TÉCNICO LISBOA

A Hybrid Heuristic for a Flexible Job Shop Problem in Logoplaste Santa Iria

João Pedro Ferreira Pedrosa

Thesis to obtain the Master of Science Degree in

Industrial Engineering and Management

Supervisors: Prof. Tânia Rute Xavier de Matos Pinto Varela

Prof. Nelson Fernando Chibeles Pereira Martins

Examination Committee

Chairperson: Prof. Mónica Duarte Correia de Oliveira

Supervisor: Prof. Nelson Fernando Chibeles Pereira Martins

Member of the Committee: Prof. Graça Maria Marques da Silva Gonçalves

December 2019

Resumo

A presente dissertação foi desenvolvida no âmbito do Mestrado em Engenharia e Gestão Industrial, no Instituto Superior Técnico. O caso de estudo explora uma multinacional portuguesa no segmento da produção de plásticos, nomeadamente o Grupo Logoplaste, que é neste momento o líder do Mercado português.

Com a crescente pressão dos mercados e reduzidas margens de lucro, as empresas veem-se obrigadas a lidar com vários desafios para continuarem competitivas, o que desencadeia o desenvolvimento e implementação de abordagens inovadoras, para reagir com informação mais rápida e precisa. Logoplaste Santa Iria (LSI) é uma fábrica subsidiária do Grupo Logoplaste que produz embalagens para a FIMA e Lactogal, produtores portugueses de cremes de barrar e laticínios, respetivamente. Atualmente, o planeamento de produção na LSI é feito manualmente por um grupo de trabalhadores e posteriormente validado pelo gestor de fábrica sem qualquer suporte tecnológico. Um software na fábrica que ajudasse no planeamento de produção pode gerar grandes melhorias a nível de lucro total e despesas.

O objetivo desta dissertação é desenvolver um algoritmo que ajude a LSI a melhorar o planeamento de produção e escalonamento. Para esse efeito, um algoritmo mono-objetivo para minimizar a *tardiness* e um algoritmo bi-objetivo para minimizar a *tardiness* e o *makespan* são desenvolvidos para resolver um *flexible job shop problem*. Os algoritmos consistem numa metodologia proposta que incorpora um algoritmo genético com uma componente de pesquisa local para resolver o problema de acordo com duas estratégias de produção diferentes, *make-to-order* e *make-to-stock*.

Palavras-Chave: Flexible Job Shop, Logoplaste Santa Iria, Production Scheduling, Metaheuristics, Genetic Algorithm, Hybridization

Abstract

The present master thesis was developed within the Industrial Engineering and Management program at the Instituto Superior Técnico, Universidade de Lisboa. The case study explores a Portuguese Multinational in the plastics production segment, namely the Logoplaste Group, which is the leader in the Portuguese market.

With the increasing pressure of the markets and low profit margins, companies must deal with several challenges to remain competitive triggering the development and implementation of innovative approaches to react with accurate information and faster. Logoplaste Santa Iria is a Logoplaste Group subsidiary factory that produces containers for FIMA and for Lactogal (LSI), Portuguese producers of spreads and dairy products, respectively. Currently the production planning of LSI is done manually by a group of employees and then validated by the factory manager without the support of any IT tool. An IT software in the factory to help the production planning can have major improvements in terms of total profit and expenses.

The goal of this master thesis is to develop an algorithm that will help LSI to improve its production scheduling stage. To this end, a mono-objective algorithm minimizing the tardiness and a bi-objective algorithm minimizing both tardiness and makespan are developed for a flexible job shop scheduling problem. The algorithms consist on a proposed methodology that comprises a genetic algorithm with a local search feature to solve the problem under two different production strategies, make-to-order and make-to-stock.

Key-words: Flexible Job Shop, Logoplaste Santa Iria, Production Scheduling, Metaheuristics, Genetic Algorithm, Hybridization

Acknowledgements

I would like to thank Prof. Tânia Pinto Varela and Prof. Nelson Chibelles for their guidance during the development of this work. I am grateful for their availability and valuable support.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

BO_BGA - Bi-Objective Genetic Algorithm

C1 - Machine gene-string

C2 - Operation gene-string

CO - Crossover

CWSA - Classic Weighted Method

EA - Evolutionary approach

FJSSP - Flexible Job Shop Scheduling Problem

FSA - Fuzzy Simulated Annealing

G - Generations Number

GA - Genetic Algorithm

HC - Hill Climbing

JA - Johnson's algorithm

JSSP - Job Shop Scheduling Problem

L - Lower bound

LS - Local Search

LSI - Logoplaste Santa Iria

M - Machines Number

MO_BGA - Mono-Objective Genetic Algorithm

MR - Mutation Rate

MTO - Make-To-Order

MTS - Make-To-Stock

N - Population Size

NWSA - Normalized Weighted Method

O - Orders Number

OF - Objective Function

OF1 - Tardiness Objective Function

OF2 - Makespan Objective Function

P - Products Number

PSO - Particle Swarm Optimization

SA - Simulated Annealing

TS - Tabu search

VBA - Visual Basic for Applications

$\mathbf{W_m}$ - Weight for Makespan

$\mathbf{W_t}$ - Weight for Tardiness

1. Introduction

This chapter aims to provide an outline of the document, focusing on the background of the problem under analysis, and including the scope definition and master's thesis objectives. It is divided into four parts for easier understanding. The first part presents the background and context of the problem. The second part details the methodology to follow in this master thesis. The third part of this chapter introduces the objectives that are to be attained by the end of the thesis. The last part shows the structure to be followed by the document.

1.1 Problem Contextualization

Recently, new companies have been trying to penetrate the markets and with this competitiveness growth, new approaches need to be taken so that the companies can remain profitable. One example of that strategy is already used by some companies through the vertical integration which consists of turning into the company's internal processes tasks that depend from third parties, such as: packaging, obtaining raw materials, logistics, etc. It can be especially helpful in unstable markets where the prices are very volatile.

Following this market development, technology has also been upgrading a lot. Currently it is possible to use IT solutions to enhance company's efficiency by tracking operations in the factories, improving product's quality control and scheduling operations, which triggers the investment in their information technologies. Companies are using software to manage and plan every operation, not only in the factory but also all pre and post parts involving the collection of raw materials and the delivery of final products. This way, software models integrate the stages of production, warehouse and distribution to get better results and undertake more complexity.

In particular, Logoplaste is part of the plastic packaging industry and the market currently very saturated. Due to high competitiveness, the final prices that are applied to the consumer are the key factor for measuring companies' success. The market is also defined by having low unitary margins so the total volume that a company sells will dictate the results. Because of this, Logoplaste is interested in being informed about all new technological advances in order to improve its potential.

Logoplaste has been trying to implement new software tools in factories to have a better control and planning methodologies. The most recent example of that is the implementation of the SAP management software that the company is using currently but it still doesn't support production planning at some factories.

The goal of this master thesis is to develop an algorithm that will help the company to improve its production scheduling stage. Two production strategies will be performed, one considering the inventory (make-to-stock) and one disregarding the inventory (make-to-order). This algorithm will take into account both theoretical concepts and actual practices in the factory. This way, it is guaranteed that the mechanism will include an optimization model for the scheduling of production tasks based on the company's *modus*

operandi and portfolio of orders. This thesis uses Logoplaste Santa Iria factory (LSI) as case study, which is located inside Unilever's headquarters at Póvoa de Santa Iria, Portugal.

1.2 Methodology

The methodology adopted in this master thesis is presented in Figure 1.

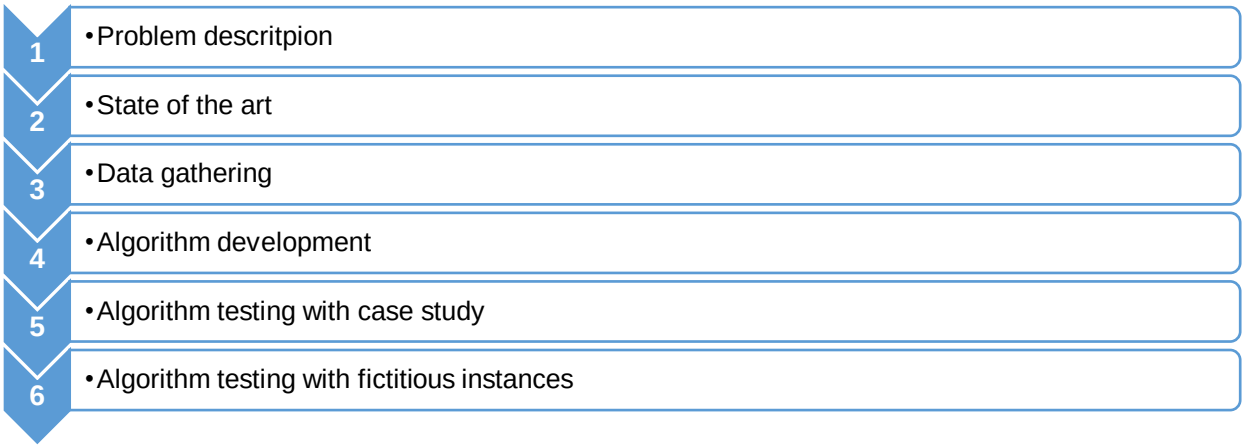

Figure 1: *Master dissertation methodology.*

First stage - Problem description

This stage describes the problem. The core production process (plastic injection method) and the production planning at LSI are detailed.

Second stage - State of the art

In this stage a state of the art review on scheduling concepts related with the information highlighted in the first stage is done. It starts with a description of a job shop problem and the ideas behind exact and approximate algorithms followed by meta-heuristic approaches (simulated annealing, tabu search, genetic algorithm and particle swarm optimization) for scheduling that can be adapted to the problem under consideration. Finally, the hybridization concept is also reviewed.

Third stage - Data gathering

In the third stage, the processes of the factory are documented in order to better design and validate the algorithm that will be developed. The processes will be used to generate the algorithm inputs.

Fourth stage - Algorithm development

The algorithm will be developed using Excel's Visual Basic tool and considers as input the factory data which includes the machines info, product details, orders data and for some cases the stock levels.

Fifth stage - Algorithm testing with case study

This stage consists on doing several analysis in order to validate the algorithm results using the case study input and a tuning process is done to select the parameters' values showing better results.

Sixth stage - Algorithm testing with fictitious instances

The sixth stage tests the algorithm with fictitious instances in order to analyze how it reacts under situations with different complexities.

1.3 Master Thesis Objectives

The main objectives of the present Master Thesis are:

- Characterization of the problem.
- Describe Logoplaste group, focusing on LSI's manufacture processes and planning.
- Explain what is motivating LSI do develop this work.
- State of the art review of methodologies and models for scheduling problems, exploring both trajectory and population families based heuristics.
- Define Make-To-Order and Make-To-Stock polices and review mono and bi-objective approaches to solve them.
- Identify an approach that has not been used yet in literature to solve the current problem. This gap will add value to the scientific community.
- Select the best way to create a tool to make the factory's scheduling.
- Develop an algorithm to assist the decision making in the factory regarding the production scheduling.
- Explore the algorithm in two different production strategies (make-to-stock and make-to-order) as well as mono-objective and bi-objective approaches.
- Test the algorithm with the case study real data and with fictitious situations.

1.4 Master Thesis Structure

The present document structure is shown in Figure 2 and is detailed below.

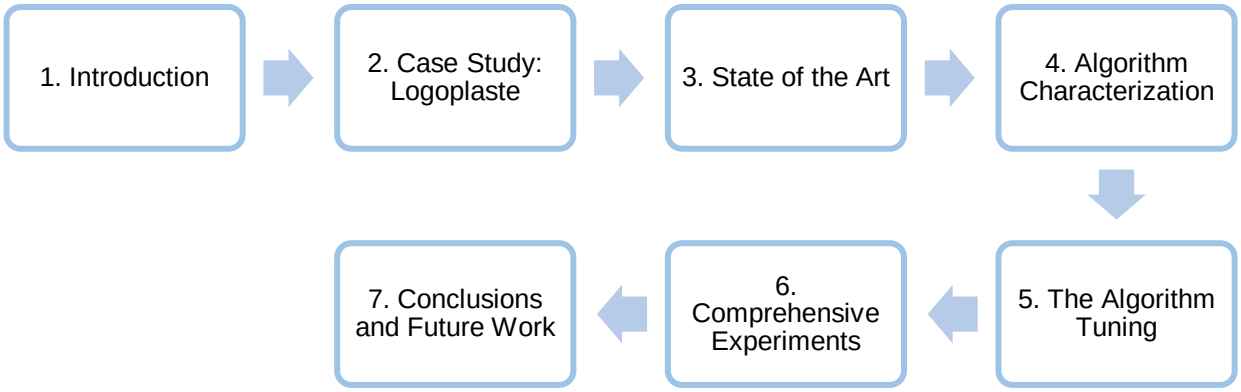

Figure 2: *Master thesis structure*

1. **Introduction** is the present chapter, where an introduction to the dissertation theme was performed, as well as the problem context and motivation. The study objectives and respective document structure are also described.
2. **Case Study: Logoplaste** presents a description of the company and how it functions. All the processes are detailed, referring all productive stages, storage and planning. Finally, an analysis on the actual procedures is carried out.
3. **State of the Art** reviews scheduling concepts related to this thesis. It will focus on several ways to solve production scheduling problems.
4. **Algorithm Characterization** chapter characterizes all procedures and principles of the developed algorithms. The production strategies make-to-order and make-to-stock are also presented in this chapter.
5. **The Algorithm Tuning** tunes several parameters in order to maximize performance and achieve desired results. The input data used in the fifth chapter is the same of the case study to make sure that the algorithm is adapted to solve a real factory scenario.
6. **Comprehensive Experiments** chapter uses fictitious situations to test the algorithms and analyze its results. Instances with several complexities are created to investigate until which point the algorithm performs consistently.
7. **Conclusions and Future Work** is the seventh and last chapter where conclusions are presented, and future work is outlined.

2. Case Study

This chapter aims to present the company and provide the contextualization and description of the problem under study, with greater detail.

Firstly, Logoplaste Group will be presented, with a briefly description of its structure, companies in the group and products. Secondly, LSI is explored in terms of production processes and everything necessary to create the final products. Thirdly, the current production scheduling at LSI is addressed. Finally, a resume of the problem to be explored will be provided, as well as some conclusions.

2.1 Introduction

Logoplaste was created in 1976 and currently is the leader of the plastics Portuguese market. Its work focuses on producing plastic bottles and containers using different types of raw materials such as HDPE (high-density polyethylene), PP (polypropylene) and PET (polyethylene terephthalate). Since the very beginning Logoplaste adopted an international expansion strategy due to the fast growing market. Nowadays, it has an important role in countries like UK, Brazil, France, Spain with a total of more than 60 factories worldwide.

The production of PET (using pre-forms) plastic bottles was the business unit with the highest impact in Logoplaste's profit. This is the reason why directors decided to implement a new factory in 1997 just to meet their PET pre-forms needs. Currently this factory produces more than one billion units, not only in Logoplaste's but also directly in client's facilities.

Logoplaste Group consists on a group of different subsidiaries that are constantly in contact with Logoplaste headquarters in Cascais, Portugal. This is the core of all business because both Logomolde and Innovation Lab are located there. Logomolde is responsible for producing, testing and maintaining factories' molds whereas Innovation Lab is capable of designing and developing prototypes that follow customers' requisites.

In this industry, production can be categorized based on the client distance: in-house, nearby and at different geographical locations. In in-house category the producer sets all his tools and equipment up in client's factory and the production line is directly connected to the customer's output line. The nearby production consists on having the supplier's factory close to the client's premises and everything is produced in the factory and then moved to the final destination. Finally, if the entities are located in different geographical locations the process is quite similar to the nearby category but it results in higher transportation costs.

2.2 Production Process

LSI is a factory that produces PP plastic containers for consumable products. Currently it operates for FIMA which is part of the Unilever Jerónimo Martins Group.

The production process is characterized by using an injection method. Every plastic product produced is composed by a bottom part and a cover one. Both of them are made of PP and produced at LSI.

The process requires three different inputs: PP, dye and printed label. All raw materials are kept in the factory's warehouse and the labels are maintained under special climate conditions in order not to suffer any damages.

The PP and the dye are mixed in a helical shaft and then the electrical resistors provide energy so that all the mixture can melt. The quantity of raw materials is fed in a continuously and smoothly way to produce an evenly distributed mixture for the desired color. When the liquid achieves the right viscosity, it can be uniformly putted in the mold's head, as presented in Figure 3.

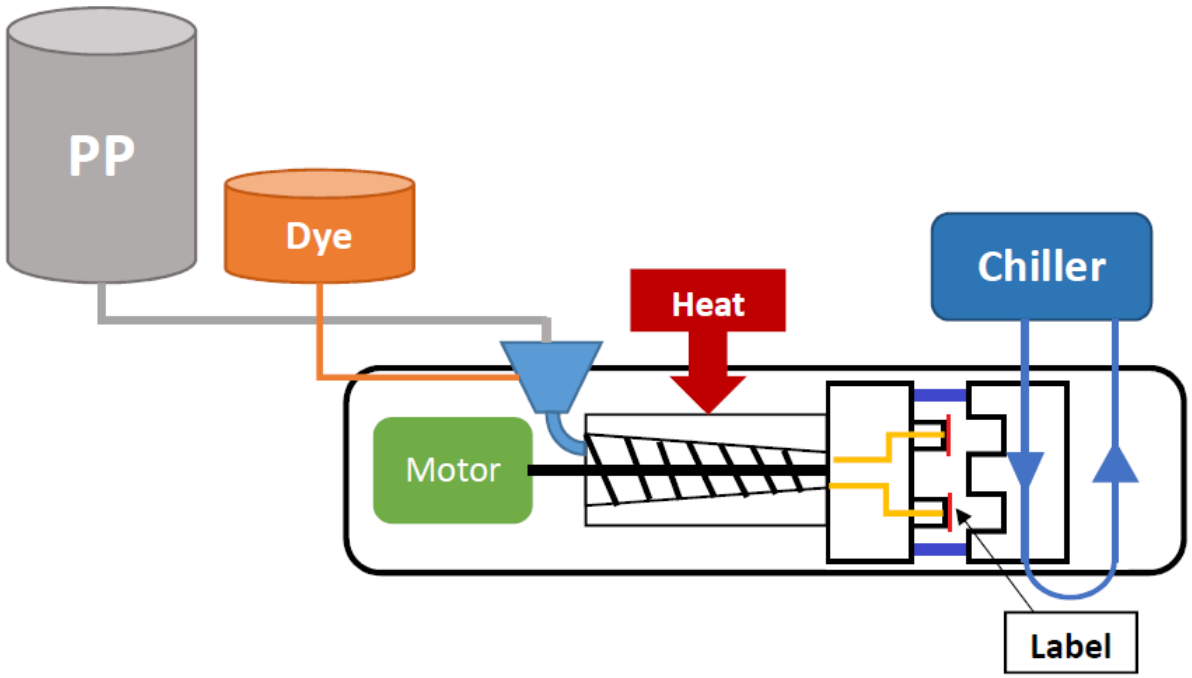

Figure 3: *Injection machine operation diagram*

(Marques et al. 2015)

During the process, both halves of the mould (the head and the cavity) work together joining and separating in order to produce the same number of bottoms or covers as cavities. It is possible to choose molds with one, two or four cavities. Whenever the mold closes, the mixture is introduced in the mold through its head and at the same time, cold water is surrounding the outside part of it. Thanks to that, it is possible to prevent the mold from damages due to the air removal process that is executed by the factory workers.

The containers' labels are placed before the PP injection instead of putting them after it. Air suckers have the responsibility to put the labels in the mold's head and then static electricity fixes them. Using this procedure, the label fuses with the PP absorbing it in the cover/bottom. When final products are ready, they are placed in piles and then on pallets. Each pallet has only products from the exact same type/reference. The number of products in each pallet depends on the size of the goods. A pallet of bottoms will always have a fewer number of products than a pallet of covers because bottoms are bigger. The pallets are then stored in a warehouse and moved to the client when possible.

Considering that both warehouses and factories deal with food products, hygiene and safety are significantly important. There are several guidelines, rules and standards that must be achieved so that everything goes well. Because of this, LSI keeps doing bacterial tests on its products.

The rule behind the warehouse operation is FIFO - first in first out. It means that the first product produced it the first one going to the client.

In order to prevent breakdowns and verify all the requirements, the molds are inspected often to make sure everything goes along with the specifications. In order to improve LSI factory's efficiency and productivity three units of each mold are utilized: one of them is always operating, the second one is under maintenance and the last one is putted aside for prevention.

In LSI a total number of 30 products that are produced for FIMA in which 15 are bottoms and 15 are covers (Table 1). Every product is distinguished by its mold type, dye color and label. Although all products have different labels, they may have same mold type and color.

Table 1: *List of all the product references produced at LSI*

TAMPOS IMPRESSOS
PLANTA 16x500G CV
PLANTA 30x250G CV
PLANTA 3X(10X250GRS) _ 55% EXPOR
PLANTA SABOR A MANTEIGA 10X250GRS
Tulicreme Avelã 16x200g
TULICREME CHOCOLATE 16x200
BECEL GOLD 3X(10X250G)
PLANTA SOJA 10X250G
BECEL Magra 3X(10X250G)
FLORA LIGHT 10X250G-SPAIN
FLORA Gourmet Spain
BECEL 3X(10X250G)
PLANTA 3X(10X250G)
FLORA 3X(10X250G)
PLANTA CÁLCIO 3X(10X250GRS)

FUNDOS IMPRESSOS
PLANTA 16x500G CV
PLANTA 30x250G CV
PLANTA 3X(10X250GRS) _ 55% EXPOR
PLANTA SABOR A MANTEIGA 10X250GRS
Tulicreme Avelã 16x200g
TULICREME CHOCOLATE 16x200
BECEL GOLD 3X(10X250G)
PLANTA SOJA 10X250G
BECEL Magra 3X(10X250G)
FLORA LIGHT 10X250G-SPAIN
FLORA Gourmet Spain
BECEL 3X(10X250G)
PLANTA 3X(10X250G)
FLORA 3X(10X250G)
PLANTA CÁLCIO 3X(10X250GRS)

The total productive force dedicated to FIMA is only 8 machines that operate with 11 different types of molds. The covers can only be produced by the 5 machines while the bottoms can only be produced by 3 machines. No machines are prepared to produce bottoms and covers.

Regarding products' production, there are two important variables to take into consideration: processing times and changeover times. Processing time consist in the necessary time to produce a certain product in a certain machine. Changeover time refers to the time needed to prepare a certain machine in order to produce a different product. It not only depends in the machine itself but also in products that make part of the transition. The three types of changeovers are: molds, dye and label change (the total operation time decreases in this order). Mold change happens when products have different shapes, taking up to several hours. Dye swap occurs when products have different colors and the same shape, taking 100 minutes. Label change refers to the change when products have the same shape and color but different label, requiring a few minutes. As a final remark, all changeover times depend on machine's complexities and sizes.

2.3 Production scheduling at LSI

LSI's production planning encompasses three different impact levels which vary in terms of time horizon and areas they deal with: strategical, tactical and operational. Strategic level is associated with long term decisions, tactical level addresses medium term decisions and operational level deals with short term decisions. Figure 4 shows the decisional sub-system decomposed into these three levels: strategic, tactical and operational.

The production scheduling is assigned to a group of employees of the factory and also to the factory manager. The information needed are stock levels at LSI and FIMA, a weekly delivery plan, demand forecasts, warehouse capacity limit and raw material availability (labels, dye and PP). Consequently, the planning process has to integrate three main stages: raw material availability at the warehouse, manufacture stage and final product storage at the warehouse.

Regarding the tactic level of LSI, in the beginning of the year the factory receives the annual demand plan from FIMA. It gives forecasts in terms of initial yearly production and it helps the manager to define annual raw material needs, annual sales and capacity needs.

The operational level involves weekly decisions. In this part it is decided the quantities of each product that will be produced. Basically, in this part we assign machines to resources and it's decided in which machines will be produced each reference.

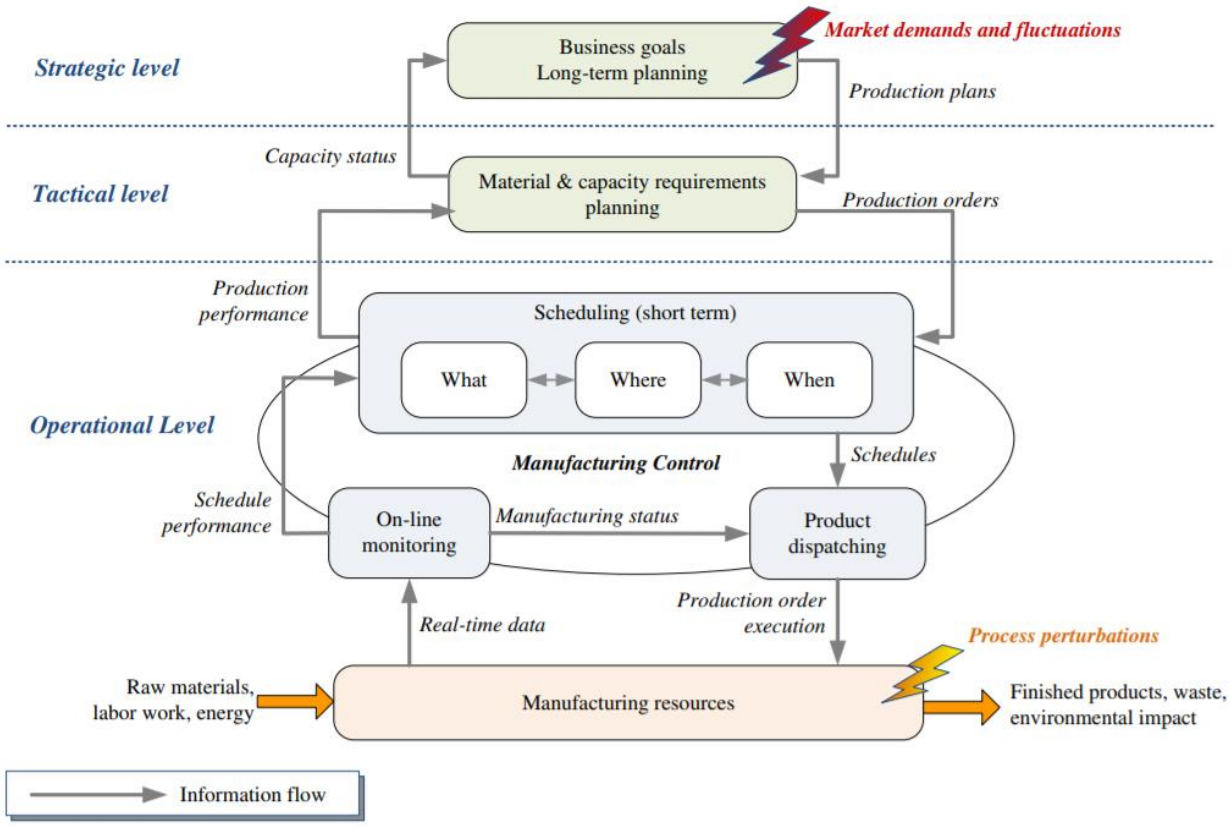

Figure 4: *Typical decisional levels in manufacturing systems*

(Rey 2014)

Before getting in detail about how LSI production planning is done, it is necessary to explain how FIMA operates. FIMA deals with products with a short expiry date and LSI is located very close to FIMA, they tend to keep low stock levels of finished products. Due to this, production in FIMA relies much more in its real demand than on forecasting. FIMA postpones production closer to deadlines in order to avoid the products do get damaged. This kind of approach can be very risk, if the orders' forecast has a last minute a change in FIMA, LSI needs to adapt quickly, so that everything goes smoothly and without production breaks.

Because of the risk described in the previous paragraph the company tried to mitigate that by keeping a safety stock in both FIMA and LSI. Each reference will have a certain safety stock assigned and it is added in the weekly delivery plan. In the annual plan, for each reference it is possible to see how many weeks of demand the safety stock should fulfil.

Whenever there are changes in products' labels (due to changing of labels descriptions or aesthetic), all finished products with old labels are discarded. In these circumstances, safety stock shelters LSI from any losses.

The main aspects behind production planning, starting by annual and finishing in weekly planning:

1. Every week, LSI receives three files: annual demand forecast and actual stock levels, raw weekly plan, updated weekly plan.
2. The annual demand forecast that was received in the previous step will be updated every week;
3. FIMA sends to LSI the weekly delivery plan for the following week;
4. LSI does the production planning for each week considering all the files and data from FIMA;
5. The production planning incorporates the schedule for every machine in the factory;
6. The stock levels are defined for each product reference;
7. LSI is only responsible for its stock;
8. Bearing in mind that both FIMA and LSI are closely located, the lead times are neglected.
9. They just consider relevant the following aspects: processing times, setup times and stock levels.

The steps of making a production scheduling plan (during the same week) are:

1. Day 1: Raw weekly delivery plan for week 1 is received;
2. Day 2: Updated annual demand forecast is received with actual stock levels at FIMA and LSI;
3. Day 2: SAP annual plan (from week 1) is made using the weekly delivery plan, the updated annual forecast and the stock levels;
4. Day 2: Production plan from week 0 to week n (depending on when the products are needed and the deadlines) is created.
5. Day 5: Updated weekly delivery plan for week 1 is received.

Despite the SAP system having control of the raw material stock levels, all the stages described before are executed manually without any IT software, specially the production scheduling. This is the main gap in the factory. If there was a software capable of getting inputs like orders, stock levels, machines setup times, etc. and come up with an optimum schedule for each machine, the company would be able to reduce production setup times and maximizing production. This kind of decisions are of extreme importance and complexity which reinforces the use of a software.

2.4 Chapter Conclusions

Logoplaste Santa Iria involves a simple production process but a very complex scheduling planning. In this chapter it was possible to see that the production planning is not supported by any IT software. It is just performed by a team of workers with no support tools and confirmed by the factory manager. This team takes an overview of all orders and guesses which schedule will be ideal. However, this problem has a huge complexity and even if the manager proposes an orders' line-up that has proved to be satisfactory in the end it does not mean that there are not a million better options.

An IT software in the factory to help the production planning can have major consequences in terms of total profit and expenses. Instead of relying just in the factory manager and in his non consistent skills, there is the need of getting always a near optimal solution.

In order to fill this gap, in this thesis it will develop a tool to do a production scheduling considering all factors inherent to the factory. Among these factors there are: product specifications by reference, machine specifications, weekly demand and stock levels. The goal is to get an algorithm that integrates all these factors so that it can be used every week with several different scenarios. The work that will be developed in this thesis will only influence the weekly production of the factory so it will focus on operational decisions.

The next chapter will review existing methodologies to solve this type of scheduling problems and then select the most convenient one for this case.

3. State of the Art

The literature review has the aim to explain all the fundaments and theory base for the development of the master thesis. In order to cover all the important knowledge regarding the problem, the structure of this chapter will be as follow: overview of job shop scheduling problem and its categories, analyze state-of-the-art scheduling algorithms as well as their advantages and flaws and finally, come up with the most promise algorithm to solve the problem.

Despite is available in the literature a lot of articles regarding solving techniques for scheduling problems, most of them just are applied over a specific case, which makes very difficult the approaches´ comparison because every algorithm has its own variations and the cases never have the same characteristics. That is why it is important to cover the widespread characteristics of all techniques and not only examples of their actual application.

3.1 Job Shop Problems and Algorithms

A combinatorial optimization problem is represented by two main characteristics (Widmer et al. 2010)**:**

- One or more objective functions, which are minimized or maximized (the solution must belong to a specific solution space).
- A set of constraints by which the solution space is defined.

Inside the combinatorial optimization problems, it is possible to find the Job Shop Scheduling Problem (JSSP), which is a NP-hard problem (Lageweg et al. 1977) . It means that there is no efficient algorithm to solve it and that is why so much work has been done recently in order to improve methodologies and ways to solve it.

JSSP can be described (Arisha et al. 2001) as the following:

- To create a single product in a factory, it is necessary to execute several operations.
- A job is a set of operations required to produce a product. In the case study's context, a job corresponds to an order.
- An operation can only be executed by a certain machine from the set of all machines in the factory.
- The duration of a task/operation depends on the machine that is executing it.
- The JSSP consists on finding the operation sequence for each machine so that it is possible to minimize one or more objective functions.

Figure 5 shows one possible way to represent a JSSP solution using a Gantt Chart in which the *x* axis represents the execution time and the *y* axis the different machines. For the job 1 (j_1) to be concluded, an operation in machine 1 (m_1) followed by an operation in machine 2 (m_2) and an operation in machine 3 (m_3)

must be performed. Regarding the second job (j_2), it requires an operation in m_2 followed by an operation in m_3. Finally, job 3 (j_3) consists in only one operation in m_1.

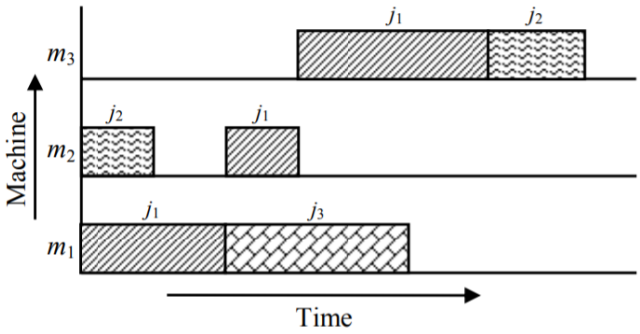

Figure 5: *Partial Gantt chart*

(Hasan et al. 2007)

One important part of the JSSP is defining the constraints and the most used ones are precedence, capacity and due date (Abdolrazzagh-Nezhad & Abdullah 2017) . Precedence constraints refers that for a job to be well executed it must flow in a specific order (e.g. before a product goes to machine B it needs to go first to machine A). Capacity constraints says if a machine can execute different tasks at the same time. Finally, release date, as it suggests, mentions what is the deadline of the job execution.

Another part of the JSSP is the objective functions that can be minimizing the makespan or minimizing the tardiness, for instance. Makespan represents the total duration between the first job and the last one. Tardiness is the amount of time between the job's deadline and its completion.

Inside the JSSP group of problems it is possible to find different classes: deterministic JSSP, stochastic JSSP, fuzzy JSSP and flexible JSSP (Pinedo 2016). The main difference between JSSP and the flexible JSSP (FJSSP) is that JSSP assumes that only one machine is able to run a particular operation. The FJSSP, on the other hand, assumes a more flexible set up in which an operation can be executed by a set of different machines. The case study of this dissertation fits on a FJSSP because from the 8 available machines in the factory, covers can by produced by any of the 5 first machines and bottoms can be produced by any of the 3 last machines.

Literature refers that there are two means to solve a scheduling problem: using exact and approximate algorithms (Y. B. Yang 1977).

Exact algorithms guarantee that they will find the best or optimal solution for the problem. An exact solution is obtained if all possible combinations of assignments are determined. Approximate algorithms use intelligent ways to avoid exploring all the solution space and consequently get solutions faster by accepting

worse solution sometimes. The solution of an approximate algorithm may not correspond to the global optimum but it guarantees that is reasonably good solution. The next section details these two methods.

3.1.1 Exact vs Approximate

Johnson's algorithm (JA) is probably the most well-known exact algorithm in the scheduling field and was developed in 1954. In 1976 was discovered a way to solve three-machine problems based in JA idea (Burns & Rooker 1976). Exact algorithms also include branch and bound techniques which consists on using dynamically constructed tree structures as a way of representing all feasible solution space. (Hariri & Pons 1991) proposed a branch and bound technique to solve the general job shop problem minimizing the makespan. Although the good results shown, there was still the possibility of improving and finding a new approach to obtain tighter lower bounds.

When the problem is relatively small, it is reasonable to use exact algorithms. However, as the problem gets more and more complex, it may take a huge amount of time to complete the algorithm. For the general job shop problem there are $(n!)^m$ possible solutions in which n and m refer to the number of jobs and machines, respectively (Jain et al. 1998). In a problem with 10 jobs and 5 machines, there will be 6.2923×10^{183} possible solutions. Even though many of these possibilities will not be feasible, a complete identification of all feasible combinations is almost impractical.

Because of exact methods' lack of performance, approximate methods started to arise among scientific community and have gained extreme popularity due to the fact today it is indispensable to find efficient solutions for scheduling problems. The importance of approximate methods was enhanced by (Glover & Greenberg 1989) as they explained that directed tree search methods for combinatorial difficult problems were not wholly satisfactory. They also referred that a metaheuristic is the best way to improve schedule optimization algorithms because they stablish an appropriate bilateral linkage between operations research and artificial intelligence.

Metaheuristics are generic strategies that define algorithmic procedures to efficiently find solutions that satisfy combinatorial optimization problem (Sörensen et al. 2017). The main difference between heuristics and metaheuristics is that metaheuristics are approximation approach problem-independent strategies and their aim is to dictate some already proved guidelines to solve certain groups of problems. Metaheuristics are not greedy and they can even accept temporary bad values so that they can make a better evaluation of the solution space.

The advantages of using metaheuristics rather than exact methods shows by (Nesmachnow 2014) are:

- Quickly selection of solutions that fit in the search space.
- Quickly selection of high-quality solutions that have better performance than most of other ones.
- Analyze all the solution space without getting stuck on local optimums.

Metaheuristics are clearly the best way to solve NP-hard problems and it matches the ideas behind industry 4.0. which is a concept based on improving mass production in minimum needed time, idealized by (Schwab 2016). Trigged by industry 4.0 context and, the need of getting fast answers, the rest of this chapter will focus on metaheuristics.

It is possible to divide metaheuristics in two different approaches: local search and evolutionary. The first family is based on trajectories whereas the second one focused on populations. These families will be discussed next.

Trajectory based metaheuristic or local search (LS) family works with one solution in each iteration. The Figure 6 shows the general procedure of local search and the main steps are:

- Selection of a random value in the solution space that will define the starting point, s_0;
- Choose a neighbor of s_0 using a method that is metaheuristic specific and problem-dependent;
- Replace s_0 with the new neighbor s_1 if s_1 has better performance than s_0;
- The search continues in order to get close to the global optimum s^*, until a final condition is met;

Simulated Annealing, Tabu Search and Variable Neighborhood Search are examples of this family.

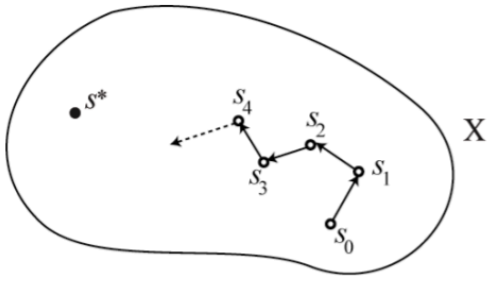

Figure 6: *Exploration of solution space X using LS procedure*

(Widmer et al. 2010)

Evolutionary approach (EA) or population-based metaheuristics were inspired in natural phenomena and started to catch researcher's attention some decades ago. Opposing trajectory-based metaheuristics, EA uses several individuals in each iteration, the so-called population. Usually, the size of population keeps the same size during all algorithm and the main goal is to continuously improve individuals' quality using natural evolution ideas. It also considers possible relations between individuals in order to get insights about their behavior. Figure 7 demonstrates the idea behind EA and the main steps are:

- Selection of a random initial population with the following elements: $s_{0,1}$, $s_{0,2}$, $s_{0,3}$, $s_{0,4}$. In each solution, the first index refers to the generation number and the second one to the individual number;

- Create a new population based in the previous one with a metaheuristic dependent approach that can be crossover and mutation in Genetic Algorithm or velocities and social influences in Particle Swarm Optimization.
- The search continues in order to get close to the global optimum *s**, until a final condition is met;
- Finally, a pareto front should be obtained which is a set of optimal solutions of a Multi Objective problem (Hong & Lee 2018). This happens because the solution to a multi objective optimization problem with multiple contradictory objective functions consists of sets of compromised objectives.

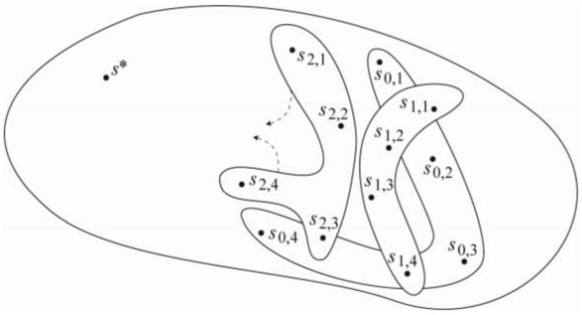

Figure 7: *Exploration of solution space using a population with four individuals.*

(Widmer et al. 2010)

These two metaheuristics' families are used in literature to solve scheduling problems with Make-To-Stock (MTS) and Make-To-Order (MTO) policies. However, there has been relatively less attention paid to MTO manufacturing (Kingsman et al. 1996). The main difference between these two architectures is that in MTS the product is already available in the stock when the order arrives and in MTO all production takes place after an order is received. MTO policy it's harder to work with because it puts the company in a competitive environment in determining how to respond to a customer, especially how to determine due dates (Weng et al. 2008). The future proposed method for solving the case study will need to take into account these two policies.

The following sections will explore some of the metaheuristics that have been most used to solve scheduling problems: simulated annealing, tabu search, genetic algorithm and particle swarm optimization (Teoh et al. 2015). Finally, the fusion of the previous methods will be also reviewed.

3.1.2 Trajectory-based Heuristic: *Simulated Annealing*

Simulated annealing (SA) is a trajectory-based metaheuristic based on the idea that metals and glasses become less flexible as temperature goes down. It is a nature-inspired metaheuristic that represents the process of annealing in metallurgy. SA principle was introduced by (Kirkpatrick et al. 1983) and (Laha 2008) explains this metaheuristic very well.

Figure 8 shows the pseudocode of SA. *T* represents the temperature and R_{max} the predefined number of iterations. As stated by the acceptance probability formula, $e^{-\Delta/T}$, the lower the temperature the lower the probability to accept new solutions. With fitness variation it works the other way around, the lower the variation the higher probability to accept it. In this case the goal is to minimize the objective function *f(x)*.

1: Generate initial solution x^c, initialise $R_{\max}$ and T
2: **for** $r = 1$ to $R_{\max}$ **do**
3: **while** stopping criteria not met **do**
4: Compute $x^n \in \mathcal{N}(x^c)$ (neighbour to current solution)
5: Compute $\Delta = f(x^n) - f(x^c)$ and generate u (uniform random variable)
6: **if** $(\Delta < 0)$ or $(e^{-\Delta/T} > u)$ **then** $x^c = x^n$
7: **end while**
8: Reduce T
9: **end for**

Figure 8: *SA algorithm pseudocode*

(Schumann 2011)

There are no guidelines to assign a correct value to the initial temperature and as a result it is crucial to be aware the problem context. For instance, if the aim is to accept several bad quality solutions, a high temperature must be set in the beginning. Regarding the cooling factor, it can influence final results because it is directly related with how many different regions in the solution space will be exploited (Gerber & Bornn 2018)**.** The feature on SA that allows it to avoid local optimums is the probability of accepting degrading solutions. Without this step it would be impossible to navigate towards the global optimum.

Although SA has the same algorithm base in every problem, it is possible to improve and add new features to the algorithm to get better results. (Yazdani et al. 2009) suggested a SA algorithm that uses one objective function in order to minimize the makespan. The authors concluded that it would be better to split the problem in two different phases: assignment and sequencing. Assign machines to jobs and the second one defines the tasks scheduling. Two different neighborhood structures were used for each problem division. Moreover, a specific approach was used to define an initial solution, compromising minimum processing times and localizations techniques. Comparing to benchmark problems, good quality solutions were presented. (Antonio Cruz-Chávez et al. 2017) also proposed an improvement to the traditional SA structure applied to mono-objective. In this case, the cooling factor behaves accordingly to the standard deviation of a sample of random feasible solutions. It was proven that with this feature, SA converges more quickly to good solutions.

In 2013, three ways of dealing with bi-objective approaches were explored: classic weighted method (CWSA), normalized weighted method (NWSA) and fuzzy simulated annealing (FSA) (Jolai et al. 2013). It was concluded that NWSA and FSA can result in better multi-objective solutions for the FJSSP.

3.1.3 Trajectory-based Heuristic: *Tabu Search*

Tabu search (TS) has an approach quite similar to SA but instead of having a probability to accept worse solutions, it has a tabu list that keeps track of all previous data, representing forbidden steps.

The main steps of TS are shown in Figure 9:

```
TabuSearch(Pr) // Pr is an optimisation problem

S := randomly generated solution of Pr;
Best := S;
While(termination criterion not met)
    S := best solution in the neighbourhood of S*;
    If(Cost(S) < Cost(Best))
        Best := S;
    EndIf
    update tabu list;
EndWhile
Return Best;

*the neighbourhood of S does not include solutions obtained using
 those moves which are contained in the tabu list and do not sati-
 sfy the aspiration criterion.
```

Figure 9: *TS algorithm pseudocode.*

(Montemanni et al. 2003)

The tabu list makes it possible to avoid cycles and sometimes forces the algorithm to accept worse solutions, avoiding local optimums. Regarding the list, it can adopt different sizes and it depends on how much solution space is desired to exploit. A small length tabu list will result on a narrow exploration of the space and the search will cycle around a local minimum (Glover et al. 1993). Alternatively, a long tabu list will be able to discover more different solutions throughout the search space (Dell'amico & Trubian 1993).

The following articles show interesting improvements to the traditional TS algorithm applied to FJSSP. In 1994, a special technique was used to select the first initial solution using a beam search (Hurink et al. 1994). The core of this process consists on analyzing a fixed number of incomplete viable schedules at the same time. Final results outperformed benchmark problems. (Dell'amico & Trubian 1993) used a disjunctive graph to represent all neighborhood of the problem. This graph agglomerates several vertices that serve as operations. They can be connected either by directed edges (when task B needs to be performed after

task A) or by undirected edges (when there is not temporal dependence). This way, any feasible solution will always create a path that goes toward the global optimum.

Regarding multi-objective applications of TS, most of literature use the classic weighed sum method in order to deal with FJSSP. (J. Li et al. 2010) tested a TS algorithm using classic weighed sum method to minimize three objectives: makespan, total workload of machines and the workload of the critical machine. It showed superiority compared to other competing approaches.

3.1.4 Population-based Heuristic: *Genetic Algorithm*

In order to represent Darwin's theory of evolution of species, genetic algorithm (GA) was created by John Holland between the fifties and sixties (Holland 1992). The structures that are used in GA are genes and chromosomes. Genes are the basic unit to solve the problem (a value for a specific variable) and chromosomes consist on a group of genes (constitute a solution). A population is an agglomeration of chromosomes.

Figure 10 displays the main operations in a GA metaheuristic.

```
(1) initialise population;
(2) evaluate population;
(3) while (!stopCondition) do
(4)     select the best-fit individuals for reproduction;
(5)     breed new individuals through crossover and mutation operations;
(6)     evaluate the individual fitness of new individuals;
(7)     replace least-fit population with new individuals;
```

Figure 10: *GA algorithm pseudocode.*

(Rashid et al. 2013)

In order to produce offspring, a crossover (CO) operator is used. Crossover consists on an exchange of genes from parents, resulting on a child chromosome. The most relevant CO types are: single point, k-point and uniform. Single-point CO consists on choosing the exact point where genes are going to be swapped. K-point CO allows to divide chromosomes as many times as desired. Finally, in uniform CO each gene is chosen from either parent with equal probability. (Magalhães-Mendes 2013) applied several crossover methods to JSSP and concluded that single point was the one showing better global results. (Umbarkar & Sheth 2015) also presented insights regarding ten different crossover techniques when using the generic GA.

In mutation, random parts of chromosomes are selected and changed. It is the same concept of biological mutation on living beings. This procedure avoids the GA to get stuck on local optimums and explores more

space. The use of the mutation operator is controlled by a probability parameter (mutation rate) which is usually small: between 1 and 10% (Werner 2017). The problem with mutation is that is possible to decrease chromosomes' quality and lead the population in the wrong way (Beheshti & Shamsuddin 2013). To avoid that, (Abraham et al. 2006) came up with an elitism method that first copies the best chromosome (or a few best chromosomes) to new population. It prevents losing best-found solutions.

After crossover and mutation, the best-fit individuals are selected in order to repeat all the process again. Usually the algorithm finishes when we achieve a certain execution time or when it reaches a reasonably good fitness value (Kumar et al. 2017).

The following articles show interesting improvements to different parts of GA. (Lee et al. 1998) created an enhanced crossover operator in order to solve scheduling problems with precedence constraints. It is precedence-preserving order-based crossover (POX) and always generates only feasible chromosomes. (Pezzella et al. 2008) used an approach of localization based of processing times and machines' workload to generate initial population. It provides better final results compared to using a random feasible initial population.

Regarding multi-objective genetic algorithms, there are two approaches to solve them: combining different objectives into a weighted sum and using a pareto approach. The most used one is pareto approach which aim at satisfying two goals: first, converge towards the Pareto front and also obtain diversified solutions scattered all over the Pareto front (Hsu et al. 2002). However, in 1995 it was proposed an improved way of using the weighed sum approach. If the weights are constant the search direction in GAs is fixed but with random weights various search directions are utilized (Murata & Ishibuchi 1995). This approach outperforms the traditional weighted sum methods.

(Morinaga et al. 2014) solved a FJSSP with MTO policy using a GA to minimize tardiness and setup-worker load. It was used a classic weighted sum approach to deal with the bi-objective problem. An island model was explored as an optimization method. This model allows to efficiently distribute genetic algorithms over multiple processors while introducing a new genetic operator, the migration operator. It improved the overall algorithmic performance and permits to effectively provide MTO products and services.

3.1.5 Population-based Heuristic: *Particle Swarm Optimization*

Particle swarm optimization (PSO) is a metaheuristic that uses the power of using several agents, called particles, and all the communication between them. It was developed by (Kennedy & Eberhart 1995) and it is inspired in natural behavior of birds as a community.

Figure 11 displays the pseudocode of a traditional PSO metaheuristic.

```
For each particle
   Initialize particle
END
Do
   For each particle
      Calculate fitness value
      If the fitness value is better than the best fitness value (pBest) in history
         set current value as the new pBest
   End

   Choose the particle with the best fitness value of all the particles as the gBest
   For each particle
      Calculate particle velocity according equation
      Update particle position according equation
   End
While maximum iterations or minimum error criteria is not attained
```

Figure 11: *PSO algorithm pseudocode.*

(Hudaib & AL Hwaitat 2017)

In a population with *N* particles:

- x_i refers to the position (exact location) of the *ith* particle in the space.
- v_i refers to the velocity or speed of *ith* particle. It will influence the next position of the particle. Usually, it is defined which is the maximum velocity, v_{max}, that particles are allowed to reach.
- b_i is the best position that this *ith* particle already found.
- *g* consists on a vector that records the best position discovered by all population so far.

PSO has three main steps: evaluate fitness of each particle, update individual and global bests and finally, update velocity and position of each particle (Eslami et al. 2012). The next formulas refer to velocity and position update, respectively:

(1) $$v_i(t+1) = v_i(t) + PI \times r_1[b_i(t) - x_i(t)] + SI \times r_2[g(t) - x_i(t)]$$

(2) $$x_i(t+1) = x_i(t) + v_i(t+1)$$

Eq. (1) calculates the velocity of the particle in the next moment, $v_i(t+1)$. $v_i(t)$ represents the velocity in the current moment, *PI* stands for personal influence and indicates how much the particle will focus on his history and previous knowledge, *SI* means social influence showing how much the entity will follow the group, r_1 and r_2 are two random numbers originated from a uniform distribution and finally $x_i(t+1)$ represents

the position in the next moment and $x_i(t)$ in the current moment (Tharwat et al. 2017). On the other hand, Eq. (2) generates the new position of the particle.

The next articles show some modifications in PSO in order to improve convergence and prevent an explosion of the swarm. (Shi & Eberhart 1998) proposed a new parameter *w* for the PSO called inertia weight which multiplies the $v_i(t)$ and controls the exploration of the search space. Low values of *w* limit particles' search region and high values allow movements with larger velocities and, consequently, seeking the global optimum neighborhood. They concluded that this parameter in the range [0.9,1.2] will have a better performance. Further research was done about inertia weights by (Bansal et al. 2011) concluding that from 15 different inertia weight strategies, Chaotic Inertia Weight is the best strategy for better accuracy and Random Inertia Weight strategy is best for better efficiency. Chaotic Inertia Weight formula is represented in Eq. (3) and Eq. (4): *w1* and *w2* are the original value and the final value of inertia weight, *MAXiter* and *iter* are the maximum iterative time and the current iterative time and z is a random number in the interval of (0, 1). Random Inertia Weight formula is shown in Eq. (5) where *Rand()* generates a random number in the interval of (0, 1).

(3) $$z = 4 \times z \times (1 - z)$$

(4) $$w = (w_1 - w_2) \times \frac{MAXiter - iter}{MAXiter} + w_2 \times z$$

(5) $$w = 0.5 + \frac{Rand()}{2}$$

(Van den Bergh & Engelbrecht 2002) came up with PSO version with strong local convergence. It is called Guaranteed Convergence Particle Swarm Optimizer (GCPSO) and performs much better with smaller number of particles, compared to the original PSO. (He et al. 2004) presented a PSO with passive congregation to improve performance. Passive congregation is an important biological force preserving swarm integrity and allows the sharing of information among individuals of the swarm. This approach was tested using benchmark instances and the results showed an improved search performance. (Pratchayaborirak & Kachitvichyanukul 2011) performed a two-stage PSO variation. In the first stage, four swarms are executed in sequence using PSO and considering the same objective function. When a swarm comes to an end, a percentage of particles migrate to the next swarm and are joined with newly initialized particles. In the second stage, an equal number of particles are randomly selected from the previous four swarms in order to build a new swarm. Finally, the PSO is executed one last time. This procedure allows the share of information between swarms; hence it is possible to speed-up the convergence of solution. For the single objective cases, makespan and weighted tardiness were considered. For the multi-objective cases, three criteria were considered: makespan, weighted earliness, and weighted tardiness. Compared to previously published results, two-stage PSO provided equal or better solution quality in much shorter computational time.

3.2 Metaheuristic Families Drawbacks

Before comparing all algorithms described before, the main components of any metaheuristic are worthy to explain: diversification and intensification. Diversification consists on exploring solutions in different areas whereas intensification focus on one specific area and tries to improve solutions' quality (Lozano & García-Martínez 2010). Diversification is often achieved by the use of randomization which enables the algorithm to jump out of any local optimum so as to explore the search globally. Intensification uses the local knowledge of the search and solutions already found to concentrate on local regions. A classic technique for of intensification is the hill-climbing (Kholladi & Ryma 2011).

(X. S. Yang et al. 2014) claims that intensification tends to increase the speed of convergence and diversification decreases it. On the other hand, too much diversification increases the probability of getting the global optimum with a reduced efficiency while strong intensification will lead the algorithm to be trapped in a local optimum. Therefore, a fine balance between intensification and diversification may dictate the performance of the algorithm.

Starting by local search metaheuristics, the main problem inherent to them is the lack of diversification. They are very good at local search ability but unsatisfactory in global search ability. This happens because both SA and TS start with a random value and then they immediately begin to converge to specific solutions. However, there are some ways to instigate diversification in TS. One technique is called restart diversification and consists in executing again the algorithm with a certain initial solution (different one from the other time). Another method of diversification consists in biasing the selection of next candidate solutions (Potvin & Gendreau 2010). The problem is that all these methods need to be very well planned and tested because there are no guidelines in how to apply them to this FJSSP.

Population-based metaheuristics have the opposite problem. Both PSO and GA have a lack of local search ability but on the other hand, they exploit very well all the solution space. Concerning GA, the principle behind all the space coverage is mutation and crossover because it enables the generation of different solutions caring out some similar basis. The problem is that it may need a lot of time on the same region in order to obtain the minimum. Apart from that, in GA it is difficult to identify the best values for parameters such as population size, mutation strategy, mutation rate and selection method (Deepa & Sivanandam 2008). The higher the number of parameters, the higher will be the difficulty of selecting them due to the greater number of combinatorial possibilities. Regarding PSO, the major drawbacks are premature character, i.e. it could converge to local minimum and problem-dependency performance (Eslami et al. 2012).

Table 2 summarizes the main advantages and disadvantages of the SA, TS, GA and PSO metaheuristics.

Table 2: *MH algorithms' advantages and disadvantages*

Algorithm	*Advantages*	*Disadvantages*
SA	Escapes from being trapped at a local optimum in pursuit of a global optimum (Wang et al. 2015).	Long running time (Dharmapriya et al. 2014).
TS	A bad strategic choice can often yield more information than a good random choice because of search history (Glover & Marti 2006).	Tendency to fall into local optimization (Huang & Han Yu 2013).
GA	It does not need a good initial knowledge of the problem and has an ability to prevent from fall into a local optimum with the help of mutation (Zukhri & Paputungan 2013)	Easy to fall into premature convergence to a local optimum and longer processing time for a problem with large data (Zukhri & Paputungan 2013).
PSO	Easy to implement (Xu et al. 2015).	Hard to do the parameter tuning due to PSO's poor exploration in literature.

Because of all handicaps associated in both trajectory and population-based metaheuristics, hybrid metaheuristics started to arise. The next section focuses on metaheuristics' hybridizations.

3.3 Hybrid Approaches

Hybridization consist of mixing two or more different approaches the same problem, aiming at getting better results. Usually local search heuristics are joint with population-based algorithms. This new school of metaheuristics often outperform the usage of single algorithms (Genova et al. 2015). Furthermore, when hybrid approaches are developed, typically the problem is divided and instead of solving both routing and scheduling at the same time, a hierarchical approach is adopted. This paradigm allows to substantially reduce the complexity of the problem.

To explore this path the following articles show implementations of hybrid metaheuristics in FJSSP. (Xia & Wu 2005) proposed a hybrid optimization. The model combines the advantages of PSO in terms of global search and SA of local search. The aim was to optimize makespan, total workload and machine specific workload using a weighted sum. PSO was used to do the routing part of the problem and the SA for operation scheduling. The encoding part based in priorities was one of key aspects in the implementation.

The results showed that the proposed algorithm was a viable and effective approach for the multi-objective FJSSP, especially for problems on a large scale.

(Azardoost & Imanipour 2011) proposed an algorithm to minimize makespan, maximum workload and total workload based on tabu search, simulated annealing and genetic algorithms. The approach consists on using TA and SA to generate the initial population for the GA. The obtained results showed that algorithm had the ability to achieve close to optimal points at suitable time for different issues in different sizes.

(X. Li & Gao 2016) demonstrated a hybrid metaheuristic using TS and GA with the objective to minimize makespan. In this article it was chosen a two-string encoding method: one string representing the assignment of machines to operations and the other one showing the temporal position of them (scheduling). The major characteristic of this method was that any permutation of the first-string conducts to a feasible decoded solution. In each iteration of GA, after producing offspring, each chromosome is converted to a feasible schedule solution and this value will behave as TS input. This feature enables a good exploitation of the solution space and it allows to control the trade-off between intensification and diversification. After comparing the results with other state-of-the-art reported algorithms, the conclusion was reached that this algorithm has a competitive computational time, efficiency and effectiveness. (Min et al. 2019) used a GA based algorithm integrated with SA and PSO to minimize energy consumption and makespan. SA is associated with mutation operator which avoids getting trapped in suboptimal solutions and crossover is inspired in PSO allowing offspring to be influenced by the best individual. Instead using only the two parents during the crossover, some genes of the best chromosome are also used to generate offspring. The algorithm could solve the problem effectively and efficiently.

(Zhang et al. 2011) made a hybridization on PSO and TS for a scheduling problem considering multiple objectives, such as the total cost of earliness/tardiness penalty, tardiness penalty in delivery time window, production, inventory matching and order cancelation penalty. It was developed in order to work under MTO and MTS environment. The authors proposed heuristic rules to repair infeasible solutions. The hybrid PSO+TS compared with PSO and TB individually provides better solutions while being computationally satisfactory.

3.4 Chapter Conclusions

In this chapter, a set of important related topics to the problem in study were explored. The contextualization of an optimization problem and specially the FJSSP were made in terms of characteristics and ways to solve them. Two different metaheuristic families and several algorithms were explored.

From the analyzed literature some conclusions that can be drawn:

- Approximate methods are currently the best way to solve scheduling problems in which are included the following algorithms: SA, TS, GA and PSO. All of them have their own advantages

and disadvantages depending if they belong to trajectories or populations family. Consequently, different levels of diversification and intensification are achieved.

- When dealing with multi-objective problems, population-based metaheuristics come up with several optimal solutions for a set of compromised objectives. Outputting different solutions with the same performance can be a relevant aspect for the factory manager since he would be able to deal with possible extrinsic aspects of the factory.
- The two explored metaheuristics in the population-based family were GA and PSO. Both may achieve reasonably good performances but GA has more related literature. Due to PSO's poor exploration, the parameters selection becomes a tricky task.
- GA was already successfully applied to multi-objective FJSSP under MTO and MTS production strategies. In literature, several articles already used a weighted sum with constant weights to the multiple objective functions to solve the problem. However, a weighted sum using randomly specified weights for each iteration was never applied to a FJSSP. It would be relevant to the current dissertation and for the scientific community to know what kind of solutions would this approach lead to.
- Hybridization using various metaheuristics usually guides towards better final results. For that reason, GA mixed with a LS approach will result on a more satisfactory solution because it would create a key balance between GA's diversification and LS's intensification.

Previous conclusions show why GA+LS hybridization is the most promising approach to solve the case study.

4. Algorithm Characterization

This chapter characterize the algorithm and its strategies and is divided in four sections. The first section gives an overview of the GA and LS hybridization. The second one covers all procedures of the mono-objective algorithm, using the tardiness as objective function (OF). The third section explores the Multi-objective algorithm, minimizing tardiness and makespan, as OFs. Ultimately, MTO and MTS strategies are described showing how they are incorporated in the algorithms.

4.1 Mono-objective Metaheuristic Algorithm (MO_BGA)

The approach that will be used is based on a classic GA to fit into a production scheduling problem. As a population-based metaheuristic, GA requires a starting procedure to generate a feasible initial solution, a neighbor generation technique, a crossover method and a mutation operator to prevent an early stop on a local optimum. Then a local search procedure will be used to do a better intensification of the neighborhood feasible space.

Since the problem under study is a production scheduling problem, the main objective is to ensure that all orders are manufactured on time. Therefore, the first objective that will be taken into consideration is the minimization of the tardiness resulting on a mono-objective approach to the problem. The tardiness and other important notations are represented below:

Number of jobs: n

Completion time of job i, C_i (finishing time of an order)

(5) Lateness of job i: $L_i = C_i - d_i$, d_i as due date

(6) Tardiness of job i: $T_i = \max\{0, L_i\}, \quad L_i > 0$

(7) Earliness of job i: $E_i = \max\{0, -L_i\}$

(8) Total Tardiness: $T = \sum_{i=1}^{n} T_i$

(9) Makespan: $C_{max} = \max\{C_1, C_2, C_3, \dots, C_n\}$

The mono-objective algorithm comprehends an encoding scheme and six different major procedures. The encoding scheme is the way genes and chromosomes are represented and consequently, it covers all the mechanism behind GA. The procedures are sequential operations of the algorithm. The first procedure generates the initial population. The second addresses the fitness calculation for each chromosome and selects some of the best ones while the third is related with generating offspring. The fourth procedure applies mutation to the new population and the fifth applies a local search to the best chromosome. The

last procedure analyses the stop criterion. The algorithm procedures, defined as MO_BGA algorithm, are presented in Figure 14 and will be further addressed with greater detail. An illustrative example is used to illustrate all the procedures:

- The quantity of products that the factory produces (P), the number of operating machines (M) and the number of orders to produce (O), shown in Table 3.

Table 3: *Number of products, machines and orders.*

Products	5
Machines	2
Orders	5

- Table 4 indicates the necessary time needed to swap between products, called setup time (PxP matrix). This is a symmetric matrix which assumes that that time to changeover from product 1 to product 2 is the same as the time to changeover from product 2 to product 1. However, in most cases an asymmetric matrix is used. For instance, a situation when is necessary to paint products with different colors, the changeover time from white to black could be different from changeover time from black to white.

Table 4: *Setup times (min).*

Setup times		Product				
		1	2	3	4	5
Product	1	0	10	300	100	300
	2	10	0	100	100	300
	3	300	100	0	100	10
	4	100	100	100	0	100
	5	300	300	10	100	0

- All data regarding the orders are illustrated in Table 5: order number, order's product number, the quantity to be produced and the due date (min).

Table 5: *Orders information.*

Order	1	2	3	4	5
Product	2	3	4	1	5
Quantity	10000	6000	8000	6000	2000
Due Date	200	500	500	300	400

- Table 6 defines the processing times (MxP matrix). In this example, P3 cannot be produced by M2 so a large number is assigned to that position ("10000"). Even if a chromosome decides to produce P3 in M2, it will be immediately discarded in the next iteration due to its unreasonable processing time.

Table 6: *Product processing times (min)*

Processing Times (min)	P1	P2	P3	P4	P5
M1	1	1	10000	0,04	0,02
M2	0,02	0,005	0,03	1	1

- Finally, table 7 points out the parameters inherent to the GA mechanism. Population size (N) represents the fixed number of chromosomes in each generation, generations number (G) are the total number of new generations created and the mutation rate (MR) consists in the probability of applying the mutation operator in each chromosome.

Table 7: *GA parameters*

Population Size	30
Generations number	5
Mutation Rate	0,5

Encoding Scheme

In GA, an encoding function is used for mapping the solution space to the coding space and the decoding function maps the coding space to the solution space. The object variables in the solution space are used to evaluate the fitness of chromosomes and to select them whereas the string codes in the coding space are used to perform all genetic operations and the local search. Figure 12 shows this encoding – decoding method.

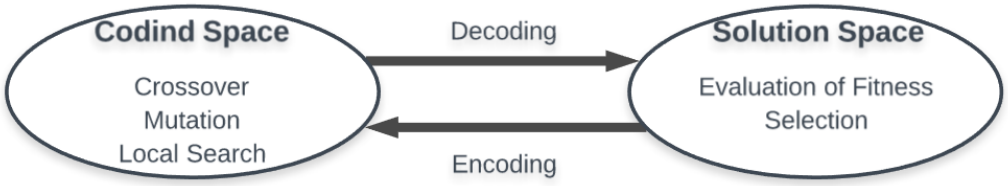

Figure 12: *Encoding – Decoding method.*

In the population, a multi-string encoding rule is used for each chromosome containing the necessary information about the machine selection and the order of the operations. Each chromosome is composed

of two strings, **Figure 13**. Figure 13 does not use the same data as the illustrative example explained previously. The first one represents the machine gene-string (C1) and the second one indicates an operation gene-string (C2). The total length (*L*) of each chromosome equals two times the total number of orders and both C1 and C2 operation gene-string have a size of *L/2* (Figure 13).

Figure 13 characterize five orders {O_1, O_2, O_3, O_4, O_5} that need to be produced, using one of three different machines {M_1, M_2, M_3}, so $L = 2 \times 5 = 10$. In C1, the value of each gene represents the corresponding machine and in C2 each gene refers the order that will be produced as well as the index of the C1 machine associated. For example:

- The 1st position "4" of C2 represents the fourth index of C1, "1", showing that O_4 will be produced by M_1.
- The 2nd position "3" of C2 represents the third position of C1, "3", showing that O_3 is assigned to M_3.
- The 3rd position "1" of C2 (meaning the third operation to be executed) denotes the first index of C1, "2", and this way O_1 is produced by M_2.

Similarly, regarding the rest of C2, the encoding and decoding operations can be also obtained. Therefore, the final sequence of operations with the assigned machines can be drawn: $O_4(M_1)$ - $O_3(M_3)$ - $O_1(M_2)$ - $O_5(M_3)$ - $O_2(M_2)$.

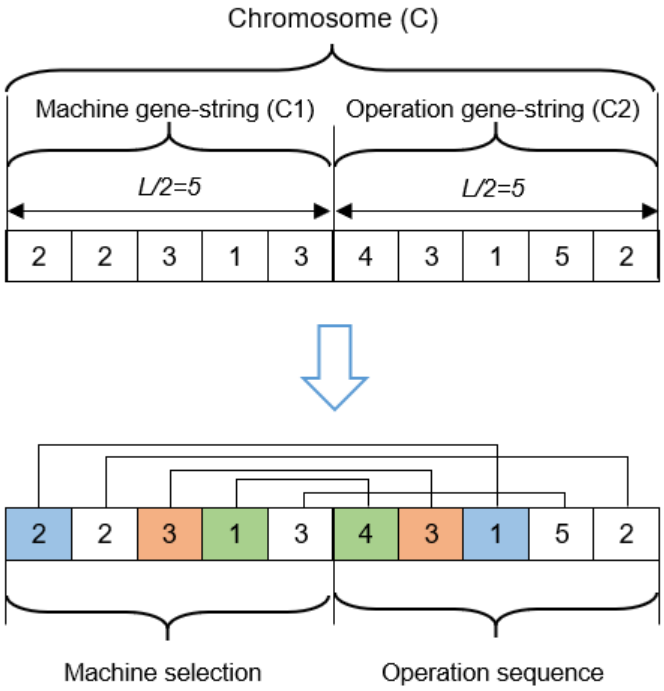

Figure 13: Chromosome representation.

The MO_BGA algorithm Framework is characterized in **Figure 14:**

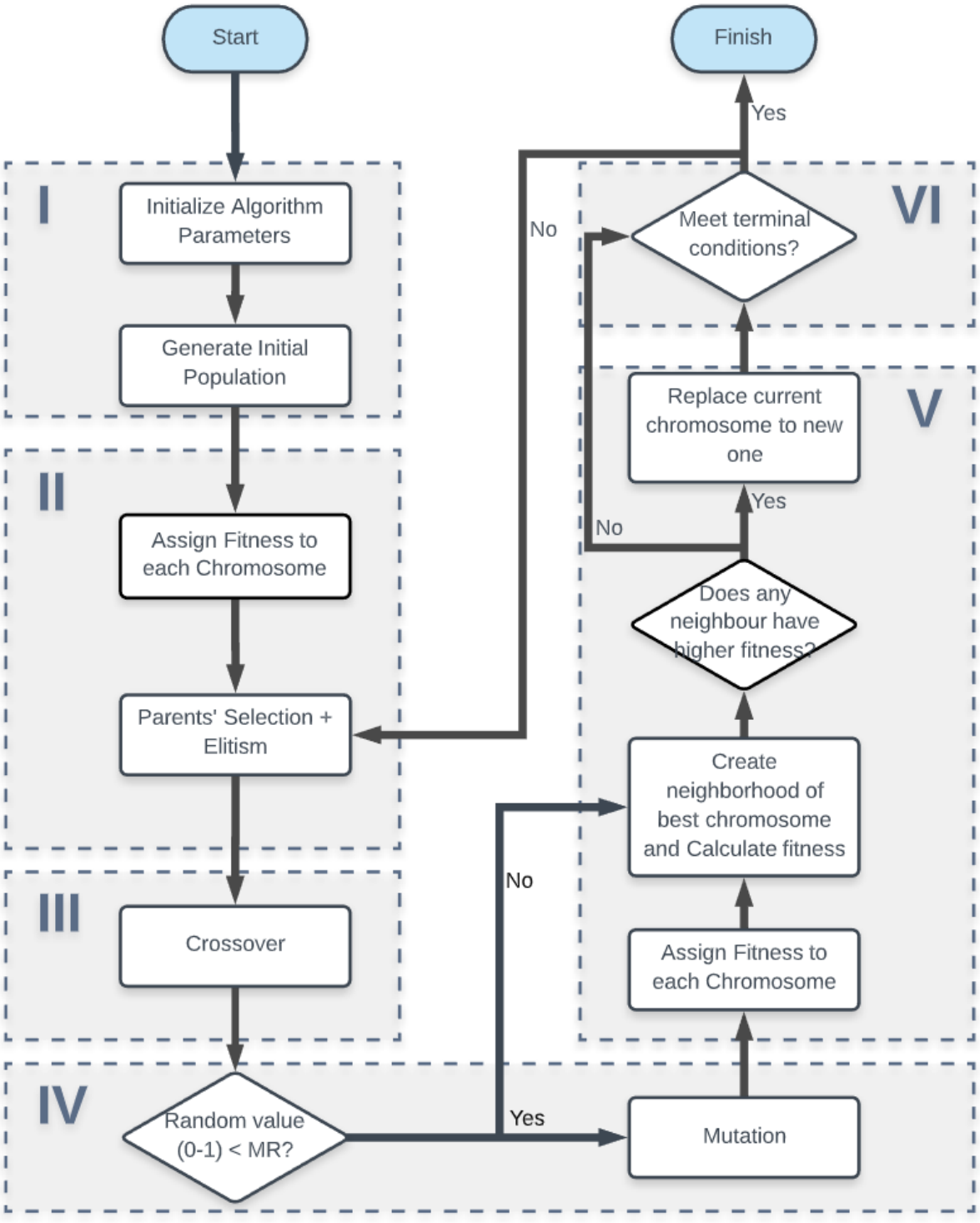

Figure 14: *Schematic representation of the MO_BGA algorithm. *MR stands for Mutation Rate.*

Procedure I – GA Initialization

The first procedure (I) addresses the initialization of the GA parameters and generation of an initial solution for the algorithm to start. It consists on the first two operations of Figure 14.

The initialization embraces the treatment of the input data with all values of Tables 3 to 7 being placed in the respective data structures, followed by the initial solution generation. In order to create the first population, two different methods are used, one for C1 and other for C2. In C1 the numbers in the genes can be repeated because several orders can be done in the same machine. However, in C2 a sequence of non-repeated numbers needs to be generated because each order is just executed once. Therefore, C2 is first populated with a ordered sequence of numbers and then the Fisher-Yates shuffle algorithm (Fisher & Yates 1963) is used to come up with a random initial solution. The algorithm randomly permutes J elements by exchanging each element e_i with a random element from *i* to J (Black 2019). Fisher-Yates shuffle's time complexity is *O(n)* meaning it runs in linear time. The pseudocode of Figure 15 details the generation of an initial chromosome. To produce the whole population, it is necessary to repeat the pseudocode N times.

```
Algorithm: Generate initial chromosome
   Input: NumberMachines, NumberOrders
   Output: Chromosome
 1 begin
 2     C1 = integer array with size of NumberOrders;
 3     C2 = integer array with size of NumberOrders;
 4     for each position in C1 do
 5         generate random integer number between 1 and NumberMachines;
 6     Populate C2 with non-repeated numbers from 1 to NumberOrders;
 7     for i from NumberOrders downto 1 do
 8         j = generate random integer number between 1 and i;
 9         exchange C2[j] and C2[i]
10     Chromosome = C1 and C2 joined
```

Figure 15: *Allgorithm to generate initial population.*

With all the inputs, parameters initialization and initial chromosome generation algorithm, an initial population is achieved. The best chromosome of the population is found in Table 8. The Table 8 includes the allocation of the several orders to the machines and the corresponding production sequence. Machine 1 will produce orders 5 and 3 while machine 2 will produce orders 4, 2 followed by 1. From now on chromosomes will represent solutions for the studied FJSSP - the scheduling of a set of production jobs (orders) on the available machines.

Table 8: *Initial solution - Procedure I*

Machine	Nº of Orders	1	2	3	4	5
M1	2	O5	O3			
M2	3	O4	O2	O1		

Procedure II – Fitness and Parents' Selection

The initial solution generation is followed by the second procedure (II) which comprises the fitness evaluation of each chromosome and parents' selection.

Based on the illustrative example, Table 9 shows the completion time, due date and delay of each order. Whenever a machine executes the first order, the completion time of previous order doesn't exist and the setup time is negligible, only remaining the processing time. To illustrate, the completion time of O5 is just the processing time of P5 in M1 ,0.02 (Table 6), times the quantity to be produced, 2000 (Table 5), resulting in 40 minutes. On the other hand, O3 completion time can be calculated by the sum of:

- The completion time of the previous order (O5) in the same machine (M1): 40 (Table 9)
- The setup time between product 5 and 4: 100 (Table 4)
- The processing time of P4 in M1, 0.04 (Table 6), multiplied by the quantity of O3, 8000 (Table 5): 320

Therefore, the completion time of O3 is 40 + 100 + 320 = 460.

If the Lateness <0 means the order did not exceed the due date and the value 0 is registered in the tardiness of that order. In the illustrative example, the order 5 has a due date of 400 minutes and is completed in 40 minutes. This order has an earliness of 360 minutes. Finally, Order 1 and Order 2 are the only delayed orders, with a total tardiness of 650 minutes.

Table 9: *Initial solution orders completion, due dates and delays*

Order Nº	O1	O2	O3	O4	O5
Product produced	P2	P3	P4	P1	P5
Completion Time	750	600	460	120	40
Due Date	-200	-500	-500	-300	-400
Tardiness	550	100	0	0	0

The method described quantify the total tardiness of each chromosome in the population, representing the fitness function.

In every population it is possible to rank individuals by order of performance, starting with the ones with lowest tardiness and finishing in the ones with highest tardiness. The way fitness is calculated ensures the solutions with the lowest Tardiness will be the ones selected to generate the following generation.

The generation of the new population has two phases: elitism and crossover. The elitism prevents losing best-found solutions by copying the best chromosome to the new generation. For the crossover, the α best individuals are selected in the first place. The α is calculated by solving the simple one-variable linear programing in figure 16. The first constraint is based on the combination's formula. The formulation in the figure 16 calculates the minimum number of parents necessary to be joined together and produce N-1 new chromosomes (minus one chromosome because of elitism). The multiplication by 2 is used because it is different to mate Chromosome1-Chromosome2 or Chromosome2-Chromosome1, as will be explained forth.

$$
\begin{aligned}
&Minimize && \alpha \\
&subject\ to, && N - 1 \leq \left(\frac{\alpha!}{2!\,(\alpha-2)!}\right) * 2 \\
& && \alpha \geq 2 \\
&and && \alpha \in \mathbb{Z}
\end{aligned}
$$

Figure 16: *Define the number of parents*

Back to the illustrative example, a population of 30 individuals is used (Table 7) so *n* equals to 6. As a result, the 6 best chromosomes will be used to generate offspring employing the crossover operator.

Procedure III - Crossover

This procedure uses a crossover operator to generate the new population. Based in the above-mentioned selection, the *n* best chromosomes are placed in an increasing order of objective function value. After that, the first chromosome is mated with the remaining ones, then the second chromosome is mated with the rest of them and so on.

The procedure of crossover operator for the machine gene-string is described as follows, and a crossover operator instance is shown in Figure 17 (a).

- Step 1: Establish two parent individuals (P1, P2) for the machine gene-string in terms of chromosomes.
- Step 2: Randomly select *L/4* (half the size of machine gene-string corresponds to 2.5 so the value 3 were considered) integer gene positions in P1 to generate a position set A.

- Step 3: Append the corresponding gene elements of the position set A in P1 to a new individual O. Check the remaining empty positions in O and copy those positions of P2 into O to obtain the final offspring individual.

The procedure of crossover operator for the operation gene-string is described below, and a crossover operator instance is shown in Figure 17 (b).

- Step 1: Determine two parent individuals (P1, P2) for the machine gene-string in terms of chromosomes.
- Step 2: Select randomly L/4 (half the size of machine gene-string corresponds to 2.5 so the value 3 were considered) integer gene positions in P1 to generate a position set A.
- Step 3: Append the corresponding gene elements of the position set A in P1 to a new individual O in order to generate a child chromosome.
- Step 4: Check the genes in P2 that are not already in O, and orderly copy them to the empty positions of O.

After selecting two parent individuals, the crossover operator for the machine and operation gene-string is applied and one new individual O is generated.

As it can be seen, the position set A always refer to the selected genes of the first parent. Consequently, it is different to mate P1 with P2 or P2 with P1 because the first genes of the offspring will belong to P1 or P2, respectively.

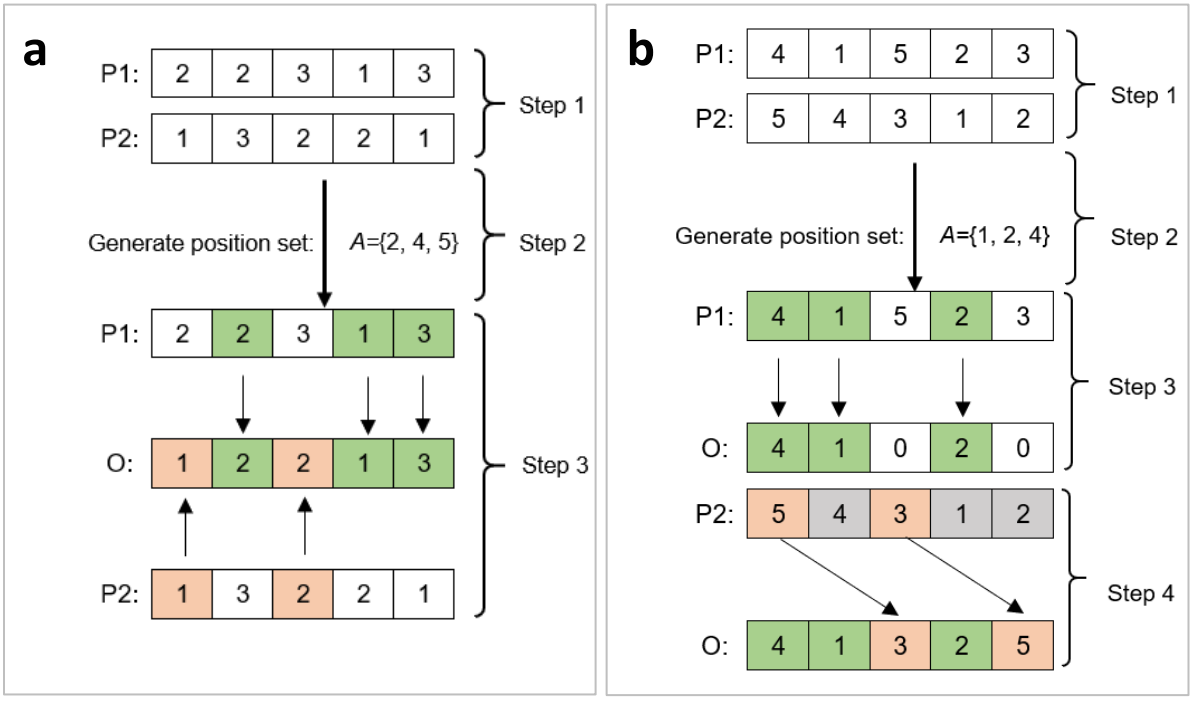

Figure 17: *Crossover operator: machine gene-string (**a**) and operation gene-string (**b**)*

The new generated chromosomes will represent viable solutions. Hopefully, some of them will have better fitness, that is lower Tardiness, than their parents

Procedure IV - Mutation

The diversification in GA is done by applying a mutation operator. Mutation is used to obtain new individuals. It applies one change in C1 and one change in C2. The parameter MR (Table 7) defines the probability of occurring a mutation. For each chromosome, a random number between 0 and 1 is generated and if that number < MR then a mutation is executed.

Figure 18 demonstrates an individual being mutated. For the machine gene-string, the new individual in created following these steps:

- Step 1: Randomly select one position in the parent individual.
- Step 2: Randomly change that value for another machine number.

For the operation gene-string, the new individual is generated as:

- Step 1: Randomly select two positions from the parent chromosome.
- Step 2: Swap the genes in the selected positions.

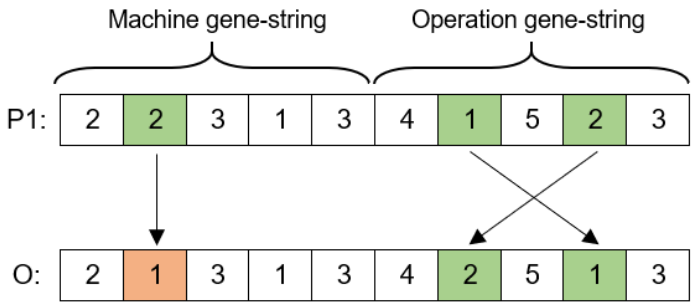

Figure 18: *Mutation operator*

The mutated chromosome may be a fundamentally different solution, noticeably worse than the original pre-mutation solution. Nevertheless, the Mutation procedure is very useful to inject diversification on the process.

Procedure V – Hill Climbing

The local search (LS) feature in the GA intensifies the algorithm. One of the most simple ways to proceed in a local search is using the Hill Climbing (HC) algorithm introduced by (Greiner 1992). The analogy for HC behavior is a climber that seeks to climb the summit of a mountain. However, the fog just allows the climber to see if the immediate surface to him is ascending or descending (Jaime et al. 2019). The HC will be the basis of LS in MO_BGA and it has 4 execution steps, clarified below.

- Step 1: Select the best chromosome (B) from the population.
- Step 2: Generate N new chromosomes (neighbors) using B as parent and applying mutation.
- Step 3: Evaluate the fitness (tardiness, OF value) of each new individual. The fitness calculation is done the same way as in procedure II.
- Step 4: If any new neighbor has lower fitness than B, B is replaced. Otherwise, this procedure has no influence.

Procedure VI – Stop Criterion

The last procedure comprehends the stop criteria control. The stop criterion for MO_BGA was set to be the number of generations (algorithm iterations). When the number of generations was already reached, it stops and returns the current population. Otherwise it goes again for the second procedure.

Based on the illustrative example, after running the algorithm for 5 generations, the stop criterion is obtained. The algorithm outputs a population whose best chromosome has an OF value of 0 which outperforms the initial one of 650. This evolution is a consequence of the scheduling. The sequence before was orders 4, 2 and 1 for M2 (Table 8). The new sequence in M2 becomes order 4, 1, followed by 2, showed in Table 10.

Table 10: *Best chromosome production sequence after the stop criterion is met*

Machine	Nº of Orders	1	2	3	4	5
M1	2	O5	O3			
M2	3	O4	~~O2~~ O1	~~O1~~ O2		
Obj1	0					

Table 11 shows the corresponding orders completion, due dates and tardiness of the new solution. For instance, O1 showed an earliness of 20 minutes because it was completed 20 minutes before the deadline, O2 had an earliness of 40 minutes and so on and so forth. Therefore, this chromosome has a OF of 0 representing an optimal solution since every order were completed before the deadline.

Table 11: *Best chromosome orders completion, due dates and tardiness after the stop criterion is met*

Order Nº	**O1**	**O2**	**O3**	**O4**	**O5**
Product produced	P2	P3	P4	P1	P5
Completion Time	180	460	460	120	40
Due Date	-200	-500	-500	-300	-400
Tardiness	0	0	0	0	0

The algorithm considers the optimality is reached when the total tardiness is zero. It is predicted that this approach will stop the algorithm as soon as the first optimal solution is found. However, it is possible to make the solution more efficient in terms of production and setup times.

Therefore, a new objective function will be implemented in the next section: the minimization of the makespan. This will trigger a bi-objective metaheuristic approach that combines both objectives. It will guarantee that the solution reaches zero tardiness and minimizes the makespan at the same time.

4.2 Bi-objective Metaheuristic Algorithm (BO_BGA)

The bi-objective algorithm (BO_BGA) is an extension of mono-objective algorithm. In this case, two objective functions are explored: the makespan and tardiness. Despite dealing with two objectives, tardiness is clearly the main one since has a high impact in the service level, by delivering every order in the right time to client, while makespan quantifies the performance impact, does not affect the client, directly. Therefore, the algorithm must give more importance to the tardiness until it finds the optimal and then stop prioritizing the tardiness and minimize the makespan. An approach suggested by (Murata & Ishibuchi 1995) is a weighed sum, that we adopted in this work. This approach applies random weights to objective functions every iteration. As can be seen in Figure 19, the random weights explore several search directions instead of a single direction, when constant weights are attached to the multiple objective functions.

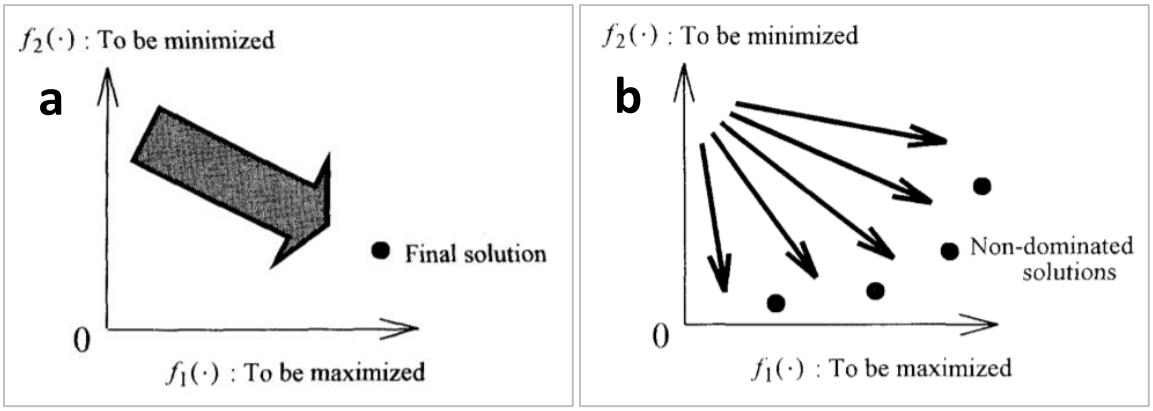

Figure 19: *Directions of search with constant weights (**a**) and random weights (**b**).*

(Murata & Ishibuchi 1995)

Literature does not solve a FJSSP with a weighted sum using random weights, so this research aims to fill this gap.

As far as we know, we are the first exploring the FJSSP with a weighted sum using random weights. The proposed formulation has impact on procedures II and V, as shown in Figure 20. In the second procedure a step to calculate weights is added and the calculation of chromosomes' fitness is done differently. In the fifth procedure also the calculation of chromosomes' fitness following a different procedure. **Figure 20** characterizes the framework and highlights the modifications for the proposed formulation, highlighting it (red contour).

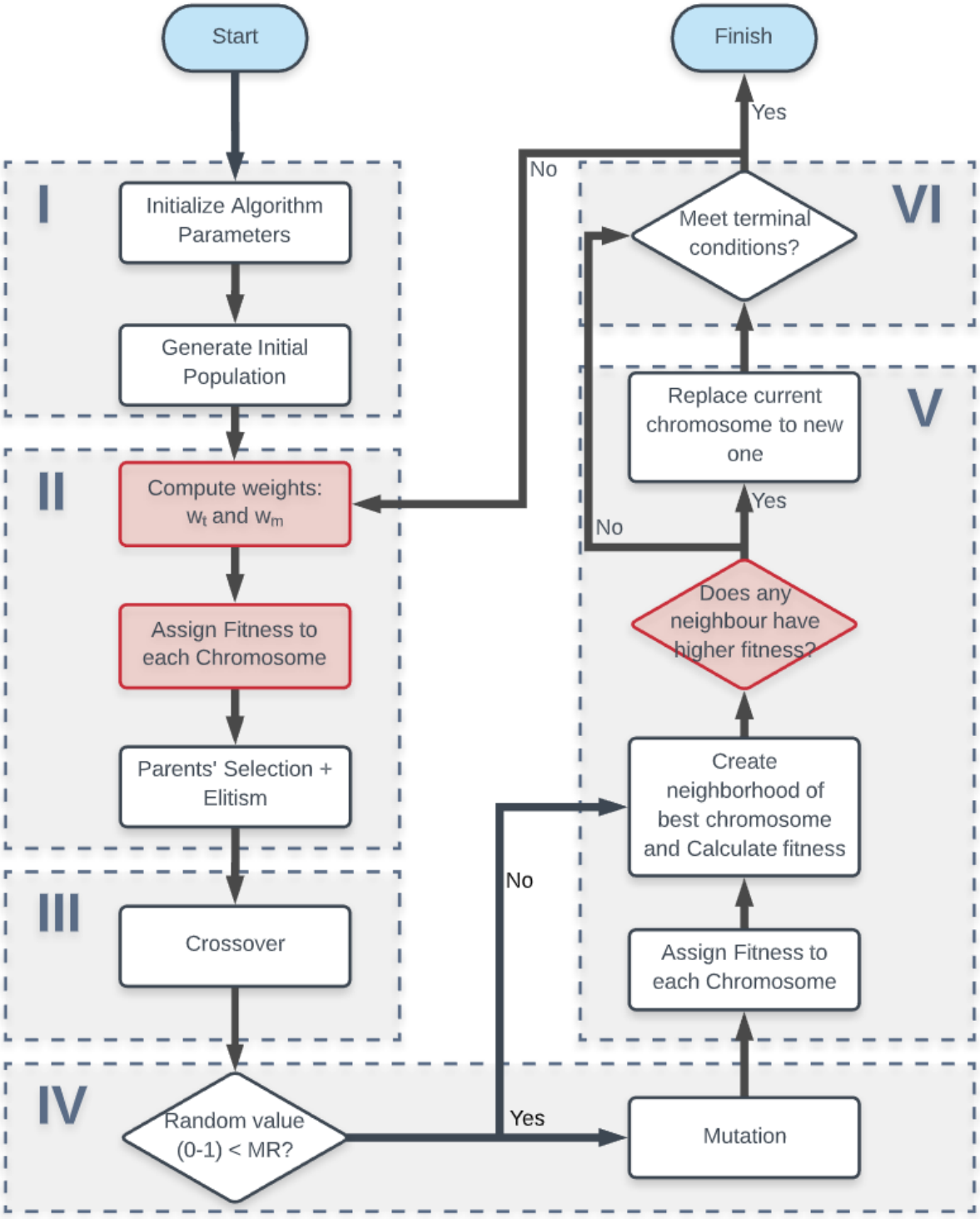

Figure 20: *Flowchart of BO_BGA*

Procedure II

This procedure starts with the calculation of the weight for tardiness (W_t) and for makespan (W_m). Figure 21 shows how they are computed. If in any previous iteration of the algorithm the zero tardiness was achieved, then the tardiness will not be prioritized meaning no rules about random weights generation will be imposed. This case follows the original idea of (Murata & Ishibuchi 1995). However, in order to enhance the tardiness importance, a new parameter is defined the lower bound (L). The lower bound will be used to control the minimum weight the tardiness can have. If the optimal tardiness was not achieved yet, then the minimum weight it can have corresponds to the value of lower bound parameter. Consequently, the search will first focus on finding the optimal tardiness OF, which has more impact to the consumer.

```
Algorithm: Weights calculation
   Input: Lowerbound
   Output: Wt, Wm
1  begin
2      if Tardiness = 0 was already achieved then
3          rand = random number between 0 and 1;
4      else
5          rand = random number between Lowerbound and 1;
6      Wt = rand;
7      Wm = 1 - rand;
```

Figure 21: *Weight calculation for tardiness (W_t) and makespan (W_m).*

After calculating W_t and W_m, the fitness is assigned to every chromosome. It is important guarantee that the sum of the weights must be equal to 1, as shown in Eq. (10).

The total tardiness and makespan are calculated using Eq. (8) and Eq. (9), respectively. Finally, the Eq. (11) is used to determine the fitness.

(10) $$W_t + W_m = 1$$

(11) $$Fitness = W_t \times Tardiness + W_m \times Makespan$$

The following steps in second procedure, the selection phase, is done the same way as in the mono-objective formulation.

Procedure V

In the MO_BGA fifth procedure the evaluation of the fitness is done differently. The fitness is computed with the Eq. 11 using the same weights calculated in the second procedure. Keeping the same weight values during every algorithm iteration allows the solutions to be tested under the same circumstances.

4.3 Production Strategies

Plastics industry works with large stock levels to fulfill costumer service level and face uncertainty, which is Make-To-Stock strategy (MTS). However, MTS has the handicap of having high inventory costs, the fact that it applies a lot of pressure in the warehouse manager since he has the responsibility of keeping the most efficient stock level to maintain the low inventory costs. Therefore, the Make-To-Order strategy (MTO), overcome the MTS drawback.

The two strategies are explored in the algorithm to validate their behavior as well as their advantages and disadvantages. The production strategies are detailed in the following sections.

4.3.1 Make-To-Order Strategy

MTO is a production strategy that does not consider any finished products in the warehouse. The production schedule is based on the received orders, and the production starts from the moment an order is received.

Production under MTO strategy requires a thigh schedule, since there is no stock available. A higher operational efficiency is needed to make it possible to produce higher volumes of order is the shortest period.

The main advantage of this strategy is that stocks are eliminated and consequently, all inventory costs are reduced. However, MTO has two inherent drawbacks. Firstly, the uncertainty of the market may create unplanned demand peaks. Thus, the risk of failing due dates grows substantially. Secondly, MTO has higher production setup times.

For instance, if a machine needs to produce the following sequence of products:

1. Order 1: Product 1
2. Order 2: Product 2
3. Order 3: Product 1

The machine will produce product 1, followed by product 2 finishing with product 1, resulting on unnecessary setup times between orders.

The way that the algorithm was explained in sections 4.1 and 4.2 already incorporates the MTO strategy. No stock values were taken into consideration and orders were treated independently. Next section shows how MTS is applied.

4.3.2 Make-To-Stock Strategy

In order to explore MTS in the algorithm, both Mono-objective and Bi-objective algorithms are extended in Procedure I. An order adjustment needs to be done and a new set of inventory availability information is introduced. In both framework of Figure 14 and Figure 20 the MTS modification takes place in the first step of Procedure I entitled “Initialize Algorithm Parameters”.

To illustrate the strategy a motivating example is used:

- The facility receives actual orders from each product: quantity and due date, Table 12.

Table 12: *Motivating example of a set of initial orders*

Actual Orders	O1	O2	O3	O4	O5	O6
Product	P4	P2	P4	P1	P3	P1
Quantity	5000	3000	3000	20000	2000	6000
Due Date	40	20	10	25	35	50

- The information regarding the initial stock and stock level limits for the finished products is received. The initial stock is the quantity of finished products in the warehouse. Final stock levels are calculated by subtracting the aggregated order quantities of the same product of Table 12 to the initial stock. Stock level limits define the minimum and maximum quantities of finished products to face unpredictable demand peaks and manage the uncertainty in the demand market. This information is presented in Table 13.

Table 13: *Stocks information*

Product	Initial Stock	Final Stock	Minimum	Maximum
P1	15000	-11000	10000	16000
P2	10000	7000	5000	12000
P3	22000	20000	19000	25000
P4	10000	2000	8000	12000

- After all the stocks characterization are done (as Table 13), three different situations may occur, as shown in Figure 22.

- If the final stock is between the minimum and the maximum, no production is needed.
- If the final stock is positive and below minimum, there were enough finished products in the warehouse to satisfy the order right ways (so due dates are irrelevant). However, it is necessary to restock that product until it reaches the maximum value.
- Finally, if the final stock is negative, the production needs to reach products' maximum value but, and satisfy the orders due dates. The due date assigned to that order will be the shortest for the actual orders in the same product, to be satisfy the service level and order will delivered on time.

```
Algorithm: Data treatment for MTS
   Input:  Table 12, Table 13
   Output: Production Orders
1  begin
2      for each product do
3          if minimum < final stock < maximum then
4              No production needed;
5          if 0 < final stock < minimum then
6              Produce enough product for the stock level to reach its
                maximum limit without due date;
7          if final stock < 0 then
8              Produce enough product for the stock level to reach its
                maximum limit with due date;
```

Figure 22: *Production orders calculation pseudocode*

- With the previous step applied to every product, a final production orders table is achieved, Table 14.

Table 14: *Production orders after data treatment*

Production Orders	**O1'**	**O2'**	**O3'**	**O4'**
Product	P1	P2	P3	P4
Quantity	27000	0	23000	10000
Due Date	25	-	-	-

As can be seen, the 6 actual orders of Table 12 turned into 3 production orders in Table 14 (O2' does not count because has 0 quantity). From this production orders, only O1' was produced considering the due date because the final stock of P1 was negative (Table 13).

After the production orders have been calculated, the algorithm is ready to follow to the next step. Henceforth, the chromosomes' initialization and the remain procedures (II, III, IV, V and VI) are the same as in sections 4.1 and 4.2.

5. The Algorithm Tuning

This chapter includes three sections, section 5.1, 5.2 and 5.3. The first section presents the algorithm and details the input data. Section 5.2 consists on a sensitivity analysis to select the parameters' values that produce a better performance in the algorithm, followed by section 5.3 that analyses the results achieved by the proposed algorithms. The algorithms were run using an Intel Core i7-6700HQ, 2.60GHz, 8 GB RAM.

5.1 Input Data

The case study data was applied in all formulations in this chapter, seeking to find the final production schedules and then select the optimal one.

It characterizes the manufacturing environment of the factory in order to produce a schedule for 30 products, 40 orders and using 8 machines. Within the products, 15 are bottoms, that are produced in machines 1 to 5, and 15 are covers which are produced in machines 6 to 8. The products have different labels, colors and shapes. Every product switch has a setup time and every machine has a processing time. Ultimately, orders data (product, quantity and due date) and warehouse stock information is also provided in Appendix 1.

The GA data depends on the approach that is being used. The mono-objective approach uses three parameters, population size, generations number and mutation rate, whereas the bi-objective approach uses the same three parameters of the mono-objective plus the lower bound (Figure 23).

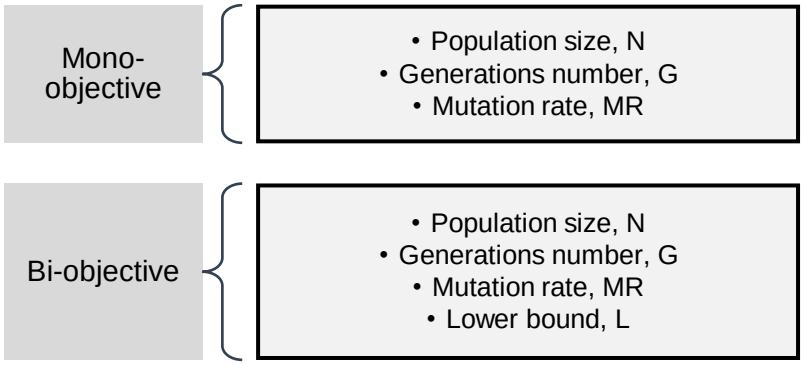

Figure 23: Sets of algorithm inputs for MO and BO approach.

As shown in Table 15, parameters values have a huge impact in algorithm's CPU time and solutions quality affecting the final performance. (Gen & Cheng 1997) stated that parameters of GAs depend largely on the characteristics of each problem. Therefore, different sets of parameters must be tested in the algorithm to investigate its behavior and, consequently, select the values tailored to the case.

Table 15: Impact of parameters' values on GA performance

Parameters	*Too low*	*Too high*
Population size	Finds a sub optimal solution and converge to that solution (Gotshall & Rylander 2002)	The number of generations to convergence increases without much increase in accuracy (Gotshall & Rylander 2002)
Mutation rate	Cannot help to improve genetic diversity and consequently get stuck in local minima (Bettemir 2011)	Will be detrimental on the good candidate solutions and prevent convergence to optimum (Bettemir 2011)
Generations number	The algorithm will stop without having enough time to realize its search possibilities (Tsoy 2003)	High CPU time (Boyabatli & Sabuncuoglu 2004)

Considering the information provided in Table 15, the next section performs a sensitivity analysis to the algorithm using the input data presented in the current section in order to select the parameters which produce the better results.

5.2 Tuning GA - Sensitivity Analysis

This section aims to calibrate the algorithm's parameters. Literature refers various techniques to deal with this process and the main ones are grid search and random search.

Grid search consists on defining several values for each parameter and then all combinations of values are tested. The set of parameters presenting the best performance is chosen. The pattern followed here is similar to the grid, where all the values are placed in the form of a matrix. The main advantage of it is its simplicity to implement and is non iterative (i.e. future simulations do not depend on past simulations) (Verwaeren et al. 2015).

Random search consists on a technique where random combinations of parameters are tested to find the best solution for the algorithm. The main advantages of random search are:

- That randomly chosen trials are more efficient for parameter optimization than trials on a grid. (Bergstra & Bengio 2012) showed empirically and theoretically.
- Random search allows statistical independence of every trial (Sipper et al. 2018)

Although random search sounds promising, it has a dangerous drawback that concerns the high variance that it yields during computing, possibly leading to non-satisfactory results. (Sipper et al. 2018) applied random search for several GA and some parameter values showed no tendency to aggregate anywhere. Due to this reason, a grid search was used in this dissertation.

The algorithm uses random numbers in the initial solution generation, crossover, mutation and in the local search. These random numbers are obtained by using the *Rnd* function in Microsoft's Visual Basic for Applications (VBA). This function uses a linear congruential generator for pseudo-random number generation. It outputs a sequence of numbers whose properties are like random numbers; however, they are not truly random because it is completely determined by an initial value called seed. Although there are ways to generate truly random numbers, pseudo-random numbers are used in some parts of this chapter because of their reproducibility. Applying the same seed in the beginning of the algorithm guarantees that every execution uses the same set of random numbers. Therefore, different instances can be tested under the same conditions ensuring unbiased results. The necessary VBA instructions to produce fixed seed and random seed random numbers can be found in Appendix 2.

The sensitivity analysis was conducted by combining several sets of parameters and analyzing the ones that produced better results in an acceptable CPU time. The analysis starts with the mono-objective algorithm for both production strategies and then the bi-objective approach for both strategies as well. Details are presented in the following section.

5.2.1 Mono-Objective Approach: Make-To-Order

The sensitive analysis firstly explores the mutation rate (MR) and the population size (N), followed by the number of generations (G).

Several sets for number of generations, population size and mutation rates were used for the sensitive analysis, as shown in Table 16. All combinations of values in the table will be studied in order to identify which one produces the best performance. In this analysis, efficiency refers to the necessary time to get to the solution and effectiveness indicates the number of optimal solutions reached.

Table 16: Number of mutation rate, population size and generations number used for the sensitivity analysis

Generations number (G)	Population size (N)	Mutation rate (MR)
250	10	0.5
500	30	0.7
750	50	0.9
1000	70	
1250		

Appendix 3 show the results of the grid search using a fixed seed. A population size of 10 individuals never achieves optimal solutions (zero tardiness) and a population of 70 requires a lot of iterations to find the best solutions which most of the time lack on effectiveness. This situation confirms the results of (Abu-Lebdeh & Benekohal, 1999) that smaller populations usually converge in fewer generations than large populations do. Finally, the population size of 50 presents a higher number of optimal solutions found when compared with a population of 30 individuals.

Regarding the mutation rate, the algorithm achieves good results with all values tested (0.5, 0.7 and 0.9). However, for populations of 50 individuals with a mutation rate of 0.5 the algorithm finds optimal solutions in fewer iterations. Therefore, this analysis shows the best tuning results using N = 50 and MR = 0.5.

In order to define the generations number that has the best performance, 50 tests with a random seed were performed for the values 250, 500, 750, 1000 and 1250. As shown in Table 17, the algorithm performs consistently for generation numbers ranging from 750 onwards with a percentage of optimal solutions equal or higher than 80%. A generation number of 1250 don't show any improvement compared to the 1000 generations so two values can be highlighted, 750 and 1000 having optimal solutions percentage of 80 and 86%, respectively. Although G = 1000 has a higher OF_1 average (10 minutes difference), it achieves the best solution in fewer iterations (245,2) and has a higher optimal solution percentage (86%), so it is expected to perform well in the case of a slight increase in the algorithm complexity (ex. more orders). Therefore G = 1000 is the value that best fits the algorithm, for this data set.

Table 17: *MO_BGA MTO statistical analysis*

N = 50; MR = 0.5		G=250	G=500	G=750	G=1000	G=1250
OF1 (min)	Optimal solution	0	0	0	0	0
	Average	63,1	50,9	**19,2**	**29,1**	31
	Maximum	1020	410	330	430	470
	Standard Deviation	168,5	99,7	60,9	92,5	90,7
	% Optimal solutions	64%	64%	**80%**	**86%**	84%
CPU time (s)	Minimum	5,8	12,5	17,2	22,8	29
	Average	7,7	15,9	19,7	25,4	32,1
	Maximum	11,4	18,52	30,25	34,8	43,5
	Standard Deviation	1,3	1,1	3,5	3,1	2,2
Iteration of best OF1 found	Minimum	95	97	90	85	95
	Average	164,9	215,5	275,5	**245,2**	309,3
	Maximum	250	495	696	969	1039
	Standard Deviation	43,9	94,3	181,1	183,2	273,3

Figure 24 shows the selected parameters (G = 1000; N = 50; MR = 0.5) algorithm performance based on the minimization of tardiness (OF_1) vs number of iterations. The OF_1 converges fast in the first 16% of iterations, until iteration 156 (red vertical line in Figure 24), and then the final decrease of the OF_1 value is at iteration 223 (black vertical line in Figure 24). After that, the algorithm keeps on searching for more solutions but there is no room for improvement. These results were obtained in one algorithm execution, requiring 27,68 CPU seconds.

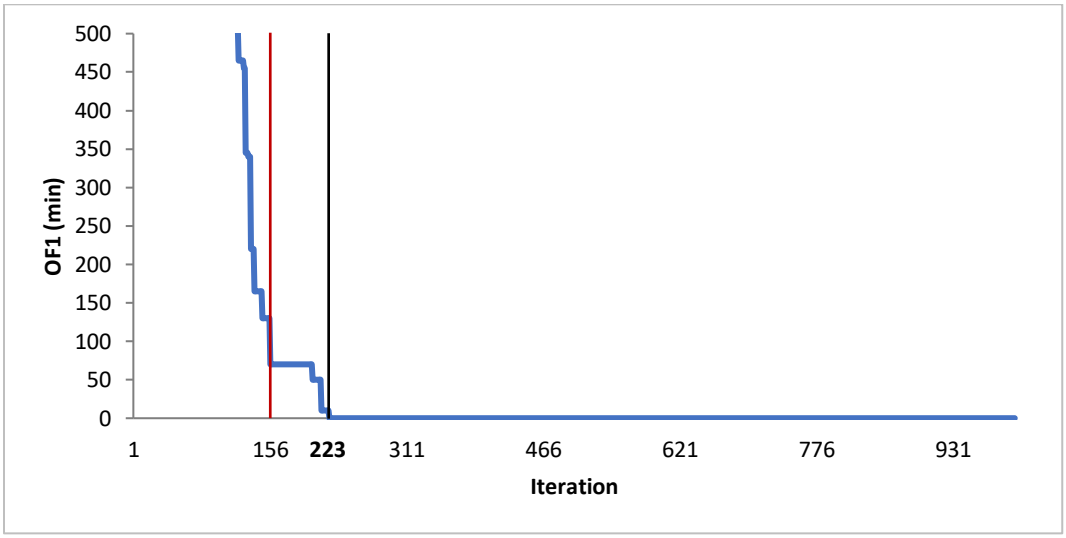

Figure 24: *MO_BGA MTO OF_1 evolution*

For more detailed analysis, 100 executions using random seed were explored, and the frequency distribution is presented in Figure 25 (using G = 1000; N = 50; MR = 0.5). The algorithm can detect the zero tardiness solutions (OF value) in 80 executions and showed an average delay of 28 minutes. However, in some cases it could not find an optimal value and one delay took values higher than a working shift, 560 minutes, suggesting additional tuning.

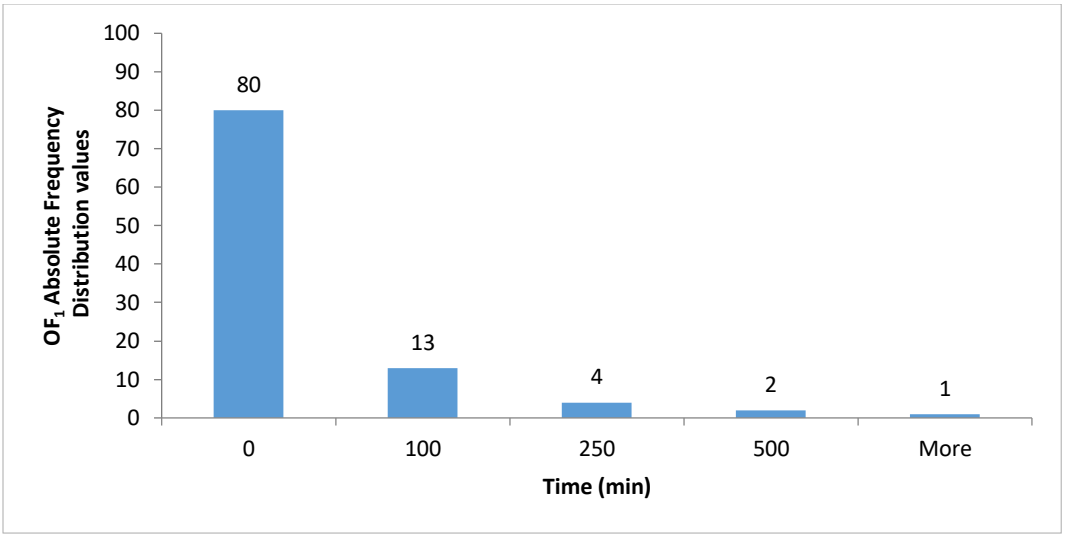

Figure 25: *Absolute frequency distribution of OF_1 optimal value*

5.2.2 Mono-Objective Approach: Make-To-Stock

The MTS strategy considers the stock levels of the products which makes the data input more complex but reduces the algorithm complexity. The complexity is decreased because due dates only become relevant for products in shortage and actual orders are turned into a smaller number of production orders. From the 30 actual orders from the case study (appendix 1), only 22 orders need to be produced, as can be seen in appendix 4.

The sensitivity analysis follows the same methodology as the MTO. Several sets of parameter combinations will be tested in order to discover which combination produces the best results within an acceptable CPU time. Five values of generations number (250, 500, 750, 1000, 1250), four population sizes (10, 30, 50, 70) and three mutation rates (0.5, 0.7, 0.9) were carried out.

Appendix 3 show the results of the grid search using the values mentioned before. The algorithm achieved optimal solutions for all parameter sets expect the ones grouping a population size of 10 and a mutation rate of 0.5. As can be seen, it was much easier to find optimal solutions in this strategy. The maximum complexity that the algorithm can face resembles the MTO strategy. Therefore, the same tuning parameter set were assumed for the population size and mutation rate, 50 and 0.5 respectively.

In order to achieve the most satisfactory number of generations, 50 tests with a random seed were performed for each generations number of 250, 500, 750, 1000 and 1250. As shown in Table 18 the algorithm achieves consistent results for generations number ranging from 750 onwards with percentage of optimal solutions equal or higher than 90%. The G = 1000 shows the lowest average of OF_1 (17,4 minutes) and requires the lowest number of iterations to find the best solution (98,9 iterations). In terms of

CPU time, as the generations number increase the average execution time rises 2 seconds between consecutive values which is not very significant. Therefore, the set of chosen parameters are *G* = 1000, *N* = 50, *MR* = 0.5.

Table 18: *MO_BGA MTS statistical analysis*

N = 50; MR = 0.5		G=250	G=500	G=750	G=1000	G=1250
OF1 (min)	Optimal solution	0	0	0	0	0
	Average	56,4	29,8	20	**17,4**	20,4
	Maximum	740	250	220	250	220
	Standard Deviation	134,8	74,9	60,7	59,9	61,9
	% Optimal solutions	80%	86%	90%	92%	90%
CPU time (s)	Minimum	1,7	3,5	5,4	7,2	9,1
	Average	1,9	4,2	6,4	8,6	10,5
	Maximum	2,9	7,7	10,3	13	16,4
	Standard Deviation	0,2	1,1	1,1	1,6	2,1
Iteration of best OF1 found	Minimum	23	23	23	22	23
	Average	55	105,8	146,3	**98,9**	156
	Maximum	177	408	621	415	1195
	Standard Deviation	32,7	107,9	177,2	107,9	278,5

Figure 26 show the algorithm evolution (in one execution) with a parameter set comprehending 1000 generations, a population of 50 individuals and a mutation rate of 50%. The results were obtained in 7.53 seconds of CPU time. The optimal solution is found after the algorithm has visited 81 solutions (black vertical line in Figure 26). The fact that this solution was found in the first 8% of iterations shows that the algorithm has a considerable generations margin to handle more complex input data that can happen if there are shortages in the stock levels of more products.

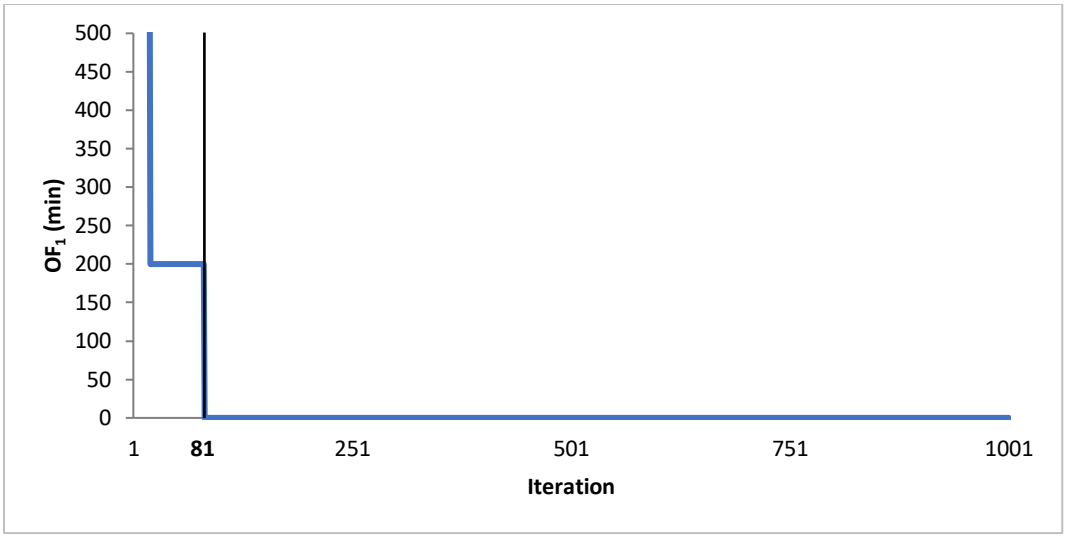

Figure 26: *MO_BGA MTS OF_1 evolution*

Random seed trials were used to evaluate the performance of the algorithm with the selected parameters set (*G* = 1000; *N* = 50; *MR* = 0.5) and after 100 executions the frequency distribution is indicated in Figure 27. The algorithm reaches the optimal value 96% of the executions and having a satisfactory average delay of 8,8 minutes. All the non-zero delay solutions finished with a 220 minutes delay meaning the algorithm got stuck in the same sub-optimal solution. Given that results proven to be acceptable, no further tuning will be addressed.

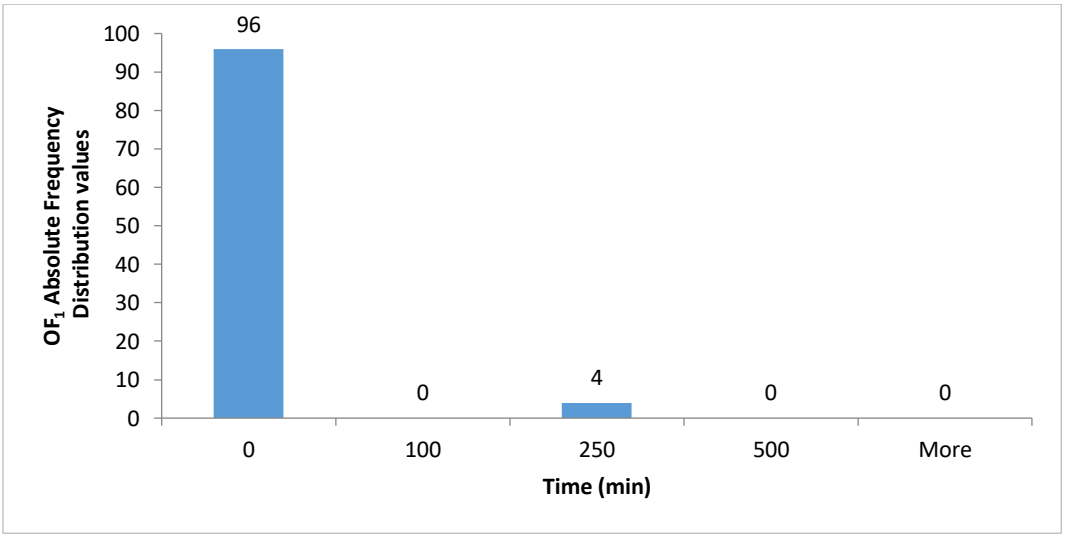

Figure 27: *Absolute frequency distribution of OF_1 optimal value*

Comparing this algorithm with the MTO strategy, the effect of the complexity becomes clear. The MTS strategy can find optimal solutions in fewer iterations and requiring less CPU time. The developed algorithm produced optimal solutions more than 90% of the executions, presenting promising results.

In the following section the results for the Bi-Objective approaches are explored.

5.2.3 Bi-Objective Approach: Make-To-Order

This section addresses the bi-objective approach where the make-to-order operational strategy is explored. The bi-objective differs from the mono-objective in procedures II and V (Figure 20) when the chromosomes fitness is computed. Both tardiness (OF_1) and makespan (OF_2) are considered for the fitness but until the optimal tardiness is not achieved a higher importance is assigned to OF_1. This mechanism is attained using a new lower bound (L) parameter which fixes the minimum weight that OF_1 can have. Although weights are randomly chosen, tardiness weight can never be lower than L. When the optimal tardiness is found, no more weight restrictions are applied.

The tuning of Lower bound parameter uses the parameters G = 1000, N = 50 and MR = 0.5, already tuned in for the MO_BGA, and explores the behavior of 0, 0.2, 0.4, 0.6, 0.8 and 1 values. For each lower bound value 50 trials with random seed were executed, and its statistical analysis are shown in Table 19.

In this case, contrary to what was shown in Tables 17 and 18, the CPU time is not relevant since the lower bound does not affect the execution time. For that reason, the studied indicators are the final OF_1 and OF_2 values and the iterations in which the best solutions were found for both OF.

The L = 1 was the one showing the worst performance with the highest makespan average (1667,4 min). In this case, the algorithm was giving maximum importance to OF_1 so that until the moment the a zero-tardiness solution was found, the weights were always 1 and 0 for OF_1 and OF_2, respectively.

The L = 0.6 displayed the lowest value for OF_1 average (8,6 minutes) and for OF_1 standard deviation (28,2) representing a great behavior of the tardiness function. In terms of makespan it also reflected the lowest OF_2 standard deviation (112,8) meaning the data points tend to be close to the average (1510,2 minutes), which is an acceptable value. Therefore, the chosen parameters are G = 1000, N = 50, MR = 0.5 and L = 0.6.

Table 19: *BO_BGA MTO statistical analysis*

G = 1000; N = 50; MR = 0.5		L = 0	L = 0.2	L = 0.4	L = 0.6	L = 0.8	L = 1
OF1 (min)	Optimal solution	0	0	0	0	0	0
	Average	11,7	16,25	14	**8,6**	11,25	14,8
	Maximum	310	172,5	250	160	180	480
	Standard Deviation	54,2	43,3	49,3	**28,2**	31,4	69,7
	% Optimal solutions	88%	82%	82%	84%	82%	90%
OF2 (min)	Minimum	1320	1315	1283,5	**1320**	1305	1310
	Average	1497,1	1522,4	1501,5	**1510,2**	1501,1	**1667,4**
	Maximum	2190	2160	1972,5	1800	1870	5265
	Standard Deviation	156,1	170	139,6	**112,8**	128,77	642,3
Iteration of best OF1 found	Minimum	78	81	68	61	66	71
	Average	156,5	208	144	161,9	194,9	158,8
	Maximum	529	917	419	742	924	600
	Standard Deviation	85,2	177,7	82,7	132	183,4	115,1
Iteration of best OF2 found	Minimum	65	69	69	54	73	57
	Average	343,8	405,8	378,2	403,5	380,6	404,4
	Maximum	967	967	989	944	972	994
	Standard Deviation	243,2	311,9	290,7	262,1	258,4	293,5

The evolution of OF_1 and OF_2 values for one algorithm execution with the tuned parameter values (G=1000, N=50, MR=0.5 and L=0.6) is presented in Figure 28. The 0 tardiness was achieved in the iteration 142 (black vertical line in Figure 28), the time when the lower bound parameter stopped being used and OF_1 was no longer prioritized. From that moment, the algorithm had more freedom to minimize makespan and results show a 22% reduction (370 minutes, approx. 6 hours) in OF_2, since it went from 1690 to 1320 minutes. This reduction highlights the importance of the second objective function for optimizing the production scheduling at the factory since all machines will be available for new production orders 6 hours earlier with no delays.

In the first 142 iterations OF_2 increased several times meaning that the algorithm accepted lower quality solutions in terms of makespan. In the mono-objective algorithm this situation did not occur because it was dealing with only one objective function and the elitism feature prevented to jump to worse solutions. However, the bi-objective formulation applies random weights to both OF every iteration, hence one solution

may have a good fitness in one iteration but in the next one it is discarded because a different importance was considered for OF.

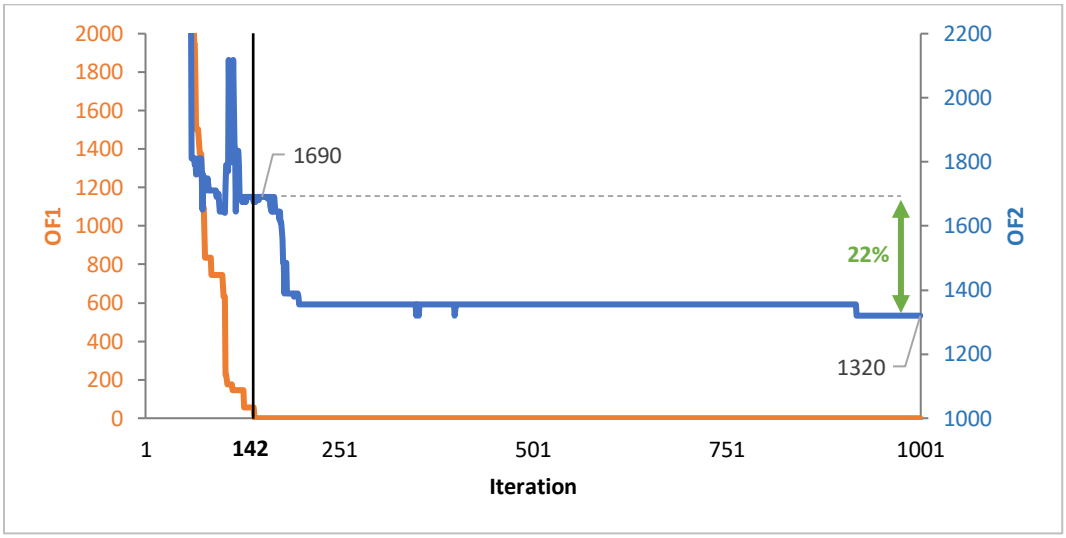

Figure 28: *BO_BGA MTO tardiness (OF_1) and makespan (OF_2) evolution*

The most satisfying production plan for the minimization of the tardiness OF_1 and makespan OF_2 is presented in Figure 29 (using the parameters G = 1000, N = 50, MR = 0.5 and L = 0.6). It achieved 0 tardiness and the lowest makespan (1320 minutes). Each machine has their own sequence of orders to produce resulting in different completion times of the last order and total setup times.

This production plan produces a total of 2850 minutes (approx. 48 hours) of setup time that is shared between 8 machines. Since reduced setup times help to minimize OF_2, the algorithm tries to sequence orders of the same product one after the other (see products 5, 12, 14 and 20) if the corresponding due dates allow. These situations lead to a 0 setup time and are highlighted in Figure 29 with a blue fill.

Machine Mi (i=1,…8)	Operations sequence							Completion time of the last order of Mi	Total setup times of Mi
	1	2	3	4	5	6	7		
M1	*O19, P11*	*O21, P13*	*O24, P12*	*O16, P12*				1320	20
M2	*O14, P10*	*O11, P7*	*O15, P10*					1180	20
M3	*O8, P6*	*O12, P8*	*O13, P9*	*O2, P3*				1315	610
M4	*O9, P2*	*O4, P1*	*O1, P2*	*O3, P4*	*O10, P5*	*O5, P5*	*O7, P4*	1170	340
M5	*O18, P14*	*O22, P14*	*O20, P15*	*O17, P13*	*O25, P15*	*O6, P3*		1305	330
M6	*O30, P20*	*O29, P20*	*O38, P21*	*O28, P19*	*O37, P27*			1180	210
M7	*O31, P22*	*O35, P25*	*O40, P24*	*O36, P26*	*O39, P23*			995	800
M8	*O34, P28*	*O33, P29*	*O32, P30*	*O23, P16*	*O26, P17*	*O27, P18*		1215	520
OF1	0								
OF2	1320								

Figure 29: *The most satisfying production schedule for BO$_B$GA MTO.*

The bi-objective algorithm described in this section presented solutions with a good performance since they required a low CPU time and achieved optimal values for tardiness and good values for makespan. The following section presents the results obtained using a Make-To-Stock strategy.

5.2.4 Bi-Objective Approach: Make-To-Stock

The Make-To-Stock strategy allows the algorithm to consider stock levels. This feature reduces the complexity of the problem based on the information stock of products available to fulfill the customer needs avoiding another order in production.

The first tests performed of MTS production strategy in the mono-objective formulation showed that CPU time reduced significantly due to the lower number of orders (22 orders instead of 40) and the decrease of orders with due dates (due dates are just applied for products in shortage). From the 22 production orders in appendix 4, only 8 have due dates.

The first three parameters used in this section resulted from the MO$_B$GA MTS algorithm (G = 1000, N = 50 and MR = 0.5). For the selection of the lower bound parameter, 50 executions were carried out using a random seed for the following values, 0, 0.2, 0.4, 0.6, 0.8 and 1, shown in Table 20.

From a preliminary analysis, all lower bound values achieved at least 90% of optimal solutions for tardiness. Regarding the OF_2 average values, from the L = 0.2 onwards values increase more than 590 minutes (approx. 10 hours) leading to non-satisfactory makespan solutions. Both values, L = 0 and L = 0.2, the

have similar performances in terms of OF_1 and OF_2 values, but the latter achieves the optimal tardiness in fewer iterations (70,9) and shows a lower standard deviation (70,8). Therefore, the chosen parameters are G = 1000, N = 50, MR = 50 and L = 0.2.

Table 20: *BO_BGA MTS statistical analysis*

G = 1000; N = 50; MR = 0.5		L = 0	L = 0.2	L = 0.4	L = 0.6	L = 0.8	L = 1
OF1 (min)	Optimal solution	0	0	0	0	0	0
	Average	12,4	12,4	8	8	20	4
	Maximum	220	220	200	200	200	200
	Standard Deviation	49,6	49,6	39,5	39,5	60,6	28,3
	% Optimal solutions	94%	94%	96%	96%	90%	98%
OF2 (min)	Minimum	1430	1520	1520	1530	1530	1497,5
	Average	1922,5	1939	2534,7	3723,5	3093,1	3438,2
	Maximum	13560	13560	20075	20360	20375	72275
	Standard Deviation	1685,82	1681,1	3429,1	5310,9	4720	10281
Iteration of best OF1 found	Minimum	18	20	19	18	17	15
	Average	73,4	**70,9**	109,2	119,5	98,4	86,2
	Maximum	600	305	909	824	543	749
	Standard Deviation	107,6	**70,8**	189,4	176,6	130,1	127,5
Iteration of best OF2 found	Minimum	42	23	30	34	20	65
	Average	386,5	382,9	360	313,9	386,2	439,8
	Maximum	994	980	952	977	946	979
	Standard Deviation	294,2	300,1	286,9	265,3	280,9	269,4

The evolution of OF_1 and OF_2 values for one algorithm execution with the selected parameter values (G = 1000, N = 50, MR = 0.5 and L = 0.2) shows a reduction of 20% (422,5 minutes, approx. 7 hours) in the makespan objective function from the point the algorithm reached the first zero tardiness solution in iteration 33 (black vertical line in Figure 30). In this reduction the makespan value went from 2130 to 1707,5 minutes. This reduction optimized OF_2 from 35,5 hours to 28,5 hours. This solution was obtained within approx. 11,56 seconds and the results are displayed in Figure 30.

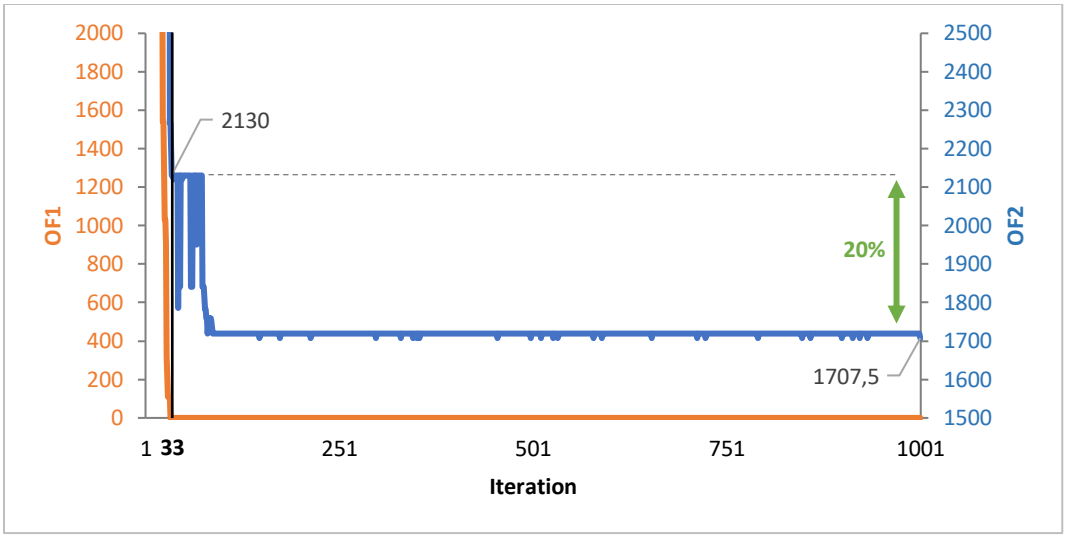

Figure 30: *BO_BGA MTS tardiness (OF_1) and makespan (OF_2) evolution*

For this algorithm the initials orders are aggregated by product meaning that all production orders consist on different products. Figure 31 shows the production plan of the previous algorithm execution and it produced a total of 1770 minutes of machine setup distributed between all 8 machines (using the parameters *G* = 1000, *N* = 50, *MR* = 50 and *L* = 0.2). For instance, machine 5 produced a total of 320 minutes of setup time due to one mould changeover (300 minutes) between products 15 and 2 and two dye changeovers (2x10 minutes) between products 11, 14 and 15. For this machine, the fourth production order scheduled (O2, P2) was completed after 1707,5 minutes (approx. 29 hours) representing the final OF_2 value since all other machines presented a lower completion time of the last order.

Machine Mi (i=1,…8)	**Operations sequence**							**Completion time of the last order of Mi**	**Total setup times of Mi**
	1	**2**	**3**	**4**	**5**	**6**	**7**		
M1	*O12, P13*	*O11, P12*						1550	10
M2	*O6, P7*	*O7, P8*	*O1, P1*					1610	310
M3	*O8, P9*	*O5, P5*	*O3, P3*					1260	600
M4	*O4, P4*	*O9, P10*						1720	300
M5	*O10, P11*	*O13, P14*	*O14, P15*	*O2, P2*				1707,5	320
M6	*O18, P20*	*O19, P21*	*O17, P19*					1530	110
M7	*O20, P22*	*O21, P23*	*O22, P25*					375	110
M8	*O15, P17*	*O16, P18*						580	10
OF1	0								
OF2	1707,5								

Figure 31: *BO_BGA MTS production plan.*

5.3 Results Analysis Conclusions

In this study, two genetic algorithm approaches were explored, mono-objective and bi-objective. Each approach was tested under two different production strategies, Make-To-Order and Make-To-Stock to solve the case study instance. The MTO showed higher complexity requiring more iterations to find optimal solutions and more CPU time as well. The different complexities led to different outputs and performances.

For the mono-objective formulation, the MTS approach demanded less computational time to obtain optimal solutions with an average of 8,6 seconds compared to the 29,1 seconds on average in MTO for the same goal. Also, the average number of iterations to find the optimum went from an average of 245,2 to 98,9 for MTO and MTS, respectively. The parameter set resulting on the best performance for both production strategies are G = 1000, N = 50 and MR = 0.5.

For the bi-objective formulation, good results were obtained in terms of minimizing both tardiness and makespan. A zero-tardiness solution was found, the algorithm was able to reduce the makespan in 22% for MTO and 20% for MTS. The set of parameters that produced the best results are:

- Make-To-Order: G = 1000, N = 50, MR = 0.5 and L = 0.6
- Make-To-Stock: G = 1000, N = 50, MR = 0.5 and L = 0.2

The bi-objective MTS showed better performance dealing with a L = 0.2 than with L = 0.6 of the MTO approach. This is explained by the lower complexity caused by the stock introduction making it easier for the algorithm to achieve optimal tardiness solution without enhancing OF_1's importance.

After the analysis done to the algorithm in order to tune parameters, the results obtained produced quality and reliable schedules and guaranteeing no production delays at least in 80% of the cases. The MTS analysis suggests since the factory has a lot of stock for each product because the actual orders do not trigger enough production orders to enable exploring the full potential of the algorithm. Nevertheless, the algorithm is a great value-added to the factory's operations.

The production planning in the factory is done by a team of four employees during several hours and is a critical task. The results showed with this algorithm require a few seconds and only one person while the production plan develop allows the machine operators to detect production delays and assess their impact in the production sequence.

The goal of this dissertation was to develop a robust algorithm that fitted in the factory operations in the existing circumstances requiring no extra developments. This was achieved because applying this algorithm to a scheduling problem requires no effort. The algorithm not only generates a zero-tardiness schedule but also optimizes the makespan which enables the factory became more efficient in their daily operations and consequently save money and resources on the long term.

The next chapter tests different input data with different complexities in the already tuned algorithm.

6. Comprehensive Experiments

This chapter aims to stress the developed algorithm, BO_BGA, with instances of different characteristics and problem sizes. This set of instances will allow to verify how versatile the algorithm is under severe situations. Thus, section 6.1 characterizes five instances based on the real case study, and real input data. In section 6.2, randomly generated instances are proposed and tested with several machines and orders to analyze the algorithm performance. Finally, in section 6.3 some conclusions are drawn.

6.1 Based on real Instances

In the previous chapter, it was possible to conclude that with real factory input data, the algorithm reaches acceptable solutions in a reasonable amount a time. However, it would be interesting to know how the algorithm reacts in a more complex environment. For this reason, five instances were created in order to analyze its performance with increase complexity. These instances vary in terms of orders' quantity, due dates, number of machines and setup times. The same five instances were tested with a symmetric setup time matrix like the one shown in appendix 1 and with an asymmetric setup time matrix presented in appendix 6. Instances are described in Table 21.

Table 21: *Instances characteristics.*

Instance 1	Instance 2
Same 8 machines of the case study and orders data in appendix 5: • Number of orders: 70 • Orders quantities: vary between 500 and 4000 • Orders due dates: vary between 500 and 3000 minutes Compared to the real case data (appendix 1), 30 more orders are received with lower quantities and keeping the same range of due date values.	Same 8 machines of the case study and orders data in appendix 5: • Number of orders: 30 • Orders quantities: vary between 1000 and 20000 • Orders due dates: vary between 300 and 1500 minutes The orders from 8 to 12 have the shortest due dates (300). Compared to the real case data, this one has 10 fewer orders but with tighter due dates.
Instance 3	**Instance 4**
Same 8 machines of the case study and orders data in appendix 5: • Number of orders: 40 • Orders quantities: 5000 to all • Orders due dates: vary between 500 and 1500 minutes Compared to the real case data, here all orders demand the same quantity.	In this instance, one machine from the factory breaks down. Same data of the case study is used (appendix 1) except that machine M5 cannot be used.
Instance 5	
Same 8 machines of the case study and orders data in appendix 5: • Number of orders: 100 • Orders quantities: vary between 1000 and 15000 • Orders due dates: vary between 500 and 4000 minutes Compared to the real case data, 60 more orders are received leading to a much higher complexity.	

Section 6.1.1 tests the five instances previously presented with both symmetric and asymmetric setup time matrixes in the BO_BGA with a Make-to-Stock production strategy.

6.1.1 Bi-Objective Approach: Make-To-Stock

In order to analyze the behavior of the MTS approach, 50 runs with random seed of the same instance were performed, and the results are shown in Table 22. The stock data considered for all instances can be found in appendix 7.

Table 22: *Statistical analysis of the five instances for MTS.*

			Tardiness		Makespan	CPU Time
	Instance	**(Order x Machine)**	**% Optimal values**	**Average (min)**	**Average (min)**	**Average (seconds)**
Symmetric setup time matrix	1	70 x 8	100%	0	1288,7	9,9
	2	30 x 8	0%	461,6	962,2	2,5
	3	40 x 8	98%	6	953,8	2,8
	4	40 x 7	100%	0	2378,8	3,3
	5	100 x 8	100%	0	**2872,4**	13,2
Asymmetric setup time matrix	1	70 x 8	100%	0	1311,8	9,8
	2	30 x 8	0%	372,9	1108	2,5
	3	40 x 8	100%	0	**753,6**	2,8
	4	40 x 7	100%	0	**1536,8**	3
	5	100 x 8	100%	0	3563,9	13,3

According to Table 22, the algorithm achieved a nearly perfect performance in all instances except in the second one. In this instance, the goal was to test how the algorithm deals with tight due dates. However, since MTS aggregates orders of the same products and sets the due date as the shortest for the actual orders in the same product, it may be impossible to deliver everything on time. That is what happens in this case. Even assigning the production orders with the shortest due dates as the first operation of the machines with the lowest production time, a positive tardiness will be invariably reached.

Concerning the setup time matrixes, using a symmetric or an asymmetric matrix can lead to different results. In instances 1 and 2 the differences were not relevant. For instances 3 and 4, lower makespan averages were achieved with an asymmetric setup time matrix, 753.6 and 1536.8 min, respectively. However, for instance 5, the best makespan average result was attained using a symmetric matrix with a value of 2872.4 minutes.

Regarding the complexity of the BO_BGA MTS, it will never be higher than the complexity of the fifth instance tested in this section. The algorithm initial orders are aggregated by product meaning that all production orders consist on different products. Independently of the number of actual orders, there will never be more than 30 production orders.

The next section tests the MTO production strategy which represents a higher complexity due to the fact that orders are not aggregated meaning all of them must be produced individually. Therefore, a higher level of combinations is obtained, increasing the complexity.

6.1.2 Bi-Objective Approach: Make-To-Order

The current section follows the same methodology of the previous one. Hence, 50 runs with a random seed were tested for each instance and the results are presented on Table 22.

Table 23: *Statistical analysis of the five instances for MTO*

			Tardiness		Makespan	CPU Time
	Instance	**(Order x Machine)**	**% Optimal values**	**Average (min)**	**Average (min)**	**Average (seconds)**
Symmetric setup time matrix	1	70 x 8	98%	0,2	1160,1	143,1
	2	30 x 8	86%	19,4	1018,2	17,1
	3	40 x 8	48%	**46**	1040,3	32,1
	4	40 x 7	**48%**	130,3	1950	34,9
	5	100 x 8	0%	718,4	2892,5	364,3
Asymmetric setup time matrix	1	70 x 8	**96%**	9,2	1133,4	148,8
	2	30 x 8	78%	27,5	1123,2	19,4
	3	40 x 8	84%	3,1	837,9	34,9
	4	40 x 7	0%	164,6	1792	31,6
	5	100 x 8	16%	439	3031,9	426,7

Regarding the results of the first instance, the algorithm managed to achieve a nearly perfect performance. The fact that it was dealing with 70 orders required a higher CPU time than the real case with 40 orders. However, for both setup time matrixes, at least 96% of the trials the algorithm found optimal tardiness solutions (Table 23). This first instance consists on a high number of orders with low quantities and there was no problem solving it. Figure 32 shows one execution of the first instance with a symmetric setup time matrix requiring 145,4 CPU seconds. The zero tardiness was attained after 180 iterations (black vertical line in Figure 32) and from that moment the makespan went from 1940 to 1130 minutes.

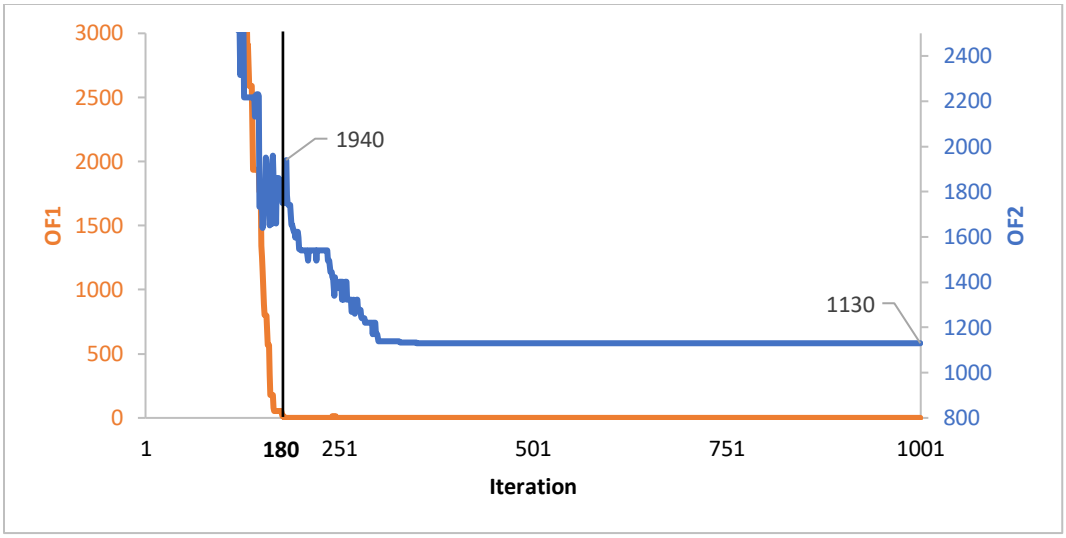

Figure 32: *BO_BGA MTS tardiness (OF_1) and makespan (OF_2) evolution for the first instance*

The second instance aimed to find out if 30 orders with tight due dates were an obstacle for the algorithm. According to Table 23, 86% of the solutions achieved the optimal value for tardiness for a symmetric setup time matrix and 78% for an asymmetric matrix. Figure 33 presents the production scheduling, in a Gantt chart with a symmetric setup times, required 19,7 CPU seconds to run. The orders 8, 9, 10, 11 and 12 that were the ones having the shortest due date (300) were efficiently assigned as the first orders to be produced by the machines (orders with blue fill). The production ended up with a makespan of 1008 minutes.

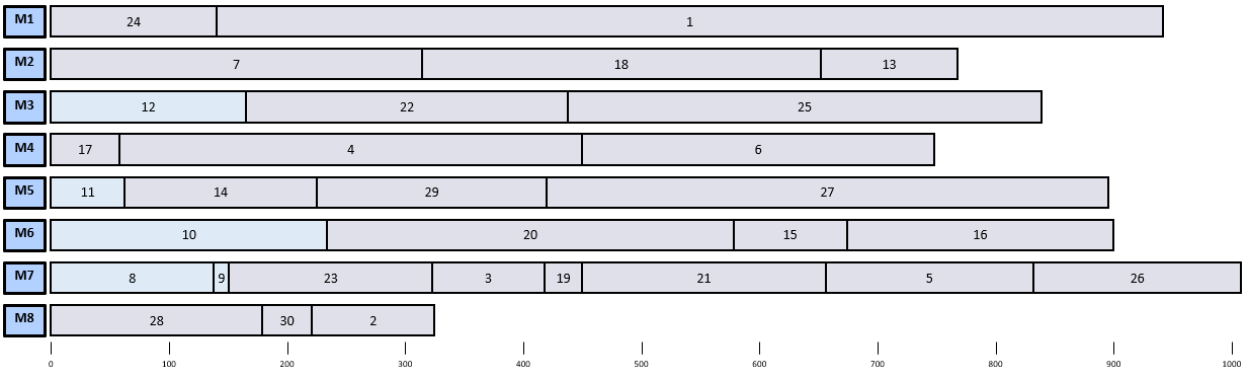

Figure 33: *Production plan for the second instance*

Concerning the third instance, only 48% of the solutions reached an optimal tardiness with a symmetric setup time matrix (Table 22). Nevertheless, the average tardiness was 46 minutes which is an acceptable value. Regarding the asymmetric setup time matrix, it outperformed the symmetric matrix with a percentage of optimal values of 84% with an insignificant tardiness average of 3.1 minutes.

The fourth instance had a percentage of optimal values of 48% for a symmetric setup time matrix (Table 23). Without one machine the algorithm struggles to find zero tardiness solutions and after 50 executions the average tardiness was 130,3 minutes. Figure 34 shows a zero-tardiness solution for this instance. It took 39,9 CPU seconds and finished with a makespan of 2000 minutes. Regarding the asymmetric setup time matrix, no optimal tardiness results were achieved.

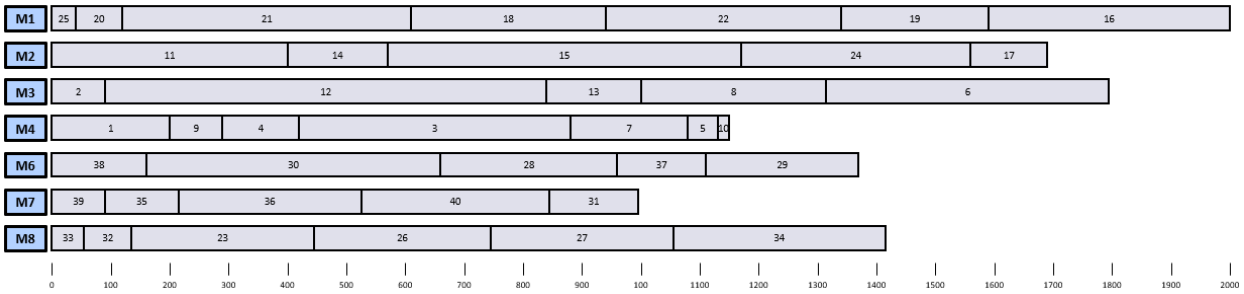

Figure 34: *Production plan for the fourth instance*

The fifth instance aims to analyze how the algorithm deals with complex input data. From all 50 executions, the asymmetric setup matrix and the symmetric showed 16% and 0% of optimal tardiness solutions, respectively (Table 23). In order to compare the schedule plans of a MTO and a MTS for the fifth instance, Figure 35 shows two Gantt charts with a 0 tardiness solution for both production strategies with an asymmetric setup time matrix. The solution in Figure 35 (a) using MTO required 426,7 CPU seconds to run and finished with a makespan of 2944 minutes. Figure 35 (b) using MTS took 12,8 CPU seconds and achieve a makespan of 2634 minutes. The 100 actual orders of instance 5 turned into 29 production orders for the MTS strategy (there are no orders for product 2) leading to a significant complexity reduction compared to the MTO. Also, the production orders of MTS are much longer than MTO's because orders are aggregated by product.

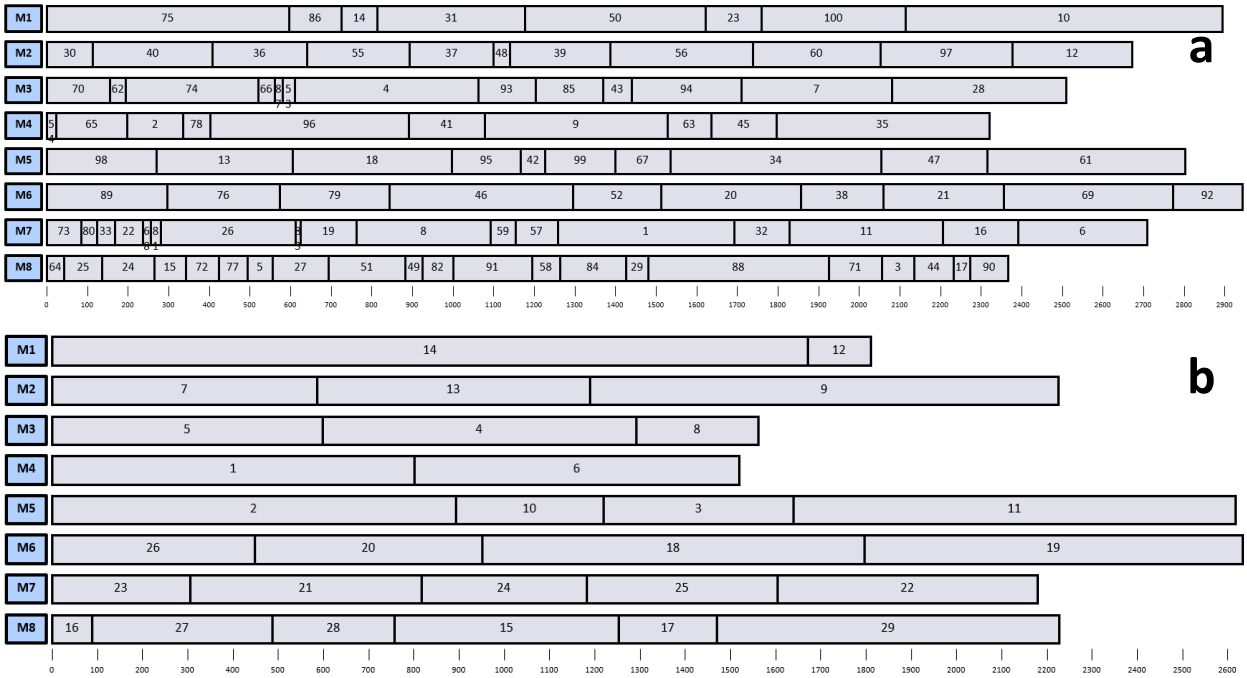

Figure 35: *Gantt charts with zero tardiness for fifth instance: MTO (**a**) and MTS (**b**).*

6.2 Randomly Generated Instances

This section analyzes how the algorithm behaves when different complexities of data are considered. In order to achieve different complexities, several sets of orders and machines were defined. The goal is to identify until which point the algorithm performs consistently satisfactory results.

Based on the nowadays trend, mass customization, and the MTO production strategy following higher complexity and challenge, this section will cope only with the MTO.

During all executions the number of available products is 30. Therefore, an order is randomly generated under the following criteria:

- The product $\in \mathbb{N}$, [1, 30];
- The quantity $\in \mathbb{N}$, [500, 20000];
- The due date $\in \mathbb{N}$, [500, 4000];

Regarding the machines, two layouts are used. One just produces bottoms (the first 15 products), and the other just produces covers (the last 15 products). In the case that the number of machines is even, the same number of machines producing bottoms and covers are used. In the case there is an odd number of machines, there is one more machine producing covers than bottoms.

Table 24 shows statistical results for different *O x M* (Order x Machine) instances. For each *O x M* instance, 50 executions were made always with a new set of random generated orders. Some conclusions can be drawn from Table 24, especially when the number of orders for the same number of machines reaches a critical instance. A critical instance can be defined as the moment until the algorithm performs consistently and from this point on, the percentage of optimal values of tardiness, the average tardiness and the average makespan suffer a sudden exponential growth. These situations can be found in the following problem instances:

- Two machines, between 5 and 10 orders. The average tardiness increased from 17,3 to 712,4 minutes, meaning that for two machines the maximum number of orders is 5. The critical instance is the number 1.
- Five machines, between 30 and 35 orders. From 10 to 30 orders the percentages of optimal tardiness values were slightly decreasing. However, between instances 7 and 8 the optimum tardiness solutions goes from 62% to 38% which is a significant reduction. The critical instance in this case is the number 7.
- Eight machines, between 50 and 60 orders. Here the average tardiness increases approx. 440% (50,1 to 220,2 minutes) setting the critical instance to number 12.
- Eleven machines, between 70 and 80 orders. From 40 to 70 orders the lower percentage of optimum tardiness recorded was 74% but for 80 orders the value decreases to 32%. Also, the average tardiness suffers an increase of approx. 404% (38,1 to 154,1 minutes). Therefore, the critical instance is set to number 17.

An important remark regarding the statistical analysis is that when a problem instance presents percentages of optimal tardiness between 60 and 70, it does not necessarily mean that the algorithm failed to find satisfactory solutions. Since the orders are randomly generated, there is the chance of creating a set of orders that are impossible to fulfil on time.

Table 24 also suggests that as the number of orders and machines increase, the CPU time does not increase in an exponential way. Due to this reason, even if the input data size is larger than the one tested in this section, the algorithm will not require an impractical amount of time to run.

Table 24: *Different problem instances statistical analysis*

Problem instance	(Order x Machine)	Tardiness		Makespan	CPU Time
		% Optimal values	Average (min)	Average (min)	Average (seconds)
1	5 x 2	94%	**17,3**	1371,7	1,4
2	10 x 2	40%	**712,4**	2932,1	2,9
3	10 x 5	100%	0	1014,3	2,9
4	15 x 5	92%	8,6	1264,8	4,9
5	20 x 5	84%	32,4	1717,5	8
6	25 x 5	76%	46,6	1888,6	11,7
7	30 x 5	**62%**	196,3	2197,9	17,7
8	35 x 5	**38%**	292,4	2607,8	26,4
9	20 x 8	100%	0	388	8,3
10	30 x 8	88%	10,3	1382,4	18,9
11	40 x 8	82%	54,2	1743,7	36,5
12	50 x 8	70%	**50,1**	2198,9	66,1
13	60 x 8	46%	**220,2**	2550,2	93
14	40 x 11	100%	0	1291,8	38,2
15	50 x 11	92%	14,3	1592,6	55,61
16	60 x 11	86%	25,7	1905,6	90,2
17	70 x 11	**74%**	38,1	2259,4	138,3
18	80 x 11	**32%**	154,1	2593,1	201,8

All critical instances found in Table 24 (with a blue fill) are represented in Figure 36. According to the graph of Figure 36, any combination of orders and machines below the line means that the algorithm can produce satisfactory results. The dotted line in the graph shows the trendline of the data. Excel uses the method of least squares to find a line that best fits the points. The R-squared value equals 0.9979, which is a good fit. The closer to 1, the better the line fits the data.

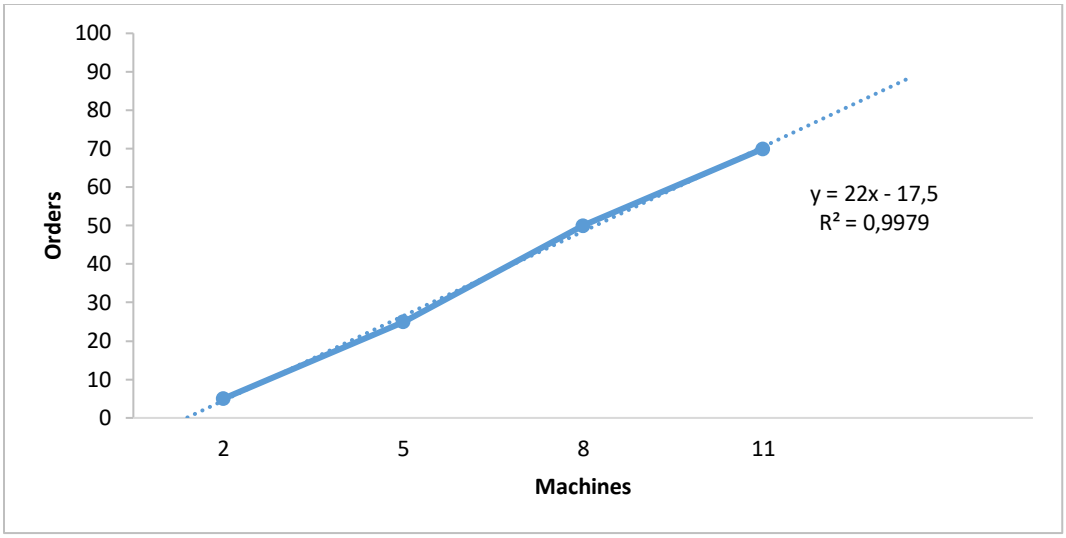

Figure 36: *Critical values for Machine-Orders instances*

The results achieved in this section (Figure 36) help to define the productive capacity (maximum possible output) of the factory under consideration. The random orders were generated with ranges of values based in real situations and machines had the same characteristics of the case study's machines. Therefore, it is easy to calculate the productive capacity of the factory since it corresponds to the critical instances identified previously. For a certain of number machines, the trendline of Figure 36 shows the maximum number of orders that the factory can satisfy consistently. If that maximum number is surpassed, the factory's performance may be jeopardized.

6.3 Chapter Conclusions

In this chapter, a set of data based in real and random instances with increasingly complexities were tested for both production strategies, MTO and MTS. The goal was to stress the algorithm and verify how it behaves under five different instances. These instances had different orders number, quantities, machines number and due dates.

Regarding the MTS strategy, the second instance was the only one failing to achieve satisfactory results, meaning no solutions with optimal tardiness were found. This second instance was characterized by having 30 orders with very tight due dates. It was possible to conclude that aggregating the orders of the same product can be an advantage and a disadvantage at the same time. On one hand, aggregating orders allows to reduce production setup times and actual orders are turned into a smaller number of production orders and hence, lower complexity. On the other hand, when due dates are tight it may be impossible to deliver everything on time because the algorithm needs to produce the total amount of the same product in

one sitting. This is the situation that happened with the second instance with MTO. No optimum tardiness was attained because it was impossible.

Concerning the MTO results, the first two instances were solved efficiently with at least 78% of the results with an optimum tardiness found for both symmetric and asymmetric setup time matrix. The last three instances largely depend on the setup matrix used. While the third and fifth instance showed better results with an asymmetric setup time, the fourth instance had a superior performance with a symmetric matrix. Finally, dealing with a machine breakdown or 100 orders, instance 4 and 5 respectively, were the most challenging ones for the algorithm.

In this chapter an analysis with increasing *O x M* (Order x Machine) instances was also performed. Four critical instances were found and, consequently, the factory's productive capacity was defined. The productive capacity consists on a function that defines the maximum number of orders that the factory can fulfil with a certain number of machines. During situations with a significant variance in the orders size, the consequences of a machine acquisition or sale can be predicted using that function, facilitating the decision making.

7. Conclusions and Future Work

This dissertation is about Logoplaste Santa Iria, which is a factory that produces plastic containers and operates for FIMA. The market of the plastic industry is now very saturated and due to high competitiveness, the final prices that are applied to the consumer are the key factor for measuring companies' success. Currently, the production scheduling in LSI is done manually by a group of employees and the factory manager without the support of any IT software. LSI needs to have a reliable and fast production scheduling tool, and this is the motivation behind this work.

The goal of this master thesis was to develop an algorithm that would help the company to improve its production scheduling stage. The algorithm was based on a genetic algorithm with two approaches, mono-objective and bi-objective. The mono-objective approach minimizes the tardiness while the bi-objective approach seeks to minimize the tardiness and makespan (giving more importance to the tardiness). Each approach was tested for two different production strategies, make-to-order (MTO) and make-to-stock (MTS). MTO considers no inventory while MTS deals with stock levels of finished products. When orders arrive, the MTS production strategy analyzes the stocks and decides which quantity of each product needs to be produced.

The case study data was used to tune the algorithm parameters and after a sensitivity analysis, the best algorithm results were achieved. The mono and bi-objective approaches using both production strategies showed at least 84% of the solutions achieving the optimal tardiness and no more than 29,1 minutes of average delay. Despite the good performance for both MTO and MTS, the MTS production strategy presented better results due to its reduced complexity. In MTS, not only the orders are aggregated by product, but it also deals with stock levels meaning that sometimes there is enough quantity in the inventory to fulfil an order, requiring no production.

Five additional instances were created to stress the algorithm and investigate how it reacts in different situations. The results concluded that for very tight due dates the MTS may not be the best option since it aggregates all orders of the same product. Also, the results of the algorithms largely depend on the setup time matrix used (symmetric or asymmetric).

Obviously, the algorithm should be tested in the factory. It is recommended the involvement of LSI's production planner on this process. The use of a set of typical LSI weekly orders would allow the comparison of production schedules created by the company's decision makers with the corresponding solutions generated by the proposed algorithm. It is expected that some adjustments to the algorithm may be necessary during these tests, but these actions were beyond the current dissertation's scope and were not planned.

From the end-user's perspective, this dissertation produces the planned results since the developed algorithms for the production planning simplify the methodologies currently used at the factory. The factory

has now a new tool available to support its scheduling tasks. After investigating the developed work, the following conclusions were drawn:

- Metaheuristic approaches are well suited to solve scheduling problems with the same complexity as the case study in short time.
- Make-To-Stock production strategy is the one that best fits the case study. Although MTS has the handicap of having high inventory costs, it reduces the problem complexity significantly and allows to handle the uncertainty of the market properly.
- The parameters of each algorithm largely depend on the complexity of the input data.

The idea of efficiency and productivity can be found in all industry sectors. Scheduling decisions have a huge impact in these concepts and are increasingly important for companies. Although the work developed in this master thesis aimed to solve a specific case study, it can be easily adapted to fit different kinds of optimization problems.

The future work would involve the development of a new methodology to solve the same case study to serve as a benchmark for comparison. It should also perform a comparative analysis with the metaheuristic developed in this dissertation. Hence, more accurate results would be provided, and the importance and applicability of metaheuristics would be highlighted.

References

Abdolrazzagh-Nezhad, M., & Abdullah, S. (2017). Job Shop Scheduling: Classification, Constraints and Objective Functions. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, *11*(4), 417–422.

Abraham, A., Nedjah, N., & Mourelle, L. D. M. (2006). Evolutionary computation: From genetic algorithms to genetic programming. *Studies in Computational Intelligence*, *13*, 1–20.

Abu-Lebdeh, G., & Benekohal, R. F. (1999). Convergence Variability and Population Sizing in Micro-Genetic Algorithms. *Computer-Aided Civil and Infrastructure Engineering*, *14*(5), 321–334.

Arisha, A., Young, P., & El Baradie, M. (2001). *Job Shop Scheduling Problem: an Overview*. *International Conference for Flexible Automation and Intelligent Manufacturing (FAIM 01)*.

Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011). Inertia Weight strategies in Particle Swarm Optimization. In *2011 Third World Congress on Nature and Biologically Inspired Computing* (pp. 633–640).

Beheshti, Z., & Shamsuddin, S. M. (2013). *A review of population-based meta-heuristic algorithm*. *International Journal of Advances in Soft Computing and Its Applications* (Vol. 5).

Bergh, F. van den, & Engelbrecht, A. P. (2002). A new locally convergent particle swarm optimiser. In *IEEE International Conference on Systems, Man and Cybernetics* (Vol. 3, p. 6 pp. vol.3).

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-parameter Optimization. *J. Mach. Learn. Res.*, *13*, 281–305.

Bettemir, Ö. H. (2011). Experimental Design for Genetic Algorithm Simulated Annealing for Time Cost Trade-off Problems. *International Journal of Engineering and Applied Sciences*, *3*, 15–26.

Black, P. E. (2019). Fisher-Yates shuffle. Retrieved June 6, 2019, from https://www.nist.gov/dads/HTML/fisherYatesShuffle.html

Boyabatli, O., & Sabuncuoglu, I. (2004). Parameter Selection in Genetic Algorithms. *Journal of Systemics, Cybernetics and Informatics*, *4*(2), 78–83.

Burns, F., & Rooker, J. (1976). Johnson ' s Three-Machine Flow-Shop Conjecture. *Operations Research*, *24(3)*(April 2019), 578–580.

Cruz-Chávez, M. A., Martínez-Rangel, M. G., & Cruz-Rosales, M. H. (2017). Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem. *International Transactions in Operational Research*, *24*(5), 1119–1137.

Dai, M., Tang, D., Giret, A., & Salido, M. A. (2019). Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robotics and Computer-Integrated Manufacturing*, *59*, 143–157.

Deepa, S. N., & Sivanandam, S. N. (2008). *Introduction to Genetic Algorithms*. Springer US.

Dell'amico, M., & Trubian, M. (1993). *Applying tabu search to the job-shop scheduling problem**. *Annals of Operations Research* (Vol. 41).

Dharmapriya, S., Kulathunga, A., & Jayathilake, P. (2014). A Comparative Study of Improved Simulation Annealing On Multi- Depot Vehicle Routing Problem with Time Windows. *Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management*, (Kellehauge 2008), 1849–1858.

Eslami, M., Shareef, H., Khajehzadeh, M., & Mohamed, A. (2012). From the SelectedWorks of Dr. Mahdiyeh Eslami (مهدیه) اسالمی A Survey of the State of the Art in Particle Swarm Optimization. *Research Journal of Applied Sciences, Engineering and Technology*, *4*(9), 1181–1197.

Fisher, R. A., & Yates, F. (1963). Statistical tables for biological, agricultural, and medical research. New York: Hafner Pub. Co.

Gen, M., & Cheng, R. (1997). *Genetic Algorithms and Engineering Design*. New York: John Wiley & Sons.

Genova, K., Kirilov, L., & Guliashki, V. (2015). A Survey of Solving Approaches for Multiple Objective Flexible Job Shop Scheduling Problems. *Cybernetics and Information Technologies*, *15*(2), 3–22.

Gerber, M., & Bornn, L. (2018). Convergence results for a class of time-varying simulated annealing algorithms. *Stochastic Processes and Their Applications*, *128*(4), 1073–1094.

Glover, F., & Greenberg, H. J. (1989). New approaches for heuristic search: A bilateral linkage with artificial intelligence. *European Journal of Operational Research*, *39*(2), 119–130.

Glover, F., & Marti, R. (2006). Tabu Search. In E. Alba & R. Martí (Eds.), *Metaheuristic Procedures for Training Neutral Networks* (pp. 53–69). Boston, MA: Springer US.

Glover, F., Taillard, E., & Taillard, E. (1993). A user's guide to tabu search. *Annals of Operations Research*, *41*(1), 1–28.

Gotshall, S., & Rylander, B. (2002). *Optimal Population Size and the Genetic Algorithm*.

Greiner, R. (1992). *Probabilistic Hill-Climbing: Theory and Applications*. *Proceedings of the Ninth Canadian Conference on Artificial Intelligence*.

Hariri, A. M. A., & Pons, C. N. (1991). A Branch and Bound Algorithm for Job-Shop Scheduling, *3*(1), 201–209.

Hasan, S. M. K., Sarker, R., & Cornforth, D. (2007). Modified genetic algorithm for job-shop scheduling: A gap utilization technique. In *2007 IEEE Congress on Evolutionary Computation* (pp. 3804–3811).

He, S., Wu, Q. H., Wen, J. Y., Saunders, J. R., & Paton, R. C. (2004). A particle swarm optimizer with passive congregation. *Biosystems*, *78*(1), 135–147.

Holland, J. H. (1992). *Genetic Algorithms - scientificamerican0792-66.pdf*. *Scientific American*. Retrieved from http://www.nature.com.ezaccess.libraries.psu.edu/scientificamerican/journal/v267/n1/pdf/scientificamerican0792-66.pdf

Hong, Y., & Lee, C. W. (2018). Pareto fronts for multiobjective optimal design of the lithium-ion battery cell. *Journal of Energy Storage*, *17*, 507–514.

Hsu, T., Dupas, R., Jolly, D., & Goncalves, G. (2002). Evaluation of mutation heuristics for solving a multiobjective flexible job shop by an evolutionary algorithm. In *IEEE International Conference on Systems, Man and Cybernetics* (Vol. 5, p. 6 pp. vol.5).

Huang, R.-H., & Han Yu, T. (2013). *A Novel Tabu Search Algorithm for Solving Robust Multiple Resource-Constrained Project Scheduling Problem*. *International Journal of Computer and Communication Engineering*.

Hudaib, A. A., & AL Hwaitat, A. K. (2017). Movement Particle Swarm Optimization Algorithm. *Modern Applied Science*, *12*(1), 148.

Hurink, J., Jurisch, B., & Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, *15*(4), 205–215.

Jaime, J., Hernández-Wong, J., Nogal, U., Rojas, A., Calderón, A., Rojas-Trigos, J. B., … Marin, E. (2019). Phase resolved method using the Hill-Climbing Metaheuristic Algorithm applied for the spectral separation from photoacoustic spectra of chilli pepper skin and yellow corn pericarp. *Measurement*, *138*, 143–148.

Jain, A. S., Meeran, S., Jain, S. A., & Meeran, S. (1998). A State-of-the-Art Review of Job-Shop. *Mechanical Engineering*, (1), 2–48.

Jolai, F., Asefi, H., Rabiee, M., & Ramezani, P. (2013). Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem. *Scientia Iranica*, *20*(3), 861–872.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (Vol. 4, pp. 1942–1948 vol.4).

Kholladi, M.-K., & Ryma, G. (2011). *PARALLEL HYBRID META-HEURISTIC: GENETIC ALGORITHM WITH HILL CLIMBING TO RESOLVE THE QAP*.

Kingsman, B., Hendry, L., Mercer, A., & de Souza, A. (1996). Responding to customer enquiries in make-to-order companies Problems and solutions. *International Journal of Production Economics*, *46–47*, 219–231.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*, 671–680.

Kumar, S., Mittal, S., & Singh, M. (2017). A Comparative Study of Metaheuristics based Task Scheduling in Distributed Environment. *Indian Journal of Science and Technology*, *10*(26), 1–13.

Lageweg, B. J., Lenstra, J. K. K., & Rinnnooy-Kan, A. H. G. H. G. (1977). Job-shop scheduling by implicit enumeration. *Management Science*.

Laha, D. (2011). *Heuristics and Metaheuristics for Solving Scheduling Problems*. *Handbook of Computational Intelligence in Manufacturing and Production Management*.

Lee, K.-M., Yamakawa, T., & Keon-Myung Lee. (2002). A genetic algorithm for general machine scheduling problems (pp. 60–66).

Li, J., Pan, Q., Xie, S., Gao, K., & Wang, Y. (2011). *A hybrid algorithm for multi-objective job shop scheduling problem*. *Proceedings of the 2011 Chinese Control and Decision Conference, CCDC 2011*.

Li, J. Q., Pan, Q. K., & Liang, Y. C. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, *59*(4), 647–662.

Li, X., & Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, *174*, 93–110.

Lozano, M., & García-Martínez, C. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers and Operations Research*, *37*(3), 481–497.

Magalhães-Mendes, J. (2013). A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS Transactions on Computers*, *12*(4), 164–173.

Marques, A. S., Chibeles-Martins, N., & Pinto-Varela, T. (2015). Simulated Annealing for Production Scheduling: A Case Study BT - Operations Research and Big Data: IO2015-XVII Congress of Portuguese Association of Operational Research (APDIO). In A. P. F. D. B. Póvoa & J. L. de Miranda (Eds.) (pp. 107–114). Cham: Springer International Publishing.

Montemanni, R., Moon, J. N. J., & Smith, D. H. (2003). An improved tabu search algorithm for the fixed-spectrum frequency-assignment problem. *IEEE Transactions on Vehicular Technology*, *52*(4), 891–901.

Morinaga, Y., Nagao, M., & Sano, M. (2014). Optimization of flexible job-shop scheduling with weighted tardiness and setup-worker load balance in make-to-order manufacturing. In *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)* (pp. 87–94).

Murata, T., & Ishibuchi, H. (1995). MOGA: Multi-objective genetic algorithms. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation* (Vol. 1, pp. 289-).

Nesmachnow, S. (2014). *An overview of metaheuristics: Accurate and efficient methods for optimisation*. *International Journal of Metaheuristics* (Vol. 3).

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers and Operations Research*, *35*(10), 3202–3212.

Pinedo, M. L. (2016). *Scheduling Theory, Algorithms, and Systems (5th ed.) [Pinedo 2016-02-11]*. Springer US.

Potvin, J.-Y., & Gendreau, M. (2010). *Handbook of Metaheuristics*. *International Series in Operations Research &Management Science*.

Pratchayaborirak, T., & Kachitvichyanukul, V. (2011). A two-stage pso algorithm for job shop scheduling problem. *International Journal of Management Science and Engineering Management*, *6*(2), 83–92.

Rashid, M. A., Newton, M. A. H., Hoque, M. T., & Sattar, A. (2013). Mixing Energy Models in Genetic Algorithms for On-Lattice Protein Structure Prediction. *BioMed Research International*, *2013*(September), 1–15.

Rey, G. Z. (2014). Reducing Myopic Behavior in FMS Control : A Semi-Heterarchical SimulationOptimization Approach. *Simulation Modelling Practice and Theory*, *46*, 53–75.

Schumann, E. (2011). Simulated Annealing. Retrieved from http://comisef.wikidot.com/concept:simulatedannealing

Schwab, K. (2016). *The Fourth Industrial Revolution*. Currency.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)* (pp. 69–73).

Sipper, M., Fu, W., Ahuja, K., & Moore, J. H. (2018). Investigating the parameter space of evolutionary algorithms. *BioData Mining*, *11*, 2.

Sörensen, K., Sevaux, M., & Glover, F. (2017). A History of Metaheuristics. In *Handbook of Heuristics* (pp. 1–27). Springer US.

Teoh, C. K., Wibowo, A., & Ngadiman, M. S. (2015). Review of state of the art for metaheuristic techniques in Academic Scheduling Problems. *Artificial Intelligence Review*, *44*(1).

Tharwat, A., Gaber, T., Hassanien, A. E., & El Elnaghi, B. (2017). *Particle Swarm Optimization-A Tutorial*.

Tsoy, Y. R. (2003). The influence of population size and search time limit on genetic algorithm. In *7th Korea-Russia International Symposium on Science and Technology, Proceedings KORUS 2003. (IEEE Cat. No.03EX737)* (Vol. 3, pp. 181–187 vol.3).

Umbarkar, D. A., & Sheth, P. (2015). *CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW*. *ICTACT Journal on Soft Computing (Volume: 6 , Issue: 1)* (Vol. 6).

Verwaeren, J., Van der Weeën, P., & De Baets, B. (2015). A search grid for parameter optimization as a byproduct of model sensitivity analysis. *Applied Mathematics and Computation*, *261*, 8–27.

Wang, C., Mu, D., Zhao, F., & Sutherland, J. W. (2015). A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering*, *83*, 111–122.

Weng, M. X., Wu, Z., Qi, G., & Zheng, L. (2008). Multi-agent-based workload control for make-to- order manufacturing. *International Journal of Production Research*, *46*, 2197–2213.

Werner, F. (2017). A survey of genetic algorithms for shop scheduling problems. *Mathematical Research Summaries*, *2*, 15.

Widmer, M., Hertz, A., & Costa, D. (2010). *Metaheuristics and Scheduling*. *Production Scheduling*.

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, *48*(2), 409–425.

Xu, S.-H., Liu, J.-P., Zhang, F.-H., Wang, L., & Sun, L.-J. (2015). A Combination of Genetic Algorithm and Particle Swarm Optimization for Vehicle Routing Problem with Time Windows. *Sensors (Basel, Switzerland)*, *15*(9), 21033–21053.

Yang, X. S., Deb, S., & Fong, S. (2014). Metaheuristic algorithms: Optimal balance of intensification and diversification. *Applied Mathematics and Information Sciences*, *8*(3), 977–983.

Yang, Y. B. (1977). *Methods and Techniques Used for Job Shop Scheduling*.

Yazdani, M., Gholami, M., Zandieh, M., & Mousakhani, M. (2009). A simulated annealing algorithm for flexible job-shop scheduling problem. *Journal of Applied Sciences*, *9*, 662–670.

Zhang, T., Zhang, Y.-J., Zheng, Q., & Pardalos, P. (2012). A Hybrid Particle Swarm Optimization and Tabu Search Algorithm for Order Planning Problems of Steel Factory based on the Make-To-Stock and Make-To-Order Management. *Journal of Industrial and Management Optimization*, *7*, 31–51.

Zukhri, Z., & Paputungan, I. V. (2013). A HYBRID OPTIMIZATION ALGORITHM BASED ON GENETIC ALGORITHM AND ANT COLONY OPTIMIZATION. *International Journal of Artificial Intelligence & Applications*, *4*(5), 63–75.

Appendix

Appendix 1 - Case Study Data

Numbers of products, machines and orders

Products	30
Machines	8
Orders	40

Setup times durations

Setup Types	Dye Change	Mold Change	Label Change
Duration (min)	100	300	10

Setup times

Setup Times (min)	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
P1	0	10	10	300	300	300	300	300	300	300	300	300	300	300	300
P2	10	0	10	300	300	300	300	300	300	300	300	300	300	300	300
P3	10	10	0	300	300	300	300	300	300	300	300	300	300	300	300
P4	300	300	300	0	10	10	300	300	300	300	300	300	300	300	300
P5	300	300	300	10	0	10	300	300	300	300	300	300	300	300	300
P6	300	300	300	10	10	0	300	300	300	300	300	300	300	300	300
P7	300	300	300	300	300	300	0	10	10	10	300	300	300	300	300
P8	300	300	300	300	300	300	10	0	10	10	300	300	300	300	300
P9	300	300	300	300	300	300	10	10	0	10	300	300	300	300	300
P10	300	300	300	300	300	300	10	10	10	0	300	300	300	300	300
P11	300	300	300	300	300	300	300	300	300	300	0	10	10	10	10
P12	300	300	300	300	300	300	300	300	300	300	10	0	10	10	10
P13	300	300	300	300	300	300	300	300	300	300	10	10	0	10	10
P14	300	300	300	300	300	300	300	300	300	300	10	10	10	0	10
P15	300	300	300	300	300	300	300	300	300	300	10	10	10	10	0

Setup Times (min)	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30
P16	0	100	100	300	300	300	300	300	300	300	300	300	300	300	300
P17	100	0	10	300	300	300	300	300	300	300	300	300	300	300	300
P18	100	10	0	300	300	300	300	300	300	300	300	300	300	300	300
P19	300	300	300	0	100	10	300	300	300	300	300	100	300	300	300
P20	300	300	300	100	0	100	300	300	300	300	300	10	300	300	300
P21	300	300	300	10	100	0	300	300	300	300	300	100	300	300	300
P22	300	300	300	300	300	300	0	10	100	100	300	300	300	300	300
P23	300	300	300	300	300	300	10	0	100	100	300	300	300	300	300
P24	300	300	300	300	300	300	100	100	0	100	300	300	300	300	300
P25	300	300	300	300	300	300	100	100	100	0	300	300	300	300	300
P26	300	300	300	300	300	300	300	300	300	300	0	300	300	300	300
P27	300	300	300	300	100	10	100	300	300	300	300	0	300	300	300
P28	300	300	300	300	300	300	300	300	300	300	300	300	0	100	100
P29	300	300	300	300	300	300	300	300	300	300	300	300	100	0	10
P30	300	300	300	300	300	300	300	300	300	300	300	300	100	10	0

Processing times

Processing Times (min)	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
M1	1	1	1	1	1	1	1	1	1	1	0,04	0,04	0,04	0,04	0,04
M2	0,02	0,02	0,02	1	1	1	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02
M3	1	1	0,015	1	0,015	0,015	1	0,015	0,015	1	1	1	1	1	1
M4	0,02	0,02	0,02	0,02	0,02	1	0,02	0,02	0,02	0,02	1	1	1	1	1
M5	0,025	0,025	0,025	0,025	0,025	1	1	1	1	1	0,025	0,025	0,025	0,025	0,025

Processing Times (min)	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30
M6	1	1	1	0,02	0,02	0,02	1	1	1	1	1	0,02	1	1	1
M7	1	1	1	1	1	1	0,07	0,07	0,07	0,07	0,07	1	1	1	1
M8	0,01	0,01	0,01	1	1	1	1	1	1	1	1	1	0,01	0,01	0,01

Orders data

Order	1	2	3	4	5	6	7	8	9	10
Product	2	3	4	1	5	3	4	6	2	5
Quantity	10000	6000	8000	6000	2000	12000	10000	1000	4500	1000
Due Date	2500	1900	1800	1950	1350	1500	1700	1850	100	1400

Order	11	12	13	14	15	16	17	18	19	20
Product	7	8	9	10	10	12	13	14	11	15
Quantity	20000	30000	10000	8000	30000	10000	6000	8000	6000	2000
Due Date	2500	2000	4500	1600	1200	1500	2500	2000	3000	1000

Order	21	22	23	24	25	26	27	28	29	30
Product	13	14	16	12	15	17	18	19	20	20
Quantity	12000	10000	1000	4500	1000	20000	30000	10000	8000	20000
Due Date	2000	2000	1500	7000	2000	1500	8000	1950	1700	3000

Order	31	32	33	34	35	36	37	38	39	40
Product	22	30	29	28	25	26	27	21	23	24
Quantity	5000	7000	5500	6000	2500	1000	2500	8000	9000	2000
Due Date	1800	1500	1550	1500	1500	1900	5000	1800	1450	1250

Inventory info

Finished Product	Quantity	Stock level	
		Min	Max
1	30000	25000	32000
2	25000	24000	30000
3	10000	9000	11000
4	15000	15000	20000
5	12000	10000	14000
6	25000	24000	28000
7	13000	10000	15000
8	5000	4000	10000
9	20000	18000	30000
10	50000	50000	60000
11	30000	25000	32000
12	25000	24000	30000
13	10000	9000	11000
14	15000	15000	20000
15	12000	10000	14000
16	25000	24000	28000
17	13000	10000	15000
18	5000	4000	10000
19	20000	18000	30000
20	50000	50000	60000
21	15000	15000	20000
22	12000	10000	14000

Finished Product	Quantity	Stock level	
		Min	Max
23	25000	24000	28000
24	13000	10000	15000
25	5000	4000	10000
26	20000	18000	30000
27	50000	50000	60000
28	30000	25000	32000
29	25000	24000	30000
30	50000	50000	60000

Appendix 2 - Random Number Generator Procedure

The necessary VBA lines in the beginning of the algorithm to generate random numbers with a fixed seed are the following:

$Rnd(-1)$

$Randomize(1000)$

The $Rnd()$ with a negative argument sets the function to repeat sequences of random numbers. $Randomize()$ function defines the seed. Henceforth, the set of random numbers generated will be the same for all the tests performed to the algorithm.

A variable seed in all experiments can be achieved by adding the following instruction:

$Randomize()$

Appendix 3 - Mono-Objective Results

In order to tune MO$_B$GA parameters, a grid search was done using a fixed seed. All combinations of the following parameters values were tested:

- Generations number: 250, 500, 750, 1000 and 1250.
- Population number: 10, 30, 50 and 70.
- Mutation rate: 50%, 70% and 90%.

The next table shows the results for the MO$_B$GA grid search with a MTO production strategy.

Execution number	Generations number G	Population number P	Mutation Rate MR (%)	CPU time	Objective function OF1 (tardiness)	Iteration of best OF1 found	Objective function OF2 (makespan)
1	250	10	50	1,2	9340	249	8390
2	250	10	70	1,14	11235	249	9305
3	250	10	90	1,32	7795	235	8005
4	250	30	50	3,66	5	212	2090
5	250	30	70	3,61	30	147	1830
6	250	30	90	3,63	45	188	2090
7	250	50	50	6,36	60	165	1620
8	250	50	70	6,18	0	214	1840
9	250	50	90	6,45	5	109	1705
10	250	70	50	8,95	310	123	1822,5
11	250	70	70	8,76	310	80	2230
12	250	70	90	11,21	40	107	1420
13	500	10	50	2,54	90	411	1680
14	500	10	70	2,43	625	486	2327,5
15	500	10	90	3,67	125	491	2055
16	500	30	50	10,29	5	212	1850
17	500	30	70	7,03	0	147	1830
18	500	30	90	7,26	0	188	1960
19	500	50	50	13,19	0	165	1620
20	500	50	70	11,73	0	214	1900
21	500	50	90	12,76	5	109	1960
22	500	70	50	16,63	310	123	1670
23	500	70	70	17,41	310	80	2270
24	500	70	90	17,48	40	107	1570
25	750	10	50	3,63	60	715	1930
26	750	10	70	3,38	280	707	2415
27	750	10	90	4,03	75	699	1880
28	750	30	50	11,8	5	212	2207,5
29	750	30	70	11,91	0	147	1800
30	750	30	90	10,96	0	188	1990
31	750	50	50	18,37	0	165	1620
32	750	50	70	19,3	0	214	1900
33	750	50	90	18,79	0	562	1610
34	750	70	50	29,07	310	123	1822,5
35	750	70	70	32,25	310	80	2270
36	750	70	90	29,69	0	740	1600
37	1000	10	50	5,06	60	715	1920
38	1000	10	70	6,73	235	833	2405
39	1000	10	90	5,49	55	991	1985
40	1000	30	50	18,48	5	212	2207,5

Execution number	Generations number G	Population number P	Mutation Rate MR (%)	CPU time	Objective function OF1 (tardiness)	Iteration of best OF1 found	Objective function OF2 (makespan)
41	1000	30	70	16,84	0	147	1800
42	1000	30	90	15,09	0	188	2090
43	1000	50	50	29,75	0	165	1620
44	1000	50	70	28,49	0	214	1900
45	1000	50	90	25,93	0	562	1820
46	1000	70	50	41,61	0	979	1712,5
47	1000	70	70	37,62	310	80	2230
48	1000	70	90	43,43	0	740	1670
49	1250	10	50	6,1	60	715	1930
50	1250	10	70	6,24	235	833	2405
51	1250	10	90	6,75	40	1026	1985
52	1250	30	50	20,23	5	212	2207,5
53	1250	30	70	21,05	0	147	1830
54	1250	30	90	18,65	0	188	1990
55	1250	50	50	32,95	0	165	1620
56	1250	50	70	41,07	0	214	1900
57	1250	50	90	48,45	0	562	1580
58	1250	70	50	53,08	0	979	1912,5
59	1250	70	70	47,45	310	80	2140
60	1250	70	90	46,64	0	740	1555

The next table shows the results for the MO$_B$GA grid search with a MTS production strategy.

Execution number	Generations number G	Population number P	Mutation Rate MR (%)	CPU time	Objective function OF1 (tardiness)	Iteration of best OF1 found	Objective function OF2 (makespan)
1	250	10	50	0,38	34200	68	55300
2	250	10	70	0,35	0	128	56300
3	250	10	90	0,37	0	143	38000
4	250	30	50	1,14	0	216	48500
5	250	30	70	1,15	0	41	48000
6	250	30	90	1,14	0	30	20620
7	250	50	50	2,04	0	32	22670
8	250	50	70	2,21	0	31	33010
9	250	50	90	2,22	0	31	21290
10	250	70	50	3,53	0	26	13420
11	250	70	70	3,55	0	78	2790
12	250	70	90	3,81	0	24	38370
13	500	10	50	1,17	600	266	51980
14	500	10	70	0,91	0	128	56300
15	500	10	90	1,19	0	143	51100
16	500	30	50	3,69	0	216	48500
17	500	30	70	2,8	0	41	48920
18	500	30	90	3,32	0	30	28760
19	500	50	50	6,39	0	32	22210
20	500	50	70	6,24	0	31	33010
21	500	50	90	5,3	0	31	49700
22	500	70	50	7,04	0	26	51220
23	500	70	70	7,02	0	78	39170
24	500	70	90	7,06	0	24	13500
25	750	10	50	1,33	600	266	51980
26	750	10	70	1,43	0	128	33890
27	750	10	90	1,42	0	143	51100
28	750	30	50	4,65	0	216	49270
29	750	30	70	4,38	0	41	48000
30	750	30	90	4,47	0	30	20480
31	750	50	50	7,46	0	32	21700
32	750	50	70	7,66	0	31	33010
33	750	50	90	7,6	0	31	50180
34	750	70	50	10,34	0	26	51220
35	750	70	70	10,38	0	78	51895
36	750	70	90	10,75	0	24	13500
37	1000	10	50	1,81	600	266	51980
38	1000	10	70	1,89	0	128	56300

Execution number	Generations number G	Population number P	Mutation Rate MR (%)	CPU time	Objective function OF1 (tardiness)	Iteration of best OF1 found	Objective function OF2 (makespan)
39	1000	10	90	1,88	0	143	15480
40	1000	30	50	5,9	0	216	49270
41	1000	30	70	5,81	0	41	49220
42	1000	30	90	5,73	0	30	20380
43	1000	50	50	9,99	0	32	20480
44	1000	50	70	10,8	0	31	49060
45	1000	50	90	9,43	0	31	48375
46	1000	70	50	14,8	0	26	46680
47	1000	70	70	13,43	0	78	51475
48	1000	70	90	14,41	0	24	13200
49	1250	10	50	2,13	600	266	51980
50	1250	10	70	2,61	0	128	48000
51	1250	10	90	2,71	0	143	51100
52	1250	30	50	7,12	0	216	49060
53	1250	30	70	7,46	0	41	48000
54	1250	30	90	8,79	0	30	20480
55	1250	50	50	13,07	0	32	21060
56	1250	50	70	11,93	0	31	48875
57	1250	50	90	13,3	0	31	49410
58	1250	70	50	19,48	0	26	51220
59	1250	70	70	16,04	0	78	38875
60	1250	70	90	15,4	0	24	38790

Appendix 4 - MTS Production Orders

In a MTS production strategy, actual orders are converted in production orders. The next tables show the production orders originated from the actual orders of appendix 1.

Order	1	2	3	4	5	6	7	8	9	10
Product	1	2	3	4	5	7	8	9	10	11
Quantity	8000	19500	19000	23000	5000	22000	35000	20000	48000	8000
Due Date	-	-	1500	1000	-	700	1200	-	-	-

Order	11	12	13	14	15	16	17	18	19	20
Product	12	13	14	15	17	18	19	20	21	22
Quantity	19500	19000	23000	5000	22000	35000	20000	38000	13000	7000
Due Date	-	900	950	-	1450	1400	-	-	-	-

Order	21	22
Product	23	25
Quantity	12000	7500
Due Date	-	-

Appendix 5 - Based on real Instances

Five instances with different characteristics and complexities were created to stress the algorithm.

The first instance orders are described in the table below:

Order	1	2	3	4	5	6	7	8	9	10
Product	1	27	13	14	10	29	25	15	24	12
Quantity	1200	2600	500	2200	1800	3100	700	3200	900	3000
Due Date	2400	2700	1400	2300	2400	1100	700	1800	2200	1100

Order	11	12	13	14	15	16	17	18	19	20
Product	25	7	25	7	30	23	12	11	24	11
Quantity	3700	1200	2500	1500	2400	600	3400	3500	1400	3200
Due Date	2000	500	2200	2500	600	800	2300	2800	1200	600

Order	21	22	23	24	25	26	27	28	29	30
Product	29	27	14	22	13	28	7	6	17	17
Quantity	3200	600	500	3500	3400	1800	3200	2600	800	2300
Due Date	2200	600	1900	800	2300	1800	2500	800	500	2200

Order	31	32	33	34	35	36	37	38	39	40
Product	22	15	9	3	20	14	26	18	21	10
Quantity	1800	1800	2700	2200	1900	3800	1200	1000	2300	1500
Due Date	1800	2100	2300	2200	2400	2900	700	900	1900	2900

Order	41	42	43	44	45	46	47	48	49	50
Product	16	15	20	26	3	20	30	4	11	21
Quantity	600	500	3500	1900	3900	1800	3600	2100	3300	2400
Due Date	2500	2200	1500	1300	2600	1800	2900	800	1000	500

Order	51	52	53	54	55	56	57	58	59	60
Product	12	11	28	7	21	29	8	26	4	28
Quantity	1200	1700	3600	2000	3800	1400	600	1500	1100	2500
Due Date	1100	1700	1200	1900	2500	1900	700	2200	1600	2400

Order	61	62	63	64	65	66	67	68	69	70
Product	30	29	19	14	13	3	20	25	11	9
Quantity	2500	3200	2700	900	3100	1600	3200	2500	1600	2300
Due Date	600	1700	2400	1300	2700	2900	1600	2800	2500	1500

The second instance orders are described in the table below:

Order	1	2	3	4	5	6	7	8	9	10
Product	11	18	25	10	23	10	13	23	23	21
Quantity	19800	9400	9500	4600	17600	14900	15700	13800	1300	11700
Due Date	1300	1200	1000	500	1000	900	500	300	300	300

Order	11	12	13	14	15	16	17	18	19	20
Product	14	9	10	13	21	19	4	7	25	19
Quantity	2500	11000	5300	6100	4300	10800	2900	1900	3200	16700
Due Date	300	300	1300	1000	1300	1200	700	700	1000	900

Order	21	22	23	24	25	26	27	28	29	30
Product	23	9	25	13	6	22	11	17	12	17
Quantity	10600	18200	7200	3500	6700	16600	18600	17900	7400	4200
Due Date	1200	900	500	500	1200	1200	900	800	1100	1300

The third instance orders are described in the table below:

Order	1	2	3	4	5	6	7	8	9	10
Product	16	13	8	12	5	19	29	12	14	25
Quantity	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
Due Date	800	500	500	900	700	600	1000	1200	1000	700

Order	11	12	13	14	15	16	17	18	19	20
Product	3	4	20	3	14	22	3	14	10	6
Quantity	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
Due Date	800	1000	1300	1100	600	1000	900	700	1000	500

Order	21	22	23	24	25	26	27	28	29	30
Product	14	23	8	10	8	11	7	23	17	13
Quantity	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
Due Date	1300	1100	500	700	700	900	1000	800	1100	600

Order	31	32	33	34	35	36	37	38	39	40
Product	28	5	30	11	17	25	30	13	13	17
Quantity	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
Due Date	800	700	500	1400	1300	1200	900	600	1300	1000

The fifth instance orders are described in the table below:

Order	1	2	3	4	5	6	7	8	9	10
Product	22	10	30	6	29	23	6	25	7	15
Quantity	13400	6900	8000	10100	6200	1700	4600	3000	7500	12000
Due Date	2700	500	2200	1500	2100	3900	3500	1600	1700	3600

Order	11	12	13	14	15	16	17	18	19	20
Product	22	9	14	14	16	24	16	3	24	19
Quantity	7800	9700	1400	2200	7700	8500	3200	3700	13800	12200
Due Date	2500	3400	2700	900	900	3100	3800	2400	3800	2100

Order	21	22	23	24	25	26	27	28	29	30
Product	19	23	13	16	28	24	29	5	18	12
Quantity	14800	5800	3200	2800	9400	3200	13800	8600	5500	5700
Due Date	3600	1100	1800	600	800	3300	3200	2600	2300	700

Order	31	32	33	34	35	36	37	38	39	40
Product	15	24	25	5	1	8	11	19	1	10
Quantity	1600	3700	3500	8800	11200	6600	9800	10200	11900	14200
Due Date	3800	3600	700	2600	3200	900	1500	2500	1600	900

Order	41	42	43	44	45	46	47	48	49	50
Product	5	4	6	30	7	27	15	12	28	15
Quantity	8800	2000	4700	9600	8000	7600	6400	1500	3200	11100
Due Date	2000	1900	3200	3000	3400	3000	3900	1400	1800	2700

Order	51	52	53	54	55	56	57	58	59	60
Product	17	21	9	10	3	10	23	16	23	15
Quantity	8900	10300	2000	1200	12200	2500	10400	5800	5100	10700
Due Date	900	2400	1600	1200	2700	3800	1700	1300	1600	2700

Order	61	62	63	64	65	66	67	68	69	70
Product	1	3	7	28	10	9	12	23	21	3
Quantity	7500	2600	5400	4300	8700	2700	5000	2100	5800	10400
Due Date	3500	900	3400	1900	2200	1200	1800	1800	3800	1300

Order	71	72	73	74	75	76	77	78	79	80
Product	30	30	26	9	12	20	29	10	20	26
Quantity	3100	7100	8500	1700	14900	13400	6100	3300	13500	3800
Due Date	2200	800	800	700	1900	600	1000	2700	1600	700

Order	81	82	83	84	85	86	87	88	89	90
Product	23	28	24	18	6	14	9	28	27	30
Quantity	2300	7500	1300	6300	11100	3000	1300	14400	14800	8300
Due Date	1300	1500	900	1600	1600	1000	1800	2100	1000	3700

Order	91	92	93	94	95	96	97	98	99	100
Product	30	21	6	8	3	1	8	4	11	14
Quantity	9500	8600	9400	11400	6800	9500	11300	10800	2900	8600
Due Date	1600	3400	2100	3500	2200	1100	3100	500	1600	2300

Appendix 6 - Asymmetric Setup Time Matrix

The tables below show an asymmetric setup time matrix. In this case, contrary to the symmetric matrix of appendix 1, the necessary time to changeover from product 1 to product 2 is can be different from the time to changeover from product 2 to product 1.

Setup Times (min)	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
P1	0	300	300	300	10	300	300	300	300	300	300	300	300	300	300
P2	10	0	10	300	10	300	300	300	300	100	300	300	10	300	10
P3	300	300	0	10	100	300	300	300	300	300	10	300	300	300	300
P4	300	300	300	0	300	300	300	300	300	300	100	300	100	300	300
P5	300	300	300	300	0	300	300	300	10	300	300	300	300	10	100
P6	300	300	10	10	300	0	300	100	300	300	100	10	300	300	100
P7	300	300	100	10	300	10	0	300	300	100	100	300	300	10	300
P8	100	300	10	300	300	300	300	0	100	10	300	10	300	300	300
P9	300	300	10	300	300	300	300	300	0	300	300	300	10	100	300
P10	300	300	10	300	300	300	300	100	300	0	10	10	300	300	100
P11	300	10	300	100	300	300	300	10	300	300	0	10	300	300	300
P12	10	300	300	300	300	300	10	300	300	10	300	0	300	10	300
P13	300	300	300	10	300	300	100	300	10	300	300	300	0	10	10
P14	300	300	300	300	300	300	10	10	300	300	10	300	100	0	300
P15	300	300	300	300	300	300	10	100	300	100	300	300	10	300	0

Setup Times (min)	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30
P16	0	300	100	100	300	300	300	300	100	300	300	300	300	300	10
P17	300	0	300	300	10	10	300	100	100	300	300	300	10	300	300
P18	300	300	0	300	300	10	300	300	10	10	10	300	300	300	300
P19	300	300	300	0	300	300	10	300	300	300	300	300	300	300	100
P20	10	100	300	300	0	300	100	300	10	10	300	300	300	300	300
P21	300	100	300	100	100	0	300	300	300	100	10	300	300	300	300
P22	300	300	10	300	300	300	0	300	100	300	10	300	300	10	300
P23	300	100	300	100	300	10	300	0	300	10	300	300	300	300	100
P24	300	10	300	300	10	300	300	300	0	300	300	300	300	300	100
P25	300	10	300	100	10	300	300	10	300	0	300	300	300	10	300
P26	300	100	10	100	10	300	300	300	100	10	0	300	10	10	100
P27	300	300	300	300	10	10	10	100	300	10	300	0	100	300	300
P28	100	300	300	300	10	300	300	300	300	300	300	300	0	10	100
P29	300	100	300	300	300	10	10	100	300	300	300	300	300	0	300
P30	300	300	300	300	300	300	300	300	300	300	300	300	100	10	0

Appendix 7 - Stock Levels

The next table presents the inventory info used to test the instances of appendix 5 with a MTS production strategy.

Finished Product	Quantity	Stock level	
		Min	Max
1	5100	8000	5100
2	30000	24000	30000
3	11000	9000	11000
4	20000	15000	20000
5	14000	10000	14000
6	28000	24000	28000
7	15000	10000	15000
8	10000	4000	10000
9	30000	18000	30000
10	60000	50000	60000
11	32000	25000	32000
12	30000	24000	30000
13	11000	9000	11000
14	20000	15000	20000
15	14000	10000	14000
16	28000	24000	28000
17	15000	10000	15000
18	10000	4000	10000
19	30000	18000	30000
20	60000	50000	60000
21	20000	15000	20000
22	14000	10000	14000
23	28000	24000	28000
24	15000	10000	15000
25	10000	4000	10000
26	30000	18000	30000
27	60000	50000	60000
28	32000	25000	32000
29	30000	24000	30000
30	60000	50000	60000