4	106.440	96.876	9.564	92.1951166	14.245
5	157.731	152.556	5.175	152.3781711	5.353
6	109.193	102.819	6.374	98.79553204	10.397
7	138.330	135.851	2.479	135.6691358	2.661
8	89.238	80.976	8.262	77.52540466	11.713
9	114.482	111.376	3.106	108.1516958	6.331
10	99.933	92.981	6.952	89.42274479	10.510
11	92.291	85.085	7.206	81.92722451	10.364
12	83.255	75.593	7.661	72.89978357	10.355
Maximum Difference			9.564		14.245
Average Difference			5.556		8.833
Minimum Difference			1.350		2.121

Figure G.4 Monthly overall loading (both existing machines and ICM) – Comparison between SA, TS related to the initial solution – Increased demand

Month	Initial Schedule	Simulated Annealing	Tabu Search
1	93.957	100	100
2	100	100	100
3	92.546	100	99.948
4	84.749	96.876	91.850
5	100	100	100
6	91.646	100	98.439
7	100	100	100
8	70.482	80.976	77.224
9	95.710	100	100
10	83.209	92.981	89.135
11	75.559	85.085	81.659
12	66.729	75.593	72.632

Figure G.5 Monthly overall loading (existing machines) – Comparison between SA and TS and the initial solution – Increased demand

In Figure G.4 we notice that Tabu Search decreases the overall monthly loading the most in all 12 months. Moreover, in Figure G.5, we see that both algorithms lead to the same results in 4 months, while for the rest, Tabu Search results in a smaller loading. When checking the number of units assigned on the ICM after running the two metaheuristics, we notice that in 7 out of 12 months Tabu Search results in less such volumes. Given that, in some cases, Tabu Search manages to fit more volumes while also resulting into a smaller loading, we conclude that it is able to release machines with lower cycle times than Simulated Annealing. We expect such results, given that we notice a similar behavior in our previous test. At that point, both algorithms manage to decrease the loading of the ICM to 0, but Tabu Search manages to release machines with lower cycle times and lead to a smaller loading of the existing machines.

Although Tabu Search leads to fewer units assigned on the ICM in 7 months, in the 5 months when Simulated Annealing outperforms Tabu Search, no units are left on the ICM. Hence, we conclude that the lower loading, for Tabu Search, we display in Figure G.5 in months 4, 8, 10, 11, and 12 is a consequence of having fewer units allocated on the existing machines. Given that the existing machines are not fully loaded in these months, we believe that running Tabu Search with an increased number of iterations can lead to results at least as good as the ones of Simulated Annealing.

Month	Volume IS (Existing Machines)	Volume SA (Existing Machines)	Volume TS (Existing Machines)	Volume IS (ICM)	Volume SA (ICM)	Volume TS (ICM)
1	1609058	1683015	1697138	416647	47449	33326
2	1525996	1533457	1537720	386487	154443	150180
3	1765700	1829126	1901469	455335	77611	5268
4	1601959	1740322	1738122	196402	0	2200
5	1755036	1786948	1788043	672077	324066	322971
6	1640436	1731249	1746432	335136	17383	2200
7	1634901	1649660	1650745	504479	213442	212357
8	1555313	1692045	1689845	222372	0	2200
9	1901405	1953525	1976250	536528	80171	57446
10	1881840	2009851	2007651	636842	0	2200
11	1862295	1999521	1997321	517887	0	2200
12	1645191	1780720	1778520	253831	0	2200

Figure G.6 Number of units (both existing machines and ICM) – Comparison between SA, TS and the initial solution – Increased demand

To summarize the results, in terms of loading, in Figure G.7 we show an overview with the average loading, average difference and average relative difference.

	Initial Schedule	Simulated Annealing	Tabu Search
Average Loading	112.373	106.816	103.540
Average Difference		5.556	8.833
Average Relative Difference		5.363	8.495

Figure G.7 Overview of the differences between SA and TS when compared with the initial schedule

In a similar manner as before, we show the running times of both metaheuristics in Figure G.8.

Month	Simulated Annealing	Tabu Search
1	206.266	88.891
2	552.297	189.25
3	348.75	104.078
4	4.281	90.766
5	1226.922	612.828
6	348.391	129.937
7	499.235	245.328
8	1.031	35.235
9	281.062	99.532
10	1.422	45.359
11	0.485	18.094
12	0.656	17.719
Max	1226.922	612.828
Average	289.233	139.751
Min	0.485	17.719

Figure G.8 Running times – Comparison between SA and TS

Appendix H:Local Search Parameters for Releasing a New Machine

In this section we show the results on which we base our choices for local search parameters. This section focuses on the Simulated Annealing and Tabu Search algorithms for releasing a new machine.

Appendix H.1: Simulated Annealing

First, we consider the case when 5 machines are initially available and we allow the algorithm to order a new one. We start by setting the Markov chain length to 50. Keeping this value constant, we vary the starting temperature. In Figure H.1 we display the acceptance probability values for different starting temperatures.

Starting Temperature	Accepted Transitions	Proposed Transitions	Ratio
1	48	50	0.96
5	48	50	0.96
10	48	50	0.96
20	48	50	0.96
30	48	50	0.96
40	48	50	0.96
50	48	50	0.96
60	48	50	0.96
80	48	50	0.96
100	48	50	0.96

Figure H.1 Acceptance probability for different starting temperatures – 5 machines

At this point, our algorithm selects products currently assigned on the ICM for which it should release the newly added machine. Hence, no matter what product the algorithm picks, most of the transitions will be accepted. As we see in Figure H.1, the acceptance probability is constant no matter what the initial temperature is. For our next steps we pick an initial temperature of 20. Moreover, we keep the final temperature to its previous value of 0.5.

In Figure H.2 we display the monthly overall loadings and the running times for different values of the cooling factor.

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Running Time (1 month) (s)	Runing Time (1 month) (min)	Running Time (12 months) (min)
20	0.5	0.1	50	124.361	90.857	13.203	0.220	2.641
20	0.5	0.2	50	124.361	90.857	13.907	0.232	2.781
20	0.5	0.3	50	124.361	90.857	18.531	0.309	3.706
20	0.5	0.4	50	124.361	90.857	16.468	0.274	3.294
20	0.5	0.5	50	124.361	90.857	15.406	0.257	3.081
20	0.5	0.6	50	124.361	90.857	14.421	0.240	2.884
20	0.5	0.7	50	124.361	90.857	16.766	0.279	3.353
20	0.5	0.8	50	124.361	90.857	18.625	0.310	3.725
20	0.5	0.9	50	124.361	90.857	19.156	0.319	3.831

Figure H.2 Overall loading and running times different cooling factors (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 50) – 5 machines

We notice that no matter the value of the cooling factor the monthly overall loading is the same. This being the case, we change the Markov chain length to check if we can obtain better results. Figures H.3 and H.4 show the results for a Markov chain length of 10 and 100.

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Running Time (1 month) (s)	Runing Time (1 month) (min)	Running Time (12 months) (min)
20	0.5	0.1	10	124.361	97.630	7.219	0.120	1.444
20	0.5	0.2	10	124.361	96.412	10.109	0.168	2.022
20	0.5	0.3	10	124.361	92.213	13.140	0.219	2.628
20	0.5	0.4	10	124.361	90.857	15.391	0.257	3.078
20	0.5	0.5	10	124.361	90.857	15.250	0.254	3.050
20	0.5	0.6	10	124.361	90.857	18.672	0.311	3.734
20	0.5	0.7	10	124.361	90.857	17.468	0.291	3.494
20	0.5	0.8	10	124.361	90.857	14.343	0.239	2.869
20	0.5	0.9	10	124.361	90.857	15.188	0.253	3.038

Figure H.3 Overall loading and running times different cooling factors (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 10) – 5 machines

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Running Time (1 month) (s)	Runing Time (1 month) (min)	Running Time (12 months) (min)
20	0.5	0.1	100	124.361	90.857	16.156	0.269	3.231
20	0.5	0.2	100	124.361	90.857	16.016	0.267	3.203
20	0.5	0.3	100	124.361	90.857	17.453	0.291	3.491
20	0.5	0.4	100	124.361	90.857	13.906	0.232	2.781
20	0.5	0.5	100	124.361	90.857	16.250	0.271	3.250
20	0.5	0.6	100	124.361	90.857	18.328	0.305	3.666
20	0.5	0.7	100	124.361	90.857	16.203	0.270	3.241
20	0.5	0.8	100	124.361	90.857	19.625	0.327	3.925
20	0.5	0.9	100	124.361	90.857	24.000	0.400	4.800

Figure H.4 Overall loading and running times different cooling factors (Starting Temperature 20, Stopping Temperature 0.5, Length of Markov Chain 100) – 5 machines

In Figure H.3 we see that even when using a Markov chain length of 10 we can obtain the same monthly overall loading for a cooling factor greater or equal to 0.4. Considering the Markov chain length of 100 (Figure H.4), we find the same monthly overall loading no matter the value of the cooling factor.

Next, we decrease the number of available machines to 4 and, just as before, we allow the algorithm to order a single machine. In Figure H.5 we compare the results when considering the loading of both existing machines and ICM, while in Figure H.6 we show the results related only to the existing machines.

Furthermore, In Figure H.7 we show the number of units allocated on both the existing machines and the ICM.

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Running Time (1 month) (s)	Running Time (1 month) (min)
10	0.5	0.1	50	222.471	133.979	56.085	0.935
10	0.5	0.2	50	222.471	133.172	63.694	1.062
10	0.5	0.3	50	222.471	133.554	92.563	1.543
10	0.5	0.4	50	222.471	132.901	141.984	2.366
10	0.5	0.5	50	222.471	132.901	173.887	2.898
10	0.5	0.6	50	222.471	132.901	224.381	3.740
10	0.5	0.7	50	222.471	132.901	383.050	6.384
10	0.5	0.8	50	222.471	132.901	541.800	9.030
10	0.5	0.9	50	222.471	132.901	1098.331	18.306

Figure H.5 Overall loading (both existing machines and ICM) and running times different cooling factors (Starting Temperature 10, Stopping Temperature 0.5, Length of Markov Chain 50) – 4 machines

T0	T1	Alpha	L	Loading Initial Schedule	Loading SA	Running Time (1 month) (s)	Running Time (1 month) (min)
10	0.5	0.1	50	100	100	56.085	0.935
10	0.5	0.2	50	100	100	63.694	1.062
10	0.5	0.3	50	100	100	92.563	1.543
10	0.5	0.4	50	100	100	141.984	2.366
10	0.5	0.5	50	100	100	173.887	2.898
10	0.5	0.6	50	100	100	224.381	3.740
10	0.5	0.7	50	100	100	383.050	6.384
10	0.5	0.8	50	100	100	541.800	9.030
10	0.5	0.9	50	100	100	1098.331	18.306

Figure H.6 Overall loading (existing machines) and running times different cooling factors (Starting Temperature 10, Stopping Temperature 0.5, Length of Markov Chain 50) – 4 machines

T0	T1	Alpha	L	Volume IS(Existing Machines)	Volume SA (Existing Machines)	Volume IS (ICM)	Volume SA (ICM)
10	0.5	0.1	50	741817	1013966.8	416647	144497.2
10	0.5	0.2	50	741817	1017399.6	416647	141064.4
10	0.5	0.3	50	741817	1015776	416647	142688
10	0.5	0.4	50	741817	1018551	416647	139913
10	0.5	0.5	50	741817	1018551	416647	139913
10	0.5	0.6	50	741817	1018551	416647	139913
10	0.5	0.7	50	741817	1018551	416647	139913
10	0.5	0.8	50	741817	1018551	416647	139913
10	0.5	0.9	50	741817	1018551	416647	139913

Figure H.7 Number of units (both existing machines and ICM) for different cooling factors (Starting Temperature 10, Stopping Temperature 0.5, Length of Markov Chain 50) – 4 machines

Appendix H.2: Tabu Search

For this algorithm we start by varying the number of iterations from 5 to 70. For this test we keep the tabu list length constant. We display the result in Figure H.8

Number Of Iterations	Tabu List Length	Loading Initial Schedule	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)	Running Time (12 months) (min)
5	5	124.361	97.820	126.969	2.116	25.394
10	5	124.361	95.050	199.5	3.325	39.900
20	5	124.361	92.364	317.046	5.284	63.409
30	5	124.361	91.423	417.86	6.964	83.572
50	5	124.361	90.857	475.266	7.921	95.053
70	5	124.361	90.857	451.75	7.529	90.350

Figure H.8 Overall loading and running times for different number of iterations (tabu list length 5) – 5 machines

We observe that when performing 50 iterations, or more, the algorithm finds the same monthly overall loading as the one from Simulated Annealing. Hence, we select 50 as the number of iterations. Next, we vary the tabu list length from 1 to the chosen number of iterations and show the results in Figure H.9. Just as in the case of releasing existing machines, we notice no effect of the tabu list length.

Iterations	Tabu List Length	Loading Initial Schedule	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)	Running Time (12 months) (min)
50	1	124.361	90.857	454.312	7.572	90.862
50	5	124.361	90.857	475.266	7.921	95.053
50	10	124.361	90.857	469.354	7.823	93.871
50	20	124.361	90.857	439.845	7.331	87.969
50	30	124.361	90.857	479.419	7.990	95.884
50	40	124.361	90.857	436.547	7.276	87.309
50	50	124.361	90.857	398.734	6.646	79.747

Figure H.9 Overall loading and running times for different lengths of the tabu list (50 iterations) – 5 machines

Next, considering the case when only 4 machines are available, we start by varying the number of iterations. This time, we keep the length of the tabu list constant to 30% of the number of iterations. In Figure H.10 shows the results when considering the loading of both existing machines and ICM, while in Figure H.11 we show the results related only to the existing machines. Furthermore, In Figure H.12 we show the number of units allocated on both the existing machines and the ICM.

Number Of Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule (Current)	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
10	3	30	222.422	132.901	417.078	6.951
30	9	30	222.422	132.901	1267.281	21.121
50	15	30	222.422	132.901	2000.047	33.334
70	21	30	222.422	132.901	2799.265	46.654
90	27	30	222.422	132.901	3805.203	63.420
100	30	30	222.422	132.901	4047.047	67.451

Figure H.10 Overall loading (both existing machines and ICM) and running times for different number of iterations (Tabu list length set to 30% of the number of iterations) – 4 machines

Number Of Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule (Current)	Loading Tabu Search	Running Time (1 month) (s)	Running Time (1 month) (min)
10	3	30	100.0	100.0	417.078	6.951
30	9	30	100.0	100.0	1267.281	21.121
50	15	30	100.0	100.0	2000.047	33.334
70	21	30	100.0	100.0	2799.265	46.654
90	27	30	100.0	100.0	3805.203	63.420
100	30	30	100.0	100.0	4047.047	67.451

Figure H.11 Overall loading (existing machines) and running times for different number of iterations (Tabu list length set to 30% of the number of iterations) – 4 machines

Number Of Iterations	Tabu List Length	Percentage from Iterations	Volume IS (Existing Machines)	Volume TS (Existing Machines)	Volume IS (ICM)	Volume TS (ICM)
10	3	30	741817	1018551	416647	139913
30	9	30	741817	1018551	416647	139913
50	15	30	741817	1018551	416647	139913
70	21	30	741817	1018551	416647	139913
90	27	30	741817	1018551	416647	139913
100	30	30	741817	1018551	416647	139913

Figure H.12 Number of units (both existing machines and ICM) for different number of iterations (Tabu list length set to 30% of the number of iterations) – 4 machines

From the previous figure we see that the results do not change when running more iterations. Hence, to test for different tabu list lengths we decide to run a total of 10 iterations. Figure H.13 shows the results when considering the loading of both existing machines and ICM, while in Figure H.14 we show the results related only to the existing machines. Furthermore, In Figure H.15 we show the number of units allocated on both the existing machines and the ICM.

Number of Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule (Current)	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
10	1	10	222.422	132.901	350.343	5.839
10	2	20	222.422	132.901	335.062	5.584
10	3	30	222.422	132.901	323.672	5.395
10	4	40	222.422	132.901	332.484	5.541
10	5	50	222.422	132.901	321.937	5.366
10	6	60	222.422	132.901	362.031	6.034
10	7	70	222.422	132.901	350.375	5.840
10	8	80	222.422	132.901	357.141	5.952
10	9	90	222.422	132.901	314.688	5.245
10	10	100	222.422	132.901	306.609	5.110

Figure H.13 Overall loading (both existing machines and ICM) and running times for different lengths of the tabu list (10 iterations) – 4 machines

Number of Iterations	Tabu List Length	Percentage from Iterations	Loading Initial Schedule (Current)	Loading TS	Running Time (1 month) (s)	Running Time (1 month) (min)
10	1	10	100.0	100.0	350.343	5.839
10	2	20	100.0	100.0	335.062	5.584
10	3	30	100.0	100.0	323.672	5.395
10	4	40	100.0	100.0	332.484	5.541
10	5	50	100.0	100.0	321.937	5.366
10	6	60	100.0	100.0	362.031	6.034
10	7	70	100.0	100.0	350.375	5.840
10	8	80	100.0	100.0	357.141	5.952
10	9	90	100.0	100.0	314.688	5.245
10	10	100	100.0	100.0	306.609	5.110

Figure H.14 Overall loading (existing machines) and running times for different lengths of the tabu list (10 iterations) – 4 machines

Number of Iterations	Tabu List Length	Percentage from Iterations	Volume IS (Existing Machines)	Volume TS(Existing Machines)	Volume IS (ICM)	Volume TS (ICM)
10	1	10	741817	1018551	416647	139913
10	2	20	741817	1018551	416647	139913
10	3	30	741817	1018551	416647	139913
10	4	40	741817	1018551	416647	139913
10	5	50	741817	1018551	416647	139913
10	6	60	741817	1018551	416647	139913
10	7	70	741817	1018551	416647	139913
10	8	80	741817	1018551	416647	139913
10	9	90	741817	1018551	416647	139913
10	10	100	741817	1018551	416647	139913

Figure H.15 Number of units (both existing machines and ICM) for different lengths of the tabu list (10 iterations) – 4 machines

Given the results, we can conclude that, for this test, 10 iterations are enough to fully load the existing machines, and that changing the length of the tabu list does not lead to different results.

Appendix I:Additional Results – Releasing a New Machine

In Figure I.1 we display the monthly loading of both the exiting machines and the ICM. For the initial schedule we calculate this loading by only considering 5 machines available, while for the results of SA and TS we consider 6. Next, in Figure I.2 we display the running times of both metaheuristics.

Month	Initial Schedule	Simulated Annealing	Difference (Init - SA)	Tabu Search	Difference (Init - TS)
1	124.361	90.857	33.504	90.857	33.504
2	128.166	89.386	38.779	89.386	38.779
3	121.741	93.095	28.646	93.095	28.646
4	89.003	89.003	0.000	89.003	0.000
5	165.520	120.511	45.008	114.991	50.528
6	97.076	97.076	0.000	97.076	0.000
7	154.802	102.703	52.099	102.703	52.099
8	96.232	96.232	0.000	96.232	0.000
9	168.723	119.260	49.462	117.811	50.912
10	178.212	132.932	45.280	121.903	56.308
11	153.906	103.338	50.568	102.629	51.277
12	111.504	86.636	24.868	86.636	24.868
Maximum Difference			52.099		56.308
Average Difference			30.684		32.243
Minimum Difference			0.000		0.000

Figure I.1 Monthly overall loading (both existing machines and ICM) – Comparison between SA, TS related to the initial solution – 5 machines

Month	Simulated Annealing	Tabu Search
1	15.188	402.313
2	4.875	86.547
3	10.937	276.156
4	-	-
5	43.625	1042.906
6	-	-
7	3.485	60.094
8	-	-
9	3.89	89.328
10	3.625	94.813
11	3.797	56.75
12	1.5	22.922
Max	43.625	1042.906
Average	10.102	236.870
Min	1.500	22.922

Figure I.2 Running times – Comparison between SA and TS – 5 machines

Next, we show the same results for the test in which we only have 4 machines available. Figure I.3 shows the results when considering the loading of both existing machines and ICM, while in Figure I.4 we show the results related only to the existing machines. Furthermore, In Figure I.5 we show the number of units allocated on both the existing machines and the ICM.

Month	Initial Schedule	Simulated Annealing	Difference (Init-SA)	Tabu Search	Difference(Init-TS)
0	222.471	132.901	89.570	132.901	89.570
1	222.345	142.636	79.708	142.636	79.708
2	220.891	144.440	76.451	144.440	76.451
3	153.882	90.527	63.355	107.359	46.523
4	290.741	195.625	95.116	195.625	95.116
5	195.114	121.402	73.712	127.238	67.876
6	248.289	168.372	79.917	168.372	79.917
7	161.007	94.648	66.359	113.884	47.123
8	252.271	168.300	83.972	164.065	88.206
9	287.196	192.643	94.553	192.643	94.553
10	242.081	160.750	81.331	160.750	81.331
11	169.638	105.509	64.129	118.323	51.316
Maximum Differece			95.116		95.116
Average Difference			79.014		74.808
Minimum Difference			63.355		46.523

Figure I.3 Monthly overall loading (both existing machines and ICM) – Comparison between SA, TS related to the initial solution – 4 machines

Month	Initial Schedule	Simulated Annealing	Tabu Search
0	100	100	100
1	100	100	100
2	100	100	100
3	100	90.527	90.199
4	100	100	100
5	100	100	96.109
6	100	100	100
7	100	94.648	89.948
8	100	100	100
9	100	100	100
10	100	100	100
11	100	100	91.457

Figure I.4 Monthly overall loading (existing machines) – Comparison between SA, TS and the initial solution – 4 machines

From Figure I.3 we notice that in 7 out of 12 months both algorithms reach the same results. However, in the other 5 months, Simulated Annealing outperforms Tabu Search in most of the cases. In these months we notice that the loading, considering both existing machines and the ICM, is lower for Simulated

Annealing, which means that it manages to reduce the loading of the ICM more than Tabu Search. As a consequence it also manages to load the existing machines more than TS. This is exactly what we observe in Figure I.4 where we see a higher loading for SA. Furthermore, the results we show in Figure I.5, the volumes assigned on both existing machines and ICM on a monthly basis, show that Simulated Annealing manages to fit more units within existing machines. Despite these results, we believe that Tabu Search is limited by the number of iterations we use and that increasing this value, when needed, has the potential to lead to results at least as good as the ones of Simulated Annealing.

Month	Volume IS (Existing Machines)	Volume SA (Existing Machines)	Volume TS (Existing Machines)	Volume IS (ICM)	Volume SA (ICM)	Volume TS (ICM)
0	741817	1018551	1018551	416647	139913	139913
1	729413	947540	947540	386487	168360	168360
2	879402	1125507	1125507	455335	209230	209230
3	971920	1168322	1090134	196402	0	78188
4	866937	1117845	1117845	672077	421169	421169
5	841496	1082367	1039527	335136	94265	137105
6	786623	1000350	1000350	504479	290752	290752
7	897673	1120045	1010985	222372	0	109060
8	925168	1160878	1179529	536528	300818	282167
9	801009	1043886	1043886	636842	393965	393965
10	909634	1150728	1150728	517887	276793	276793
11	954889	1183619	1086316	253831	25101	122404

Figure I.5 Number of units (both existing machines and ICM) – Comparison between SA, TS and the initial solution – 4 machines

To summarize the results, in terms of loading, in Figure I.6 we show an overview with the average loading, average difference and average relative difference.

	Initial Schedule	Simulated Annealing	Tabu Search
Average Loading	222.161	143.146	147.353
Average Difference		79.014	74.808
Average Relativ Difference		36.116	33.470

Figure I.6 Overview of the differences between SA and TS when compared with the initial schedule

In Figure I.7 we provide an overview of the running times incurred by each algorithm.

Month	Simulated Annealing	Tabu Search
0	114.547	298.203
1	55.141	133.781
2	64.859	235.172
3	7.875	27.859
4	110.765	481.969
5	91.828	407.000
6	75.344	186.875
7	14.516	8.407
8	4.282	105.859
9	170.031	555.718
10	13.969	17.235
11	69.437	299.016
Max	170.031	555.718
Average	74.048	254.832
Min	4.282	8.407

Figure I.7 Running times – Comparison between SA and TS

Appendix J: Flowchart – Entire Heuristic

Extensions multiple facilities by just setting machine releases to the good values.

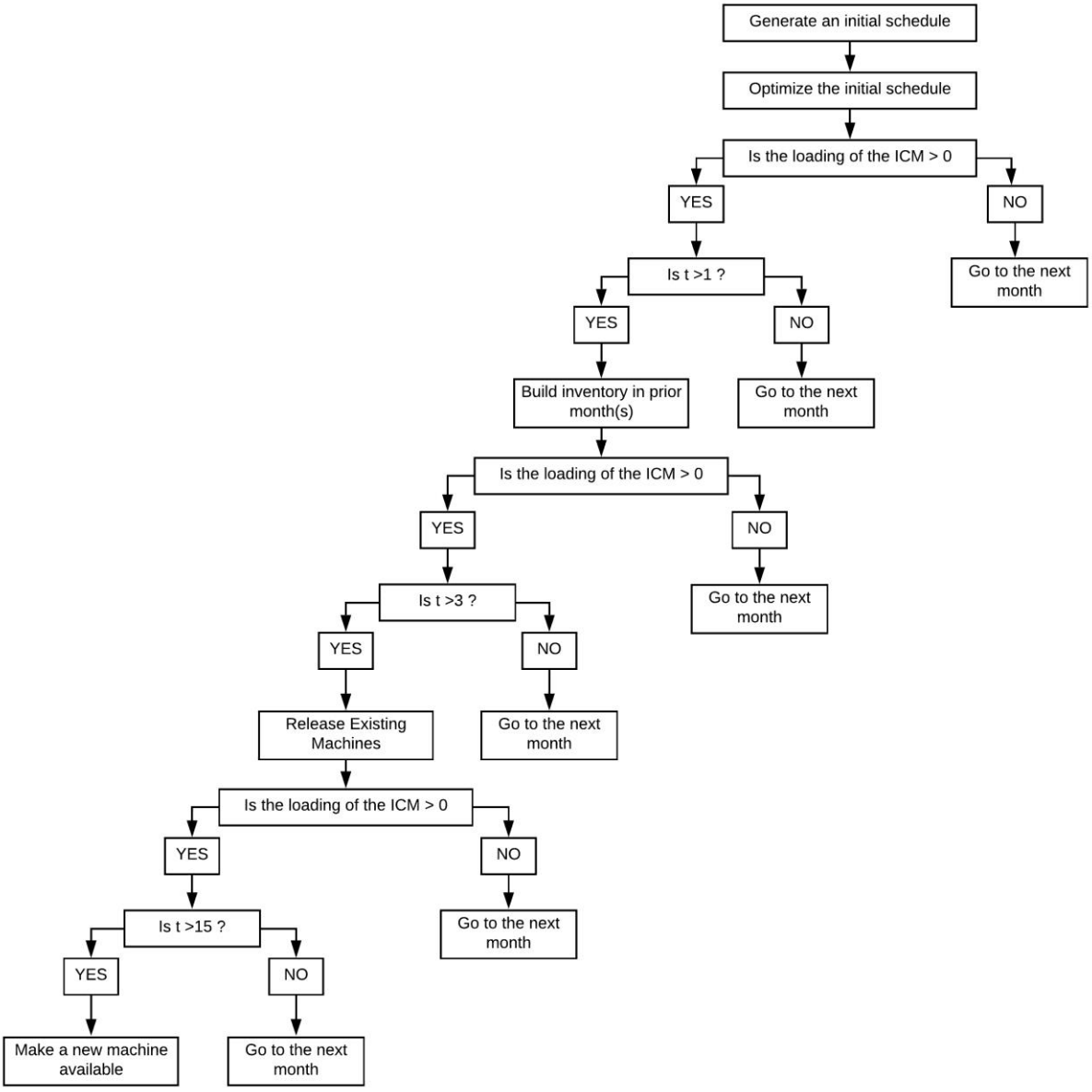

Figure J.1 Flowchart showing the steps of the entire heuristic