Faculdade de Engenharia da Universidade do Porto

Optimization and Simulation of Manufacturing Systems

Luís Filipe de Jesus Vieira Pereira

FOR JURY EVALUATION

Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Supervisors: Jorge Pinho de Sousa and Samuel Moniz

June 27, 2016

Let all things be done decently and in order
I Corinthians

Abstract

Production planning and scheduling are critical in matters of cost reduction, resource sustainability and performance improvement, these being crucial factors due to the highly competitive nature of today's industry. The technological development allowed the emergence of hybrid methods of optimization and simulation to perform the referred actions with better results.

In this dissertation, a set of methods related with planning and/or scheduling are presented, many of which are a hybridization of optimization and simulation. From those methods, some were chosen as inspiration to the work developed in this dissertation, Kim & Kim (2001) being one of those influences. However, to that work it is intended to add WIP effects, setup times, and scheduling components, as well as improving the quality of its results. The proposed methodology consists in the interaction between a Mixed Integer Programming model and a simulation model, the latter being used as an evaluator of the solution proposed by the first, while also implementing the developed scheduling technique. In the case of an infeasible solution, the simulation results will serve as input to the MIP model to adjust some parameters in an intent to produce more realistic results. The proposed methodology was applied to a case study concerning a project-oriented company with production driven by customer demand.

Results prove the proposed methodology to be satisfactory and of good quality in handling the case study's problems, namely its bottleneck features. Moreover, the results support the proposed methodology as a viable alternative for production planning and scheduling problems in similar situations.

Acknowledgements

There are people without whom this dissertation would not have been written and to whom I am greatly thankful.

My parents, whose support and dedication kept me motivated and encouraged me to tackle the gloomiest problems that surged throughout this dissertation.

My supervisor and co-supervisor for their knowledge, motivation, enthusiasm, and mostly for their time. Their advices were crucial to the success of this dissertation.

My brothers, for the constant challenges that shaped me to be dedicated and persistent, traits that were fundamental to embrace this thesis as a challenge and to always try to do my best.

List of Contents

List of Figures

List of Tables

Abbreviations and Symbols

AME	Analytical Model Enhancement
ATO	Assemble-to-Order
BOM	Bill of Materials
CF	Clearing function
DES	Discrete-Event Simulation
DoE	Design of Experiment
FFD	Fractional factory design
FIFO	First In First Out
GA	Genetic algorithms
HPP	Hierarchical production planning
JIT	Just in Time
KPI	Key Performance Indicators
LIFO	Last In First Out
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
MIP	Mixed Integer Programming
MOO	Multi-Objective Optimization
MTO	Make-to-Order
MTS	Make-to-Stock
MPPFLP	Multi-Product, Multi-Period Facility Location Problem
OM	Operations Management
PLE	Personal Learning Edition
PML	Process Modelling Library
ROSA	Recursive Optimization-Simulation Approach
SM	Scientific Management
SPSA	Simultaneous Perturbation Stochastic Approximation
WIP	Work in Process

Chapter 1

Introduction

Production planning and scheduling are critical functions in manufacturing systems in matters of operational costs reduction, resource sustainability, and performance improvement. Market's increased accessibility and consequent competitiveness encouraged a generalized investment in continuous improvement. Improvements in modelling and the verified leap in technology allow the development of new methods to tackle these problems.

This chapter covers in some detail, the following aspects of the work: the motivation for this project; theory and concepts context; objectives and questions to answer; brief methodology explanation; and document structure presentation.

1.1 Motivation

Today's strong and ever-growing interconnection led to market globalization consequently increasing its competitiveness. Customer options enlarged significantly and companies had to intensely compete to conquer each client. This ferocious rivalry stimulated continuous improvement methodologies inside corporations. Production systems, integrant parts of companies and industries, felt the need to attain and maintain high levels of quality and efficiency to be attractive to the market.

Production planning and scheduling are typically critical functions regarding operational costs reduction and overall system improvement. Nevertheless, operational complexity has highly increased with time and analytical methods used in the past are no longer viable due to the unpractical solving times. Hence, alternative approaches are of uttermost necessity to manage planning and scheduling problems related to high complexity and size production systems.

Traditionally, simulation was used in cases where the complexity or nonlinear nature of systems could not be handled by analytical methods. Nonetheless, simulation is a tool for solution analysis and optimal solutions can neither be obtained through simulation nor proved to be optimal with resource to it. Therefore, simulation allowed for strong modeling without guarantee of optimality.

Advances in modelling techniques and the simultaneous increase in computational power and decrease in computational costs allowed both simulation and optimization techniques, usually considered to be separate or alternative, to be combined in a hybrid manner.

The relatively recent development start of these hybrid methods is an opportunity to explore due to the high potential this combination represents to solve production planning and scheduling problems. The methodology will be briefly presented in section 1.4.

1.2 Concepts and Context

1.2.1 Operations Management and Dynamics of Production Systems

This dissertation is comprised in the Operations Management (OM) discipline that came from Scientific Management (SM), the first management discipline dating back to the late 19th century. Despite not being the first person to show interest and seek to rationalize the practice of management, Frederick W. Taylor (1856-1915) was the first to generate "the sustained interest, active following, and systematic framework necessary to plausibly proclaim management as a discipline" Hopp and Spearman (2011) [1]. Taylor defended that planning and doing are distinct activities that should be addressed by different job categories. This principle is the backbone of modern management. Besides Taylor, many were the contributors to SM and OM and are detailed in [1].

In [1], many definitions used in production systems are explained. Notwithstanding the importance of them all, only a few will be detailed in this dissertation due to its pertinence.

Hopp and Spearman (2011) [1] use the term *workstation* to refer a "collection of one or more machines or manual stations that perform (essentially) identical functions. (…) In process-oriented layouts, workstations are physically organized according to the operations they perform". In this dissertation the term WorkCentre is used as a synonymous of *workstation*.

When the term product is used in this dissertation it refers to the synonymous term *part* described as "a piece of raw material, a component, a subassembly, or an assembly that is worked on at the workstations in a plant" [1]. The same way, non-elementary components are *subassemblies*, elementary components are *components* and final products are *end items*.

The terms *routing, order, raw material,* and *lead time* are also explained.

Routing "describes the sequence of workstations passed through by a part.". *Order* might be of two types: external or internal. External orders are *customer orders* which represent customer requests "for a particular part number, in a particular quantity, to be delivered on a particular date". *Raw material* "refers to parts purchased from outside the plant".

The *lead time* is described, for a determined line or routing, as "the time allocated for production of a part on that routing or line", however this is not the definition used in this dissertation. In this dissertation, lead time is used as the total time it took the part to go from its production start point to its finish point, being composed by the production time plus the waiting time.

Hopp and Spearman (2011) [1] describe Work in Process (WIP) as jobs "that have not yet arrived at an inventory location". WIP is a critical factor for manufacturing and production system's performance as it affects throughput (TH). TH is defined as "average output of a production process" but it can be further detailed as "the average quantity of *good* (nondefective) parts (…) produced per unit time". This concept is important to understand

Little's Law (12), stating that "at every WIP level, WIP is equal to the product of throughput and cycle time" (CT).

$$WIP = TH * CT \quad (1)$$

The *cycle time* used in the definition refers to the term lead time in this dissertation. This law translates that when the systems reaches its maximum throughput capacity an increment in WIP will lead to an increase in lead time. Little's law is not actually a *law* but a *tautology* meaning the proposed relation is not accurate in all systems. Nonetheless, it can be used as a "conjecture about the nature of manufacturing systems" to understand the influence of WIP in such systems. In the studied case (that will be presented in section 3.2), lead time does not vary linearly with WIP because the different products have different processing time, being the wait time dependent not only on the number of products but also on the type of operation those products will go through. Therefore, the lead time does not change linearly due to variability.

Hopp and Spearman (2011) [1] formally define variability as "the *quality of nonuniformity of a class of entities*. (…) In manufacturing systems, there are many attributes in which variability is of interest. Physical dimensions, process times, (…), setup times, and so on". Variability can either be controllable or random. Controllable variation occurs "as a direct result of decisions" whereas random variation "is a consequence of events beyond our immediate control".

Variation has a nefarious influence on cycle time (in this dissertation referred to as lead time). For a single station, it is comprised by move time plus queue time plus setup time plus process time. For assembly operations, waiting all components is added to the equation. Variation highly influences waiting times and, as generally concluded in many studies, the waiting times tend to be the highest factors in the described equations. Therefore, control over variability is of highest importance.

In [1], there are sections entirely dedicated to variability and served as references for this dissertation. However, due to the extension of the subject, variability and its consequences will not be further detailed here.

1.2.1.1 Push and Pull

Push and *Pull* are concepts widely used in manufacturing and production systems descriptions, sometimes not in a precise and even contradictory manner.

Taiichi Ohno, the father of Just in Time (JIT) used the term *pull* in a very general sense in [2]:

"Manufacturers and workplaces can no longer base production on desktop planning alone and then distribute, or *push*, them onto the market. It has become a matter of course for customers, or users, each with a different value system, to stand in the frontline of the marketplace and, so to speak, *pull* the goods they need, in the amount and at the time they need them" [2].

However, the interpretations of this concept are diverse, and this wide range of definitions is present, for example in [3-8].

Bonney *et al.* (1999) [9] analysis some of those interpretations and opts for a definition based on the information flows used for control. When the control information flow is in the opposite direction to the material flow, the system is of type *pull*. When the control information

flow is in the same direction as the material flow, the system is of type *push*. This definition is also used in this dissertation.

Information control information is the production trigger. In a *push* strategy, when a WorkCentre finishes its operation on a certain part it triggers the production order for the next WorkCentre in that part's routing. In a *pull strategy*, when a WorkCentre is to execute an order it triggers the orders related to its dependencies, either material or operational.

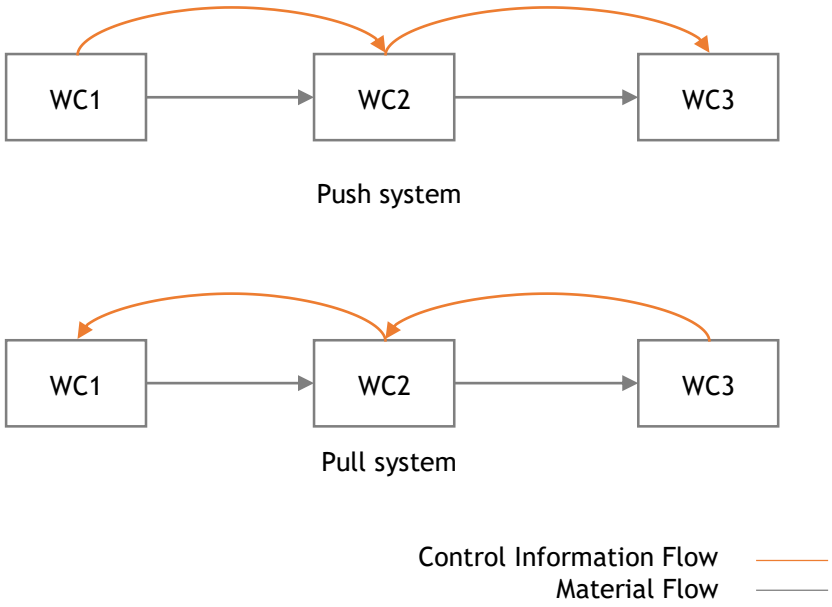


Figure 1.1 - Push vs Pull flows

Customer orders serve as control information inputs. Considering Figure 1.1, for the push system, customer orders would enter in WC1 (first WorkCentre) and the information would spread from that point onward. For the pull system, customer orders would enter WC3 (third and last WorkCentre) and would propagate from that point backward.

The benefits of the *pull* system are presented in [1] and are divided in: reduced manufacturing costs, reduced variability, improved quality, flexibility maintenance, and facilitation of work ahead.

Systems often encompass both push and pull features, originating push-pull strategies. While push and pull systems are generally (but not only) associated with Make-to-Stock (MTS) and Make-to-Order (MTO) strategies, respectively, push-pull systems are related to Assemble-to-Order (ATO) strategies.

ATO strategies are developed around a decoupling point (or more) from where the production strategy changes. This strategy is especially benefic in systems where different combination of a set of components allows the production of many different products suited to customer needs. A good example is ice-cream based on flavors. From a relatively small set of flavors, many different compositions can be made. If there are 5 flavors and customer can choose up to 3 scoops, there are 155 possibilities (5*5*5 for three scoops plus 5*5 for two scoops plus 5 for one scoop).

These systems work in MTS strategy up to the decoupling point (being the selling point in the ice-cream case) and in MTO strategy from that point onward. These systems benefit from both economies of scale and possibility of customization based on customer demand.

1.2.2 Planning and Scheduling

Independently of the strategy in use, virtually all manufacturing systems are desired to provide "on-time delivery, minimal work in process, short customer lead times, and maximum utilization of resources" [1]. Unluckily, these goals conflict. Production scheduling aims at striking a profitable balance among these conflicting objectives.

The goals of production scheduling might be meeting due dates, maximizing utilization, or reducing WIP and Cycle Times (in this dissertation referred to as lead times). Anyhow, this dissertation's interest is mainly on the first goal. Due date performance can be evaluate using service level, fill rate, lateness, and tardiness. These and many other scheduling-related concepts are described in [1]. In this dissertation tardiness is used together with another measure to influence optimization parameters (explained in section 3.5). Scheduling defines the sequence on which production orders and operations will be executed. Besides the already defined goals of scheduling, its results impact setups, one of the main causes of controllable variability in a system.

Scheduling is based on a plan, which is produced by production planning tasks.

The basic problem of production planning in manufacturing environments "involves viewing the production system as a conglomerate of resource groups" (WorkCentres) "and allocating the capacity of production resources (…) among different products over time, coordinating the associated inventories and raw material inputs so that known or predicted customer demand is met in the best possible manner" [10]. The "best possible manner" is not a very scientific description of the objective and requires a better definition to form the basis of an optimization model. Generally, this objective is minimizing the total expected costs or maximizing the total profit of the system over the considered time interval. However, in many systems, the second option is not easily calculated, thus the first option is the most common and generalized approach.

System's increase in complexity led to a two planning level approach where the upper level generates an aggregate production plan while the lower level produces a detailed scheduling of the work orders within the production units based on the production plan produced by the upper level.

In short, a production plan describes what to produce in a specific time interval and the production schedule determines the moment for each of the productions and operations comprised in the production plan. Therefore, production planning and scheduling are related and are crucial tasks of manufacturing systems.

1.2.3 Modelling and Solving

All the presented concepts would be useless if they could not be applied and analyzed. System modelling is therefore of extreme relevance for the OM discipline. As referred, manufacturing systems are growing in complexity. Not only that, but they are getting more integrated and sophisticated. "Production planning models are very often Mixed Integer Programming (MIP) models, because of problem features such as set-up costs and times, start-up costs and times, machine assignment decisions, and so on." [11]. MIP models, in opposite of Linear Programming (LP) ones, are able to capture the discrete nature of some decisions, due to integrality constraints. Most modelling decisions and techniques were based on [11] and [10].

The mathematical formulation used in this project is present in section 3.3.1.

Mathematical models are used in a generalized manner as they are, in theory, solvable. If a real system is possible to be mathematically modelled, for instance as a MIP model, then it is theoretically solvable. In this dissertation, the Branch and Bound algorithm design paradigm is used. It was first proposed by Land and Doig (1960) [12] and named "Branch and Bound" by Little *et al.* (1963) [13]. A Branch and Bound algorithm is based on a systematic enumeration of candidate solutions through state space search. The set of candidate solutions can be described as a rooted tree where the root of the tree contains the full set. Each branch of such tree represents a subset of the solution set. To avoid the search of the entire universe of candidate solutions, before enumerating the candidate solutions of a branch, the algorithm checks that branch against upper and lower estimated bounds. If that branch cannot produce a better solution than the best one found it is discarded.

The used Branch and Bound is incorporated in the MIP solver used that will be presented in section 3.3.2.

1.3 Objectives of the Research

Increased market competitiveness led industries towards a continuous struggle to overcome each other and successfully conquer clients. Consequently, the investigation and research in areas related to this problematic increased significantly. This dissertation also aims at answering this problematic. Being planning and scheduling operations at the core of the market competitiveness problematic, this dissertation's main objective is to tackle both these problems. Converting this general goal into questions to be answered, this dissertation would pose the following questions:

- How to use optimization and simulation methods to improve production planning and scheduling in manufacturing systems?
- How to perform the interaction between optimization and simulation to obtain the maximum benefits from the hybridization?

Nevertheless, the second question is an extension of the first one, as the interaction is included in the utilization of the two methods. However, as it will be presented in chapter 2, the main difference between hybrid optimization-simulation methods lays on the interactions between both rather than on each method's formulation. For that reason, the second question was posed separately.

1.4 Methodology

Optimization, as the name states, consists on the search and prove of the optimal solution for a certain problem. However, when the problem in analysis is of great size and complexity, the solve time is unpractical. Moreover, typical mathematical models are not able to incorporate uncertainty as well as stochastic characteristics of the system in their formulation.

Simulation, on the other hand, is able to analyze extremely complex nonlinear and stochastic systems in a practical time. Nevertheless, simulation is mostly a tool for scenarios evaluation and is not fit to find the optimal solution for a certain problem nor prove the optimality of a certain solution.

Optimization and simulation, when analyzing the pros and cons of each method, are almost symmetrical, i.e., the strengths of a method serve as solutions for the weaknesses of the other. Henceforth, the hybrid use of both creates a stronger method.

Such hybridization might be performed in different ways, as reviewed by Figueira and Almada-Lobo (2014) [14]. To tackle production planning and scheduling problems an optimization-based approach was chosen. Based on [14], the proposed methodology is categorized as a Recursive Optimization-Simulation Approach (ROSA) from the category of Analytical Model Enhancement (AME) approaches.

This type of approach "consists on running recursively a relatively east (typically linear) analytical model and a (more detailed) simulation model. Simulation uses the solution generated by optimization and computes particular performance measures. (…) The values of these measures are then introduced again into the analytical model, refining its parameters. The recursive process ends after a stopping criterion is met." [14]. Despite being a MIP model instead of a LP model, the proposed methodology is still encompassed in this category.

As referred by Figueira and Almada-Lobo (2014) [14], each model abstraction level is different, i.e., the detail level in the optimization model is lower than on the simulation model. This modelling option is used to avoid the increase in complexity which optimization solve times suffer from. However, since the simulation model will comprise the features not included in the optimization model, the parameter adjustment made between iterations will serve a similar purpose as the inclusion of such features in the optimization model.

The proposed approach is applied to a case study of a manufacturing system working on a project-based strategy where customer demand drives production. Final products are assembled from standard components built from processed metal sheets (raw material). Previous system strategy presents many opportunities that the proposed approach aims to solve. One of the problems in the system is the bottleneck WorkCentre being so majorly due to long setup times. Besides the bottleneck WorkCentre there are five other WorkCentres, defined by the type of operations developed in them (process-oriented layout).

The iterative approach and the case study to which it is applied will be further detailed in chapter 3.

1.5 Structure of this dissertation

This introductory chapter will be followed by a literature review where techniques used to tackle planning and or scheduling problems are presented and analyzed. Afterwards, the proposed methodology is exposed in chapter 3 including the presentation of the case study. Following this chapter, results of several instances are presented and analyzed. Finally, the dissertation closes (chapter 5) with the statement of the most important results and conclusions, as well as future work that could be applied to the proposed approach.

Chapter 2

Literature Review

The combination of increased market competitiveness and technology progress pushed developers and researchers towards an intensive search and development process for better tools to handle production planning and scheduling problems. The literature on such methods and approaches is vast making unviable the inclusion and analysis of its entirety. Hybrid approaches, referred in the previous chapter as generally better than their isolated parts, were the main focus of this research appearing, however, some methods that do not fit this category but were kept due to the value they present.

Information and references were selected and gathered using an iterative approach.

Primarily, the search method was carried out with resource to Google Scholar and a set of keywords. The search results were then analyzed by the sequence: Abstract and Introduction, Conclusion, and finally the description of the approach.

If valuable aspects were found during this first analysis process, the reference was kept and marked for more detailed analysis and its references were used as a second level for the search cycle.

The third step was to search for citations of the highest quality articles or books from the selected ones to find more recent work based on the same foundations.

In some of the iterations of the presented cycle, the found articles would consist in state-of-the-art texts, taxonomies or reviews of the best methods to solve this type of problems, at the time of its writing. Those articles also served as reference for the gathering of quality literature.

After a few iterations of such cycle, the resultant articles and books started to become repetitive and that was when the gathering moment was considered sufficient, beginning the second, more detailed and exhaustive, moment of analysis. Afterwards, the references considered to be of most relevance were grouped according to the nature of the proposed methods.

Table 2.1 - Literature Search Method Keywords Set

Production planning	Production scheduling	Simulation	Optimization
Modelling	Multi Product Multi Period	State of the art	WIP

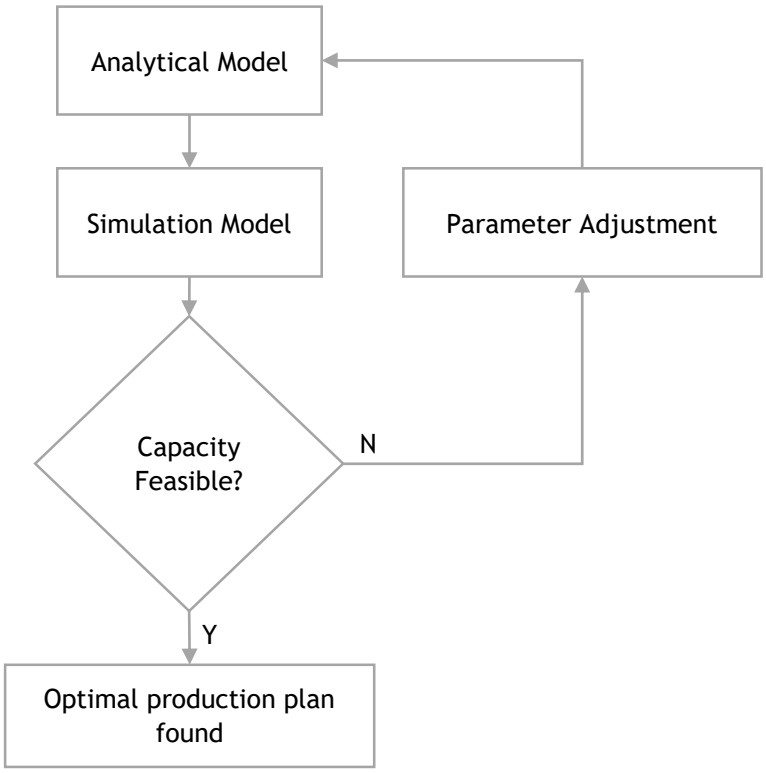
2.1 Optimization-Simulation Methods

The solutions based on some kind of hybridization of optimization and simulation methods are of great interest as they are the most similar to this dissertation's proposed approach.

Understanding that the disadvantages from both optimization (analytical) and simulation methods could be mitigated by the advantages of the other, Byrne and Bakir (1999) [15] studied and developed an hybrid model between the two types of solution for the Multi Period Multi Product Production Planning Problem. The approach is based on an iterative process where the optimization results are evaluated by the simulation model If the results are not valid, the simulation model will adjust a specific set of parameters of the optimization model and repeat the process with more restrict conditions. When the results from the analytical method are valid, that resulting production plan is "both mathematically optimal and practically feasible" [15]. This article presents the base concept for several approaches, including the one proposed in this dissertation. Although the authors used LP as the analytical tool for their case study, the general concept allows for any type of optimization method to be applied.

Kim and Kim (2001) [16] leveraged on this approach and extends it by changing the LP model formulation considering factors that directly affect the capacity and workload of the resources and allowing production orders launched in a determined period to be stretched further into future periods. The authors state that the extended method proposed converges in few iterations consistently. In fact, its behavior is explored by Irdem *et al*. (2010) [17] and the method does converge consistently in a relatively small number of iterations, apart from a few cases where cycling between two solutions was observed. The authors compared two methods using a case study in the semiconductor manufacturing context where the complexity and size of the problem are immense.

Figure 2.1 - General hybrid modelling procedure



Consequently, the method proposed in [16] presented a major disadvantage due to the need for a detailed simulation model of the production facility which must be run multiple times per iteration.

Bearing the Just-in-Time concept in mind, Byrne and Hossain (2005) [18] propose a new LP formulation to use in the hybrid approach proposed in [15] and improved in [16]. Such a formulation allows the partitioning of orders into a set of smaller dimension ones.

Almeder *et al.* (2009) [19] present a solution approach to support the operational decisions for supply chain networks. The developed general framework consists in applying a LP or MIP formulation in the context of a Discrete-Event Simulation (DES). Additionally, the authors empirically show an iterative combination of simulation and LP to be competitive when compared to deterministic MIP-models, in the context of stochastic supply chains. In the proposed approach, the simulation model enrolls as the master process, controlling the data communication and the LP/MIP-solver. Since the framework can be applied to stochastic situations, it may contain stochastic and nonlinear elements. Therefore, the simulation must be run several times and its results must be combined. Depending on the parameters and its influence on other elements of the optimization model, the combination rules are different.

The approach begins with several simulation runs to generate initial parameter values for the optimization model and their results are ignored in further iterations. The combined results are calculated and stored in the database. The optimization model is executed with base on those values and its solution is stored. Decision rules are computed based on this solution and new simulation runs are executed.

The authors' tests prove the combination of simulation and optimization methods to be "worthwhile" and advantageous when compared to the more traditional alternatives and the separate utilization of the methods.

Lee and Kim (2002) [20] proposed a solution similar to [19] and previously presented. The authors aimed to solve the incapacity of analytic models to correctly represent the dynamic and uncertain behavior of real supply chain systems. The suggested hybrid approach combines analytic and simulation models considering the operation time parameter in the analytical model as a dynamic factor adjusted by the results of the simulation model. While Almeder *et al.* (2009) [19] focus on obtaining a robust production, supply and transport plan considering stochastic and nonlinear operations and costs, estimating delays and cost-influential factors based on simulation experimentation, Lee and Kim (2002) [20] aim to obtain more realistic capacity estimates for the optimization model.

Bang and Kim (2010) [21] suggest "a two-level hierarchical production planning (HPP) method in which the higher level (aggregated level) decision is made for production planning and the lower level (disaggregate level) decision is made for detailed scheduling". The proposed method consists on a three-step iterative process. For each iteration, first, a production plan is produced from the LP model. Then, a priority rule-based scheduling method is used for operations scheduling in the fab. Finally, the resulting schedule and production plan are evaluated with resource to a DES.

The method was developed for the semiconductor wafer fabrication context, characterized by high production rate and variety, and the results proved it to work better than traditional approaches and other commonly used methods. One of the authors considerations was the occurrence of unexpected events and its effect on the method since the LP modelling approach assumes deterministic situations in the facility. Under those circumstances, the plan is expected to change but doing so in a frequent matter may cause instability and confusion.

Therefore, the preferred method was to obtain a new plan in case of a major disturbance occurs with updated information of the fab. Otherwise, if the disturbance is minor, the current plan is used if the difference between the plan and actual production information does not exceed a predetermined level.

While the method was developed for the semiconductor wafer fabrication context, the methodology is general enough to be applied to situations where the variety and amount of production are inferior. However, the results and effectiveness might differ.

Such an adaptation can also be made to the solution proposed by Kropp *et al.* (1978) [22] due to the value of the general concept. The authors compare their hybrid approach to the isolated use of simulation and optimization methods recurring to a hypothetical health care environment.

The concept is iterative consisting in the evaluation of the optimization model results with simulation. Then, the approach proceeds to find relationships between the nonlinear variables of simulation and the variables that are common to both models through linear regression performed on the results of a number of simulation runs. These linear relationships will then be applied to the optimization model as constraints to reflect a "non-cost objective of the facility". The process will iterate until the achievement of the desired result of the evaluation. The presented technique can be adapted to different contexts with small changes in the formulation and a quality simulation model of the desired system, as in every hybrid technique.

Acar *et al.* (2009) [23] propose an approach that also benefits from the referred versatility of application to different situations with small changes in the formulation and a quality simulation model. The authors develop a generalized MIP formulation able to interact with a simulation model. The generalized formulation interaction with simulation is based on the computation and evaluation of candidate solutions based on the results of previous runs. To test the suggested solution, a multi-product, multi-period facility location problem (MPP-FLP) was used as a case study and the results were promising and showed benefits versus the majority of alternative solutions.

2.2 Meta-Heuristics

From the previous literature analysis there is a common trait most of the authors refer as a problem or disadvantage of the hybrid optimization simulation approaches. As the size and complexity of the problem increase, the optimization model run time becomes unaffordable. To solve this problem, authors tend to increase the level of abstraction of the optimization model, however, it may not be desirable since it signifies a decrease of information and reliability. Such high complexity and size problems often do not require an optimal solution due to the unpracticality of its computation and accept quality sub-optimal solutions as an alternative.

To approach these and other combinatorial optimization problems, since the early 80's a lot of interest has be placed in the development and application of meta-heuristics, from which, as an example, will be highlighted the genetic algorithms (GA), considered good solution search strategies. Nonetheless, in order to model a real production planning problem stochastic and nonlinear parameters must be included. Most of those real problems are not simple enough for GA to be applied. Thus, to solve this situation, the hybridization concept is applied, combining meta-heuristics, GA in this case, with simulation methods.

The hybridization concept with resource to meta-heuristics is similar to the one with optimization methods, as the simulation is used as evaluation tool and considers the stochastic and nonlinear parameters of the system, and the meta-heuristic is used to search for optimal (or sub-optimal) solutions for the problem and its model is affected by the simulation results.

Jeong *et al*. (2006) [24] developed a hybrid solution where the GA is used to optimize schedules and the simulation is used to minimize the maximum completion time of the last job while reflecting stochastic characteristics with the fixed input from the GA. The authors considered the completion time to be the simulation runtime, which is the overall time spent to execute all operations based on the production schedules generated by the GA.

The operation time in the GA model is adjusted according to the simulation results. With this new values, the GA regenerates new operation schedules. This process is executed until the difference between the preceding simulation runtime and the current runtime is acceptable.

Li *et al*. (2009) [25] propose an approach using GA and Design of Experiment (DoE) in an iterative manner. Their proposal is presented in the remanufacturing context, which differs from the general production systems. However, the concept can be adapted to fit such systems. Understanding that the major disadvantage of GA is the probability of skipping the optimal solutions around a certain individual solution when the optimal does not strictly fit the selected criteria for the next generation. The authors use fractional factory design (FFD) to find the extrema of each cell and develop a method to overcome this drawback. The solution candidates for the GA are provided by the FDD and the GA will continue the search process until the stop condition is met, considering the corresponding extrema of each cell. The authors state that the use of FFD improve the traditional use of GA in two ways. First, it ensures the local optima is found for each cell, improving the searching accuracy. Second, due to its fractional nature, improves the searching efficiency.

Liu *et al*. (2011) [26] adapted the Multi-Objective Optimization (MOO) MatLab function and used it in cooperation with a simulation model to solve production planning problems. The MOO function is based on an elitist GA and is adapted to the problem and interaction with the simulation model to “search for a set of release plans that are near-Pareto optimal” [26]. The multiple objective are the mean and variance of total cost. The simulation model is also used as the objectives evaluator, similarly to previously referred proposals. The proposed method is mainly directed to help decision making in circumstances where there is the necessity to “weigh the trade-off between average cost and the risk associated with that cost”[26], as it provides more detailed information than the obtained from single-objective optimization. However, the authors understand their results to be possible starting points for other algorithms or to require further investigation through ranking and selection procedures.

2.3 Clearing Functions

The willingness to find faster and higher quality solutions to production planning and scheduling problems of stochastic/nonlinear nature led researchers towards different approaches. Additionally, traditional models as LP, MIP and other analytical methods tend to assume fixed lead times, which has been indicated by queueing-theoretical results and practical experience to be incorrect. In fact, lead times are considered to be “load-dependent, which leads to the well-known trade-off between short lead times and high capacity utilization”[27]. This feature modelling requires the representation of the “relationship between output and

WIP or lead time in the model"[27], which can be accomplished using nonlinear, saturating clearing functions (CFs). Considering a facility divided in sections, a clearing function (CF) is the "functional relationship between some measure of WIP in a period *t* and the expected or maximum output of the [section] in period *t*" [27].

Pürgstaller and Missbauer (2012) [27] analyze the use of CF and compare their inclusion in optimization order release with traditional rule-based methods in workload control. Their conclusions prove the optimization-based methodology to largely outperform the rule-based ones.

Kacar and Uzsoy (2010) [28] work on the major problem concerning CF, its estimating. The authors compare different regression approaches based on simulation results and compare the computational results. From the results obtained from the different experiments and approaches, they conclude on the difficulty of estimating CF and the lack of a strong foundation and/or mathematical models to support such procedure.

The authors wrote a more recent article [29] on the same subject with a more detailed analysis of the problem, solution proposals and results, while presenting different and newer alternatives. On the same article, the authors propose the use of Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm to estimate CFs. In the used case study, SPSA is shown to significantly improve the production plan by either estimating "better CF parameters or by directly optimizing releases"[29].

Comparisons between production planning models using CF and alternatives are present in the literature, as in [30-31]. The results tend to show the use of CF to be superior to the alternatives however, its estimating is not general and the results cannot be considered replicable in different situations.

2.4 Conclusion

The presented methods are plausible solutions, each of them having advantages and associated difficulties/problems.

CF offer high quality model-reality relationship with a strong ability to incorporate stochastic features into the production planning approaches, but are not trivial to estimate and the results from its application in a case study cannot be generalized as there is no mathematical formulation or a strong foundation for estimating methods. Missbauer and Uzsoy (2011) [10] dedicate a subsection of their article to the analysis of the limitations of CF models.

GA, and Meta-Heuristics in general, tend to find solutions faster than the common analytical methods. Nonetheless, such methods do not guarantee the optimal solution to be found.

Approaches based on the hybridization of analytical optimization and simulation methods have strong mathematical foundations to support the analytical modelling and are guaranteed to find the optimal solution for the optimization part of the approach. Nevertheless, the runtime might become impracticable with the increase of optimization model size and complexity. The usual solution is to increase the level of abstraction and use the simulation model to include the important details and the hard-to-model parameters.

The strong foundation supporting the development of approaches belonging to the last mentioned group associated with the ease of manipulation and variation of the mathematical formulation, and the optimal solution finding guarantee, led to the choice of the hybrid optimization-simulation approach as this dissertation solution proposal for production planning and scheduling problems.

The method extended in [16] and [18] from [15] serve as the basis to the development of the method proposed in this work. It is easily adapted to different situations and the MIP formulation is flexible enough to be manipulated towards improvement.

Furthermore, the method proposed in [16] is showed in [17]. to rapidly converge towards a final feasible solution This conclusion has both advantages and aspects to be carefully considered. The convergence proves little necessary iterations of the method to obtain a feasible solution, which is positive as it is translated in practicable run times. However, such trait is also a problem, since the simulation affects the optimization parameters too intensely and the solution results are often non-optimal in the real situations. Therefore, the interaction between simulation results and optimization parameters must be reviewed and adapted to allow for a higher quality convergence of the proposed method.

In [23], the authors prove the superiority of the iterative hybrid method in comparison with the simple MIP approach. Additionally, forms of interaction between simulation and optimization are proposed and uncertainty factors are introduced in optimal decision making. The authors also state that the modelling of uncertainty factors improve the results for production planning problems. This article, in conjunction with [21], serve as inspiration for the inclusion of dynamic factors (stochastic/nonlinear) as each of them present suggestions for the modelling of different parameters: setup times and costs, defect production, and WIP effect on lead-times.

In the following chapter, the approach developed in this work is explained in detail along with the adaptations made to the proposals of the above referred articles.

Chapter 3

Simulation/Optimization Approach

The approach proposed in this thesis is categorized as hybrid, since the general idea consists in the iterative use of simulation and optimization methods. Each of the methods present different advantages and limitations specific to their nature. Hybridizing the methods aims to attenuate/eliminate the limitations and disadvantages of each method while taking full advantage of their positive features, obtaining a superior method than each of the individual parts.

Optimization methods are well-known for the guarantee of obtaining the optimal solution for the modeled problems. Additionally, the mathematical foundation is well developed and its formulation can be adapted at will with relative ease. However, nonlinear and stochastic factors, common in real situations and sometimes impactful on the behavior of the whole system, cannot be directly modeled by optimization methods. Furthermore, with the increase of model's size and complexity, these methods run time might exponentially increase and become unpractical.

Simulation methods, in contrast, can comprise a high level of detail, including nonlinear and stochastic factors, without compromising their execution time. Moreover, in situations where the richness of detail is crucial, simulation models are of uttermost utility. Despite the high modelling capacities such methods are used to evaluate deterministic situations. The search for optimal solutions cannot be purely done with resource to simulation.

In a summarized manner, the interaction between this duality of methods has the objective of overcoming each method's limitations with the other method's advantages.

This chapter will proceed with a thorough description of the intended problem to solve, followed by a detailed explanation of the optimization and simulation models construction process. Later it will conclude on the comprehensive explanation of the interaction between both models.

3.1 Problem Description

Production planning and scheduling are critical in matters of cost reduction and performance improvement, crucial factors in today's highly competitive industry. These problems can be decomposed in several key tasks.

Production planning decisions are often related to medium-term time horizons. These decisions address the determination of optimized production mixes, lot sizes, order assignment to resources and release plans. In contrast, production scheduling key tasks are related to operational aspects with short time horizons for instance, order and operation sequencing.

The problem at stake can be divided in two phases: solve the planning decisions and subsequently the scheduling ones accordingly. For instance, considering a manufacturing scenario, decide on the production mix to release on production for each period. Based on this decision, decide on the sequence of releasing from which the system benefits the most, for each period.

The interdependency between the two sorts of decisions is noticeable, being its simultaneous resolution advantageous.

Capacity restrictions tend to be linear, thus not posing as a relevant problem during system modelling. However, aside from these and other common restrictions, there are some influencing factors that are critical. The lead time of the production line, which might differ with the product type, is one of such factors. The amount of WIP in the system influences the lead time in a nonlinear manner that cannot be modelled using linear constraints. Similarly, it is known that when a resource is being used at a near-limit capacity, the lead time increases significantly. Hopp and Spearman (2011) [1] go into further detail on the referred dynamics.

The proposed methodology aims to solve production planning and scheduling problems that must attend the referred considerations.

These problems can be briefly defined as determining, for each product, the amount and moment to release it to the shop floor.

3.2 Case Study

In this dissertation we address the referred problems in a real case concerning a job-shop manufacturing system that produces industrial equipment. The production is project-oriented, meaning the end product is only manufactured once the customer places the order, which also defines the quantities and release date. The company produces each customer order according to the required features, being each order a new project. The company performs installation, maintenance and repair services at customer's site. However, in this dissertation, only the manufacturing process of the end products issue will be addressed.

The production facility is organized in four main areas (production, assembly, maintenance, and special projects) from which only the first two are concerned to the normal manufacturing process. Production and Assembly are divided in six WorkCentres (Figure 3.1). These physical areas are related with different tasks and composed by machines, and input and output buffers.

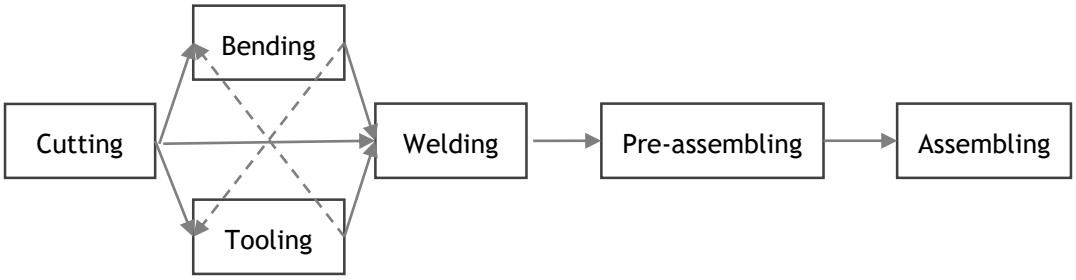


Figure 3.1 - Production and Assembly systems

The inclusion of buffers between each WorkCentre aims at ensuring the material flow between

them. The Cutting WorkCentre (WC01) comprises two machines, the Bending WorkCentre (WC02) four, the Welding WorkCentre (WC03) one, the Tooling WorkCentre (WC04) one, the Pre-assembling WorkCentre (WC05) three, and the Assembling WorkCentre (WC06) two.

At the facility, storage space does not represent a limiting factor, however excessive WIP may cause internal logistic problems, namely the significant increase of material transportation and handling times.

Material requisites, operation times as well as resource allocation considerably change from end product to end product. Hence, the facility's process flow can be categorized as a job-shop.

3.2.1 Production Process

Despite the uniqueness of each project and the consequent difference in requirements and features, the production of industrial equipment starts invariably by cutting pieces of metal in desired shapes. The cut pieces are then forwarded to the next WorkCentre, which might be the Bending, the Tooling or the Welding WorkCentres. After being welded, the pieces are sent to the Pre-assembling WorkCentre where they are combined into standard components. Finally, these standard components are assembled into final products in the Assembling WorkCentre.

The production system capacity is defined by processing units' availability.

Although having the higher number of production units, the Bending WorkCentre represents the system's bottleneck due to the long tooling set changeover times. These setup times are sequence dependent.

3.2.2 Planning Process

Long and short term planning are performed by the planning department responsible. Long term planning is considered one month in advance while short term planning comprises a one-week time horizon. The plan's update occurs at least three times per week and re-planning is performed regularly. The customer's agreement is required for any change in the plan.

While planning also involves equipment installation and maintenance planning, development of weekly production plans is the most important planning function to consider since the focus of this dissertation is in the production activities of the company. Commonly, the production plan encompasses the quantity and timing of products to be produced. The planner, resorting to his experience, computes the resources requirements to execute the production mix, estimating its overall impact on system capacity. Nevertheless, at this stage, no resource allocations are performed.

Scheduling implicates resource allocation (raw materials, components, processing units and workers) to production orders that are then released to the shop floor and is performed on a daily basis. This resource allocation to orders occurs one week in advance. Order release and sequencing, and resource allocation are done according to due dates, taking in account the current state of the shop-floor.

The occurrence of unexpected events, common in highly complex production systems, leads to rescheduling.

3.2.3 Opportunities

The described production planning presents improvement opportunities.

The plan is re-calculated and corrected on a regular basis. Additionally, it is mostly done based on experience, which might produce acceptable results but cannot guarantee the decisions optimality.

Resource allocation to orders is not an automated process and is not directly considered while developing the production plan. This allocation is performed and considered solely during the scheduling task, hindering the optimality of the order release plan and delaying the information flow.

Unexpected events, such as machine failure, defects production or worker related problems, are not considered in the production planning and scheduling tasks, forcing the re-execution of such tasks and incurring unnecessary work.

The production system has a bottleneck, the Bending WorkCentre. The time associated with the changeovers of the tooling set is the main reason for it and it is highly sequence dependent. The lack of search for production plan optimality incurs in performance reductions in this WorkCentre, compromising the entire system performance.

The development of tools that would consider not only the dependencies between resource allocation, sequencing and release of orders, but also the occurrence of unexpected situations would highly benefit the production system performance.

3.2.4 Summarized Description

Before continuing this chapter with the optimization model, it is important to conclude the present sub-section with a detailed yet summarized description of the case study.

The company at study works on a MTO production strategy, working with large projects. Each of these projects aims at manufacturing final products, which are composed by standard components originated by the Pre-assembling WorkCentre. These standard components are assembled from worked metal pieces that are produced by a defined sequence of operations taking place at a specific WorkCentre.

There are six WorkCentres, each composed by one or many machines assumed to be equal inside each WorkCentre. The system's bottleneck is at the Bending WorkCentre, mainly due to highly sequence dependent setup times.

Despite excessive WIP reducing system's productivity/performance, storage space is not a limiting factor.

The proposed approach intends to present solutions that help the decision-making process at the management levels of the company. These solutions consist in deciding on which piece to be produced, in what quantity and when.

3.3 Optimization Model

Prior to the mathematical formulation of the optimization model it is necessary to state the main relaxations and abstractions made from the real problem. As previously referred, it is not computationally affordable to solve a MIP problem containing all the information from a

complex real system. Hence, some considerations were made while modelling the problem at study as a MIP formulation.

Tooling set changeovers are critical in the Bending WorkCentre, thus its inclusion in the model. All machines in a specific WorkCentre are considered equal.

The travel time between WorkCentres is not considered as it is rather insignificant.

Raw material (metal pieces) is considered available on demand, therefore not being modeled as a constraint to the system.

In production systems, profit is related to costs and in our case, being a project-oriented system, this dependency is even further noticeable. Therefore, the objective function of our formulation will be the minimization of costs. Costs are divided into three categories: production, inventory and backlogging, related to the costs of producing a certain product, storing a product, and not fulfilling a release order in the correct period of time, respectively.

3.3.1 Mathematical Formulation

The format for the mathematical formulation is as follows. Lowercase italics are used for indices, uppercase bold letters for sets, uppercase italics for variables, and lowercase Greek letters for parameters.

Table 3.1 - Formulation Elements - Indices

Indices	
$t \in \mathbf{T}$	Periods: discrete intervals of time of a certain duration
$i, j \in \mathbf{P}$	Products: types of products
$k \in \mathbf{K}$	WorkCentre: resources that process products through certain operations
l, m $\in \mathbf{O}$	Operation: specific task to be executed on a product

Table 3.2 - Formulation Elements - Sets

Sets	
$\mathbf{T}$	Periods in the modelling horizon
$\mathbf{P}$	Product types
$\mathbf{P}^B$	Basic products (worked metal pieces)
$\mathbf{P}^F$	Final products
$\mathbf{P}^k$	Products that are operated in WorkCentre *k*
$\mathbf{K}$	WorkCentres
$\mathbf{O}$	Operations
$\mathbf{O}_i$	Operations that can be performed on product *i*
$\mathbf{D}_i$	Direct successors of *i* in the Bill of Materials (BOM)
$\mathbf{BOM}_i$	Direct dependencies of *I* in the BOM

Table 3.3 - Formulation Elements - Variables

Variables	
X_{it}	Amount of product *i* to produce in period *t*
I_{it}	Inventory of product *i* at the end of period *t*
B_{it}	Backlog of product *i* at the end of period *t*
E_{kt}	Elasticity of WorkCentre *k* in period *t*
A_{it}	Represents whether or not product *i* was produced in period *t*

Table 3.4 - Formulation Elements - Parameters

Parameters	
α_{it}	Production cost of product *i* in period *t*
σ_{it}	Inventory cost of product *i* in period *t*
π_{it}	Backlog cost of product *i* in period *t*
η_k	Maximum duration of an operation in WorkCentre *k*
$\lambda_{k(i,j)}$	Tooling set changeover from operation of product *i* to operation of product *j* time at WorkCentre *k*
μ_{it}	Demand of product *i* in period *t*
τ_{ilk}	Processing time of product *i* under operation *l* at WorkCentre *k*
φ_{kt}	WorkCentre *k* total capacity in period *i*
ε_{ij}	Amount of product *i* required to produce one unit of product *j*
γ_i	Lead time of product *i*
β	Large positive number

The structure of the model is defined in terms of the previous elements.

As referred, the minimization of costs will be the objective function, comprising the production, inventory and backlog costs for each product across all periods(2).

$$\min \sum_{i \in \mathbf{P}} \sum_{t \in \mathbf{T}} (X_{it}\alpha_{it} + I_{it}\sigma_{it} + B_{it}\pi_{it}) \tag{2}$$

As in any system, the overall resource utilization cannot exceed its maximum capacity. This utilization is measured in time and has two components: time spent executing operations on a product, and time spent in tooling set changeovers whenever the type of product to operate changes. This constraint is defined in (3).

$$\sum_{i \in \mathbf{P}^k} \sum_{l \in \mathbf{O}^i} X_{it}\tau_{ilk} + \left(\sum_{i \in \mathbf{P}^k} A_{it} - 1 \right) \lambda_k \leq E_{kt} + \varphi_{kt} - E_{kt-1}, \qquad \forall k, t \tag{3}$$

Since the products in operation sequence is not known, $\sum_{i \in \mathbf{P}^k} A_{it}\lambda_k$ could lead to an overestimation of the changeovers times when the WorkCentre ends a period operating the same type of product it will begin to operate in the next period. Therefore, the subtraction of one unit is added to translate that changeovers are equal to the number of products minus one.

The right-hand side of capacity constraints includes the Elasticity factor of WorkCentres and was one of the contributions of this dissertation. To ensure that if the capacity of a WorkCentre in a period *t* is approximately enough to produce a certain product *i*, that product will be launched in production in period *t* and finished in the following period. This factor was

restrained by two conditions: it cannot exceed the time of a period (4); and it cannot exceed the time of the longest production operation in that WorkCentre (5).

$$E_{kt} \leq \varphi_{kt} \quad (4)$$
$$E_{kt} \leq \eta_{kt} \quad (5)$$

To ensure the correct material balance and flow, it is necessary to create constraints relating quantities between any pair of adjacent periods.

$$I_{it} - B_{it} = X_{it} - \mu_{it} + I_{it-1} - B_{it-1}, \forall i \in \mathbf{P}^F, t \quad (6)$$
$$I_{it} = X_{it} - \left(\sum_{j \in \mathbf{D}_i} \varepsilon_{ij} X_{jt+\gamma_i} \right) + I_{it-1}, \forall i \notin \mathbf{P}^F, t \quad (7)$$

Constraints (6) and (7) serve to carry quantity information from a period to the next one. Final and non-final products are associated with different material balance constraints. These sets of constraints differ in two aspects.

First, the demand for final products originates from customer orders (independent demand) while the demand for non-final products is created by the production orders of higher level products. Multi-level dependency between products is modeled in the proposed approach, originating the already referred BOM. Hence, the existence of dependent demand. As an example, refer to Figure 3.2. The production order of one unit of product MT will create dependent demand of product MS and MA in the ε_{ij} proportion. Similarly, MS has dependencies of products MA and ME, generating dependent demand of each in the proportion ε_{ij} per unit of MS. In the example scenario, the production of one unit of product MT would create a total dependent demand of three MA products, six ME products and two MS products.

Demand is associated with a period. Independent demand occurs in the same period the customer order is placed. However, for dependent demand, the period is influenced by the product lead time γ_i. The dependent demand of product *i* occurs γ_i periods prior to the period when the production of the product that originated the demand is produced. In other words, if product MT was to be produced in period *t*, product MS dependent demand would be related to period *t*- γ_i.

$\sum_{j \in \mathbf{D}_i} \varepsilon_{ij} X_{jt+\gamma_i}$. translates the dependent demand of product i in period t. $\mathbf{D}_i$ set represents the direct bottom-up successors of product *i*. Referring again to the Figure 3.2, $\mathbf{D}_i$ of MA is (MS, MT) Concluding, the dependent demand of product *i* in period t is obtained by the production orders of each of its direct successors *j* in period *t*+ γ_i with a multiplicative factor ε_{ij}.

Constraints (6) and (7) differ yet in another way. There is no backlog of components as

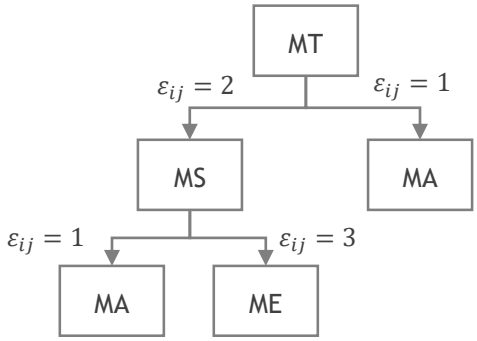


Figure 3.2 - Multi-level dependence example

there is only dependent demand of such products. However, in case of unmet demand of a final product, the demand must be fulfilled later in the time horizon. The backlog from period *t-1* carries that period unmet demand to period *t*.

The already referred binary variable A_{it} represents the occurrence of production of product *i* in period *t*. This logical condition is modeled with resource to (8).

$$\beta A_{it} \geq X_{it}, \forall i, t \quad (8)$$

Finally, there are constraints related to the non-negativity of the integer variables and the binary nature of A_{it}.

$$X_{it}, I_{it}, B_{it}, E_{kt} \geq 0, \forall i, t, k \quad (9)$$
$$A_{it} \in \{0; 1\}, \forall i, t \quad (10)$$

3.3.2 Model Implementation and Software

There are many solvers on which we could implement our model: CPLEX, XPRESS, GUROBI, among others. The choice criteria were availability and performance. Fortunately, the overall best performer solver in industry standard public benchmark tests (check [32]), Gurobi Optimizer [33], also provided an academic license.

Gurobi Optimizer is a solver for mathematical programming designed to exploit modern architectures and multi-core processors and incorporates six solvers (Table 3.5). From those the particular interest for this dissertations lays on the Linear Programming Solver and the Mixed-Integer Linear Programming solver, considering the problem at stake nature.

Gurobi support a variety of programming and modelling languages, providing high flexibility to the user. To implement the optimization model of the proposed approach the chosen programming language was Python. Python is an interpreted language, meaning it is highly flexible and can be implemented in any number of ways. Python is efficient, easy and fast.

Table 3.5 - Gurobi included solvers

Linear Programming Solver (LP)	Quadratic Programming Solver (QP)
Mixed-Integer Linear Programming solver (MILP)	Quadratically Constrained Programming solver (QCP)
Mixed-Integer Quadratic Programming solver (MIQP)	Mixed-Integer Quadratically Constrained Programming solver (MIQCP)

Python's syntax is designed to be readable, which means its writing does not require most of the structures and details other languages need. Python is able to perform in little lines of code what would require complex programming in most other languages. The language is dynamically built, allowing for better memory management: names are linked to objects instead of declared. When the object is no longer needed, the name can be linked to a different object. Data manipulation and handling is more efficient in Python (reason why it is widely becoming popular due to Big Data issues). For those and other reasons, the Python version of Gurobi Optimizer was chosen.

3.3.3 Data Input and Output

The optimization model required system's data input and it was accomplished with resource to excel files and using *xlrd* Python library to read from such files [34]. Similarly, to output the model results excel files were also the chosen resources and the *openpyxl* Python library was used to write such files [35].

The input data was related to the information on the sets and indices previously presented. Output data comprised release orders, inventory levels and backlog for each product on each period. WorkCentre utilization was also outputted for each period as well as the total lead time of each product in periods.

To save the input data on the model, the preferred data structures were dictionaries and lists. Whenever an index was related to a product, operation or WorkCentre, it was of type string. When it was related to a period, it was an integer. This indexation allowed for easier modelling, debugging and understanding than if all indexes were integer. Furthermore, whenever an index combination was inexistent, instead of attributing a null or zero value to it, it was simply ignored and not created on the respective data structure, increasing model performance.

3.3.4 BOM Handling

One of the major concerns while building the model was with BOM use and representation. First, due to the nature of the mathematical formulation, it was necessary to build two distinct structures based on the material dependence: the direct successors of a product (products that depend on it to be produced) and the dependencies of a product (products necessary to produce it). These different structures can be comprehended as the reverse of each other and, therefore, appear to be redundant. However, they are used in different situations and are in fact both necessary. Referring to the mathematical formulation, these structures match $\mathbf{D}_i$ and $\mathbf{BOM}_i$ sets, respectively.

$\mathbf{D}_i$ is used in constraints (12) while $\mathbf{BOM}_i$ is used to calculate the lead time of product i (ignoring its production time, as it is marginal compared to its dependencies lead times). $\mathbf{BOM}_i$

Pseudo-Code 1 **Longest Path Procedure**

```
function calc_max_path(BOM, i, P^B, leadTime, dependencies_leadTime):
    Max_value = 0
    for all product s in BOM_i do
        if s ∈ P^B then
            If Max_value < leadTime[s]
            Max_value = leadTime[s]
        else
            Max_pathvalue_s = calc_max_path(BOM, s, P^B, leadTime, dependencies_leadTime)
    if Max_pathvalue_s+leadTime[s] > Max_value then
        Max_value = Max_pathvalue_s+leadTime[s]
    dependencies_leadTime[i] = Max_Value
    return Max_value
end function
```

contains the information of product *i* direct dependencies. Referring to Figure 3.2, $\mathbf{BOM}_{MT}$ is composed by MS and MA.

If the material dependency structure of a certain product *i* is seen as a graph, to calculate a product lead time, considering no initial inventory of its dependencies both direct and indirect, it is necessary to calculate the longest path of the graph, considering each product lead time as the path value.

The algorithm developed to obtain this longest path is recursive and is described in *Pseudo-Code 1*.

The procedure is performed for all final products $\mathbf{P}^F$, populating the related data structures.

3.3.5 Variation of Period Duration

The developed optimization model allows for the variation of period duration, i.e., the amount of periods in a determined amount of time can change based on user input. Considering a fixed window of a week with five working days each with eight working hours, if the number of periods per week is one, each period corresponds to a week. However, in the same scenario, if the number of periods per week is five, each period corresponds to one working day. The same applies to forty periods per week where each period has a duration of one hour (Figure 3.3).

There are two main consequences to this variation in duration. First, as variables and parameters are related to periods, increasing the number of periods implies an increase of variables and parameters which translates in a higher size model. Optimization models runtime, as referred, significantly increase with model size and complexity. Therefore, increasing the number of periods severely impact the time spent solving the model which is even more critical in an iterative approach with multiple optimization model runs.

Nonetheless, the increase in number of periods has benefits. The conversion of lead times into periods implies smaller errors with the decrease of period duration. With very small period durations, the model output conveys in itself a pseudo-scheduling The benefits in decreasing period duration can be resumed as an overall increase in precision.

A more precise model better translates the system's dynamics and features on the other hand it reduces its practicality. Hence, it is necessary to find a balance point.

Varying the duration of the period implies a higher number of periods per time interval. With the intent of explaining the data manipulation related to this changes, a time interval of three weeks will be considered from now on.

Inventory and backlog costs must be adapted to this period alterations, as well as WorkCentre capacities and lead time conversion to periods.

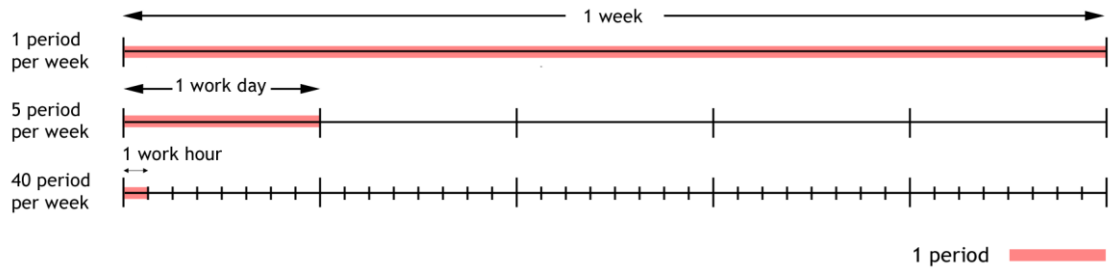


Figure 3.3 - Number of periods vs period duration

The affection of WorkCentre capacities and the lead time conversion to periods are the simplest to explain. WorkCentre capacity is defined in minutes per period. If period duration is smaller, the WorkCentre capacity will change accordingly. Lead time conversion is calculated as the result of the round up to the higher closest integer of the division between lead time and period duration.

Inventory costs can be defined as the holding cost per unit of time. If the inventory cost is x per week, it will be $x/5$ per working day and $x/40$ per working hour.

In the case study scenario, customer demand is always considered at the end of a week, thus being associated with the last period of such week. Backlog consists in the undelivered quantity of products matching a specific demand. Being so, backlog costs are only associated

to the last period of any week, independently of the amount of periods comprised in one week. In other words, the backlog cost related to the last period of a week is unchangeable and zero to all other periods.

3.4 Simulation Model

The proposed method is composed by two main parts, being simulation one of them. While it is unpractical and unviable to model the optimization model with full detail and comprising every dynamic of the real system, the simulation model can include such features without significant decrease in performance. Therefore, the simulation model can be used as an evaluator to the output of the optimization model. Those features that are present in simulation but lacking in optimization are generally limitations. The influence of WIP in the lead time is certainly the most important limitation faced in simulation, as well as in real systems.

Simulation is the last step of the proposed iterative approach and is used to evaluate the feasability of the optimization results. According to the evaluation, the approach may iterate again or terminate. Additionally, the proposed approach uses simulation to implement the developed scheduling technique, explained in section 3.5.

3.4.1 AnyLogic

Simulation has provided a constantly evolving tool to work in proximity with the real world for more than half a century. The available software is vast and equipped with different tools. The decision on which software to choose was majorly aided by Swain (2015) [36] where fifty five products from thirty one vendors are listed and compared. AnyLogic was chosen as the software to model the case study system. Besides providing a student free license called Personal Learning Edition (PLE), it is one of the most complete software in the survey and allows for full customization through Java programming.

AnyLogic is unique in its capacity to support all the most common simulation methodologies: System Dynamics, Agent Based, and Process-centric (Discrete Event) modelling. For the purpose of this dissertation, only the latter is used. The generality of manufacturing systems can be modelled using Discrete Event modelling techniques since the system can be represented as a sequence of operations being performed on entities of certain types, from products to packages, workers to machines. The term Process-centric is self-descriptive. Such modelling

focus on the process and ignores some physical level details, such as geometry, accelerations, etc. Therefore, DES or Process-centric are medium-low abstraction level modelling approaches.

Anylogic supports object-oriented model design, providing modular, hierarchical, and incremental construction of large models while allowing reusability. The native Java environment supports limitless extensibility ranging from custom Java code to external libraries and data sources. Additionaly, both the Anylogic IDE and the models have multi-platform support, working on Windows, Mac and Linux.

The Process Modelling Library (PML) is the primary Anylogic toolkit for Discrete Event modelling. The library is a collection of highly customizable objects used to define process workflows and their associated resources. Their parameters can be changed dynamically and their actions may be dependent on entity's attributes. Workflow objects have extension points that permit custom definition of actions to be performed on entities throughout the process. Most objects have "onEnter/onExit" extension points. Nevertheless, specific objects have specific extension points related to their function and allowing for further control and customization.

Complex systems benefit from the modularity capabilities of AnyLogic as they can be break down into components and modelled separately. AnyLogic allows sub-process definition, reducing the logical and visual complexity of top-level model and providing a good basis for reusability within a model or across models.

AnyLogic's main building blocks are Agents. Agent is a unit of model design that can have behavior, memory(history), timing, contacts, etc. and may represent diverse things, from people to equipment, from non-material things to organizations. Within agents, a multitude of definitions can be performed: variable, events, custom code, and the list goes on. Agents can also communicate with the external world, for instance using calling functions.

To build the case study model, some blocks inside PML were used and will be introduced in the next subsection. However, blocks related with Agent definition and Connectivity will be briefly presented first.

System's inputs are agents and, in this dissertation's particular scenario, those agents represent products. Each of these products have multiple parameters.

Figure 3.4 - Parameter Block

Parameters are agent's attributes and can be of many types, both Java primitive types and AnyLogic special types. The agent class used to define products is named Prod. Prod's parameters are represented in the following table.

Table 3.6 - Prod class parameters

materialCode	id	route_center
operationProcessingTime	Stock	next_WC
nextProcessingTime	reorderPoint	reordered
operation_route	nextOperation	order
materialID	codeComponent	amountComponent
priority	WIPonEnter	

Parameter *materialCode* is used to define the type of product. Parameter *id* is the product unique identifier. Parameter *route_center* holds the information of the WorkCentre route the product needs to go through. Parameter *next_WC* has the information of the WorkCentre to where the product needs to go next. Parameter *operation_route* holds the information of the operation sequence the product needs to pass through. Parameter *nextOperation* represents the next operation that has to be performed on the product. Parameter *operationProcessingTime* is related to parameter *operation_route* and holds the information of the duration of each operation the product needs to go through. Parameter *nextProcessingTime* represents the processing time of the next operation to be performed on the product. When the system is working on an ATO strategy, the parameter *reorderPoint* represents the level of inventory below which a reorder will be triggered and the parameter *reordered* represents whether or not that specific product has an active reorder. When working on a MTO strategy with no stock, the parameters are ignored. The parameter *order* holds the identification of the Order that triggered the production of that specific product. Parameter *WIPonEnter* represents the amount of WIP on the system when that specific product entered production. This parameter is used to relate WIP with lead time which will be later explained.

Some products have other products dependencies, thus the need to implement a BOM. That structure was implemented using the collection block to store the immediate dependencies of the product.

Figure 3.5 - Collection Block

Collections are Java classes developed to efficiently store multiple elements of a certain type. One of the advantages over Java arrays is the ability to store any number of elements. There are many types of collections: *ArrayList*, *LinkedList*, *HashSet*, *TreeSet*, etc. The simplest one is *ArrayList* which is a sort of resizable array. Each collection type has a different purpose and its choice must be based on the predominant operations that will be performed on it. Since the BOM collection will serve essentially for information storage and its size will be relatively small, the chosen type of collection was *ArrayList* due to the small operation time related with search operations for relatively small *ArrayLists*.

The parameters whose cells have a blue fill are related with BOM construction. The BOM structure of a level one product with dependencies will be constructed using Prod agents that are direct components. However, the only information required is the *codeComponent* which is equivalent to *materialCode*, the amount per unit of level one product (*amountComponent*) and the level one product *id* (*materialID*).

The parameter whose cell have a yellow fill will be explained in section 3.5.

All the information regarding parameters, BOM, orders and all external-dependent features of the system are obtained via Excel File.

Figure 3.6 - ExcelFile Block

This particular block serves as the import of Excel files to the model from which AnyLogic can read information and for which it can output data. The model comprises four of these blocks named *Info*, *infoOrders*, *LTs*, and *setupTs*, referring to the information on system and products, orders, lead times, and setup times, respectively. AnyLogic recurs to the Java API Apache POI to operate Excel files (more information on [37]).

Some information must be kept under the form of global variables that are accessible for every object and instance on the simulation model.

Figure 3.7 - Variable Block

One example of such information is the orders identifier that needs to be incremented every time a new order or reorder is created and the identifier needs to be unique. Orders were already referred to as the triggers for products production. In fact, orders translate the optimization solution release orders information. Orders and reorders are modeled using the same class Order whose parameters are presented in the following table.

Table 3.7 - Order class parameters

id	productCode	Amount
releaseDate	reorderCalls	reordersGen
parent		

Parameter *id* is the unique serial identifier of an Order object. Parameter *productCode* holds the information related to the product type to produce, matching Prod's parameter *materialCode*. *Amount* is the integer quantity of products of type *productCode* to produce. Parameter *releaseDate* holds the information of the period when the order is to be executed. The parameters whose cells are filled with blue are related with the reorder function and will be explained at the same time of the referred function.

In order to track products, orders and reorders, and machines, populations were used.

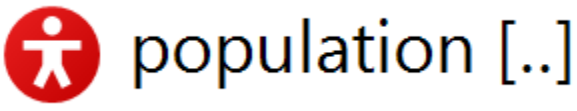


Figure 3.8 - Population Block

Populations are a special type of collection within AnyLogic aimed at storing individual agents. A population can be an *ArrayList* or a *LinkedHashSet*, being the former optimized for accesses by index and the former optimized for add/remove operations. In the case study system model, twelve populations were used for ease of information access and manipulation. Each of these populations and respective storage objective is listed in the Table 3.8.

Orders need to be executed when the simulation reaches the release date. An event is used to achieve such triggering.

Figure 3.9 - Event Block

The event block is the simplest way to schedule some action in the model and meets all the requirements to trigger orders. Events can be of three types, according to its trigger condition.

Table 3.8 - Populations and objective in Simulation model

Population name	**Objective**
machinesWC	Store the WorkCentre machines entities for simpler access to get information of setup times.
productsInProduction	Store the products in production where only products in WorkCentres 1 to 4 are considered to be in production (Cutting, Bending, Tooling and Welding WorkCentres)
productsInElementarStock	Store the products considered to be elementary (do not have product dependencies) that are in inventory
productsInCompositeStock	Store the non-elementary products that are in inventory. These products are the standard components mentioned earlier in this chapter
productsInPreAssembly	Store the products that are being operated in WorkCentre 5 (Pre-assembling WorkCentre)
productsInAssembly	Store the products that are being operated in WorkCentre 6 (Assembling WorkCentre)
deliveredProducts	Store the final products that have already been fully produced and are considered delivered
ordersOnHold	Store the orders whose release date has already occurred but could not yet be executed due to lack of inventory of its product's dependencies, stopping it from being executed
ordersOnExecution	Store the orders that are currently being executed
reordersOnExecution	Store the reorders that are currently being executed
reordersOnHold	Store the reorders that could not yet be executed due to lack of inventory of its product's dependencies
finishedOrders	Store the fulfilled orders

Timeout triggered events occur exactly in timeout time after it is started and it can expire once or occur cyclically or even be fully controlled by the user. *Rate triggered events* intent to

model a stream of independent events (Poisson stream) and are often used to model arrivals. Such an event is executed periodically with time intervals distributed exponentially with the parameter rate. If the rate is x, the event will occur on average x times per time unit. *Condition triggered events* are triggered when a certain condition becomes true. The first type of event, *timeout triggered event*, was used to model the orders trigger. Its first occurrence is the initial instant of the simulation run and the recurrence time is based on the period duration, recurring every period. Whenever the current period equals any order *releaseDate*, the event places that order in the *ordersOnHold* population and attempt to execute it. The order execution mechanic will be explained in detail later in this chapter.

3.4.2 Process Modelling Library

The PML agglomerates many blocks from which only a few were useful and required to model the case study system. First, it was necessary to input entities into the system. These entities represent products and are modeled using agents, as explained previously. To do so the Enter block was used.

Figure 3.10 - Enter Block

This block was used as the system's inputs. It is used five times: input to the first four WorkCentres (production area); input to the stock of elementary products; input to the Pre-assembling WorkCentre; input to the stock of standard components; and input to the Assembling WorkCentre. The Block in itself was left unchanged aside from the Agent type that was changed to be Prod. The agent's insertion was made from other blocks or functions and will be explained further ahead.

Agent removal from the previously indicated areas of the simulation model two techniques were applied: direct remove using Exit blocks, and removal through code.

Figure 3.11 - Exit Block

Exit blocks allow programming decision on what to do with the exiting agents. In the simulation model in analysis, the agents are removed from their previous population, added to a different one and moved to the new section of the system's logical flow.

In order to operate products, machines are necessary. Each WorkCentre is composed by one or more machines of a different type. These groups of machines are modeled using the ResourcePool Block in PML.

Figure 3.12 - ResourcePool Block

The ResourcePool block is an agglomerate of resource type agents. The number of resources comprised in such block is defined by the field *Capacity*. In this dissertation context, resources are machines and they differ between WorkCentres. Nonetheless, there is only one critical parameter for the simulation model, the setup time. Due to lack of time and to invest in a simplistic but functional model, setup times are machine dependent but not dependent on the operation sequence. The setup time and capacity values are both extracted from excel files, namely *Info* and *setupTs*.

This block has several capabilities, being the most important ones related to tasks. Besides the normal operating task, AnyLogic provides tasks that translate natural occurrences on the resources: maintenance, shifts and breaks/failures. Additionally, the user is capable of defining a custom task either by code or flowchart. These tasks are time triggered, either deterministic or probabilistically.

Having the resources modeled it is necessary to use them. The operations to perform are relatively simple and can be defined with resource to a processing time, requiring no other actions than a delay. For such operations, PML has a block name Service that seizes a resource, simulates the operation with resource to a user-defined delay and later on releases the resource seized to operate that agent.

Figure 3.13 - Service Block

Moreover, this block also comprises an entrance buffer to store on wait entities and its capacity is also user-defined. The ResourcePool to use is also defined within this block as well as the number of such resources to be used per agent. The utilization of different resource types is possible consisting on a network of resources working in cooperation. Service block's most important features are actions executed when a certain agent seizes a resource unit and when a certain agent leaves the block. The former allows for the consideration of setup times when the product to operate changes while the latter is used to update agent Prod's information related to the next WorkCentre, operation and respective processing time on its production route.

Between each WorkCentre there is a decoupling point in the form of a buffer. In AnyLogic buffers are represented with Queue blocks.

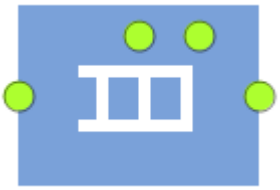


Figure 3.14 - Queue Block

Queues can be dimensioned as limitless or with a user-defined capacity. When the size of buffers is not a primary issue, limitless capacity might be used for error prevention. The queueing can be one of four types: First In First Out (FIFO), priority-based, agent comparison, or Last In First Out (LIFO). Besides decoupling points, inventory storage was implemented using Queue blocks. The chosen type was FIFO as it is the better fit considering the nature of the problem.

Processing paths vary based on product type. To implement the path decision moment, the Select Output5 block was used. This block consists on one *in* port and five *out* ports chosen based on four conditions upmost plus an else condition.

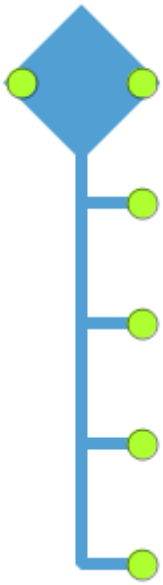


Figure 3.15 - Select Output5 Block

This specific block is not ideal. AnyLogic only provides two base decision-based flow blocks, this one and a two *out* ports one. There is no customizable option besides combining both to obtain the desired number of possible different paths. Such option is neither visual nor comprehension friendly. However, for advanced users, it is possible to fully define a new custom block with the desired amount of *out* ports. The use of this block in this model was not fully efficient since there where unused *out* ports. When the agent enters this block, the *out* port used for its exit is decided based on its parameter *next_WC*, referring to the next WorkCentre on its route.

In situations like the storage of inventory, it is necessary to keep products in the Queue block for an indefinite amount of time. To accomplish so, Hold block was used.

Figure 3.16 - Hold Block

Hold block might begin blocked or unblocked, which translates in whether the first agent to reach the exit moment of the previous block will be prevented from leaving that block or not. AnyLogic provides three Hold modes: Manual, Block automatically after N agents, and Conditional. The first mode is used with resource to the functions *block()* and *unblock()*. The second mode is self-descriptive and the *unblock()* function is used to unblock the Hold block. The third mode evaluates a condition for each agent on the enter moment and either blocks or allows its passage depending on the result of such evaluation.

In the storage of inventory, the Hold block is placed after the Queue block and was used in Manual mode and set to initially blocked. Furthermore, its *out* port is not connected to anything and when a product in stock is consumed it is removed from the Queue block using code. Thus, the Hold blocks used on the model are blocked throughout the entire simulation.

Intending on measuring product's lead time, another pair of blocks was used.

Figure 3.17 - Time Measure Start and Time Measure End Blocks

This pair of blocks is used for precise time measurement of travel time. Placing the Time Measure Start block at the beginning of the production area and the Time Measure End block at the end of such area, the measured time will be the agent's lead time.

3.4.3 Flowchart composition of the system

Model implementation started with the connection of PML blocks in a logic flowchart to represent the system's basic dynamics. The system was interpreted has having two strong decoupling points and working on a double push-pull mode. Consider Figure 3.18. When a client order arrives, the system checks if there is enough inventory of its dependencies. If so, it consumes the dependencies and send them directly to the Assembling Area. If the amount of elementary dependencies in inventory is not sufficient, the system will initiate their production in the Production Area. If the existent inventory of non-elementary dependencies is not enough, the system will attempt to produce those dependencies. To do so, it checks if there are sufficient inventory of its dependencies. If not, it repeats the already described behavior until it is able to produce every ordered product. This procedure explanation will be completed in more detail including the presentation of the used functions.

From the analysis of the Figure below, five key areas can be identified: Production, first decoupling point, Pre-assembling, second decoupling point, and Assembling. These decoupling points will be, from now on, named supermarkets as a reference to its storage purpose. Our model is also divided in five logical flowcharts.

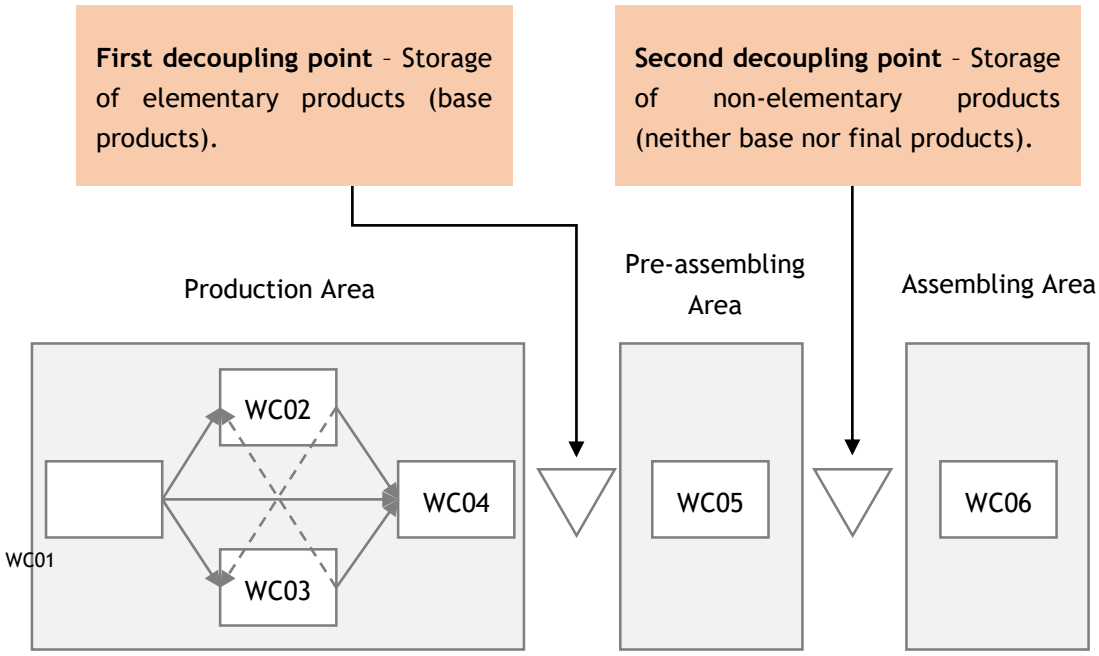


Figure 3.18 - Schematic representation of the system

The first logical flowchart models the Production area basic dynamics (Figure 3.19). Every elementary product starts its production cycle in WC01 where the raw material (metal sheet) is cut. Then, it can proceed to any of the three following WorkCentres. Similarly, from each of those three WorkCentres it can finish it production cycle or head towards the other two WorkCentres.

Each WorkCentre is represented by a Service block. Since the problem is not dimensioning the buffers between WorkCentres, the Service block incorporated queue capacity is considered limitless. The *out* port of each Service is connected to a Select Output5 block that decides on the path to follow based on the parameter *Next_WC*.

Whenever an agent leaves a Service block, three parameters are updated: *Next_WC*, *nextOperation*, and *nextProcessingTime*. Those parameters influence the behavior of the following Service block and the Select Output5 block path decision. When the agent arrives the Exit block it is removed from the population *productsInProduction*, added to *productsInElementarStock* and placed on the next area logical flowchart (first decoupling point) using the function *take(agent)* on the Enter block of the next area.

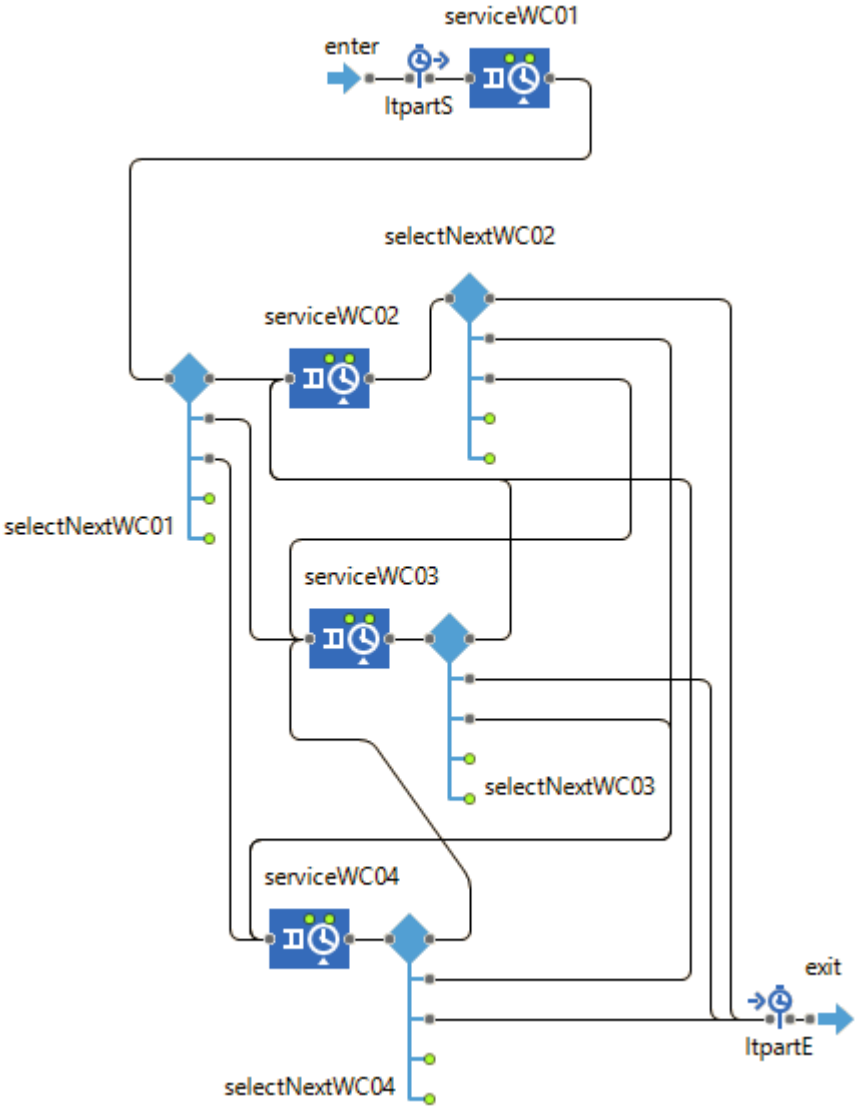


Figure 3.19 - Production area logical flowchart

There is a pair of Time Measure blocks (Start and End) that are used to retrieve the value of each elementary product lead time. When the agent enters the Time Measure End block *ltpartE* its lead time is stored in association with its *id* parameter in a Data Set.

Figure 3.20 - Data Set Block

A Data Set is an AnyLogic data structure capable of storing 2D (X,Y) data of type *double* while maintaining the minimum and maximum of the stored data for each dimension up-to-date. When the X-values record a dependency of Y the Data Set is designated *phased*, which is the case in this situation.

The next logical flowchart area represents the first supermarket and is very simple comprising solely three blocks: Enter, Queue, and Hold.

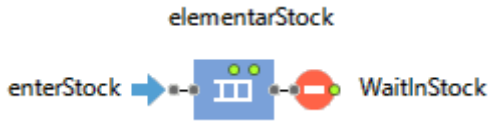


Figure 3.21 - First Supermarket logical flowchart

Produced elementary products are sent from *exit* (Figure 3.19) to *enterStock* (Figure 3.21) from where they enter *elementarStock* buffer (Queue Block). These products, also named basic components, are kept in *elementarStock* using the permanently blocked Hold block *WaitInStock*.

When a basic component enters *elementarStock*, two actions are performed. First, the stock of products of that type (*materialCode*) is incremented. It would be worthless to just update that agent's parameter *stock* as it would not be reflected on all other agents of the same type.

To overcome this problem and to store the base information for each product type obtained from the excel file *Info*, a collection of Prod agents named *products* was created. For each product type an agent is added to that collection. The stock information is always read from and updated on that reference agent. Whenever a product is placed on production, its parameter information is filled using the reference agent of the same type from the *products* collection.

The same concept is also used for orders and a collection named *orders* was used to store all the release orders to be produced. Then, using the already mentioned order triggering event *calendar_orders*, a search through collection is executed and if the release date of an order matches the current period, that order is placed on hold and the function used to execute orders is performed with that order as argument. The function will be explained later.

The other action performed when a basic component enters *elementarStock* consists on checking which reorder generated that component's production and the produced quantity of that reorder is updated. This behavior will be completed during function explanation.

Whenever a higher-level product that as a dependency present in the first supermarket in the necessary amount is placed on execution, such amount of dependencies is removed from *elementarStock* and form the population *productsInElementarStock* using code.

Next on the system's skeleton is the Pre-assembling area, composed of a set of five blocks: Enter, Time Measure Start, Service, Time Measure End, and Exit.

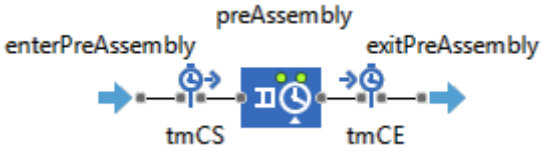


Figure 3.22 - Pre-assembling logical flowchart

Again, as in Production's logical flowchart, the Time Measure pair (Start and End) is used to measure the lead time, in this case of non-elementary products. Likewise, the Service block included queue has limitless capacity. When a product is injected on the Enter block *enterPreAssembly*, it was previously added to the population *productsInPreAssembly*. When a product enters the Exit block *exitPreAssembly* it is removed from the population

productsInPreAssembly, added to the population *productsInCompositeStock* and placed on the second supermarket logical flowchart using the function *take(agent)* in its Enter block.

The second supermarket is modelled as the first supermarket.

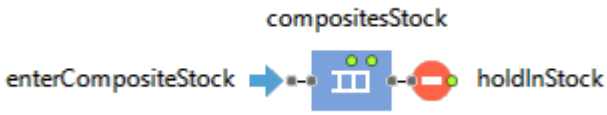


Figure 3.23 - Second Supermarket logical flowchart

Not only are the used blocks the same but the actions performed inside each of them is also the similar.

The final area of the case study system is the Assembling area.

Figure 3.24 - Assembly area logical flowchart

Exit block *output* is the end of the system. On the entrance of a final product the *output* block removes it from the population *productsInAssembly*, adds it to the population *deliveredProducts* and updates the parameter *Amount* from the order that triggered its production by reducing one unit. If the order *Amount* becomes zero, the order is removed from population *ordersOnExecution* and added to population *finishedOrders*.

Services operate using resources, thus there are six ResourcePool blocks in the model, one for each WorkCentre.

3.4.4 Data Input and Output

Data communication was made using excel files. In AnyLogic a model might have multiple Experiments being the default Simulation. Simulation runs on a special agent called *Main*. To perform any action when the simulation begins, the *Main* action *On startup* is used.

The data input starts by reading the number of periods per week and calculating the time per period in minutes dividing the number of working minutes in a week by the number of periods per week. Afterwards, the setup times for each WorkCentre machine is placed on a *HashMap* collection from where it can be easily get using the WorkCentre name. The number of machines is obtained for each WorkCentre and the capacity of the respective *ResourcePool* is set to that value.

The next data input stage is the information gathering for agent Prod, including parameter values for each product type, BOM construction and stock creation (if the system is operating in an ATO strategy) and populating the *products* collection. The startup code continues by getting orders information and populating the *orders* collection.

Data output is performed through the AnyLogic default log and using the Main action *On destroy*. The former type of data output is used for optimization model parameter adjustment through changes on the *Info* and *setupTs* excel files.

3.4.5 Function description

Four global functions were developed for the simulation model. These functions serve different purposes: execute an order (*executeOrder)*, execute a reorder (*executeReOrder*), produce a product (*produceProduct*), and reorder a product (*reorderProduct*). AnyLogic provides a special block for global function implementation.

Figure 3.25 - Function Block

Functions can be a simple action, returning nothing, or return a value based on the action result. A function may receive none or any number of arguments of any type.

3.4.5.1 produceProduct

The *produceProduct* function is the system's base function, being executed in all other three functions. The function takes four arguments and returns an integer value.

Table 3.9 - *produceProduct* arguments

Name	Type	Meaning
prdct	Prod	The Prod object from *products* collection whose *materialCode* matches the order *productCode* parameter
valueToProduce	int	Amount of product prdct to produce
level	int	Used to distinguish between non-elementary components and final products
order	Order	The order that originated this production

In the manufacturing system in study there are three type of products: elementary products that do not depend on any product other than raw materials to be produced; non-elementary components that depend on other products to be produced but are not final products; final products which depend on other products to be produced. The *produceProduct* function needs to be able to distinguish between those types as the production process differs. Elementary

products enter their production in the Production area; non-elementary components enter their production in the Pre-assembling area; final products enter their production in the Assembling area.

The first distinction moment is on whether the product to produce has dependencies or not, which is done by measuring its BOM size. Zero means it is an elementary product, otherwise it

*Pseudo-code 2 – **produceProduct** Procedure ATO strategy (with initial stock)*

```
function produceProduct(prdct, valueToProduce, level, order):
    if prdct.BOM.size()==0 then
        for i=1:valueToProduce do
            Insert a new Prod object equal to prdct in population productsInProduction
            Place that Prod object in Production area
    else
        flag=0
        for all Prod objects p in prdct.BOM do
            if p.stock >= valueToProduce * p.amountComponent then
                flag++
            else
                reorderProduct(p, level+1, order, valueToProduce*p.amountComponent)
        if flag==prdct.BOM.size() then
            for all Prod objects p in prdct.BOM do
                if p.BOM.size()==0 then
                    for j=1:valueToProduce*p.amountComponent do
                        Remove a Prod object with the same materialCode as p from elementary supermarket
                        Remove that Prod object from population productsInElementarStock
                        Reduce the stock of products with the same materialCode as p
                    if p.stock < p.reorderPoint then
                        reorderProduct(p, level+1, order, valueToProduce*p.amountComponent)
                else
                    for j=1:valueToProduce*p.amountComponent do
                        Remove a product with the same materialCode as p from non-elementary supermarket
                        Remove that Prod object from populatoin productsInCompositeStock
                        Reduce the stock of products with the same materialCode as p
                    if p.stock < p.reorderPoint then
                        reorderProduct(p, level+1, order, valueToProduce*p.amountComponent)
            If level==1 then
                Insert a new Prod object equal to prdct in population productsInAssembly
                Place that Prod object in Assembling area
            else
                Insert a new Prod object equal to prdct in population productsInPreAssemby
                Place that Prod object in Pre-assembling area
            return 1
    return 0
end function
```

has dependencies. To distinguish between final products and non-elementary components, the level argument is used: if level value is one, the product is final, else it is not.

The production of an elementary product is simple and consists on the addiction of products from prdct *materialCode* to population *productsInProduction* and their placement in the Production area, in the amount valueToProduce.

However, due to their dependencies, final products and non-elementary components have a more complex production process. First, it is necessary to verify if there is sufficient inventory amount for each of their dependencies. If such verification is positive, the dependencies are consumed in the correct amount and the product is sent to its area (depending on their type) and added to the respective population. If not, those dependencies are reordered and the current order is placed on hold.

Pseudo-code 3 – ***produceProduct*** **Procedure MTO strategy (without stock)**

```
function produceProduct(prdct, valueToProduce, level, order):
    if prdct.BOM.size()==0 then
        for i=1:valueToProduce do
            Insert a new Prod object equal to prdct in population productsInProduction
            Place that Prod object in Production area
    else
        flag=0
        for all Prod objects p in prdct.BOM do
            if p.stock >= valueToProduce * p.amountComponent then
                flag++
            else
                return 0
        if flag==prdct.BOM.size() then
            for all Prod objects p in prdct.BOM do
                if p.BOM.size()==0 then
                    for j=1:valueToProduce*p.amountComponent do
                        Remove a Prod object with the same materialCode as p from elementary
                        supermarket
                        Remove that Prod object from population productsInElementarStock
                        Reduce the stock of products with the same materialCode as p
                else
                    for j=1:valueToProduce*p.amountComponent do
                        Remove a product with the same materialCode as p from non-elementary
                        supermarket
                        Remove that Prod object from populatoin productsInCompositeStock
                        Reduce the stock of products with the same materialCode as p
            If level==1 then
                Insert a new Prod object equal to prdct in population productsInAssembly
                Place that Prod object in Assembling area
            else
                Insert a new Prod object equal to prdct in population productsInPreAssemby
                Place that Prod object in Pre-assembling area
            return 1
```

produceProduct returns zero if the production was unsuccessful and one if it was.

When working in a MTO strategy, there is no safety nor initial inventory. Therefore, the algorithm suffers some changes. Despite checking if the dependencies are all available in the desired quantity, it does not reorder them in the negative case. The order is placed on hold until those quantities have already been produced. Dependency orders are obtained through optimization output. Whenever a non-final order is completed, the final orders on hold are once again attempted to execute and, this time, if there is already sufficient inventory of all their dependencies the productions are executed, otherwise the orders are maintained on hold.

3.4.5.2 reorderProduct

reorderProduct takes four arguments and returns nothing.

Table 3.10 - *reorderProduct* arguments

Name	Type	Meaning
product	Prod	The Prod object from *products* collection whose *materialCode* matches the reorder needs
level	int	Used to pass it as argument for the function *produceProduct* call inside *reorderProduct*
parent	Order	The Order during which execution the reorder was triggered
amountToProduce	int	Amount of product to produce

In section 3.4.5.1, two different behaviors were described for *produceProduct* function based on system's strategy, MTO or ATO. Similarly, *reorderProduct* also behaves differently whether the system's strategy is MTO or ATO. If the system is working based on a ATO strategy, reorders are triggered when the stock of a product reduces bellow a predetermined value and the value to produce is calculated to be high enough so that when the reorder is fulfilled, the inventory level of such product is, on average, in the desired level. Therefore, simultaneous reorders of the same product type are not allowed.

The *reorderProduct* function, for the ATO situation, checks if there is a reorder of that product already being executed (Prod's Boolean parameter *reordered*). On a positive situation, current reorder is ignored. Otherwise the reorder is created, added to population *reordersOnExecution* and attempted to execute using function *produceProduct* where the *valueToProduce* argument is defined according to the product to produce (ignoring the input argument *amountToProduce*. If, for some reason, function *produceProduct* return is zero, the reorder is removed from population *reordersOnExecution* and added to population *reordersOnHold*. The argument *parent* refers to the order during which execution the current reorder was triggered. If the reorder is created, *parent*'s parameter *reordersGen* is incremented to hold the amount of reorders that order generated.

For the MTO situation, *reorderProduct* is ignored as dependencies production orders are obtained from optimization and the function is not used in the simulation.

*Pseudo-Code 4 – **reorderProduct** procedure ATO strategy*

```
function reorderProduct(product, level, parent, amountToProduce):
    if product.reordered == false then
        amountToProduce = value defined for products of type product.materialCode
        Insert a new Order object in population reordersOnExecution
        parent.reordersGen++
        production = produceProduct(product, amountToProduce, level, created Order object)
        if production == 0 then
            Remove the Order object previously inserted in population reordersOnExecution
            Insert that Order object in population reordersOnHold
end function
```

3.4.5.3 executeOrder and executeReOrder

executeOrder takes one argument and returns an integer value. The argument, named *order*, is of type Order and represents the order to execute.

This function is used to execute an Order, either because the current period equals such order *releaseDate* or because the Order was previously unsuccessfully executed and placed on hold and the conditions for its execution are now met.

executeReOrder is based on *executeOrder* with some minor differences. Instead of placing the order in populations related with order tracking, places it in populations related with reorder tracking. The other difference is on the value of the third argument used on function *produceProduct*. Since reorders are always of non-final products and orders are always of final products, the *produceProduct level* argument takes the value two inside *executeReOrder* and value one inside *executeOrder*. As referred in 3.4.5.1, this argument represents whether the product to be produced is a non-elementary component or a final product, changing the behavior of *produceProduct*.

*Pseudo-Code 5 – **executeOrder** procedure*

```
function executeOrder(order):
    Remove order from population ordersOnHold
    p = element from products collection whose materialCode matches order.productCode
    result = produceProduct(p, order.Amount, 1, order)
    if result == 0 then
        Insert order in population ordersOnHold
        return 0
    else
        Insert order in population ordersOnExecution
        return 1
end function
```

*Pseudo-Code 6 – **executeReOrder** procedure*

```
function executeReOrder(order):
    Remove order from population reordersOnHold
    p = element from products collection whose materialCode matches order.productCode
    result = produceProduct(p, order.Amount, 2, order)
    if result == 0 then
        Insert order in population reordersOnHold
        return 0
    else
        Insert order in population reordersOnExecution
        return 1
end function
```

3.4.6 Order and reorder control

This subsection aims to explain the implemented dynamics that did not fit the previous subsections, namely the order and reorder control.

In 3.4.5, it was explained that whenever the result from *produceProduct* was zero the order (or reorder) that triggered such function was placed on hold. However, it is necessary to explain how and when would those orders (or reorders) be triggered again. Again, the method changes according to system's strategy.

In the MTO situation, whenever a product reaches one of the supermarkets, the parameter *Amount* from the reorder that originated that product is decremented. If *Amount* equals zero the reorder is complete. Once a reorder is complete, its *parent* parameter *reorderCalls* is incremented. Whenever *reorderCalls* form an order (or reorder) reaches the value *reordersGen*, it means that the amount of reorders triggered by that order (or reorder) is complete. Hence, the function *executeOrder* (or *executeReOrder*) is performed.

In the ATO situation, reorders are triggered from inventory level instead of being triggered directly by orders. Therefore, the triggering of on hold orders and reorders is different. Again, whenever a product reaches one of the supermarkets, the parameter *Amount* from the reorder that originated that product is decremented. If *Amount* equals zero, the reorder is complete. Nevertheless, triggering occurs every time a product enters the supermarkets. If there is an order (or reorder) on hold that requires that type of product to be executed, *executeOrder* (or *executeReOrder*) is performed. If the inventory level of all components is sufficient, the order is placed on execution, else it is kept on hold.

3.5 Scheduling/Sequencing

Previously, the input mechanism was explained, including the order reading. Orders are read from an excel file (optimization results) and attempted to execute when its release date occurs. Orders from the same period are executed on the same sequence they are presented on the excel file. If the product the order is attempting to produce has dependencies and the

system is working under a MTO strategy, the dependencies will be attempted to produce following the sequence they were read from the excel file.

In simulation, setups are implemented and occur whenever the *materialCode* parameter of the product to produce in a WorkCentre changes. Bending WorkCentre is the system's bottleneck essentially due to its long toolkit set changeover times, i.e., its setup times. Therefore, reducing the number of setups is highly benefic to improve system's efficiency.

Scheduling allows for an optimized sequencing of production orders, optimizing each WorkCentre utilization. Scheduling rules and techniques are vast, nonetheless, most do not consider the system's status and are more of general use. A new technique was developed to address the case study type of problems.

Figure 3.26 helps illustrate how products are placed in production.

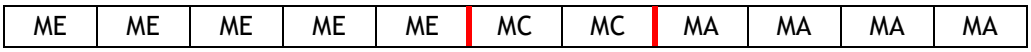


Figure 3.26 - Production sequence (with scheduling)

Figure 3.26 represents the queue at the entry of a determined WorkCentre and the first element in the queue is at the rightmost one, MA in this case. Within the same period, production orders are read alphabetically, meaning that the sequence will be sorted that way, without a scheduling technique. Setup times are considered sequence independent in our methodology. Thereby, the specific sequence inside a period is irrelevant. However, the first element in the queue for each period plays a major role in reducing setups. If the last product in queue referring to period *t* equals the first product to be produced in period *t+1*, one setup has been avoided. Moreover, if the queue is empty but the sequence can start with a product of the same type as the last produced product in the subsequent WorkCentre (or the type of the current product in production in that WorkCentre), a setup has also been avoided.

The scheduling algorithm structure is described next in Figure 3.28 and Pseudo-Code 7.

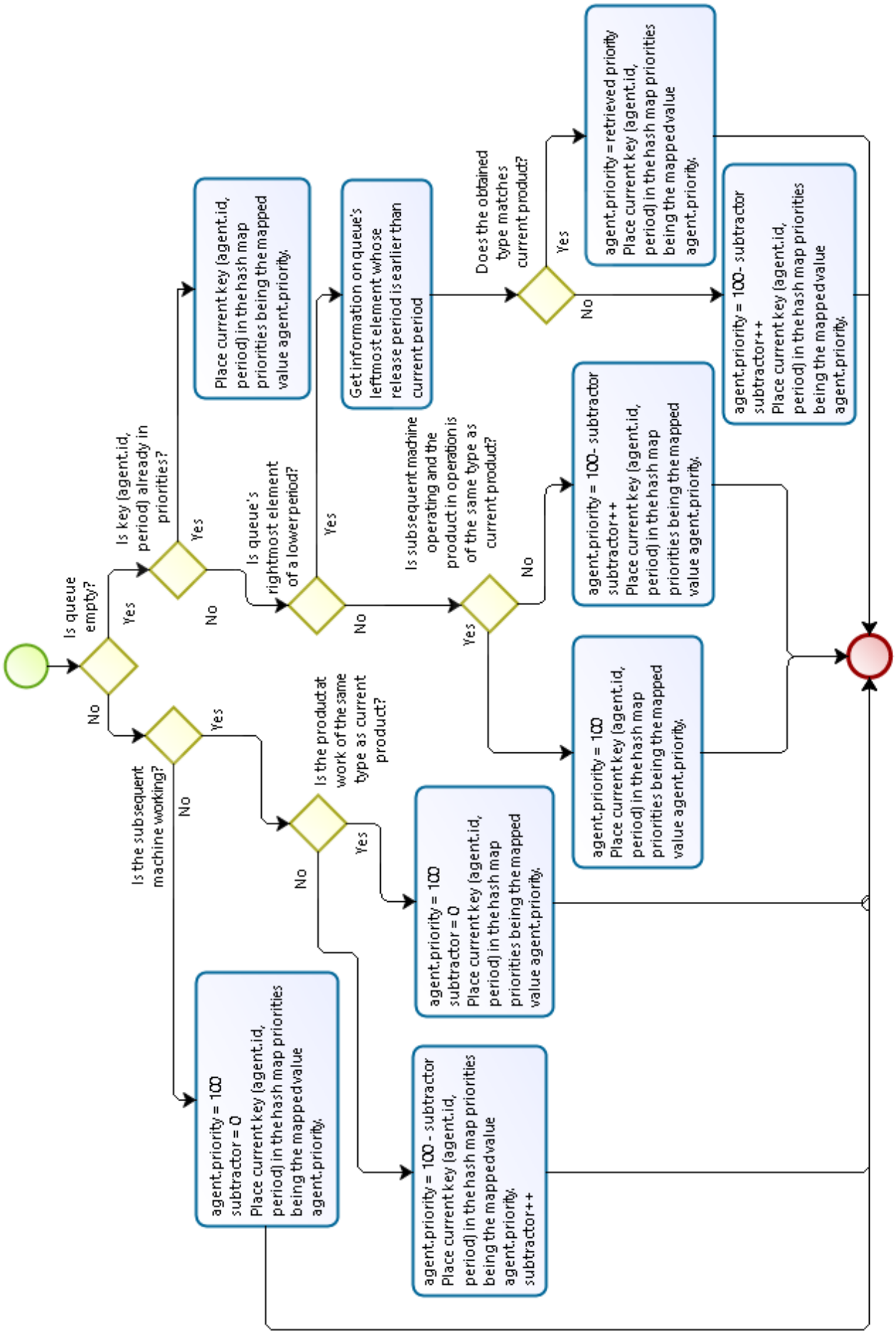


Figure 3.27 - Scheduling algorithm flowchart

There are seven moments of decision, represented by the yellow decision blocks in the flowchart.

The first one is whether or not the queue is empty. The scheduling algorithm aims at grouping similar products within periods.

If the queue is empty, the first product position can only depend on the WorkCentre. If there is a product being produced and the types differ, the *priority* of the entering product is not maximum following a rule of *maximum-variable* where this variable is named *subtractor*. Whenever a new *priority* is defined, *subtractor* is incremented and that *priority* is placed on a structure named *priorities* with the key (*id*, *period*). The product *id* represents the type of product (parameter *id* of Prod object) and *period* represents the period when it was inserted in queue.

Otherwise, if the type of product in production is the same of the product to be placed in

Pseudo-Code 7 – **Scheduling Algorithm**

```
function schedule(enteringProduct, WC, priorities,subtractor, period)
    if WC.queuesize() == 0 then
        if WC.delaysize() > 0 then
            if first product to enter production is of the same type as enteringProduct then
                enteringProduct.priority=100
                subtractor=0
            else
                enteringProduct.priority=100-subtractor
                subtractor++
        else
            enteringProduct.priority=100
            subtractor=0
    else
        if priorities.contains((enteringProduct.id, period)) then
            enteringProduct.priority=priorities.get((enteringProduct.id, period))
        else
            if there are products of earlier periods in queue then
                get information on the lower priority product of an earlier period
                if enteringProduct type equals the obtained product type then
                    enteringProduct.priority = obtained_priority
                else
                    enteringProduct.priority=100-subtractor
                    subtractor++
            else
                if first product to enter production is of the same type as enteringProduct then
                    enteringProduct.priority=100
                else
                    enteringProduct.priority=100-subtractor
                    subtractor++
    priorities.put((enteringProduct.id, period), enteringProduct.priority)
end function
```

queue, this product's *priority* will be maximum so that it goes to production right after the current product operation is finished, avoiding a setup. *Subtractor* is reset to zero.

If the WorkCentre is also empty, the first product to enter the queue has maximum *priority* and the *subtractor* is reset to zero.

On the contrary, if the queue is not empty, product's *priority* is defined by the elements in queue. If the key (*id, period*) is already in the *priorities* data structure, the product receives that *priority*. If not, the algorithm verifies if there are products in queue that entered in lower periods.

If there are no products in queue that were released in a previous period, the algorithm verifies if the product in production is of the same type as the one entering the queue. If so, gives maximum *priority* to that product and stores that *priority* in *priorities*. Else, product's *priority* is set to *maximum – subtractor*, *subtractor* is incremented and the *priority* is stored in *priorities*.

If there are products from previous periods in queue, the algorithm obtains the information of the lowest *priority* product in queue that was released on a previous period and compares the types. If they match, attributes that product's *priority* to the entering product and saves it in *priorities*. Otherwise product's *priority* is set to *maximum – subtractor*, *subtractor* is incremented and the *priority* is stored in *priorities*.

Value 100 is used for maximum *priority* as an example, being the algorithm completely independent of the used value since the parameter *priority* can be either positive, zero or negative and the *priority* comparison is made between products and not related to an external factor. If all priorities are defined based on the same referential, that referential value is irrelevant.

3.6 Simulation and Optimization Interaction

Simulation and optimization models interact in various ways. Optimization results are simulation orders input. Considering system status and the production plan, simulation performs the developed scheduling technique, improving WorkCentre utilization. Simulation, whenever the optimization results were too optimistic and order due dates were not fully satisfied, influence optimization WorkCentre capacity parameters to tighten its constraints and attempt to produce a less optimistic production plan. After the first run, simulation provides the lead time inputs for optimization. Finally, when a production plan is validated by simulation, the plan is altered to comprise the scheduling made during the simulation execution.

Order reading is direct: the period in which optimization plan releases an order is the same period when the simulation process will release that same order. However, for the products with dependencies, if there is not sufficient dependencies inventory to fulfil the order, the order is placed on hold until it can be performed.

The lead times provided from simulation are the average of the measured lead times for each product. The average was chosen instead of, for instance, maximum to diminish the oscillation effect between iteration solutions. The tightening is, therefore, controlled and smoother than with the maximum option.

WorkCentre capacity-related parameters are changed whenever the optimization production plan was not validated by simulation and for WorkCentres that verified an average utilization above 80%. WorkCentres in high average utilization were chosen as those are the

more likely to have had the most diverging behavior from what was expected in optimization. This parameter adjustment was made multiplying the initial capacity by the proportion obtained from the following formulae (11).

$$adjust = \frac{expected\ duration}{expected\ duration + \frac{\sum_{unmet\ due\ dates} delivery\ period - due\ date}{number\ of\ unmet\ due\ dates}} \quad (11)$$

This adaptation incorporates the average delay for the unmet due dates (tardiness).

When decreasing capacity from an iteration to the next, the adjustment is made as described. However, when increasing capacity, the adjustment is smoothed, as in (12).

$$\frac{current\ adjust - previous\ adjust}{2} \quad (12)$$

Both capacity parameters and lead time estimates are adjusted by simulation. When simulation tries to increase capacity, it means the previous optimization solution had excessively conservative constraints, i.e., either the WorkCentre capacities were underestimated or the lead times were overestimated (or both). Therefore, the total number of orders and per period was below the capacity of the system, resulting in a smaller amount of WIP, leading to shorter average lead times. Harsh adjustments in WorkCentre capacity, associated with lead time measurements, could lead the approach towards an oscillatory behavior between excessively conservative and excessively optimistic solutions. (12) is used to smooth adjustment and avoid/reduce such oscillatory behavior.

3.6.1 Stopping Criteria

The iterative nature of the proposed approach requires the approach to finish under some condition. However, one single condition was considered not to be sufficient to tackle every possible end scenario. Therefore, a set of conditions was developed for the approach to address the verified situations during development and test phases.

Table 3.11 instantiates all implemented conditions that lead to the completion of the proposed approach.

The three conditions do not have the same priority. The main objective of the proposed approach is to aid decision making by proposing feasible production plans and schedules. Therefore, the highest priority condition is demand being met (“Demand was met” in Table 3.11). The other two conditions are mutually exclusive as if the approach is cycling between two solutions where one is optimistic and the other rather conservative, the approach is not outputting the same solution consecutively.

Table 3.11 - Stopping conditions

Condition	Scenario where it is applicable
Solution is cycling	Methodology is iterating between two results. Might happen when the ideal results are very close to both solutions and consecutive increases and decreases in constraints lead to more or less optimistic results from optimization.
Sequence of iterations without significant changes	Three iterations were executed without significant changes in output (lead times, production plan, and simulation evaluation changed less than 3%). Might happen when demand is impossible to fulfill or period duration does not allow for better results due to lack of precision.
Demand was met	If all demand is met under the obtained and validated plan, methodology might terminate as its objective is attained.

No time related condition was implemented as run time is related with period duration and average bottleneck utilization, as seen in chapter 4, and a generalized time related condition was not possible develop.

The proposed approach is analyzed in the following chapter.

Chapter 4

Approach Assessment

With the dynamics of the case study system presented and the methodology described, the approach has been evaluated in terms of computational performance and quality of the proposed solutions, considering the impact of different scheduling rules and by changing the time discretization of the MILP model.

4.1 Impact of different period durations on the approach performance

Period duration variation has been introduced and briefly explained in section 3.3.5. In this section, a detailed analysis of this solution strategy is presented.

The period duration is associated with the number of periods per time interval. A larger amount of periods per time interval translates in smaller periods and increases the model precision. Nevertheless, the amount of periods also decreases the performance of the MILP model by increasing the number of integer variables. Therefore, a balance between model precision and performance must be attained.

Time discretization requires the conversion of lead times from time units to time periods. If a certain lead time is an exact multiple of period duration, there is no problem as the conversion is accurate. However, it is highly unlikely to verify such coincidence and the more natural occurrence is to have a lead time that is not an exact multiple of the period duration. In those cases, the conversion to periods is made such that the lead time in periods is equal or greater than the real value of the lead time.

For instance, if a determined product lead time is 500 minutes and the period duration is 240 minutes, lead time in periods is 3, corresponding to 720 minutes. In this situation the converted lead time is excessive by 220 minutes. For the same lead time, if the period duration is 120 minutes, lead time in periods is 5, corresponding to 600 minutes. In this situation the converted lead time is excessive only by 100 minutes. The increase in precision is notorious.

For this example, one could argue that a period duration of 250 minutes would be more adequate as it would allow for an exact representation (2 periods would match exactly 500 minutes). This affirmation is fairly correct however, considering this as one isolated example, one cannot make the conjecture that higher period durations are benefic. The next example will support this statement.

In systems having product interdependency, lead times are used to calculate when to trigger the production orders for such dependencies. Considering the BOM structure presented in the next figure, product MA would have to be launched in production 500 minutes prior to product MB, which, in its turn, would have to be launched in production 300 minutes prior to product MC. Therefore, product MA would have to be launched in production 800 minutes prior to product MC.

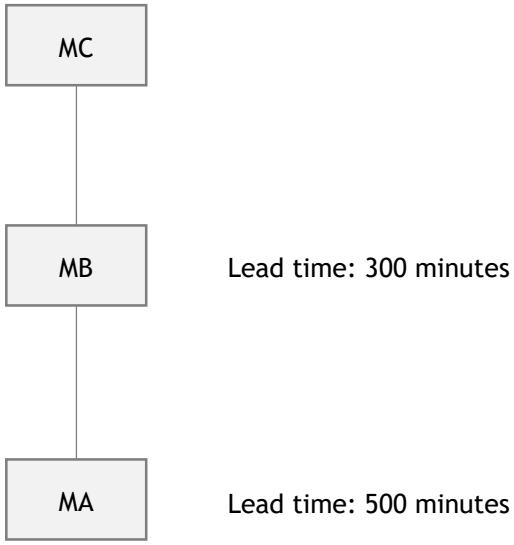


Figure 4.1 - BOM structure example

The following table represents the period conversion and total time equivalence for the three referred period durations.

Table 4.1 - Lead time conversion comparison

Period Duration	MA Lead time		MB Lead Time		Total Lead time		Excess
	Periods	Minutes	Periods	Minutes	Periods	Minutes	Minutes
120 min.	5	600	3	360	8	960	160
240 min.	3	720	2	480	5	1200	400
250 min.	2	500	2	500	4	1000	200

As expected, if lead times are not exact multiples of period duration on its majority, smaller period durations provide higher precision. When the number of possible combinations is very high, the likelihood of lead time match with a multiple of period duration is extremely low.

Additionally, lead time conversion has another impact on optimization behavior, besides the calculation of the triggering moment for the production order of dependencies. If the period duration is very high, even if the dependency lead time is very low it will be converted to at least 1 period. Products can only be produced when their dependencies are fully produced (which happens in moment of production plus lead time). If the converted lead time is very excessive, backlog might occur. In more detail, consider the following two situations:

- Product MB has product MA as its dependency; lead time of product MA is 100 minutes; period duration is 2400 minutes; demand for 1 unit of product MB exists at the end of the week (consider a week to have 2400 working minutes)

- Product MB has product MA as its dependency; lead time of product MA is 100 minutes; period duration is 240 minutes; demand for 1 unit of product MB exists at the end of the week (consider a week to 2400 working minutes)

Both situations differ only on period duration and consequent number of periods per week. In the first situation the number of periods per week is 1 whereas in the second situation the number of periods per week is 10. The following tables represents the orders, inventory, backlog and demand for that week considering only the referred products for each of the situations.

Table 4.2 - Situation 1 Optimization output

	MA	MB
	1	1
Production orders	0	0
Inventory	0	0
Backlog	0	1
Demand	0	1

Table 4.3 - Situation 2 Optimization output

	MA										MB									
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
Production orders	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
Inventory	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Backlog	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Demand	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

In situation 1, lead time is converted to 1 period. The excess time is 2300 minutes. Since the week only has 1 period and product MB can only be produced in the period after product MA is produced (which takes lead time, in this case 1 period), product MB cannot be produced and will be placed in backlog. Since there is no production order for MB, product MA production order will not be triggered.

In situation 2, lead time is converted to 1 period. The excess time is 140 minutes. Since the week has 10 periods, product MA can be produced 1 period prior to product MB, originating the result exposed in Table 4.3.

Product MB is considered to take less than one period to be produced and demand must be fulfilled at the end of the period.

The given example serves to illustrate the second problematic originated by low precision lead time conversion. In addition, the given example serves to exclude cases where there is a small number of periods per week from future tests, as the testing instance includes products with multi-level dependencies.

Variation of period duration also impact costs. For instance, inventory costs for a 120 minutes' period cannot be the same as inventory costs for a 240 minutes' period. Inventory cost is related to period duration. There is a base value of inventory costs per week. Depending on period duration, such value is divided by the number of periods per week. Production costs do not change with period duration as the cost is not related with a time interval but with the

actions inherent to production. Backlog costs are related to unmet due dates and are constant for every period duration to represent the necessity to fulfill the highest number of customer requests possible.

4.2 Proposed Scheduling vs FIFO

From the explanation given in section 3.5, the proposed scheduling technique presents advantages facing the FIFO strategy, implemented by default in AnyLogic. This subsection is dedicated to testing and analyzing the same instances in each of the strategies.

The following tables present the common data and test information. Note that customer demand exists only for final products and at the end of each week. Costs are considered equal for every period for the tests.

Table 4.4 - List of products

MA	MC	ME
MS	MR	MT

Table 4.5 - Immediate product dependency

Product Code (parent)	Product Code (dependency)	Quantity
MS	MA	2
MS	ME	3
MR	MA	1
MR	MC	2
MT	MA	1
MT	ME	2
MT	MS	1

Table 4.6 - WorkCentre information

WorkCentre	Number of machines	Working minutes per week per machine
C01	2	2400
C02	4	2400
C03	1	2400
C04	1	2400
C05	3	2400
C06	2	2400

Table 4.7 - Product related costs

Product Code	Production cost	Inventory cost	Backlog cost
MA	100	50	-
MC	145	73	-
ME	90	45	-
MS	200	100	-
MR	115	58	6900
MT	150	150	11000

Table 4.8 - Operation Sequence

Product Code	Operation	Next Operation	WorkCentre	Processing Time
MA	P01	P02	C01	5
MA	P02	P03	C02	20
MA	P03	P04	C04	3
MA	P04	-	C03	4
MC	P01	P02	C01	2
MC	P02	P03	C04	5
MC	P03	P04	C02	30
MC	P04	-	C03	5
ME	P01	P02	C01	5
ME	P02	-	C02	10
MS	P01	-	C05	5
MR	P01	-	C06	3
MT	P01	-	C06	2

Table 4.9 - WorkCentre setup times

WorkCentre	Setup time
C01	1
C02	6
C03	2
C04	1
C05	1
C06	3

Table 4.10 - Customer demand

Product Code	Week 1	Week 2	Week 3
MR	40	45	45
MT	55	50	50

Four metrics were used to compare the two scheduling rules:

- Number of setups performed in the bottleneck WorkCentre
- Total time necessary to execute all orders
- Percentage of orders that did not met their due date
- Tardiness

Each test is made twice, once for the first iteration of the proposed methodology were lead time estimates are direct input from the user; and a second time for the last iteration of the proposed methodology, happening after a stopping criteria is met.

Whenever the proposed approach initiates its execution, lead time estimates are required to be inputted by the user. Those lead times can be classified as average, pessimistic or optimistic, according to the difference from the empirically known lead times.

Average estimates on lead times are those that are not far from the values empirically obtained. The same way, pessimistic estimates are those that are much higher than the values empirically obtained. Finally, optimistic estimates are those that are significantly lower than the values empirically obtained.

A total of three tests were performed on the same instance (considering costs, capacity, demand and BOM structures) but with different period duration and initial lead time estimates, as explained above.

Table 4.11 - Test information (scheduling comparison)

Test number	Iteration	Period duration	Initial lead time estimates
1	First	120	Average
	Final		-
2	First	240	Pessimistic
	Final		-
3	First	240	Optimistic
	Final		-

The size of the instance, considering the number of variables in the optimization model is defined in Table 4.12.

Table 4.12 - Instance size on variables

Test	Period duration	Integer variables	Binary variables
1	120	1074	320
2 and 3	240	493	147

Numerical results are listed in Table 4.13.

Table 4.13 - Test results (scheduling comparison)

Test - iteration	Scheduling Rule	Number of bottleneck setups	Total execution time (min.)	Percent of delays	Tardiness (periods)
1 - First	Proposed	338	7349	45,1%	1,43
	FIFO	442	7503	54,9%	1,92
1 - Final	Proposed	192	7105	0	0
	FIFO	212	7105	0	0
2 - First	Proposed	190	6989	0	0
	FIFO	274	6989	0	0
2 - Final	Proposed	165	7097	0	0
	FIFO	238	7207	3,33%	1
3- First	Proposed	191	7141	6,67%	1
	FIFO	273	7265	13,33%	1
3 - Final	Proposed	165	7097	0	0
	FIFO	238	7207	3,33%	1

Results show that the proposed strategy reduced the number of setups in the bottleneck. The setup count is performed only for the bottleneck WorkCentre because it is the WorkCentre

with the longer setup times. Plus, the queue preceding that WorkCentre is the most likely to fill and where sequencing would have higher impact.

The proposed scheduling rule also seems to generally outperform the FIFO strategy in terms of total execution time, except in test 3. This test started with pessimistic lead time estimates, thereby the optimization results were very conservative. Backlog was verified at the end of each week (the backlog at the end of week 3 was final as there is no more periods where those orders could be produced). This result was also conservative in the moment when dependencies were placed on production. Since lead times were considered very long, dependencies were placed on production in very early periods. When optimization results were inputted in the simulation model, the real lead times were shorter. Hence, dependencies would wait longer in the supermarkets. Execution time was, therefore, not defined by the variations in waiting time in production but in waiting time in supermarket and by the production calendar.

The second and third tests was based on the same instance and period duration but with different initial lead time estimates. For both tests, the proposed methodology finished with the same solution, thus results for test 2 and 3 on the final iteration are the same. However, the second test took more iterations to attain the same results.

Comparing delay information, both percentage of delays and delay duration, the proposed strategy presents an overall superior behavior.

The first test, for the first iteration, presents high levels of delays. The smaller period duration increases the precision of the optimization model, as previously stated. Smaller period duration results in higher number of periods per week increasing the solver flexibility. Therefore, despite starting with average lead time estimates, first iteration results were too optimistic as a small difference in lead time could change its value in the conversion to periods. Plus, an order is considered delayed when it is delivered one period later. If the order was released in *t*, it is expected to be delivered in *t+1*. A delay is considered when an order is released in *t* and the delivery date of the final product from that order is delivered on a period superior to *t+1*. Since periods are smaller, deviations are more noticeable and due dates are stricter.

4.3 Result Analysis of the period length

Besides affecting costs and working minutes per period, period duration affects the performance of the proposed methodology. The decrease in period duration will, in theory, increase the model precision, therefore improving its performance. This assumption was putted to test using data presented in Tables 4.4 to 4.10. Demand values were designed to provoke a high bottleneck utilization situation, forcing the system to work close to its limit. In this type of high utilization scenarios, the optimization model takes longer to solve. In section 4.4, this effect will be tested and analyzed. The initial lead time estimates were 500 minutes for products MA, MC and ME, and 50 minutes for products MS.

Three cases were considered, where period duration is 480 minutes, 240 minutes, and 120 minutes. To compare the three scenarios (that only differ on period duration) five Key Performance Indicators (KPI) were used: total cost of final plan; percentage of delivered products; total run time of the methodology from start to finish; number of iterations the proposed approach had to execute to reach the final result; criteria that triggered methodology termination (presented in section 3.6.1).

Table 4.14 - Period duration test results

Period duration (min)	Cost	Fulfilled orders	Run time (seconds)	Iterations	Stop criteria
480	525955,6	85,96%	369,43	4	Sequence of iterations without significant changes
240	325345,0	96,84%	573,09	6	Solution is cycling
120	325333,9	100%	45552,64	6	Demand was met

As expected, solution quality increases with the decrease in period duration (and consequent increase in number of periods). Total solution cost reduces and the percentage of fulfilled orders increases.

However, run time increases with the decrease in period duration as there are more periods per time interval. Decision variables (production, inventory and backlog) are dependent on both product type and period, therefore, a rise in number of periods increases the number of decision variable and the model to solve grows in size and complexity. Optimization models are known to be harder to solve as size and complexity grows.

For the 480 minutes' period duration scenario, the final solution was found on iteration number two. This can be explained as the lower precision translates in more conservative results. The lead time conversion prevents a higher percentage of customer orders to be fulfilled and the following iterations did not influence the optimization results as lead time conversion must be always done to an integer number of periods.

The best scenario, percentage of fulfilled orders wise, is the 120 minutes' period duration one. However, the run time is considerably higher and might not be practical in real situations of higher system complexity. It is important to note that the high bottleneck utilization imposed by the high customer demand influences run time (96,56% average utilization). Section 4.4 explores this influence.

The number of integer and binary variables increases with the number of periods per week (decreasing with period duration), influencing run times. Table 4.15 matches each tested period duration to the resulting integer and binary variables.

Table 4.15 - Integer and binary variables per period duration

Period duration (min)	Integer variables	Binary variables
480	220	52
240	497	147
120	1074	320

4.4 High bottleneck utilization vs low bottleneck utilization

Preliminary tests pointed towards a relation between WorkCentre average utilization and optimization model run time. Eight instances were putted to test. Data was constant and identical to that of section 4.2, apart from customer demand, which directly influences bottleneck utilization. Period duration was 240 minutes. Lead time estimates were 250 minutes for products MA, MC and ME, and 50 minutes for products MS.

Table 4.16 contains the run time and average bottleneck utilization results for those tests ordered by average bottleneck utilization.

Table 4.16 - Utilization vs run time test results

Test number	Average bottleneck utilization	Optimization run time (seconds)
1	35,96%	0,055
2	56,91%	0,105
3	76,08%	0,180
4	81,07%	1,115
5	86,31%	1,180
6	95,92%	7,673
7	96,56%	29,76
8	96,15%	1,385

The highlighted results correspond to an excessive instance, i.e., customer demand was impossible to be met by the system at study. Considering the first seven instances, run time grows with average bottleneck utilization, being the increase in run time more impactful when the average bottleneck utilization is close to limit.

In the excessive situation, many orders are unable to be fulfilled by the manufacturing system, increasing the number of backlogs. As backlogs are unavoidable in the highlighted situation as the system is physically unable to handle the tested customer demand, the solver does not struggle to find an optimal solution considering no backlog.

Ignoring the eighth test, bottleneck utilization impact on run times could be described as presented in Figure 4.2.

Whenever WorkCentre capacity is attempted to be used close to its limit, optimization run time is expected to increase. Since the proposed approach follows an iterative scheme, the increase in optimization run time significantly impacts its total run time.

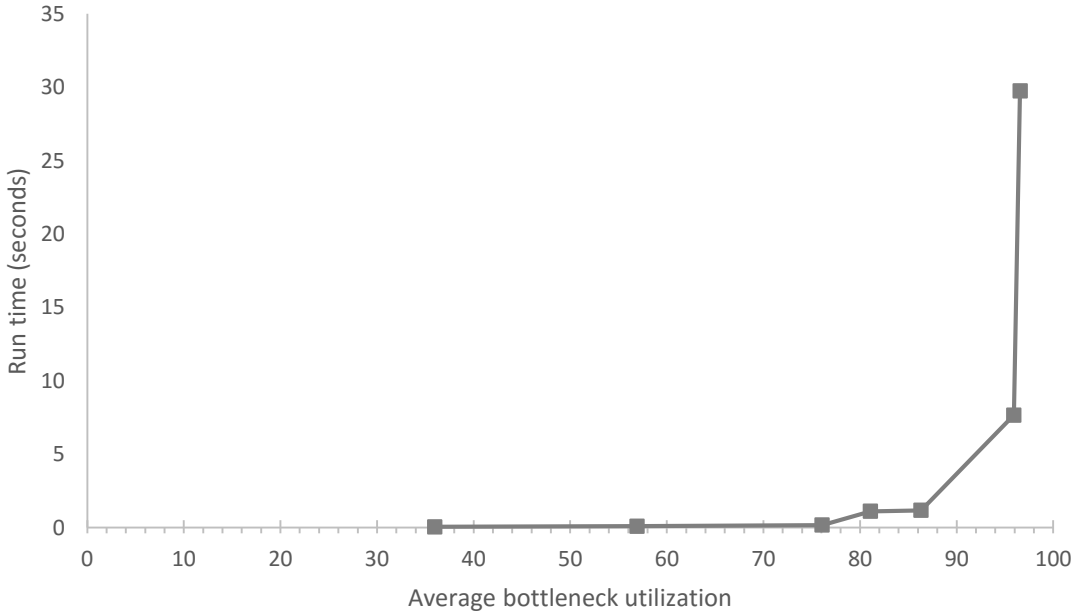


Figure 4.2 - Average bottleneck utilization impact on optimization run time

Chapter 5

Conclusion

Operational efficiency is overall seen as a competitive advantage and companies invest searching and developing methods to increase this efficiency. The growing demand for quality and customization by costumers increased the importance of production planning and scheduling.

This dissertation focused on developing a methodology to aid managers and top level employees in decision making tasks and dealing with both planning and scheduling problems related with manufacturing systems. The objective was accomplished by proposing and developing an iterative hybrid method between optimization and simulation techniques.

Answering the first research question, the proposed approach is based on a simple MIP model and a detailed simulation model of the manufacturing system. The former is used to obtain the production plans while the latter applies the developed scheduling technique to the production plan and evaluates whether or not it is feasible. If not, WorkCentre capacity is adjusted for bottleneck WorkCentres, lead time estimates are updated, and the optimization model is solved again considering the adjusted parameters.

The second research question is related to the interaction between optimization and simulation. In the proposed approach, optimization and simulation interact through parameter and lead time estimates adjustments. Furthermore, simulation performs a scheduling technique during its run using the production plan coming from optimization and the verified conditions of the system.

The proposed methodology proved to be flexible in precision selection through period duration variation, being fit for both high precision and relaxed systems. Furthermore, the proposed methodology, given a sufficient level of precision, was able to provide detailed production plans even under high WorkCentre capacity utilization (96,56% utilization of bottleneck WorkCentre).

The methodology benefits from using a set of stopping conditions instead of a single condition as it contributes to its flexibility and better handling of different situations.

Unlike other hybrid optimization-simulation approaches, the proposed methodology does not force convergence by reducing solution quality as for example [16]. The impact from simulation on optimization only affects lead times, and WorkCentre capacity when that WorkCentre is at high utilization rate.

The proposed methodology studies and develops a method for the incorporation of BOM in recursive approaches while using backlog decision variables.

Most manufacturing systems can be modelled and described using the proposed approach. The mathematical model is easily generalized and adapted to most situations and simulation models can be built considering virtually any dynamics due to technological advancements in the area. The approach is, therefore, applicable to the majority of situations and is expected to provide satisfactory results on tackling production planning problems.

5.1 Further research

Despite the satisfactory results produced by the methodology, it has only been tested on the case study and further testing on different scenarios must be performed to validate and support the approach.

The proposed approach works with unique and defined processing routes for each product. However, in real scenarios, products might follow alternative routes. The implementation of this feature in the methodology would allow for better WorkCentre utilization, deviating products towards WorkCentres with smaller waiting queues/times.

As referred, a balance between period duration and the total run time of the proposed approach must be made. Creating a systematic tool that could analyze the optimization model of the system and suggest a fitting period duration for the run time to be practical based on complexity analysis and average bottleneck utilization would be of great interest. This feature would release users from tuning tasks, increasing the independency of the approach.

Convergence was not proved nor guaranteed and a set of conditions were necessary to prevent an infinite number of iterations when the approach is being applied to certain scenarios under specific conditions. A thorough study on convergence and possible improvements to the methodology would be important to ensure the efficiency of the approach.

The proposed scheduling technique is not generalized and might not be applicable to different scenarios. An intense study on generalized scheduling techniques would have to be performed to allow the proposed methodology to be applied to the most manufacturing systems.

Chapter 6

Bibliography

[1] Hopp, W.J. and M.L. Spearman, *Factory physics*. 2011: Waveland Press.

[2] Ohno, T., *Toyota production system: beyond large-scale production*. 1988: crc Press.

[3] Ashayeri, J. and R. Kampstra, *Demand driven distribution: The logistical challenges and opportunities*. Proceedings of International Trade & Logistics, Corporate Strategies and the Global Economy. Le Havre: University of Le Havre, 2005.

[4] De Toni, A., M. Caputo, and A. Vinelli, *Production management techniques: push-pull classification and application conditions*. International Journal of Operations & Production Management, 1988. **8**(2): p. 35-51.

[5] Goddard, W. and R. Brooks, *Just-in-Time: A Goal for MRP II*. Readings in Zero Inventory, 1984.

[6] Lee, L., *A comparative study of the push and pull production systems*. International Journal of Operations & Production Management, 1989. **9**(4): p. 5-18.

[7] Venkatesh, K., et al., *A Petri net approach to investigating push and pull paradigms in flexible factory automated systems*. International Journal of Production Research, 1996. **34**(3): p. 595-620.

[8] Villa, A. and T. Watanabe, *Production management: beyond the dichotomy between 'push'and 'pull'*. Computer Integrated Manufacturing Systems, 1993. **6**(1): p. 53-63.

[9] Bonney, M., et al., *Are push and pull systems really so different?* International Journal of Production Economics, 1999. **59**(1): p. 53-64.

[10]Missbauer, H. and R. Uzsoy, *Optimization models of production planning problems*, in *Planning Production and Inventories in the Extended Enterprise*. 2011, Springer. p. 437-507.

[11]Pochet, Y. and L.A. Wolsey, *Production planning by mixed integer programming*. 2006: Springer Science & Business Media.

[12]Land, A.H. and A.G. Doig, *An automatic method of solving discrete programming problems*. Econometrica: Journal of the Econometric Society, 1960: p. 497-520.

[13]Little, J.D., et al., *An algorithm for the traveling salesman problem*. Operations research, 1963. **11**(6): p. 972-989.

[14]Figueira, G. and B. Almada-Lobo, *Hybrid simulation–optimization methods: A taxonomy and discussion*. Simulation Modelling Practice and Theory, 2014. **46**: p. 118-134.

[15]Byrne, M. and M. Bakir, *Production planning using a hybrid simulation–analytical approach*. International Journal of Production Economics, 1999. **59**(1): p. 305-311.

[16]Kim, B. and S. Kim, *Extended model for a hybrid production planning approach*. International Journal of Production Economics, 2001. **73**(2): p. 165-173.

[17]Irdem, D.F., N.B. Kacar, and R. Uzsoy, *An exploratory analysis of two iterative linear programming—simulation approaches for production planning*. Semiconductor Manufacturing, IEEE Transactions on, 2010. **23**(3): p. 442-455.

[18]Byrne, M. and M. Hossain, *Production planning: An improved hybrid approach*. International Journal of Production Economics, 2005. **93**: p. 225-229.

[19]Almeder, C., M. Preusser, and R.F. Hartl, *Simulation and optimization of supply chains: alternative or complementary approaches?* OR spectrum, 2009. **31**(1): p. 95-119.

[20]Lee, Y.H. and S.H. Kim, *Production-distribution planning in supply chain considering capacity constraints*. Computers & industrial engineering, 2002. **43**(1): p. 169-190.

[21]Bang, J.-Y. and Y.-D. Kim, *Hierarchical production planning for semiconductor wafer fabrication based on linear programming and discrete-event simulation*. Automation Science and Engineering, IEEE Transactions on, 2010. **7**(2): p. 326-336.

[22]Kropp, D.H., R.C. Carlson, and J.V. Jucker. *Use of both optimization and simulation models to analyze complex systems*. in *Proceedings of the 10th conference on Winter simulation-Volume 1*. 1978. IEEE Press.

[23]Acar, Y., S.N. Kadipasaoglu, and J.M. Day, *Incorporating uncertainty in optimal decision making: Integrating mixed integer programming and simulation to solve combinatorial problems*. Computers & Industrial Engineering, 2009. **56**(1): p. 106-112.

[24]Jeong, S.J., S.J. Lim, and K.S. Kim, *Hybrid approach to production scheduling using genetic algorithm and simulation*. The International Journal of Advanced Manufacturing Technology, 2006. **28**(1-2): p. 129-136.

[25]Li, J., M. González, and Y. Zhu, *A hybrid simulation optimization method for production planning of dedicated remanufacturing*. International Journal of Production Economics, 2009. **117**(2): p. 286-301.

[26]Liu, J., et al. *Production planning for semiconductor manufacturing via simulation optimization*. in *Proceedings of the Winter Simulation Conference*. 2011. Winter Simulation Conference.

[27]Pürgstaller, P. and H. Missbauer, *Rule-based vs. optimisation-based order release in workload control: A simulation study of a MTO manufacturer*. International Journal of Production Economics, 2012. **140**(2): p. 670-680.

[28]Kacar, N.B. and R. Uzsoy. *Estimating clearing functions from simulation data*. in *Proceedings of the Winter Simulation Conference*. 2010. Winter Simulation Conference.

[29]Kacar, N.B. and R. Uzsoy, *Estimating Clearing Functions for Production Resources Using Simulation Optimization*. Automation Science and Engineering, IEEE Transactions on, 2015. **12**(2): p. 539-552.

[30]Armbruster, D. and R. Uzsoy, *Continuous dynamic models, clearing functions, and discrete-event simulation in aggregate production planning*. Tutorials in Operations Research, INFORMS, 2012.

[31]Kacar, N.B., D.F. Irdem, and R. Uzsoy, *An experimental comparison of production planning using clearing functions and iterative linear programming-simulation algorithms*. Semiconductor Manufacturing, IEEE Transactions on, 2012. **25**(1): p. 104-117.

[32]Mittelmann, H. *Mixed Integer Linear Programming Benchmark (MIPLIB2010)*. 2016 [cited 2016 26 February]; Available from: http://plato.asu.edu/ftp/milpc.html.

[33]Gurobi. [cited 2016 March]; Available from: http://www.gurobi.com/.

[34]Machin, J. *xlrd*. [cited 2016 April]; Available from: https://pypi.python.org/pypi/xlrd.

[35]*openpyxl*. [cited 2016 April]; Available from: https://pypi.python.org/pypi/openpyxl.

[36]Swain, J.J. *Simulation Software Survey*. 2015 [cited 2016 15 March]; Available from: http://www.orms-today.org/surveys/Simulation/Simulation.html.

[37]Foundation, T.A.S. *Apache POI - the Java API for Microsoft Documents*. 2016 [cited 2016 April]; Available from: https://poi.apache.org/.