FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Hybrid Machine Learning/Simulation Approaches for Logistics Systems Optimization

Francisco Alexandre Lourenço Maia

FINAL VERSION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Américo Lopes de Azevedo

Second Supervisor: João Pedro Tavares Vieira Basto

July 22, 2020

Resumo

Hoje em dia, tem-se testemunhado um abrupto crescimento e desenvolvimento da indústria, refletido no elevado grau de complexidade e inteligência que os sistemas de produção correntes apresentam, onde se destacam os sistemas logísticos. Esta incessante procura pela inovação e melhoramento contínuo são muito recorrentes na época atual, traduzindo-se em constantes transformações no conceito da qualidade de um produto.

Deste modo, emerge a necessidade em otimizar os *layouts* fabris conduzindo a um aumento da flexibilidade face aos seus comportamentos dinâmicos. Neste seguimento surge a imprescindibilidade de aprimoramento do comportamento do veículo autónomo associado, com vista a finalidades comuns como o aumento da produtividade e minimização de custos e *lead times*.

Neste âmbito, o objetivo desta dissertação é a combinação de técnicas de Reinforcement Learning com abordagens de simulação para a otimização de um sistema logístico job-shop, no que à produtividade diz respeito.

Para além da implementação do modelo de simulação do sistema logístico, esta dissertação desenvolve também numa fase inicial comportamentos elementares a aplicar ao veículo, implementadas no próprio ambiente de simulação.

Posteriormente, dado que a área de *Machine Learning* tem obtido tanto sucesso noutras áreas tecnológicas, surgiu o desafio da introdução do conceito de rede neuronal, através da criação de uma nova entidade designada Agente e caraterizada pela técnica de aprendizagem baseada em *Reinforcement Learning*.

Por fim, nesta dissertação, para além de se concluir que a abordagem baseada em *Reinforcement Learning* proporcionou os melhores resultados de produtividade, retiraram-se ainda conclusões no que à robustez destes modelos diz respeito, a fim de avaliar a sua flexibilidade quando sujeitos a diferentes contextos, simulando um ambiente real.

Abstract

Nowadays, we have been witnessing an abrupt growth and development of the industry, reflected in the high level of complexity and intelligence that the current production systems present, in which the logistics systems stand out. This incessant search for innovation and continuous improvement are very common today, reproducing into constant changes in the product quality concept.

In this sense, the need to optimize the factory layouts emerges, leading to an increase in flexibility because of their dynamic behaviours. In this segment, there is an essential need to improve the behaviour of the associated autonomous vehicle, to reach common objectives such as increasing the productivity and minimizing costs and lead times.

In this context, the objective of this dissertation is the combination of Reinforcement Learning techniques with simulation approaches for the optimization of a job-shop logistics system, regarding productivity.

Beyond the implementation of the simulation model of the logistics system, this dissertation develops, in an initial phase, elementary behaviours to be applied to the vehicle, implemented in the simulation environment itself.

Subsequently, given that the Machine Learning area has been so successful in other technological areas, the challenge of introducing the concept of the neural network appears, through the creation of a new entity called Agent and characterized by the Reinforcement Learning technique.

Finally, in this dissertation, in addition to concluding that the Reinforcement Learning-based approach provided the best productivity results, conclusions were also drawn regarding the robustness of these models, in order to assess their flexibility when subject to different contexts, simulating a real environment.

Acknowledgements

My first words go to Engineer João Basto and Professor Américo Azevedo for all their availability, comprehension and support during this master's dissertation. Their help was fundamental in this whole process and I am very grateful to them for that.

Secondly, I would also like to acknowledge the support of the Engineers Romão Santos and Narciso Caldas for all the availability and incentive during the dissertation.

In this segment, I also have a word addressed to INESC-TEC, in particular to the Centre of Enterprise Systems Engineering and all its elements for the facilities offered throughout the dissertation and for the excellent environment provided.

I would like to thank the Faculty of Engineering of the University of Porto and its professors for everything that has been transmitted to me over these 5 years, not only academically but also personally.

I have to highlight all the friendly relationships I have created in these 5 years, namely Carlos Carvalho, with whom I shared this experience in the INESC-TEC during this final semester, as well as Fábio Queirós, Eduardo Caldas, Francisco Pires, André Oliveira, André Cipriano, Lídio Ribeiro, Maria Pereira, Alexandra Santos and Artur Almeida, among many others that I take with me for the rest of my life.

I also thank my family for all the support and motivation that helped me to go through this journey, namely my parents Francisco Maia, Maria Isilda and sister Carolina Maia.

Finally, I address a word of appreciation to all my friends for all the encouragement and stimulation during this period of my life, not only in the bad moments but also in the good ones.

Francisco Maia

A moment of pain is worth a lifetime of glory.

Louis Zamperini

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AGV	*Automated Guided Vehicle*
AI	*Artificial Intelligence*
FIFO	*First In, First Out*
IB	*Input Buffer*
IBWSx	*Input Buffer of Workstation x*
JIT	*Just-in-Time*
MHS	*Material Handling System*
OB	*Output Buffer*
OBWSx	*Output Buffer of Workstation x*
PPO	*Proximal Policy Optimization*
RL	*Reinforcement Learning*
VRP	*Vehicle Routing Problem*
WIP	*Work-in-Progress*
WS	*Workstation*

Chapter 1

Introductory Analysis

1.1 Contextualization

Lately, there has been an abrupt growth and development of the industry, in which the concept “Industry 4.0” appeared, which allowed the exchange of information between a variety of equipments in a factory. Namely regarding the optimization of internal processes, as well as a company’s products and even services, this paradigm is revolutionizing the industry worldwide. Consequently, the current need and demand for innovation and improvement follow the model based on continuous improvement. [A year without improving is a year won by competitors - J. M. Juran]. [8]

In this sense, the concept of Lean Thinking, founded by Taiichi Ohno and Eiji Toyoda, which combines the elimination of waste (Just-in-time - "Any activity that the customer is not willing to pay" - Taiichi Ohno) with the immediate reaction to any problem that could arise in during a process (Jidoka - Japanese term), has emerged in this context. The main objective of this idea is to increase customer satisfaction, creating significant changes in the manufacturing processes that contribute to their better functioning (Kaizen), increasing productivity and efficiency. [9]

And what is the reason for this incessant search for innovation and optimization of industrial processes? The answer is pretty simple, customers are the main reason. Their demands and needs have also been increasing over the past few years, both in terms of variety and quality.

At the beginning of the study of these subjects, it was considered that quality would only be related to the product specifications ("Quality is conformance to the specifications" - Philip Crosby). However, this concept has been constantly updated, considering that the primary factor is the satisfaction of the customer’s needs ("Quality is fitness for use" - Joseph Juran). For this, it is firstly necessary to infer the product specifications, followed by the identification of the probable errors that may arise during the processes and their causes, before proceeding to their elimination. [10]

The need to increase quality, decrease costs and reduce delivery times, led to the creation of several types of factory layouts, namely the functional (process-oriented, job-shop), line (linear flows, flow-shop) and fixed (the product cannot be moved).

In this sense, the industries started to adopt different modes of production, namely the job-shop, characterized by the existence of specialized areas by function, and flow-shop, defined by the

production lines. Overall, the layouts are designed to minimize Material Handling costs, eliminate bottlenecks, reduce cycle times, eliminate waste and increase process flexibility.

Currently, many industries make use of the job-shop production mode, because it is related to a high diversity of products produced in low volume (production to order). It also allows the increase of the flexibility of production processes, however, they present high WIP and queues. [11]

In order to optimize these logistics systems, Material Handling is in great focus at the present. In order to obtain shorter cycle times and lower costs in transporting raw materials between workstations (WSs), there is a need to develop new optimization algorithms. For this, it is necessary to take into account the time and space (of the warehouse, for example). In other words, it is necessary to coordinate all the tasks of each workstation in order, for example, to satisfy all orders with the shortest possible lead time. [12]

Directly associated with these material management systems are the AGVs (Automated Guided Vehicle) that allow the materials to be transported between stations, and are generally unmanned. Currently, vehicle routing problems are in the spotlight and their objective is to travel the shortest distance possible, minimizing costs. This type of problem has some common characteristics to the "Traveling Salesman" problem, in which it is supposed to visit a certain number of cities covering the shortest possible distance. In the case of industries, each vehicle has a maximum capacity.

Most of these problems can be solved using the Milk-Run system, which is a delivery system that allows us to reduce stocks, reduce waste and optimize routes. It is a delivery system in which one product is deposited and another one is collected right after, in order to save time. It should also be noted that the AGV's route is fixed. The collection of products from suppliers is carried out on a scheduled basis, in stipulated quantities, making cycle times more predictable. [13]

Besides heuristics such as Milk-Run, other approaches, based on metaheuristics, have been also developed to solve this kind of problems. [14]

Allied to these approaches, the concept of Machine Learning emerges. Since it has been quite successful in other areas of technology, why not make use of this tool and apply it to production systems? It is precisely these issues that are currently being studied and developed.

1.2 Motivation

The growing evolution of the industry due to the continued increase in competitiveness has led logistics management to be in vogue these days. In other words, its processes have undergone significant changes, both financially and temporarily, in addition to the objective of making the system more robust and secure.

The importance of reducing human-made failures has led to the use of AGVs, which, besides the safety and precision issues, also contribute to the automatization of production systems, with regard to the sequence of operations.

For this purpose, certain heuristics were created and developed that allowed, for example, to minimize the cycle times of a production line as well as the costs associated with Material

Handling. It should also be noted that a large part of the lead time is spent on transport, with only a minority focusing on production processes.

One of the most important formulations that emerged, taking into account the need to accelerate the flow of materials between locations, was the concept of Milk-Run systems. These ones allow that, in a delivery system, when a certain product is delivered to a stipulated location, another one is also collected, saving time in transportation. In short, these types of systems contribute to the integration between the logistics systems and supply chains. [15]

That said, the Milk-Run concept was later transported to a factory layout context, in which each location corresponds to a workstation. As mentioned, these heuristics ensure the resolution of problems such as Job-Shop Scheduling and Material Handling.

In order to solve these optimization problems, some tools like Machine Learning (learning algorithms such as neural networks) can be combined to solve material movement problems. This tool has a great prominence nowadays and allows to make predictions taking into account data collected previously, even in highly complex environments. In this segment, one of the paradigms of Machine Learning that emerges is Reinforcement Learning (RL). Based on the environment in question and the decisions taken by the Agent, this learning method allows receiving feedback that indicates the best decision made so far. [16]

In summary, the study of the influence that these methods have on solving dynamic vehicle routing problems associated with the transport of materials between workstations in a factory is very interesting and allows to create very effective and adaptable algorithms.

These algorithms based on Reinforcement Learning can be applied regardless of the complexity of the system, and there is no need to change the factory layout. In addition to all of this, the investment is low and the results achieved are satisfactory.

In conclusion, the constant need to optimize both production times and transport, improving the routes, leads us to question the fact that, if Machine Learning has been so successful in other areas, why not adapt it to the productive systems and combine it with simulation approaches for the optimization of logistics systems. [17]

1.3 Objectives and Research Questions

The main objective of this dissertation is the combination of techniques based on Machine Learning with simulation approaches for logistics systems optimization.

Firstly, it is necessary to model the problem, understand the factory's operation (layout), create the simulation model (through the Flexsim software tool) and, finally, define simple decision rules to command the AGV.

Subsequently, it is possible to integrate training algorithms based on Reinforcement Learning techniques, which define a completely dynamic behaviour of the logistics system, to increase the productivity and minimize the makespans (the total time to complete a sequence of tasks). These algorithms, made available by *OpenAI Baselines*, are considered the state-of-the-art of Reinforcement Learning, nowadays.

Finally, it is essential to test whether the algorithm adapts to changes in the factory, with respect to changes in the production mix or in the stochasticity of processing times.

In order to be able to accomplish these objectives, we need to answer the following research questions (RQ):

- RQ 1: How can we model a factory's operation in a simulation model with simple decision rules to command the AGV?
- RQ 2: How can we integrate training algorithms based on RL techniques with a simulation model to study the factory's productivity and makespan?
- RQ 3: How can we evaluate the robustness of the proposed approaches?

1.4 Methodological Approach

This dissertation follows the study of a set of hybrid approaches of simulation and Machine Learning, whose consequent purpose is the optimization of a logistics system, leading to an increase in its productivity and minimization of makespan.

The initial phase of the dissertation is allocated to the construction of the simulation model of the job-shop layout, using the Flexsim software, in which an automated guided vehicle will also be incorporated.

Posteriorly, there are two elementary decision rules that will be implemented, which provide the indication of the workstation (WS) where a load of an entity will be performed. The first rule, referring to the inaugural simulation model, is based on the First-in First-out (FIFO) algorithm, while the second model is a Milk-Run optimized transport system.

Then, the concept of communication between the simulation environment and an external program will be introduced, which will allow the creation of a distributed system, because the last two rules addressed were implemented in the simulation model. This external program, called Server, uses the Python language and it will communicate with the simulation environment, becoming responsible for AGV decision making. This new stage also requires that the communication between the two entities has to be preliminarily established, using the TCP communication protocol.

At the beginning of the final stage, the concepts of Machine Learning will be introduced and discussed, namely the introduction of a third entity called Agent, which implements a neural network for decision making. This network will be responsible for defining the WS where the AGV will load the respective entities. For that, the Agent receives from Flexsim, through the models subsequently developed, a set of relevant information, namely observations of the current state of the plant and the current location of the AGV. However, it should be noted that the Server is still present, assuming the role of an intermediary between the Agent and Flexsim.

Consequently, through a Reinforcement Learning algorithm, called Proximal Policy Optimization (PPO), the neural network will determine the respective WS where the AGV has to load an entity. A pre-training process for the Agent will also be studied and applied, which will allow the

improvement of its learning phase and, consequently, lead to a possible maximization of productivity ("Base Model") and minimization of makespan ("Makespan Model").

Finally, the production mixes will be changed and the concept of probability distribution will be introduced. In this sense, the models previously discussed will be applied to a stochastic environment, in order to verify if they present a high level of robustness so that they can be applied in a real context, where the processing times are subject to constant changes.

All these phases are represented in Figure 1.1.

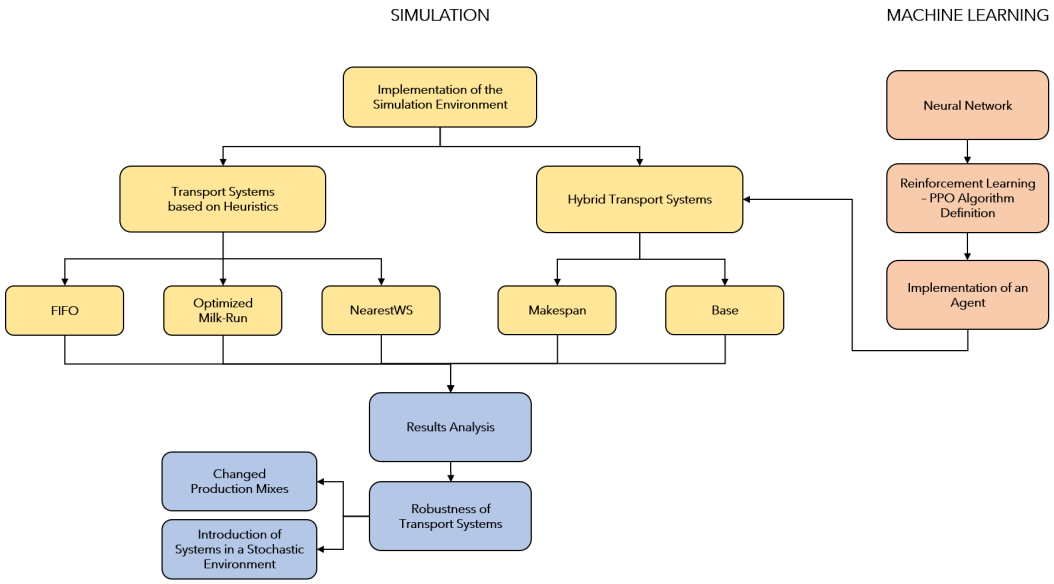

Figure 1.1: Dissertation approach method

1.5 Dissertation Organization

With regard to the organization and structure of this document, it is divided into six chapters.

The first chapter pretends to introduce the theme of the dissertation, including its contextualization, motivation, objectives and research questions, as well as the schematic of the methodological approach followed.

Chapter 2 presents the state-of-the-art of the subjects covered by the dissertation, like the industrial production systems, in particular the job-shop type, Material Handling Systems, Machine Learning concepts and discrete-event simulation.

The description and characteristics of the problem, as well as the description of its methodology, are covered in chapter 3, even as the implementation of the first three transport systems considered throughout the dissertation.

Chapter 4 reflects two additional transport systems, introducing Reinforcement Learning techniques in combination with simulation approaches. The robustness of the systems is related to the production mixes and processing times, which are also addressed.

The results obtained are exposed and treated in chapter 5 with regard to the five transport systems developed and their robustness.

Finally, chapter 6 presents the conclusions of the dissertation project and the suggestions for a future work.

Chapter 2

State-of-the-Art

This chapter aims to present the results of a bibliographic search, in order to internalize, in a simple way, the concepts that will be addressed throughout the dissertation.

Firstly, in section 2.1, the concepts of industrial production systems will be introduced in general and, later, in section 2.2, the job-shop production systems will be addressed. Then, section 2.3 presents some interpretations of the logistics systems associated with Material Handling in industrial environments.

In section 2.4 the question of Material Handling Systems (MHS) will be addressed, and in the next section (2.5) the principles referring to Milk-Run systems will be introduced.

Section 2.6 introduces the subject of Machine Learning, with special reference to the Reinforcement Learning technique.

Finally, section 2.7 presents the main concepts associated with simulation approaches.

2.1 Industrial Production Systems

The constant evolution of the market means that, nowadays, society seeks for differentiated products, which follows a perspective of diversity rather than quantity. In this way, the specialized industrial organizations have as main objective the increase of the effectiveness and efficiency of their production processes.

About typologies, there are systems whose objective is to produce products on a large scale with a low degree of variety and are also characterized by its high productivity, low qualification of their operators, reduced complexity of factory management and reduced flexibility. This type of system is called as product oriented.

On the other hand, there are systems that are oriented to the process and to the customer, in which its main goal is to satisfy the customer's needs, according to a pull perspective, and to value the quality of the product. This type of system gives more importance to the product variety instead of quantity, and it is considered more flexible and denoted by greater complexity of factory management.

Thus, production models are usually classified according to two distinct classes: make-to-order or make-to-stock.

- Make-to-order production is characterized by the fact that it is triggered to respond to an effective order;
- With regard to make-to-stock model, production is triggered taking into account a forecasted demand. [11]

2.1.1 Production Models

Over the past few years, the concepts of production systems have been changing. An example of this are the following three organizational forms of industrial production, applied during the second Industrial Revolution, in which we can see that the main common challenge continues to focus on cost optimization, however using different strategies.

Taylorism

According to Frederick Taylor, the way to increase the efficiency of the processes of a production system was observing the work performed by the workers, in order to increase and maximize the worker's performance. This method allowed to equate time and movement, increasing the efficiency of the processes. However, Taylor argued that only the administrative bodies were responsible for this, and so this idea created discomfort among workers who considered it as overexploitation. [18]

Fordism

It was created by Henry Ford who emphasizes the inclusion of the conveyor belt, which introduced a more dynamic work rhythm, resulting in a high turnover of workers. It applies to mass production systems, characterized as assembly lines. Like the previous model, it allowed both the reduction of costs and increase of the productivity of production processes. [18]

Toyotism

Founded by Taiichi Ohno, through the well-known car brand Toyota. He argues that each worker controls his own work, in which there is total trust between all levels of the company hierarchy, meaning that confidence leads to the elimination of errors in a short period of time. It also highlights the fact that each worker has the capacity to perform multiple tasks in the production process. [18]

In short, as shown in Figure 2.2, a Lean Thinking vision is highlighted, which defends the approach of zero waste, or in other words, the elimination of everything that does not add value to the customer. This paradigm allows reducing existing stocks and lead times (Just-in-time). Allied to this concept of zero waste, appears the concept of immediate reaction to any problem that could

arise during the processes, eliminating it immediately (Jidoka-Japanese term), mirrored in Figure 2.1. [19]

According to Taiichi Ohno, considered the main responsible for the creation of the Toyota's production system:

> *"All we are doing is looking at the time line, from the moment the customer gives us an order to the point when we collect the cash. And we are reducing the time line by reducing the non-value adding wastes"* - Taiichi Ohno [19]

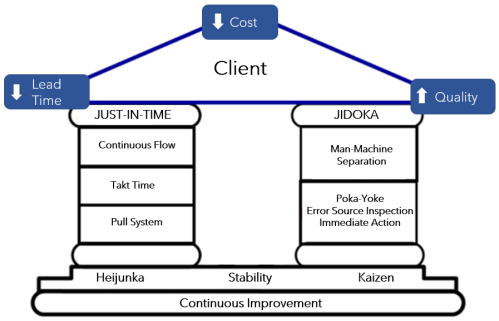

Figure 2.1: Lean Vision of Toyota's Production System, adapted from [1]

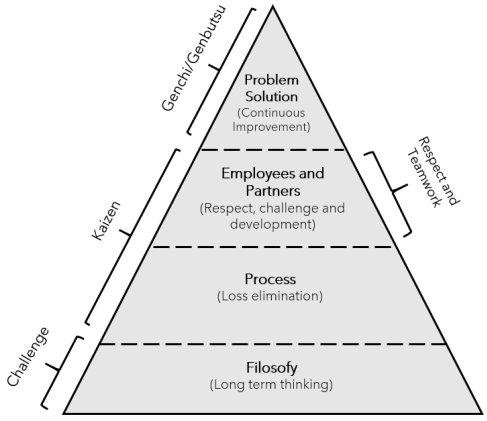

Figure 2.2: Toyota's Production System Model, adapted from [2]

2.2 Job-Shop Production System

2.2.1 Formal Definition

A job-shop scheduling problem can be described as a set J of n tasks, $J_{(i=1,...,n)} = \{J_1,...,J_n\}$, staggered on a finite set M of m machines, $\{M = M_1,...,M_m\}$. Each J_i task is fragmented into a series of m o_{ik} operations, where i represents the task J_i and k represents the machine M_k, where the o_{ik} operation will be performed. The sequence order of the machines for each task i is previously defined. It should also be noted that each o_{ik} task is associated with a non-negative p_{ik} processing time. [20]

This type of production system is characterized by its high diversity of products, produced in low volume, normally associated with the concept make-to-order. The high flexibility of its processes also allows a quick adjustment to different production mixes, in which an equipment can be used to work with different types of products. In contrast, the high WIP's (work-in-progress) that lead to long queues require more complex production planning and control.

One of the most used solutions to solve this planning issue is, for example, to join the specialized areas/sectors with the highest traffic, to reduce the distances travelled between them. In short,

these types of issues that are related to lead times and transportation costs between workstations are analysed by these job-shop systems. [11]

In the following Figure 2.3 there is an example of a job-shop layout, oriented to the process.

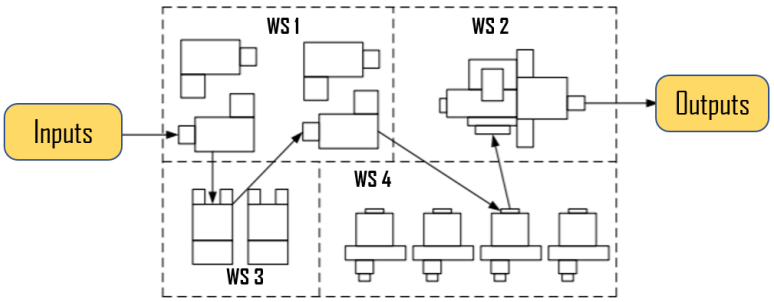

Figure 2.3: Example of a functional layout of a manufacturing process, adapted from [3]

Detailing some additional issues of this kind of job-shop model:

- Each J_i task visits a M_k machine only once;
- The operations associated with each task must be performed sequentially, following a certain pre-established order, without parallel processing;
- The setup times for machines and transports can be neglected;
- Each operation is performed only once on each associated machine;
- It is assumed that the machines are continuously available.

In this dissertation, the simulated factory has a similar production system to the classic job-shop problem. The main difference is the existence of an AGV, responsible for the transport of parts, which is carried out in a non-zero time.

2.2.2 Performance Evaluation Criteria

In this type of job-shop system, the scheduling of machines is directly associated with the evaluation of costs and performance metrics. Because it is quite difficult to evaluate the performance based only on costs, the main criterion is temporal.

Most of the criteria are based on task finish times and, in some cases, some data such as weights are added, which will allow us to assign different importance to each task, as well as delivery dates.

Some of the most frequent performance evaluation criteria in this type of systems are represented in the following list:

2.2.2.1 Makespan

This technique is one of the most used, referring to the period of time required to complete a set of production orders.

It allows to take conclusions about the total time spent, and to obtain feedbacks, through that information.

C_{max} represents the makespan and C_i is the time necessary to complete the task J_i, so that the objective is to minimize the makespan [20]:

$$C_{max} = \max_{1 \leq i \leq n} C_i \longrightarrow min \tag{2.1}$$

2.2.2.2 Machine Utilization Rate

This metric shows the relationship between the machine availability and the required capacity. The goal is to make the machine idle for as little time as possible, increasing its efficiency.

The average machine usage is represented by the following expression, MU (use of machines) [20]:

$$MU = \frac{\sum_{i=1}^{n} \sum_{k=1}^{m} p_{ik}}{m.C_{max}} \tag{2.2}$$

2.2.2.3 Flow Time

The flow time, F_i, is defined as the difference between the task completion time (c_i) and the launch time of each task (r_i). The weight assigned to each task is represented by W_i.

The objective is to minimize the so-called WIP, represented by the following expressions [20]:

$$F_i = c_i - r_i \tag{2.3}$$

$$min \sum_{i=1}^{n} W_i * F_i \tag{2.4}$$

2.2.2.4 Due Date

Taking into account the delivery date, d_i, and the task completion time, c_i, the objective is to minimize the maximum delay, $Lmax = max(c_i - d_i)$. Alternatively, sometimes the objective is, taking into account T_i (maximum between 0 and $c_i - d_i$ assuming that the completion time is always greater than the delivery date), to minimize the sum $W_i * T_i$. [20]

$$L \max_{i=1}^{n} (C_i - d_i) \tag{2.5}$$

$$T_i = max(0, c_i - d_i) \tag{2.6}$$

$$\sum_{i=1}^{n} W_i.T_i \longrightarrow min \tag{2.7}$$

2.3 Logistics Systems

There are several approaches to the transport of materials in industrial environments. According to Meyer [21]:

Point-to-Point

Also known as direct transport, this model is characterized by the fact that the deliveries and collections performed between the suppliers and the factory are carried out directly, as Figure 2.4 shows. This model is used when lots of high quantities are considered and it is recognized by its lower flexibility, however, it is simpler to plan and control. It enables to reduce the transport costs because its frequency is lower throughout the day.

Area Forwarding Services

This model allows a company to transmit the responsibility for defining routes and carrying out the transport of materials to another company, which is specialized in logistics.

Milk-Run

As Figure 2.5 shows, the Milk-Run model is a mechanism for transporting small quantities of materials of a wide variety, which allows the same transport vehicle to visit several suppliers before supplying the factory.

The routes are pre-defined and this model enables to reduce the associated transport costs, as well as the safety stocks.

It is also characterized by the higher frequency of transport, and its scheduling is much more complex than the one of the point-to-point model.

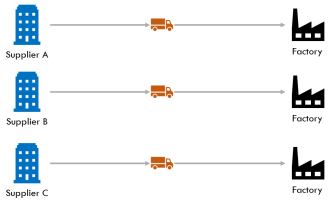

Figure 2.4: Point-to-Point Model

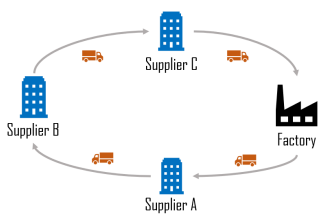

Figure 2.5: Milk-Run Model

2.4 Material Handling Systems

Currently, the main objective of the industrial environments is the elimination of the activities that do not add value to the processes.

Therefore, one of the most frequent and important problems is the Material Handling, which constitutes one of the seven wastes of Lean manufacturing, and which is directly related to the vehicle routing problems (VRP).

The Material Handling systems objectives are to [4]:

- Increase the efficiency of the material flow, supplying the materials where and when needed;
- Reduce routing costs;
- Increase productivity;
- Increase safety and working conditions.

The Lean logistics, as shown in Figure 2.6, can also be defined by three different groups: in-bound (supplier-factory), in-plant (inside the factory) and out-bound (factory-customer) [4]. It should also be noted that, in the context of this dissertation, just the Milk-Run in-plant system will be addressed.

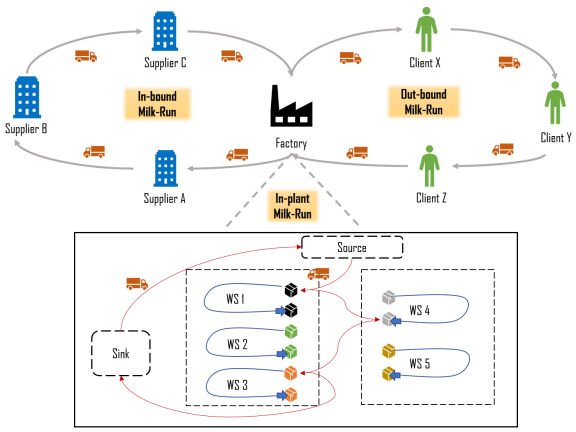

Figure 2.6: Representation of in-bound, in-plant and out-bound milk-run systems

The Lean paradigm defends that the principal goal is to minimize the transport and WIP costs. However, it should be highlighted that the costs from WIP and transport are both complementary.

Therefore, the main problems of this transport system are how to determine routes and route times.

In this segment, there are three categories represented in Figure 2.7 [4]:

- **General assignment problem** - unknown routes and times;
- **Dedicated assignment problem** - known routes and unknown times;

- **Determined time periods assignment problem** - known routes and times.

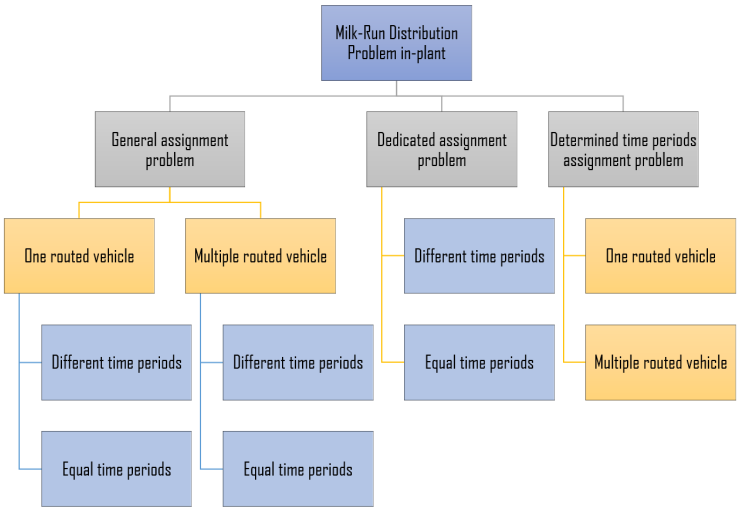

Figure 2.7: Representation of the categories of milk-run in-plant distribution problems, adapted from [4]

2.5 Milk-Run

A Milk-Run System is directly related to the concept of transportation in the industry. The need to reduce stocks and costs in transport was the reason for the creation of this model. This idea of cost reduction follows the JIT philosophy of Lean fundamentals. Briefly, this is a delivery system that allows us to optimize routes and reduce wastes between workstations in a factory plant (in-plant).

With regard to the Milk-Run system, Baudin [22] states:

> *"This concept allows to move small quantities of a large number of different items with predictable lead times and without multiplying transport costs"* [22] - (Baudin, 2004 [13])"

All these aspects led Lean manufacturers to choose to organize their transport according to fixed times and routes, in the form of a Milk-Run system. The design of these systems involves a higher complexity, which according to Meyer [21] can be described in the following three factors:

- Transport of materials;
- Frequency of transports;
- Route scheduling.

2.5.1 Advantages of a Milk-Run System

Comparing to the traditional approach, shown in Figure 2.4, the Milk-Run system presents the following advantages:

Inventory Reduction

As Figure 2.8 shows, the Milk-Run typology, in cases X and Z, allowed the inventory level to be reduced by 1/3, because the frequency of transport was increased by three units. In case Y, the frequency of transport increased twice, resulting in a decrease in the stock of 2/3 of the level of the point-to-point typology.

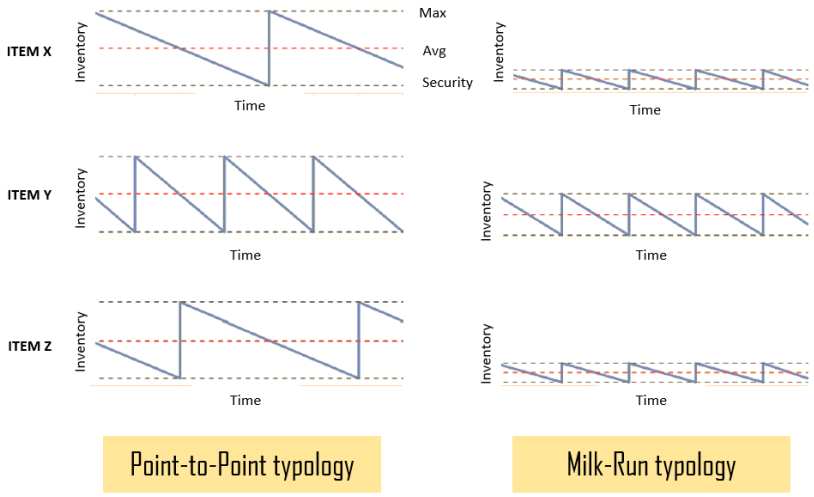

Figure 2.8: Comparison between the inventory levels of a Point-to-point and Milk-Run typology, adapted from [5]

Replenishment with Predictable Lead Times

On a daily basis there are thousands of products with different transport frequencies. However, through Figure 2.8, which presents only three different products (X, Y, Z), it is possible to generalize for other cases. So, it can be concluded that their lead times are predictable.

Because we have access to data which indicates an increase or decrease in the frequency of supply, it becomes predictable to estimate the stock variations that will occur in the future.

Better Inventory Visibility

In the case of the point-to-point deliveries, there may be a case where only the product X is transported on a large scale, causing an almost total emptying of the respective shelves. However, this situation does not represent any type of anomaly in the inventory.

In the case of the Milk-Run system, because the quantities transported are practically the same for all products, any significant variation in the amount of any type of product present on the corresponding shelf is an immediate sign of the presence of an anomaly. So, the Milk-Run typology has a better visibility of the inventory, allowing to act quickly in case of abnormality.

Improve Communication Skills with Suppliers

Using Milk-Run enables suppliers to be in regular contact with the customers, because of the frequency of supply. Therefore, it is possible to obtain feedback from consumers about the quantities and quality of the delivered products, which makes it possible to improve the delivery system and the future quality of the products. [13]

2.6 Machine Learning

The Milk-Run systems need to adapt since the current reality of the factory layouts has led this system to become dynamic so that it is necessary to develop decision algorithms. So, the concept of Machine Learning was introduced in the scope of the transport of materials in a job-shop environment, more specifically the Reinforcement Learning technique.

This concept argues that learning from interactions is a fundamental idea subjacent to almost all theories of learning and intelligence. [23] Basically, Machine Learning aims to learn based on previous data and make predictions or decisions for the future. [24]

According to Arthur Samuel, pioneer in artificial intelligence:

> *"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed"* - Arthur Samuel, 1959 [23]

Enumerating some of its applications [23]:

- Analyse product images on a production line to automatically classify them;
- Detect tumours through brain scans;
- Summarize long documents automatically.

The Machine Learning systems can be classified into three categories, according to its learning processes:

- **Supervised Learning**

  A series of examples (inputs) with the correct answer (outputs) is provided by an external supervisory Agent and, based on training, the implemented algorithm generalizes the correct answer to another set of inputs, afterwards. [25]

- **Unsupervised Learning**

  The developed algorithm tries to identify similarities between the inputs, categorizing them. One of the best-known techniques is the clustering. [25]

- **Reinforcement Learning**

  It is located between Supervised Learning and Unsupervised Learning. The algorithm is informed about the quality of the response, but it is not informed about how to correct it.

So, it is necessary to explore and experiment other different possibilities until the Agent discovers how to obtain a higher quality response.

In this particular dissertation, it is pertinent, with regard to the transport of materials in a job-shop environment, to analyse, essentially, the Reinforcement Learning field.

2.6.1 Reinforcement Learning

Learning how to control Agents directly from high-level sensory information, such as vision and speech, is one of RL's longstanding challenges. [26]

The Agent is not specifically told what actions to take, unlike what happens in other forms of Machine Learning, like Supervised Learning. Therefore, the Agent will have to find out which actions, deliberated so far, allowed him to obtain greater rewards, at the end of the learning phase. Interestingly, the actions taken in the present will affect the respective reward, as well as the future ones.

The concepts of "trial and error search" and "delayed reward" are the two most important characteristics of RL.

Unlike Supervised Learning, which is a way of learning based on examples provided taking into account the knowledge of an external supervisor, this is not appliable to an interactive learning. This is because, in interactive problems, in most cases, it is impossible to obtain examples of the desired behaviour that are correct and represent all the situations in which the Agent needs to act.

Hence, the RL allows the Agent to decide what action to take, taking into account his own experience. The Agent will have to check the decisions he has made in the past and find out if, in fact, he obtained a beneficial reward, so that he can then later carry out his action. In this sense, a new paradigm appears, in which the Agent, in addition to exploring knowledge that he already has from previous situations, also needs to explore new decisions never made before, to see if he gets a higher reward. Neither of these paradigms is considered better than the other because in both cases, the Agent will fail (obtain a lower reward) and the solution states in the critical capacity of the Agent, so he has to perform several tests, in order to find the solution that provides him with a final value corresponding to the biggest reward. [6]

2.6.2 Reinforcement Learning Characteristics

Agent

The Agent is the entity that it is responsible to make the decisions and it is called “learner” and “decision maker”.

More specifically, the Agent and the environment interact with each other in the form of discrete time intervals ($t = 0, 1, 2, ...$). As Figure 2.9 presents, for each t, the Agent receives a representation of the state of the environment s_t, such that $s_t \in S$ represents the set of all possible states. Consequently, the Agent receives a response in the form of a reward $r_{t+1} \in R$ and a new state of the environment, s_{t+1}, which is a feedback for the next decision to make a_{t+1}. [6]

Environment

The environment is responsible for informing the Agent of the current state and the reward obtained for the action taken previously. It also tells the Agent a set of all possible states.

Action

The action is the result of the decision made by the Agent. The objective is to find the best solution, which corresponds to choosing the action that allows him to obtain the highest reward because each action originates different reward values.

Reward

The reward is a feedback in which the Agent evaluates the consequences of his action taken in the previous state. It is important to refer again that the objective is to obtain the greatest possible accumulation of rewards, keeping in mind that a large reward obtained in a given state does not necessarily mean that the final accumulation of rewards will be the best. This is because, although a specific reward in a given state is the largest one, it may lead to a non-ideal situation in the future and influence negatively the following rewards. [27]

Policy π

It is a mapping strategy that allows the Agent to decide the next action to take, in order to obtain a good accumulated reward in the long term.

The RL specifies how the Agent can change his policy, taking into account his experience.

The Agent can also be classified according to policy, value function and model. A policy, π, is a mapping of states $s \in S$ and actions $a \in A(s)$, for the probability $\pi(s,a)$ of taking an action at the time of a state s. The value of the state s under a policy π is still denoted by $V^{\pi}(s)$. [6]

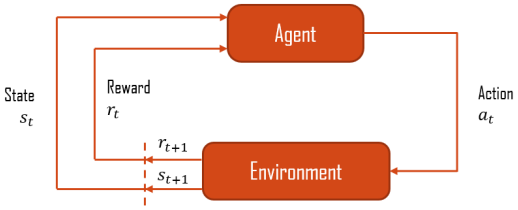

Figure 2.9: Agent-Environment interaction diagram, adapted from [6]

2.7 Simulation

Modelling is a tool that allows us to solve real context problems. Most of the time we cannot afford to experiment and test real objects, in order to obtain the best solution, since these objects are, in general, expensive and even scarce. Thus, the simulation assumes a fundamental role in this

context, in order to be able to solve problems in a practical way and without collateral damage, confirmed by the following quote adapted from [28].

> *"Modeling consists of finding the path of the problem to its solution, in a risk-free world where we can make mistakes, undo things, go back in time and start again."*
> [28]

In addition to the purpose of modelling, there are more benefits of using this method, such as [29]:

- Simulation models allow us to analyse systems and find solutions in which the analytical models fail;
- It allows testing new policies, operating procedures, decision rules, information flows without making changes to the real system;
- Discover the bottlenecks.

However, despite all these gains, the final results can sometimes be difficult to interpret. The fact that the simulation requires a lot of training is one of the least favourable points.

2.7.1 Components of a System

In order to better understand the constitution of a simulation system, this section describes its main elements. [29]

System

A set of entities that interact with each other in order to achieve the outlined objectives.

Model

An abstract representation of a system, which allows it to describe in terms of its state, entities, processes, events, activities, and others.

System State

A necessary set of variables to describe the system.

Entity

An object of interest which requires an explicit representation.

Attribute

An entity property.

Event

An instant occurrence that can change the state of the system.

Activity

The time duration that a task needs to be executed, known at the moment it starts.

Delay

The excess time interval, only known when it ends.

Clock

A variable that represents the simulated time.

A method is a structure used to map real systems in simulation models. With regard to the simulation modelling methods, this can be organized into three categories: Agent-Based, System Dynamics and Discrete-Event Modelling. The use of each one of these methods depends on the system to be implemented and its objectives.

Agent-Based

It is part of the class of computational models for simulating actions and interactions between autonomous Agents (individual or collective). Despite the lack of knowledge of the system's behaviour and inability to represent the process flow, the main objective is to verify and study the effect of the Agents on the system as a whole. [28]

System Dynamics

John Sterman [28] states that this model is a perspective and a set of conceptual tools that enables the understandment of the structure and dynamics of complex systems. This model is also a rigorous modelling method that allows to build formal simulations of complex systems and use them to design more effective policies and organizations.

Discrete-Event Modelling

It allows conceiving the modelling of a system as a discrete sequence of events in time. Each event occurs at a particular time and causes a change in the state of the system. This system requires that modelling has to be seen as a process so that it is a sequence of operations performed by Agents. [28]

2.7.2 Discrete-Event Simulation

Nowadays, this simulation method is widely used in the modelling and analysis of problems in the area of logistics systems, as it allows the study of aspects such as processes, scheduling and resource allocation. Health, business processes and military applications are also areas covered by this modelling method. Consequently, all these advances have led to a software development. [30]

In the specific case of this dissertation, the simulation will be used to evaluate the internal material flow of a manufacturing plant, in order to make conclusions regarding the factory's productivity and values of makespan, so we are able to identify possible bottlenecks.

Therefore, this tool is used as a method for analysing and solving the following problems, associated with job-shop environments:

- Evaluate the effect of changing material transport routes between workstations;
- Analyse the phenomenon of resource allocation and bottleneck prevention;
- Assess the impact of changing the elements in the layout on performance.

Through published articles dedicated to the study of this area of simulation, it is possible to draw examples of applications in distribution and transport systems.

Hugan (2001) elaborated a study that allowed him to evaluate the internal traffic of a General Motors automobile plant, in the USA, which was based on the JIT model of Lean manufacture. The simulation allowed to improve the internal routes for each type of product, as well as to estimate the average time spent by a product in the factory, from its entry to its exit. Kuo, Chen, Selikson and Lee (2001) used the simulation of discrete-events to study the flow of materials also in a manufacturing plant, which allowed them to have a deeper knowledge of operations and logistics processes. [30]

2.7.3 Construction Steps of a Simulation Model

Problem Formulation

The problem must be well formulated so that there is no doubt. There are still cases where it is necessary to proceed with a total or partial reformulation of the problem. [29]

Establishment of Objectives and General Project Plan

The objectives will be the questions to be answered through the simulation. After deciding which simulation method is the most suitable, it is necessary to list a series of alternatives to the simulation and find ways to evaluate the effectiveness of these same solutions. Besides the objectives defined for the end of each state, it is also important to mention in the plan the number of people involved, as well as the associated costs and the estimated time for each phase of the project. [29]

Model Conceptualization

According to Pritsker (1998), although it is not possible a priori to define the instructions that will lead to a successful model, there are some points of view that must be followed for the model to be successful. It is necessary to start by defining a simplistic model and, from there, make the necessary changes step by step to obtain good results. [29]

Data Collection

Data collection is directly associated with the construction of the model, since the data collected will serve as input to the model. As this collection fills large intervals of time and this is a very important aspect in the elaboration of the model, it is essential to start the collection as soon as possible. [29]

Model Translation

This phase consists of converting the model into a simulation language using specific software programs. In the specific case of this dissertation, the software used is the Flexim program. [29]

Verification and Validation

This step enables to check if the program is prepared for the simulation model, carrying out verification and debugging tests. By comparing the model with the behaviour of the current system, it is useful to use this feedback in order to improve the model. The process is repeated iteratively until the result obtained is satisfactory. [29]

Experimental Design

The alternatives previously defined in the "Establishment of Objectives and General Project Plan" phase that must be simulated, must be determined. [29]

Production and Analysis of Results

After several tests of the model, with different data, the resulted analyses are used to estimate the performance of the system that was simulated. [29]

Documentation

After the simulation, it is necessary to report two types of data: program and progress. If the program is used by others in the future, the program documentation indicates the modes of operation and behaviour of the program, as well as other fundamental aspects. In relation to the progress report, this is essential for the model to obtain credibility and certification. [29]

Implementation

The success of the implementation will depend on each phase previously referred. [29]

Chapter 3

Problem and Methodology

This chapter presents the general characteristics of the generic Material Handling problem in a job-shop system, using an AGV. Following this purpose, there are addressed questions related to the current layout of the manufacturing plant and the processing and sequencing of predefined tasks, as well as the strategies adopted to solve this type of problem. In this context, the characteristics of the simulation model of the system under analysis will also be introduced in this chapter, using the Flexsim software tool.

3.1 Problem Description

With this dissertation it is intended, in a simple and quick way, to build a simulation model which is capable to improve the functioning of industrial systems, through the creation of decision rules to provide to an AGV, based on the Material Handling paradigm. In this sense, in an initial phase, the objective is to develop the simulation model considered throughout the dissertation, which is characterized by a functional layout in a manufacturing environment, in which the positions of the workstations are already pre-defined and represented in Figure 3.1.

In this segment, an autonomous vehicle responsible for the transport of materials between the workstations will be introduced. Initially, this AGV will follow elementary decision rules such as First-in, First-out (FIFO) and, later, the Milk-Run concept.

Furthermore, a new decision rule will also be addressed (NearestWS Rule), which brings some improvements that take into account the distances between the WS and which have impact in the final productivity results.

Finally, comes up the challenge of using the Machine Learning area with the objective of creating decision rules that will allow to decrease the makespans of the industrial systems and to increase their productivity. For this purpose, an Artificial Intelligence program, called Agent, will be used and will be run in parallel with the simulation environment. The intention is, in a previous phase, to establish the communication between these two entities, through the creation of a Server, so that the simulation environment can send data containing its current state. Then, the Agent can take its own actions to perform in the respective simulation environment. In summary, this last

problem discussed in the dissertation consists of implementing training algorithms based on RL techniques, which define a completely dynamic behaviour of the logistics system, with a view to increase the plant productivity and minimize the makespans.

3.2 Problem Characteristics

3.2.1 Layout

The manufacturing plant follows a job-shop layout, characterized by its high flexibility, in which the workstations are arranged by specialized areas. As Figure 3.1 represents, there are seven WSs and an AGV. Their positions are also identified in Table 3.7, in the form of X (abscissa) and Y (ordinate) coordinates. Each station also has an associated input (IB) and output buffers (OB), in which both are three distance units apart from the WS itself.

The raw material warehouse, where the parts enter the system, is represented by a Source, and the point where the parts leave the system (finished products warehouse) is Sink's responsibility.

It should also be noted that the capacity of the AGV is just of one part and the processing capacity of each WS is also of one part, except for Source, Sink and buffers, which have unlimited capacities.

Another important aspect is the fact that the respective factory layout was already previously optimized, and this information was taken from a previously developed project. [31]

Table 3.1: Capacity and Coordinates (in metres) of each WS

Workstation	Capacity	X	Y
WS1	1	30	-20
WS2	1	0	10
WS3	1	18.873	-56.619
WS4	1	18.873	16.619
WS5	1	30	0
WS6	1	0	-50
WS7	1	30	-40
Sink	∞	0	-30
Source	∞	0	-10

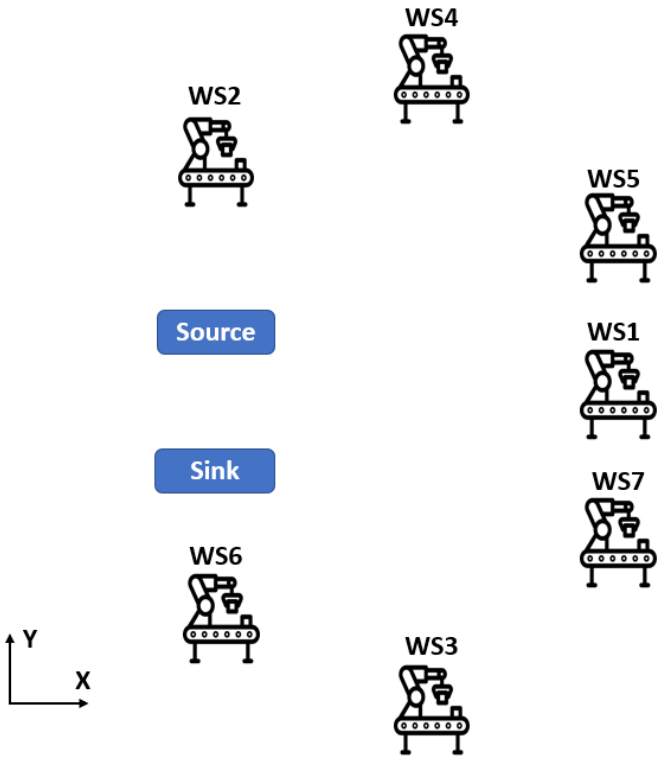

Figure 3.1: Factory Plant Layout in study

3.2.2 Processing

Associated with each WS there is a processing time that also depends on the part type in question. There are five different part types, and the following Table 3.2 shows the processing times, in minutes, for each part type, depending on the WS where they are processed.

Table 3.2: Processing Times (in minutes) for each WS

Workstation	Part 1	Part 2	Part 3	Part 4	Part 5
WS1	9	5	-	-	9
WS2	2	9	6	7	2
WS3	-	-	7	7	-
WS4	-	10	10	5	10
WS5	-	5	-	-	6
WS6	8	-	-	-	-
WS7	7	-	7	3	-

3.2.3 Sequencing

Regarding the sequence of stations that each part has to visit, including Sink and Source, these are indicated in the subsequent Table 3.3.

Table 3.3: WS Sequence for each part type

Part 1	Part 2	Part 3	Part 4	Part 5
Source	Source	Source	Source	Source
WS2	WS4	WS2	WS7	WS4
WS1	WS2	WS4	WS3	WS5
WS7	WS5	WS7	WS2	WS1
WS6	WS1	WS3	WS4	WS2
Sink	Sink	Sink	Sink	Sink

3.2.4 Production Plan

Each production order corresponds to just one part type and has a specific release time associated with it. These production orders are stated in an Excel table previously provided, which contains 2080 orders. This table has columns referring to some characteristics such as the release time, name, quantity, order ID and part type, in which the first three ones are considered mandatory for the use of the Flexsim tool. In order to clarify what was mentioned, Table 3.4 presents the first five orders of the production plan into consideration. All the orders are released from 90 to 90 seconds.

Table 3.4: Production Plan excerpt

Release Time (in minutes)	Name	Quantity	OrderID	Part Type
0	1	1	1	4
1.5	2	1	2	1
3	3	1	3	1
4.5	4	1	4	1
7.5	6	1	6	3

3.3 Discrete-Event Simulation Model – Flexsim

Following the mentioned steps in chapter 2.7.3 to build a simulation model, it is possible, through the Flexsim software tool, to simulate the behaviour of the system under study, using the discrete-event simulation category.

This software package allows the simulation of the factory's behaviour with great precision and reality. Thereby, it is possible to simulate the release times of a part, as well as the processing of each WS, so that we can draw valid conclusions about the system's behaviour.

3.3.1 Entities

These objects represent the parts that run through each one of the workstations, in which it is possible to know the current state of each part type, like the current and next WS. By selecting an object, it is possible, for example, to have access to its part type and the production order's launch number (orderID), as Figure 3.2 denotes.

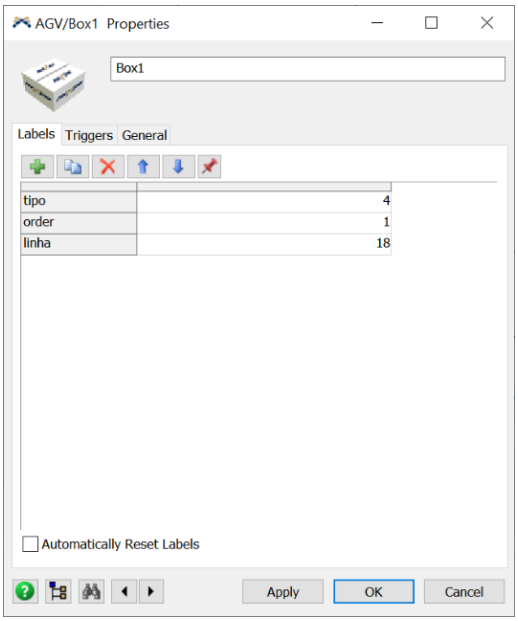

Figure 3.2: Properties of an entity - Flexsim

3.3.2 Resources

The Flexsim Library offers a wide variety of resources, whose different categories are represented in the following Figure 3.3. In particular, in this MHS problem, the resources will represent the AGV itself and its workstations.

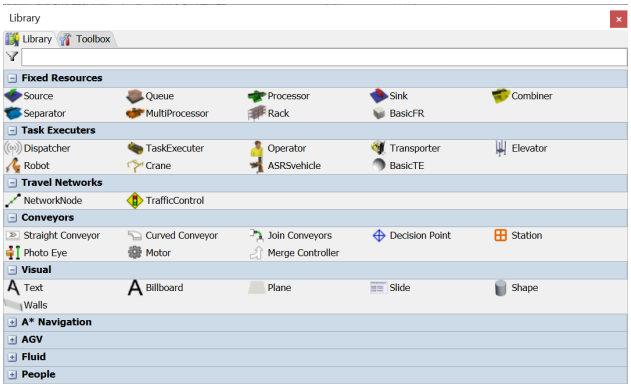

Figure 3.3: Library – Flexsim

3.3.3 Workstations

The workstations are reflected by Fixed Resources and it is important to mention that they will be represented by native Processor resources. As mentioned, each WS contains an input queue (input buffer - IB) and an output queue (output buffer - OB) which both distance 3 units apart from the respective WS, illustrated in Figure 3.4. As chapter 3.3.7 presents, it is also possible to obtain concrete information about the entry and exit rates of the parts in each of the WS.

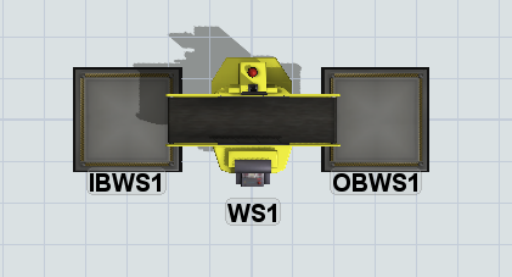

Figure 3.4: Graphical representation of a WS – Flexsim

3.3.4 Transport

As previously indicated, the transport of the entities is carried out by an AGV, which is created from an object of the Task Executer class (Flexsim Library). Thereby, it is only necessary to indicate which stations the AGV will collect (OB) and deliver (IB) parts. This information can be found in detail in the previous chapters of Processing, 3.2.2, Sequencing, 3.2.3 and Production Plan 3.2.4.

It is also worth noting that the AGV speed will be 1.50 m/s for all simulation models.

3.3.5 Load

All the parts come in into the system through the Source, however, in this particular case, this resource has been replaced by a Queue, called "Source1", which admits the same behaviour. The main reason for using a Queue object instead of a Source is that it is possible to observe the accumulation of parts throughout the simulation. At last, the resources where the parts are loaded are the Source and the output buffers.

3.3.6 Unload

The final unloading of the entities is carried out at Sink, and the intermediate unloads between WSs take place in the respective input buffers (IBWS).

3.3.7 Processes

There are essentially two tools that allow the building of a simulation model: 3D model and Process Flow. Regarding Process Flow, it always and in any circumstance overlaps the 3D model. However, the cooperation between both is essential, although, in this particular situation, it is in the Process Flow that the flow of entities and resources will be defined.

In general, all transport systems involve three large blocks: Creation of Parts (Figure 3.5), Processing of Parts (Figure 3.6) and Transport of Parts. Both phases of creation and processing of parts are common to all models covered in this dissertation.

3.3.7.1 Creation of Parts

The "Source" block defines the time when the production orders will be released, using an Excel file that contains the information for this purpose. Subsequently, the “Create Object and Type” block allows the creation of the entities and assigns them to the corresponding part type, taking into account the Label referring to the part type of the Production Plan table (Table 3.4). Taking into account the 5 different part types, there is a need to define colours for each part type, in order to distinguish them during the performance of the simulation, accomplished by the “Change Color” block:

- 1 - *Aqua*;
- 2 - *Red*;
- 3 - *Blue*;
- 4 - *Yellow*:
- 5 - *Lime*.

Considering the type of the current part, the block “Determine Line” allows to go through the Routings Table (Table 3.5) and identify the line in question, in order to follow the WS sequence of the part type in evidence.

Subsequently, after identifying the WS where the first unload will be performed, it is necessary to acquire the AGV in order to allocate it, so that the initial load can be executed in the Source and the unload in the IB of the following WS. At the end of the process, the resource is released ("Release").

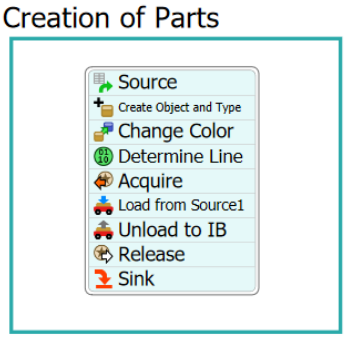

Figure 3.5: Representation of the “Creation of Parts” Block

3.3.7.2 Processing of Parts

When an entity enters the IB of a WS, it is necessary to move it to the WS itself, then operate it and, finally, move it one more time to the respective OB. However, it must be noted that the entity will be just moved from the current IB to the corresponding WS if there are no parts in

the respective WS, because its maximum capacity is unitary. These restrictions are defined by the creation of zones, focused on each one of the existing WS. At the end of the processing, the block "Exit Zone" will release the respective WS from the due entity already processed, and it will be available to receive a new part. Then the line in the Routing Table (Table 3.5) is incremented by one, in order to follow the pre-defined sequence of the part.

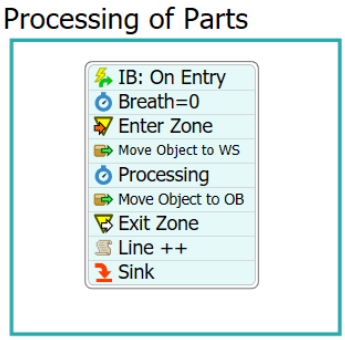

Figure 3.6: Representation of the "Processing of Parts" Block

3.4 Implementation of a FIFO Transport System

Amongst all the transport systems covered in the dissertation, this model is considered the simplest one since the decision rule to apply to the AGV follows a First-in, First-out paradigm. This means that the AGV will decide to take forward the transport of a part that was processed first, instead of a part that has been processed more recently.

According to the Process Flow method, in addition to the two blocks discussed in subchapters 3.3.7.1 (Creation of Parts) and 3.3.7.2 (Processing of Parts), the block that mentions the transport of parts is also essential (3.4.1).

3.4.1 Transport of Parts

Apart from the initial transport from Source to the IB of the first WS of each production order, addressed in the "Creation of Parts" block (3.3.7.1), this block includes all the other transports. In this segment, as soon as an entity reaches the OB of any of the 7 WSs, the AGV will be requested and, as soon as available, it will transport the part from the current WS to the next one, respecting the particular sequences. Finally, the AGV will be released, originating the creation of a new transport process.

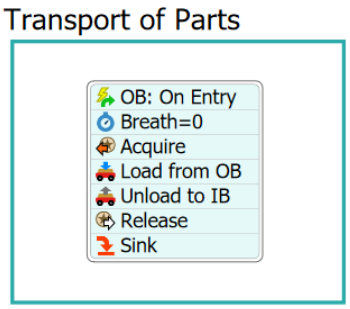

Figure 3.7: Representation of the "Transport of Parts" Block – FIFO Model

Table 3.5: Routings Table

Line	Part Type	Workstation	Processing Time (in minutes)
1	1	WS2	2
2	1	WS1	9
3	1	WS7	7
4	1	WS6	8
5	1	Sink1	0
6	2	WS4	10
7	2	WS2	9
8	2	WS5	5
9	2	WS1	5
10	2	Sink1	0
11	3	WS2	6
12	3	WS4	10
13	3	WS7	7
14	3	WS3	7
15	3	Sink1	0
16	4	WS7	3
17	4	WS3	7
18	4	WS2	7
19	4	WS4	5
20	4	Sink1	0
21	5	WS4	10
22	5	WS5	6
23	5	WS1	9
24	5	WS2	2
25	5	Sink1	0

3.5 Implementation of an Optimized Milk-Run System

The main difference between this Milk-Run system and the one previously presented in the earlier subsection is the decision rule to be applied to the AGV. While in the previous system there was no priority regarding the transport of entities, following the FIFO paradigm, in this specific context

the AGV has a predefined WS sequence, (identified in the following Table 3.6 and illustrated in Figure 3.8), that it follows in order to load the entities. In other words, if the AGV is, for example, in WS2, but that same station does not contain entities in its OB to perform the load, the AGV will check if the next WS in the AGV Routing Table (WS4) includes parts in the respective OB, and so on.

What distinguishes this optimized Milk-Run algorithm from the classic one is the fact that the AGV performs the load directly on the nearest WS, in a clockwise direction, which must necessarily have parts in its OB. On the other hand, what happens in the classic Milk-Run system is that the AGV must visit the nearest WS, also clockwise, even if it does not contain parts in the respective OB.

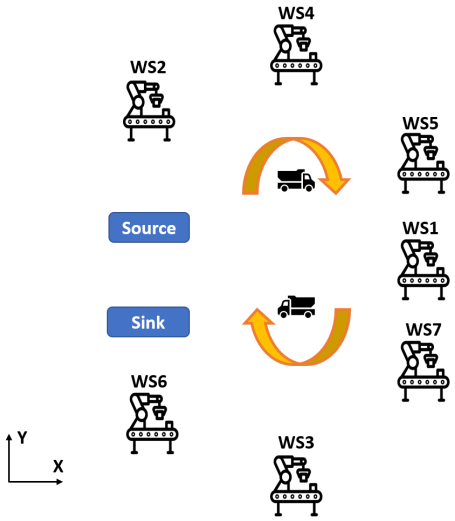

Figure 3.8: AGV Trajectory – Milk-Run Model

Table 3.6: AGV Routing

Workstation
Source
WS2
WS4
WS5
WS1
WS7
WS3
WS6

Consequently, using one more time the Process Flow method, three large blocks were also established: Creation of Parts (3.3.7.1), Processing of Parts (3.3.7.2) and Transport of Parts (3.5.1).

3.5.1 Transport of Parts

The type of the “Release token.WS” block is the “Schedule Source” and this block allows the definition of the current WS and associates it with a token called token.WS. A token is considered a class, visibly represented by a circle, with the ability to be updated throughout the execution of the Process Flow. The “Next WS?” decision block identifies the next WS where the AGV has to perform the load, more specifically its OB, as well as an indication of the correspondent IB where the entity will be unloaded.

After identifying the subsequent WS, it is necessary to reserve the AGV and then identify the oldest “entity” of the next WS. After that, it is essential to load and unload the current part and, finally, update the current WS token. Finally, the resource is released.

It should also be noted that when the AGV Routing table is completely crossed and the current line corresponds again to the current WS, it means that none of the WSs has parts in the respective OBs to be loaded and, therefore, the AGV is redirected to a state where it will be waiting until a part is processed ("Wait for Event").

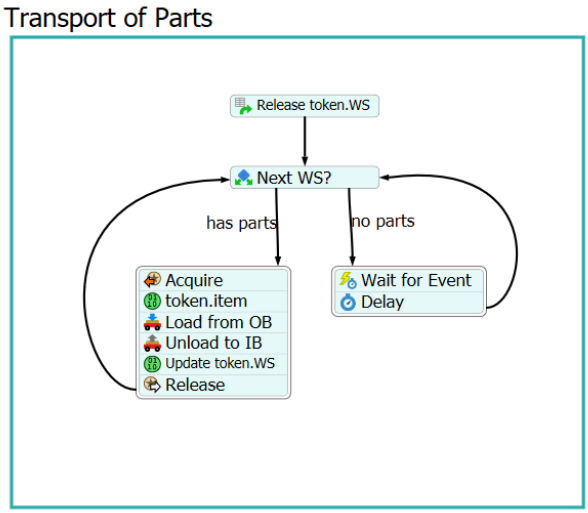

Figure 3.9: Representation of the “Transport of Parts” Block – Optimized Milk-Run Model

3.6 Implementation of the NearestWS Rule

This section intends to demonstrate a new heuristic also developed in the dissertation, called NearestWS Rule, whose main objective is to minimize the distances covered by the AGV. In terms of the decision principle, what distinguishes this heuristic from the Milk-Run algorithm (Chapter 3.5) is the fact that the AGV moves to the nearest WS with parts waiting to be transported, not following any sort of route.

As Figure 3.11 displays, if the current station is the WS4 and there are only parts already processed waiting for transport in the output buffers of WS2 and WS3, since the closest station to WS4 is WS2, the AGV will load the part present in the WS2 OB.

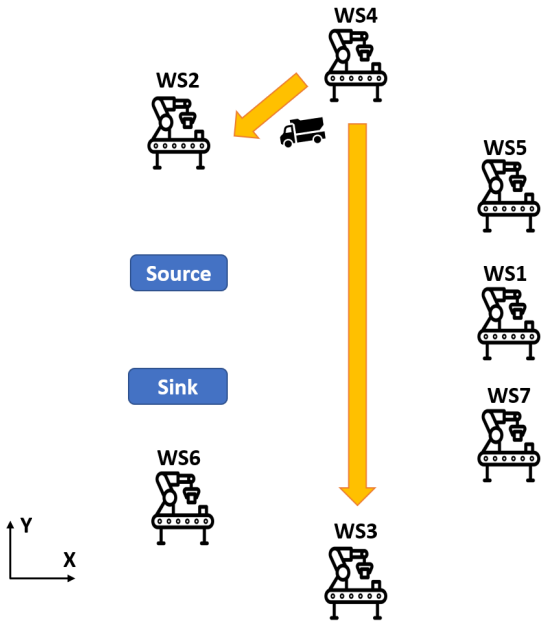

Figure 3.10: Illustrative example of the NearestWS Rule

One of the differentiating features of this new transport system is the fact that there is an external program that controls the simulation environment, receiving information from it in the form of observations, and sending actions that indicate the OB of the station where the load will be carried out. This approach is completely different from those previously studied, like the FIFO and optimized Milk-Run systems, since Flexsim will not be responsible for making the decisions, but an external program, called Server. This entity is programmed using the Python language and it is represented in Figure 3.11.

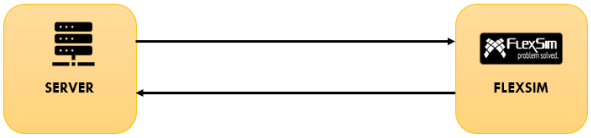

Figure 3.11: Server-Flexsim relationships

The communications between the Server and Flexsim environment, which will be a client, have to be successful. In this sense, the Transmission Control Protocol (TCP) was chosen to accomplish these communications requirements. For this, it is crucial to create Client-Server sockets, that will be discussed in the next Section 3.6.1.

3.6.1 Client-Server TCP Sockets

Sockets are an important tool used to send messages over a network. This network can be logical, local to the computer or even physically connected to an external network.

The TCP protocol, in addition to being very reliable, allows that the data sent by the client can be received and interpreted according to the order of sending by the Server. However, regarding the UDP protocol (User Datagram Protocol), it does not guarantee that the data will be received by the Server in the same sequence in which it was sent by the client. [32]

Within the scope of this dissertation, the communication between the Server and Flexsim entities requires the creation of sockets to exchange messages, respecting the TCP communication protocol. In this section, Figure 3.12 illustrates the global steps that constitute the creation and development of a Client-Server communication. The Server is considered a passive Agent because it waits for the connection request from the client. On the other hand, the client is characterized for taking the initiative and making a communication request, so it is an active entity in this whole process. [33]

Briefly, in a first instance, the Server is responsible for creating the respective socket, and then, an address and a port are immediately associated with it. Subsequently, the Server waits for the moment the client requests the connection. After the connection is successful, it is possible to send and receive messages between both entities, and at the end of this whole process of sending and receiving data, the connections are closed.

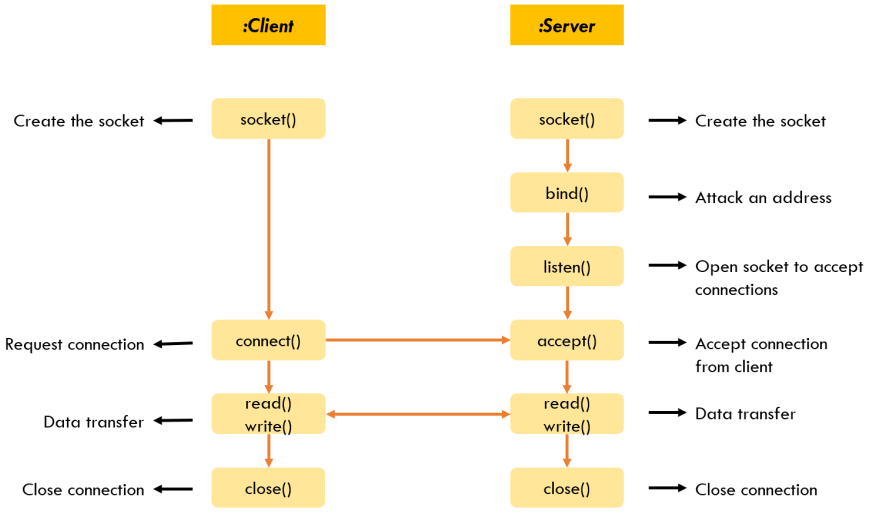

Figure 3.12: Client-Server TCP/IP communication diagram

3.6.2 Server – External Program

From an observation sent by the simulation environment that indicates the current state of each WS in terms of the presence of parts in their respective OB, as well as the current station where the AGV is placed, the Server will then send an action with the respective WS where the AGV will realize the load.

To store the current data of the simulation model in the Server, the architecture represented in the following UML diagram in Figure 3.13 was created, where three classes are defined:

- System_Data - contains the dictionaries referring to the distances between WSs and the own WSs and allows an association with the AGV class, in order to know the current station where the AGV is located;
- Workstation - allows to know if a specific WS has parts in its OB;
- AGV - contains the AGV current location;

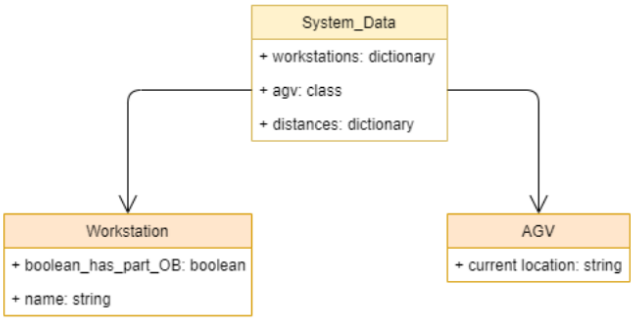

Figure 3.13: UML class diagram – NearestWS Rule

The observation sent by Flexsim is in the following form: 8 booleans + current WS, "10010000WS1", in which the 8 booleans allow to identify if each of the 7 WSs and Source have parts in their respective OB. Finally, the string allows to identify the current WS. For the specific example mentioned ("10010000WS1"), the conclusions drawn are as follows:

Table 3.7: Interpretation of the observation sent by Flexsim

Workstation	Does it have parts on OB?	Current WS
WS1	Yes	X
WS2	No	-
WS3	No	-
WS4	Yes	-
WS5	No	-
WS6	No	-
WS7	No	-
Source	No	-

It is important to note that the current station is updated whenever an unload is performed on one of the WSs, including Sink and excluding Source. In this way, Source will never assume the role of the current WS, as it is only used to perform loads. However, it is essential to know if there are parts to transport in it.

Returning to the diagram of the classes that constitute the Server, after extracting the information from the observation, it is necessary to fill all the attributes of the three classes, namely the "boolean_has_part_OB" attribute of the "Workstation" objects, as well as the current WS value, referring to the "AGV" object. Regarding the distance values, they have previously been read from an Excel file and inserted in the distance dictionary of the "System_Data" class.

After the treatment and organization of all this information, it is necessary to proceed with the implementation of the decision rule already mentioned (Figure 3.11), which allows the AGV to carry out the load in the nearest WS that contains parts in its OB.

Finally, this information that reflects the output buffer of the chosen station is sent back to the simulation environment, in the form of "OBWS" (for example "OBWS2").

3.6.3 Flexsim – Simulation Environment

3.6.3.1 Transport of Parts

After sending the observation to the Server, Flexsim waits for the response with the action containing the OBWS from the Server. However, if the Server detects that there are no parts in any of the OBs, it sends a message that orders the simulation to go to a wait state ("Wait for parts on OB"), as shown in Figure 3.14. If there are parts in at least one of the stations, the Server sends the action with the OBWS and the Flexsim accesses the oldest object present in that OB. Then, the next WS where the part will be unloaded is found.

Finally, in case that the number of parts that enter at Sink reaches the value 2080, which corresponds to the total number of released orders, the simulation ends and its state is sent to the "Sink" block.

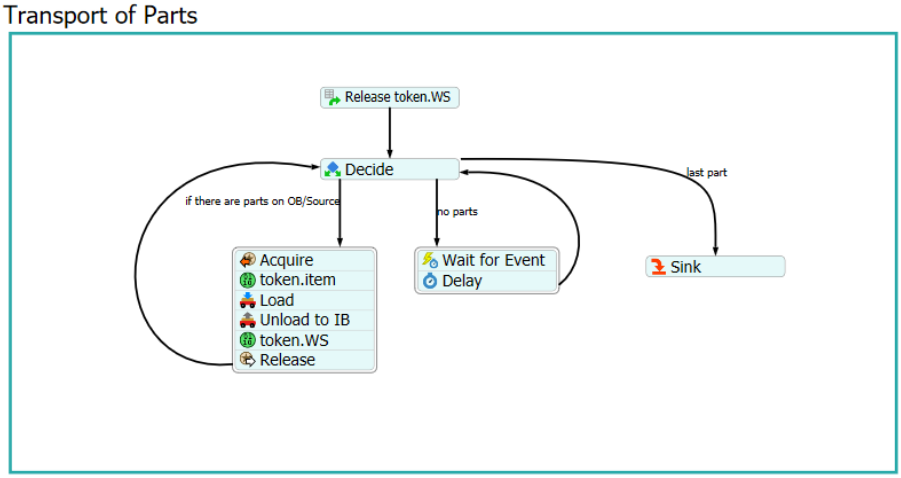

Figure 3.14: Representation of the “Transport of Parts” Block – NearestWS Rule

Chapter 4

Implementation of a Dynamic Transport System, using Reinforcement Learning Algorithms

4.1 Contextualization

The essential point that differentiates this dynamic transport system from the one previously presented in Section 3.6 is the inclusion of a third element (Agent), responsible for the decision making. Through Reinforcement Learning algorithms, the Agent is trained to define, like the previous system, the WS where the AGV will load the next entity.

In addition to this functionality, this Agent is also allowed to close and reset the Flexsim simulation program.

In this way, the Server is just an intermediary between this third entity and the Flexsim simulation environment, as shown in Figure 4.1. Like the simulation environment, previously defined as a client, the Agent will also be a client and the Server is the "go-between".

In other words, the Agent is the program that contains the code that acts on the training environment (Flexsim), executing certain functionalities.

Figure 4.1: Agent-Server-Flexsim relationships

In this logic, there are 3 possible types of interaction that the Agent can have with the simulation environment:

- Reset - its use allows the Agent to request the Flexsim reset;

- Close - enables the closure of the Flexsim;

- Step - grants the sending of an action, which represents the WS where the AGV will load the entity. The action decision is modified and improved through the neural network training by RL algorithms of the *StableBaselines* framework. This framework represents an improved and more stable version of the *OpenAI Baselines* algorithms. [34]

The only difference between the available RL algorithms is the strategy followed to redefine the policy used to define the action to be taken for a given state of the system. In the domain of this dissertation, the Proximal Policy Optimization (PPO) algorithm is currently the state-of-the-art in this area and, therefore, it was considered the most adequate to the current problem. Nevertheless, it is possible to use other algorithms in the future, with just simple changes in the Agent code.

This Agent constitutes a neural network that has as input data the observation from the simulation environment and presents as an output the representative action of the OB_Load. In the first moment, the network architecture studied was the Multi Layer Perceptron (MLP), with 16 neurons in the input layer corresponding to the observation, detailed in Tables 4.2 and 4.3, and 8 neurons referring to the output layer (action).

However, as explained in the following Sections 4.3 and 4.4, it was decided to use a Single Layer Perceptron (SLP) network, since the main objective is to make the network as simple as possible and at least replicate the performance that is given by the NearestWS Rule. These input and output network layers are also linked through connections defined as weights and which allow giving more significance to one or more neurons of the input layer.

Section 4.5 highlights the influence of the pre-training algorithms and the following sections 4.6 and 4.7 present the problems regarding Reinforcement Learning treated in this dissertation.

To conclude, in relation to the initial question about the Agent-Server-Flexsim relationships, it is necessary to establish communications between the 3 entities, using the TCP communication protocol (Section 3.6.1), through the creation of Client-Server sockets. For this purpose, the UML message diagrams represented in Figures 4.2, 4.3,4.4, 4.5 allow the identification of the set of communications and message exchanges between the 3 applications for each one of the 4 mentioned interactions.

4.2 UML Sequence Diagrams

4.2.1 Init

The Init function is only used in order to confirm the connection between the Server entity and Flexsim. For this, when the simulation program starts, it sends the message “FLEXSIM; INIT” and the Server returns a confirmation.

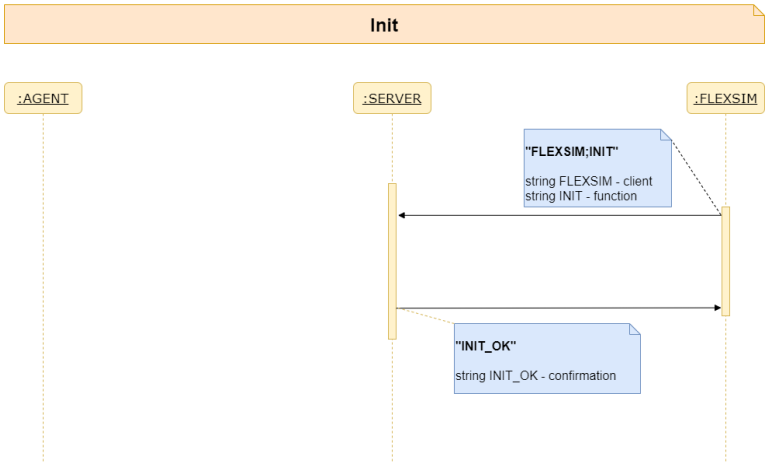

Figure 4.2: UML sequence diagram – Init

4.2.2 Reset

The request for the execution of the reset is sent by the Agent to the Server. As well as the other functionalities, the first message identifier represents the sender (Agent) so that the Server can identify it and continue with the processing of the message.

Then, after the Flexsim identifies the receipt of a reset, it will put the simulation model back to the beginning of the simulation run and build a string that will contain the indications of which WS contains entities in the respective OB and the current location of the vehicle. The constitution of this string is detailed in the next Figure 4.3.

After performing the reset, this message is sent to the Server, which will forward it to the Agent.

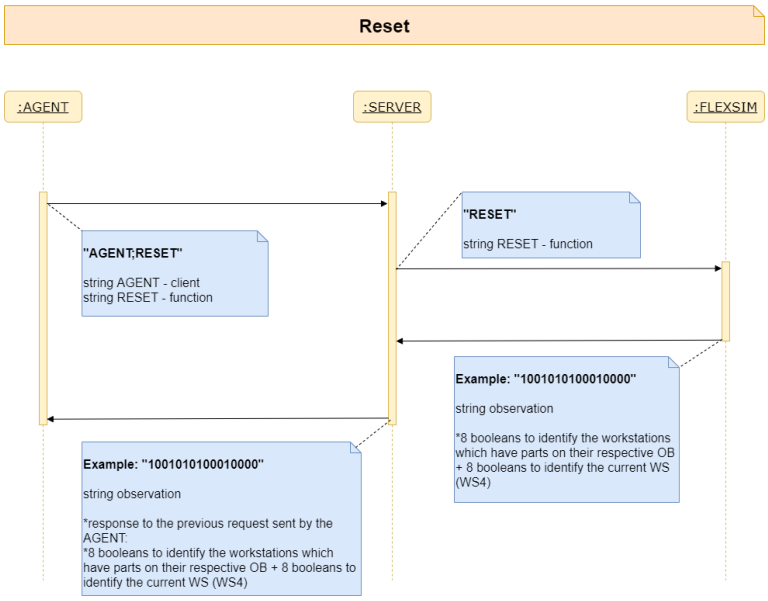

Figure 4.3: UML sequence diagram – Reset

4.2.3 Close

The first identifier of each message, in which the recipient is the Server, indicates which client is sending the information (Agent or Flexsim).

In this specific case of the close function, the Agent sends a message to the Server with the indication to end the Flexsim simulation program. After the Server receives and decodes this message, it executes a command that allows the simulation environment to be closed immediately.

After executing that respective command, the Server sends a message back to the Agent, as a way of confirming that the execution of the close command ("CLOSE_OK") was successful.

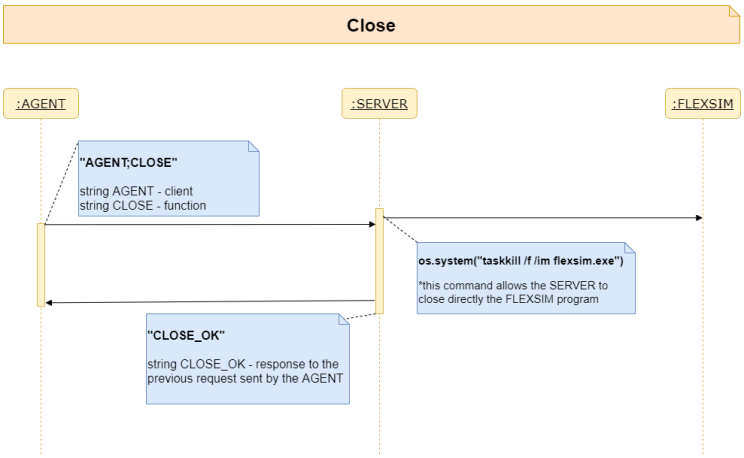

Figure 4.4: UML sequence diagram – Close

4.2.4 Step

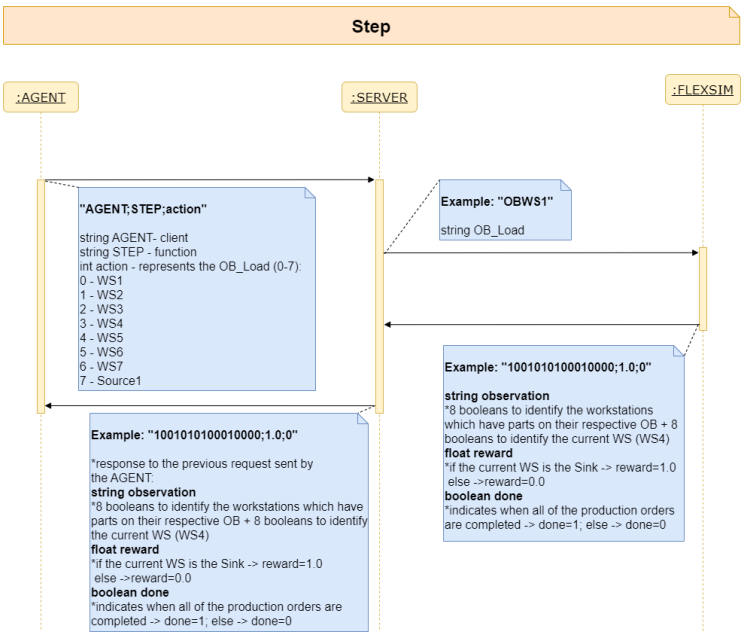

Figure 4.5: UML sequence diagram – Step

As indicated in section 4.1, the step allows the Agent to send an action that will be transmitted to the Server and later executed in Flexsim, taking into account the RL algorithm in question.

That said, the cycle starts with the sending of the step message from the Agent to the Server that extracts the "OB_Load", based on the message content. The action includes values between 0 and 7, in which each digit corresponds to a specific WS, represented in Table 4.1:

Table 4.1: Action-Workstation relationships

Action	Workstation	OB_Load
0	WS1	OBWS1
1	WS2	OBWS2
2	WS3	OBWS3
3	WS4	OBWS4
4	WS5	OBWS5
5	WS6	OBWS6
6	WS7	OBWS7
7	Source	Source

After identifying the WS or Source in question, the Server transforms this information into the OB_Load ("OBWSx" or "Source"), which is also represented in Table 4.1, and sends it to the Flexsim.

After that, the Flexsim will run the simulation model based on the action that has been provided, until an unload is done and a new action on the system is needed. At this point, the simulation environment will build a new message to send back to the Server. However, this message will also contain information that will allow the Server to identify if the result of the previous action allowed an entity to enter at Sink ("reward") and if all entities were already finished ("done").

Finally, the Server forwards this message back to the Agent, which, taking into account the feedback, will send a new step through its RL algorithm.

4.3 Neural Network General Architecture – Single Layer Perceptron

The neural networks are originated from the study of the human nervous system, characterized by its high complexity and non-linearity. Its great capacity for adaptation comes from the fact that the human brain is able to collect and organize its neurons in order to solve quite complex tasks.[7]

In this segment, as Figure 4.6 demonstrates, in addition to the input and output neurons, there are also other important properties such as the weights applied to each neuron and the activation function itself. The weights, normally between 0 and 1, allow one or more neuron(s) to get more relevance over another, so that it has more impact at the output. The activation function allows influencing the output value, introducing a non-linearity component.[7] Associated with the weights and activation function, there is also a parameter called "bias" that enables the adjust of the final value of the output, if necessary.

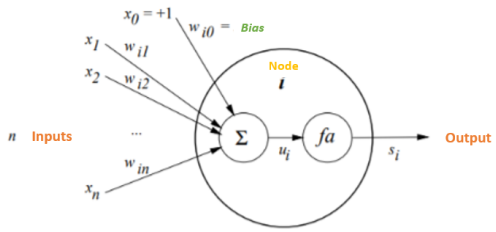

Figure 4.6: Neuron general structure, adapted from [7]

The main idea of the PPO algorithm is to make sure that the new policy is not too far from the previous one. The version used constitutes an implementation of *Stable Baselines* and in addition to using vectorized environments, it also allows that the observation and the resulting action can be of a discrete, box, mutidiscrete or multibinary type. In this specific scope, the observation assumes the box type (16,) and the action admits the discrete type (0-7). The next Figure 4.7 represents the network architecture of this dissertation.

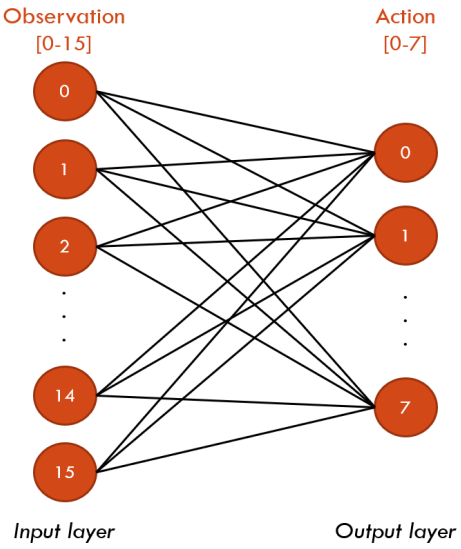

Figure 4.7: SLP network architecture

Within the observation, the first 8 booleans [0-7] indicate the WSs that have or have not parts in their respective OB (Table 4.2). The last 8 booleans [8-15] state the current AGV location, corresponding to a "one-hot encoding" (Table 4.3). Accordingly to the NearestWS Rule (Section 3.6), the stations state is updated just after an unload was performed. But, whenever an action corresponds to a WS that does not contain parts, the AGV goes there in the same way and the current WS is also updated.

As an example, if the observation is: [1001000010000000], it means that only WS1 and WS4 have parts in their OB and the current WS is WS1.

Table 4.2: Observation segmentation [0-7]

Observation [0-7]	Existence of parts
10000000	WS1
01000000	WS2
00100000	WS3
00010000	WS4
00001000	WS5
00000100	WS6
00000010	WS7
00000001	Source
00000000	There are no parts

Table 4.3: Observation segmentation [8-15]

Observation [8-15]	Actual WS
10000000	WS1
01000000	WS2
00100000	WS3
00010000	WS4
00001000	WS5
00000100	WS6
00000010	WS7
00000001	Sink
00000000	Source

4.4 Approach I – Initialization of Weights

Firstly, it was decided to define a neural network architecture and a set of weights that would allow the replication of the NearestWS Rule. This phase aimed to ensure that an SLP network, with the right set of weights, would be sufficient to obtain a performance equivalent to the results of the NearestWS Rule.

For demonstrative purposes, in Figure 4.8 it is possible to observe the behaviour of this practice, taking into account the observation content and the resulting action. Since it is just an example, there were only considered 3 WSs, whose distances between them are also illustrated in Figure 4.8.

In order to simplify the visualization, only the weights for the output neurons 1 and 2, corresponding to OBWS1 and OBWS2, are identified in the respective figures.

As the Figure 4.8 and Equation 4.3 show, the output neuron that will be activated is the first one, since it obtains the highest output value. As it makes sense, if there are parts in the WS1 and WS3 output buffers, and the current station is the WS1, then the AGV will load the part on the OB of the WS1.

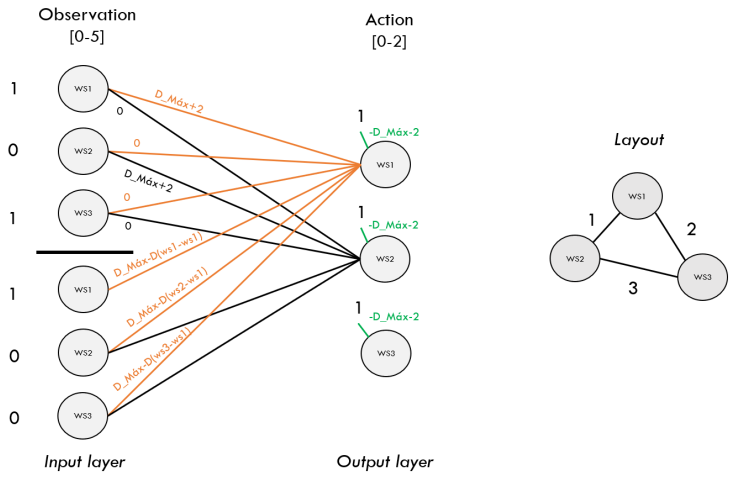

Figure 4.8: Neural Network – Example 1

$$Dmax = 3 \tag{4.1}$$

$$Bias = -Dmax - 2 = -3 - 2 = -5 \tag{4.2}$$

$$OB\_WS1 = (1*5+0*0+1*0)+(1*3+0*2+0*1)+(1*-5) = 3 \tag{4.3}$$

$$OB\_WS2 = (1*0+0*0+1*0)+(1*2+0*3+0*0)+(1*-5) = -3 \tag{4.4}$$

$$OB\_WS3 = (1*0+0*0+1*5)+(1*1+0*0+0*3)+(1*-5) = 1 \tag{4.5}$$

Like the Figure 4.9 denotes, the only case in which this network would not have the same behavior as the rule would be if the current WS was the Source, where it would be a draw between all stations with a part on their OBs (in this case: WS1 and Source).

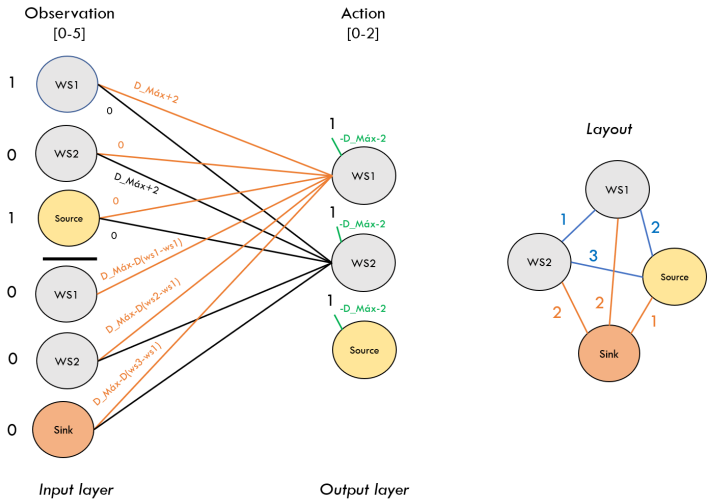

Figure 4.9: Neural Network – Example 2

$$Dmax = 3 \tag{4.6}$$

$$Bias = -Dmax - 2 = -3 - 2 = -5 \tag{4.7}$$

$$OB\_WS1 = (1*5+0*0+1*0)+(0*3+0*2+0*1)+(1*-5) = 0 \tag{4.8}$$

$$OB\_WS2 = (1*0+0*0+1*0)+(0*2+0*3+0*0)+(1*-5) = -5 \tag{4.9}$$

$$OB\_Source = (1*0+0*0+1*5)+(0*1+0*0+0*3)+(1*-5) = 0 \tag{4.10}$$

The *StableBaselines* framework, which will be used for Reinforcement Learning algorithms, does not allow the definition of initial network weights. Thus, it was not possible to create a policy with the calculated weight values. Still, this was an important exercise to ensure that this network architecture is capable of achieving performance at least equivalent to the previously defined rule. An alternative to initializing the weights is using a pre-training phase of the network, which is covered in section 4.5.

Thus, appears the necessity of studying the influence of the pre-training algorithms to observe its interference in the current problem.

4.5 Approach II – Neural Network Pre-Training

This approach allows, through the implementation of an algorithm with results already visible, in this particular case the NearestWS Rule, to have as starting point initial solutions that come from that rule.

In this phase, the behaviour that the Agent must have to perform a step in order to replicate the NearestWS Rule is defined. Then, several simulations are run using this Agent to obtain a data set with observation-action pairs. Finally, these data are used to train the SLP neural network according to Supervised Learning algorithms. This process ensures that the neural network starts its learning process with a set of weights that replicate the functioning of the rule.

For this, in order to verify the influence of this pre-training in the final solution, two models with different objectives will be approached, namely the minimization of makespan and the maximization of productivity, taking into account the PPO algorithm ("Makespan" and "Base" Models).

4.6 Makespan Model

In order to verify the correct behaviour of the Reinforcement Learning algorithm, PPO, the initial objective was to minimize the makespan. Thus, only one release order was defined for a random entity, more specifically a part type of number 4.

The main purpose of this model is to verify the correct behaviour of this transport system, observing the tendency of the final solutions to get the optimal solution. Associated with the Agent's learning process, the total number of time steps defined was 250.000 (steps), so that it is believed to be more than enough to achieve the optimal solution.

As a comparable point to the solutions achieved in the optimization process, the optimal solution can be obtained through the NearestWS and optimized Milk-Run rules, because there is just one part in the entire simulation process.

In this case, the pre-training of the network does not apply to this model since the subjacent rule would lead to an ideal behaviour already during the initial phase of the learning process, which is not intended. That said, since the objective is to analyse the behaviour of the optimization curve of the RL algorithm, the idea is that the final makespan values for each simulation performed approach the optimal solution. In this specific case, the ideal solution for a part type 4 is: 1486 seconds.

Figure 4.10 shows the Process Flow for the transport process implemented in the Flexsim simulation environment. The "Decide" block defines the next WS where the AGV has to go, taking into account the information sent by the Agent over the various steps it sends.

If there are no parts already processed waiting to be transported to the next WS, the AGV goes to a state where it is waiting for the part in question to be processed (given that in this particular model there will only be one part throughout the entire simulation).

In the case that the Agent sends a step referring to a WS that does not contain parts in his OB, since the network is in the learning process, the AGV will also go there and the current WS will be updated.

As soon as the part enters at Sink, the Flexsim will send the value corresponding to the makespan as a reward to the Server, but with the negative signal (-makespan). This particularity is justified by the fact that the purpose of the PPO algorithm is always to maximize something. In this case, the goal is to maximize (-) makespan, in other words, minimize the makespan. Still in this scenario, the third parameter, called "done", is also sent as a unit, since the part has already been entered into the Sink and the simulation is finished.

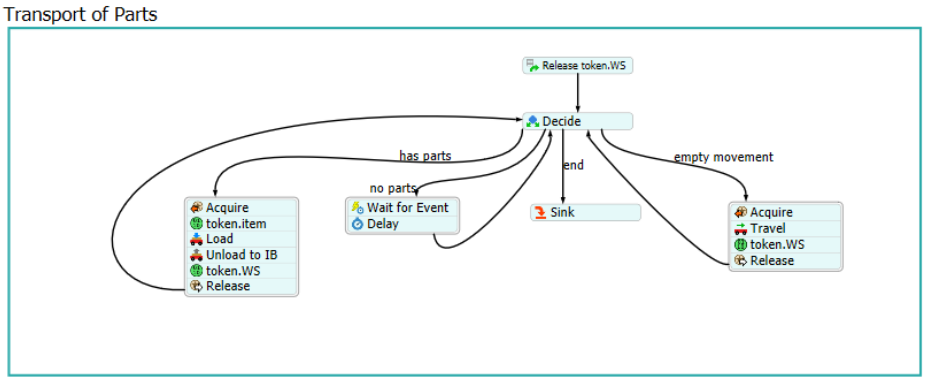

Figure 4.10: Representation of the "Transport of Parts" Block – Makespan Model

After running several simulations, it was found that the model, under the current conditions in which the makespan is only sent at the end of the simulation when the part enters at Sink, does not allow reaching the optimal solution. It was concluded, therefore, that the fact that the rewards always assume null values, with the exception of the response to the last step that includes the makespan, is not enough for the network to be able to make an adequate learning and find the ideal solution within the defined 250k time steps.

Therefore, some criteria were introduced in order to check whether it is possible to achieve the optimal solution, in particular:

- the sending of rewards between time steps with the value of a time interval initiated by a load and ended with an unload (including the processing time), in detriment of sending the total makespan only at the end of the simulation;
- the introduction of penalties, if the movements made do not involve the transportation of parts;
- the insertion of normalizations, modifying the value of the rewards, converting them in the interval [-1,0].

Later, in order to study the influence of these changes on learning performance, all of these scenarios presented in Table 4.4 were implemented and simulated.

Table 4.4: Scenarios of the Makespan Model

Scenario	Time steps	Normalization	Normalization Factor	Type of Reward	Penalty
1	250k	No	0	makespan	0
2	250k	No	0	makespan	-FN/2
3	250k	Yes	15k	makespan	0
4	250k	Yes	15k	makespan	-0.5
5	250k	No	0	between time steps	0
6	250k	No	0	between time steps	-FN/2
7	250k	Yes	FN	between time steps	0
8	250k	Yes	FN	between time steps	-0.5

FN = 4*Travel time between the two most distant stations + 2*Higher Processing Time (Part type nº 4).

For all the cases presented above in Table 4.4, the model converts the solutions obtained to the optimal solution, except for the scenario in which the rewards are sent between time steps without normalization or penalty. The question that immediately arises is related to the time that is necessary to reach the optimal solution and, in this context, it was found that normalization is the factor that allows the model to tend more quickly towards the optimal solution. These results are presented and discussed with more detail in Chapter 5.

In the following subsections are presented the steps used in the normalization and penalty methods for the types of rewards: between time steps and makespan.

4.6.1 Normalization

The main objective of normalization is to verify whether, by adjusting the values of the rewards to values between [-1, 0], the algorithm will be able to converge better to a good solution and increase its learning capacity.

For this, the normalization factor depends on the type of reward (between time steps or makespan) and application of penalties, characterized by the following Equations 4.11 and 4.12:

4.6.1.1 Type of Reward: Makespan

$$Normalization\ factor = 2 * 7.5k = 15k\ (seconds) \tag{4.11}$$

$$Reward \in [-0.5,\ 0],\ without\ penalty \tag{4.12}$$

This value attributed to the normalization factor is justified by the fact that it was observed during all the simulations previously carried out that the final value of the makespan, for a single part of type 4, never exceeded the value of 7.5k seconds, with rare exceptions. Therefore, taking into account the subsequent application of penalties (-0.5), the value considered for the normalization factor was 15k seconds, as it is twice 7.5k. Therefore, without applying the penalty, the reward assumes values between -0.5 and 0, since the purpose of optimization is to maximize (-) makespan.

4.6.1.2 Type of Reward: Between Time steps

$$Normalization\ factor = 4 * T1 + 2 * T2 \tag{4.13}$$

in which $T1$ = Travel time between the two most distant stations; $T2$ = Higher Processing Time (Part of type 4).

$$Reward \in [-0.5,\ 0],\ without\ penalty \tag{4.14}$$

As in the previous case regarding makespan, the normalization factor applied allows rewards to assume values between -0.5 and 0, without the application of penalties.

In the specific case of a part type 4:

- T1 = 48.827 seconds;
- T2 = 420 seconds.

$$Normalization\ factor = 1035.308 \tag{4.15}$$

4.6.2 Penalties

Penalties are applied when the AGV performs movements without a part. The objective is, therefore, through the application of this penalty with the value of -0.5, to make it clear to the Agent that, in this specific case, it is not a good practice to make movements without a part, given that it

is only one part in the simulation environment. That said, taking into account the type of reward, their values will be included in the following range:

$$Reward \in [-1,\ 0] \tag{4.16}$$

4.7 Base Model

After analysing the Agent's behaviour towards the model whose objective was to minimize the makespan, it was also decided to analyse the issue of maximizing productivity and create a new model that allows studying the PPO algorithm.

With regard to the Process Flow of the simulation model, all the blocks are coincident with the blocks of the previous model, referring to the makespan. The only relevant difference is the issue of the rewards that are sent from Flexsim to the Agent, intermediated by the Server. In response to a step from the Agent, the simulation environment constructs the observation that contains the current status of each WS, in terms of the presence of parts, and the current position of the AGV. However, the rewards assume a unit value whenever a part is entered at Sink. Otherwise, they are assigned a null value. In relation to the parameter "done", it will be unitary whenever the defined time horizon is reached, ending the simulation.

Moreover, whenever a reward assumes a value of 1, there is an internal counter in the simulation environment that is increased, providing an updated indication of the number of parts manufactured so far, to a later comparison of results with the previous productivity models.

In this context, unlike what happens with the makespan model, the use of pre-training becomes important in order to verify whether it has had an effect or not. The rule implemented and associated with the pre-training is based on one of the rules previously described in this dissertation, better known as the NearestWS Rule, which allows AGV to perform a load on the nearest WS that contains parts in the respective OB.

A not less important issue is to understand the effects of the introduction of penalties on productivity. For this, given that the rewards are already normalized in the interval [0, 1], more specifically assuming the values 0 or 1, the defined penalty was (-) 0.5, similar to the previous Makespan Model. That said, whenever the AGV makes an empty movement, ordered by the Agent, the reward returned will be -0.5.

Regarding the number of steps sent by the Agent, after several runs, it was concluded that the value of 10M steps would be the most indicated value and more than enough to reach the final solution. For further comparison of results, the time horizons chosen were 36 and 52 hours.

Finally, it should be noted that, whenever a "done" value is assigned to 1, the productivity value related to the simulation in question is also written in a text file, for later analysis.

4.8 Robustness of the Simulation Models

This section was created with the intention of verifying whether the models covered in the previous sections, namely the FIFO, optimized Milk-Run, NearestWS Rule and Base Model, referring to productivity, have an appropriate behaviour against possible changes in the production mix or in the processing times.

4.8.1 Production Mixes

With regard to changes in production mixes, the order in which the different part types are launched will suffer some variations and it will be randomly generated, using the Microsoft Excel software tool.

Just like what happens in the original mix, in which the probability of each part type being released is nearly 20% (Table 4.5), the same restriction applies to the next 5 different mixes generated.

Briefly, the different production mixes are generated so that the part type associated with each production order is determined randomly, but with the same final probability of being created (20%).

Table 4.5: Quantity of the different part types – Original Mix

Part Type	Quantity (units)
1	448
2	410
3	429
4	424
5	369

Therefore, as shown in Chapter 5, there are small differences in the final quantities of parts produced for each one of the mixes, due to the way they were created.

4.8.2 Processing Times

The processing times corresponding to each part type and WS were introduced in a stochastic environment, in order to follow, more concretely, a triangular probability distribution.

Assuming that T_{ij} represents the processing time of the part type i in the WS j, it was adapted to respect the triangular distribution, shown in Figure 4.11. In this segment, a variability factor α was also added to simulate possible variations that may occur in a real context.

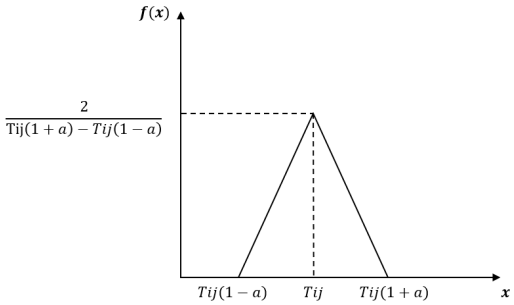

Figure 4.11: Triangular distribution function, referring to the processing times

In a rigorous way, the respective triangular distribution is defined by the following expressions:

$$f(x|a,b,c) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)}, & \text{se } a \leq x < c \\ \frac{2}{(b-a)}, & \text{se } x = c \\ \frac{2(b-x)}{(b-a)(b-c)}, & \text{se } c < x \leq b \\ 0, & \textit{otherwise} \end{cases} \tag{4.17}$$

Where a, b and c take the next values:

$$\begin{vmatrix} a = T_{ij}(1\text{-}\alpha) \\ b = T_{ij}(1\text{+}\alpha) \\ c = T_{ij} \end{vmatrix} \tag{4.18}$$

The results of this procedure, which is based on the introduction of the processing times in a stochastic environment, are described in Chapter 5.

Chapter 5

Results Analysis

In general, the purpose of this chapter is to analyse the results obtained from the implementation of the past simulation approaches, with or without the combination of Machine Learning techniques.

Therefore, this division allows us to observe all the consequences and effects of the implementation of the algorithms previously covered in chapters 3 and 4, in terms of productivity, namely with regard to the following models: FIFO, optimized Milk-Run, NearestWS Rule and implementations that involve the use of RL algorithms (Makespan and Base Models). In addition to productivity, the makespan, described in Section 4.6, will also be evaluated, but only with regard to the model referring to the PPO algorithm (Makespan Model).

Throughout all simulations, the AGV speed has always remained constant, assuming the value of 1.5 m/s, although the values associated with the total execution time have been modified for later analysis of the productivity.

In the final phase of the chapter, some pertinent questions are also presented that enable the evaluation of the robustness of all the simulation models.

It should also be noted that the software development environments involved in the dissertation, referring to computer programming and simulation, were, respectively, Pycharm (Python 3.7) and Flexsim (Educational Version 2019). In this segment, all simulations were equally performed on an HP Pavilion computer, with an Intel Core i7 processor, Quad-core up to 4 GHz and 16 GB RAM.

5.1 FIFO Model

This model is considered the simplest one of the 5 models implemented and, therefore, it is also the one that presents, without any surprise, the lowest values in terms of productivity.

Considering that the AGV speed remained constant at 1.5 m/s and the defined time horizons are 36 and 52 hours, the obtained results are expressed in the following Table 5.1.

Regardless of the number of simulations carried out, the number of parts that enter at Sink remains constant, since the decision rule applied to the AGV (FIFO) is not dynamic.

Table 5.1: Productivity related to the FIFO Model

Time Horizon (hours)	Productivity (parts)
36	177
52	262

5.2 Optimized Milk-Run Model

As can be seen in Table 5.2, the productivity values related to the optimized Milk-Run Model are considerably higher than those of the FIFO Model, as would be expected, given that the decision rule adopted by the autonomous vehicle is much more sophisticated.

Table 5.2: Productivity related to the optimized Milk-Run Model

Time Horizon (hours)	Productivity (parts)
36	266
52	389

5.3 NearestWS Rule

In contrast to the two previous models, the model referring to the NearestWS Rule, implemented through an external Server program, is characterized by a dynamic transport system, as explained in Section 3.6.

Taking into account the values presented in Table 5.3, it is possible to conclude that, among these three previous models, the optimized Milk-Run model is the one which presents the best productivity values.

Table 5.3: Productivity related to the NearestWS Rule

Time Horizon (hours)	Productivity (parts)
36	258
52	377

Apparently, although the NearestWS Rule may seem more efficient once the load is carried out at the nearest WS, it does not necessarily imply that the productivity is higher at the end, namely, when there are significant bottlenecks on the productive system.

In other words, if the load is done, for example, at the nearest WS, but that same WS contains one part in its OB to be transported to a station whose IB is already full, it can be preferable to perform the load in a WS more distant that has one part in its OB whose destination is a WS without parts in the respective IB. In this case, it will be guaranteed a higher occupancy rate of the machines, contributing in the long and short term to the increase of the number of parts that enter at Sink.

5.4 Makespan Model

As explained in Section 4.6, the study of makespan aimed to analyse the behaviour of the PPO algorithm, similar to what happens with the productivity in the Base Model.

That said, in order to understand the influence of the criteria of normalization, penalty and sending of rewards between time steps on the speed with which the model tends to the final solution, the following results obtained are presented, with respect to each one of the scenarios presented in Table 4.4, illustrated in Figures 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8 and represented in Tables 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11.

5.4.1 Scenario 1 – Type of reward: makespan, without penalties and without normalization

As can be seen from Figure 5.1 and Table 5.4, the final results of this scenario do not allow the achievement of a concrete and correct final value related to makespan. In this sense, this fact is the main reason for studying the scenarios that involve the normalization, penalties and the sending of rewards between time steps.

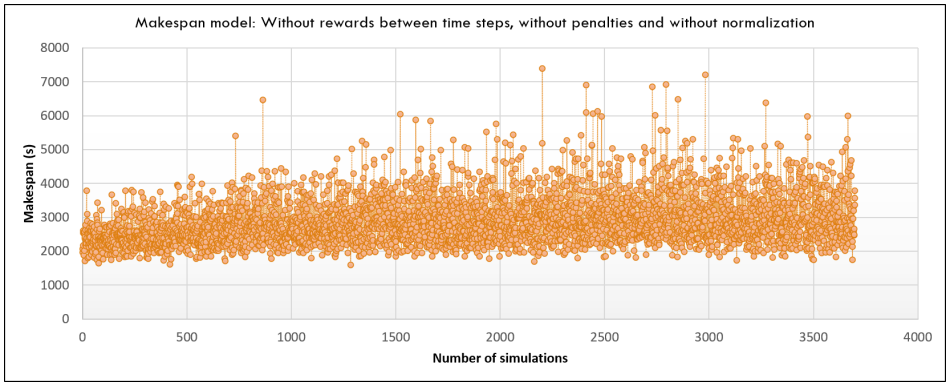

Figure 5.1: Makespan related to Scenario 1, for a total of 250k time steps

Table 5.4: Numerical interpretation of makespan referring to Scenario 1 (in seconds)

Minimum Value	Maximum Value	Final Value
1594	7389	3371

5.4.2 Scenario 2 – Type of reward: makespan, with penalties and without normalization

Looking at Figure 5.2, it is concluded that the introduction of the penalty criterion was essential for the model to move towards the optimal solution of 1486 seconds.

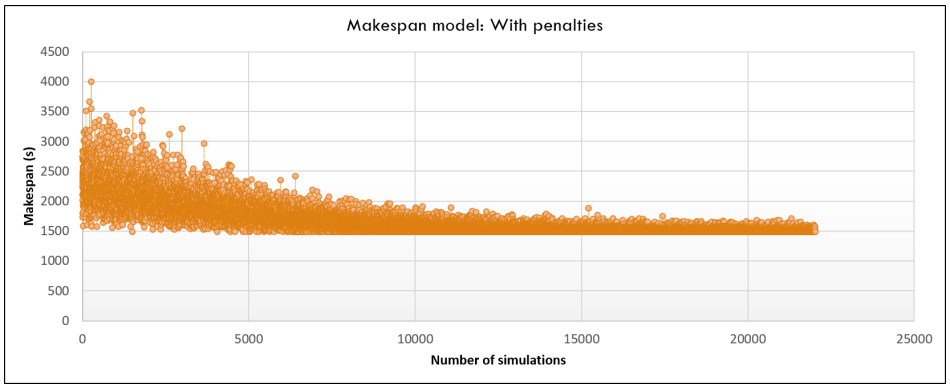

Figure 5.2: Makespan related to Scenario 2, for a total of 250k time steps

In addition to this, Table 5.5 allows us to conclude that the minimum and maximum values, referring to the learning process, are also lower when compared to the original scenario.

Table 5.5: Numerical interpretation of makespan referring to Scenario 2 (in seconds)

Minimum Value	Maximum Value	Final Value
1486	4000	1486

5.4.3 Scenario 3 – Type of reward: makespan, without penalties and with normalization

As shown in Figure 5.3 and Table 5.6, when the normalization is applied in a singular way, the model is able to converge within the 250k time steps range to the final solution, like what happens in the second scenario. However, it is also possible to verify that the curve is flatter in this current model, which allows us to conclude that the Agent's learning phase reaches the final value earlier than the second scenario, which is an advantage.

Therefore, the fourth scenario allows the study of these two parameters when applied in a simultaneous way, in order to verify the consequences in the final result, which is expected to be better than what happened in the previous scenarios individually.

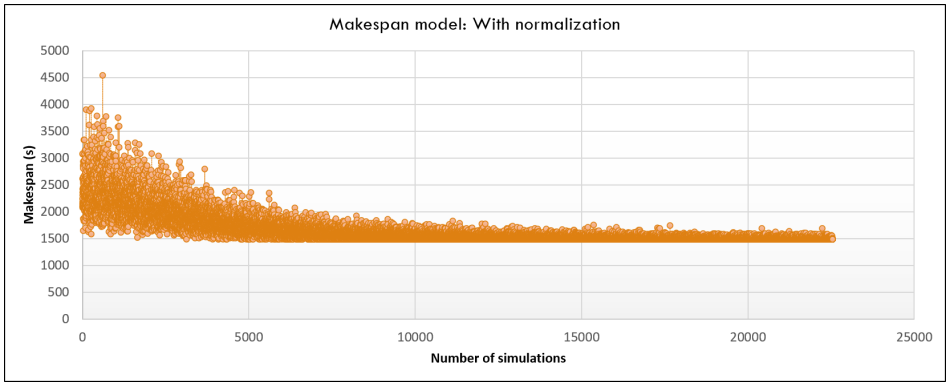

Figure 5.3: Makespan related to Scenario 3, for a total of 250k time steps

Table 5.6: Numerical interpretation of makespan referring to Scenario 3 (in seconds)

Minimum Value	Maximum Value	Final Value
1486	4534	1486

5.4.4 Scenario 4 – Type of reward: makespan, with penalties and with normalization

As predicted and illustrated in Figure 5.4 and Table 5.7, combining the two previous scenarios (2 and 3), the results are even more satisfactory, given that the model converges to the final solution earlier.

Hence, it is now time to analyse the influence of sending the rewards between time steps, through the next scenarios.

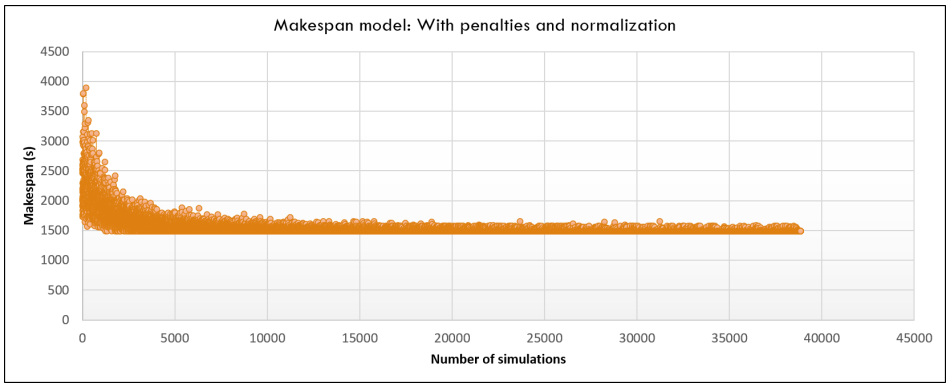

Figure 5.4: Makespan related to Scenario 4, for a total of 250k time steps

Table 5.7: Numerical interpretation of makespan referring to Scenario 4 (in seconds)

Minimum Value	Maximum Value	Final Value
1486	3896	1486

5.4.5 Scenario 5 – Type of reward: between time steps, without penalties and without normalization

Along the same lines of the results obtained in the first scenario, the isolated sending of rewards between time steps also does not allow obtaining satisfactory results, as shown in Figure 5.5. Though, the outcome for scenario 5 is even more unsatisfactory than the one of the first scenario (Table 5.8).

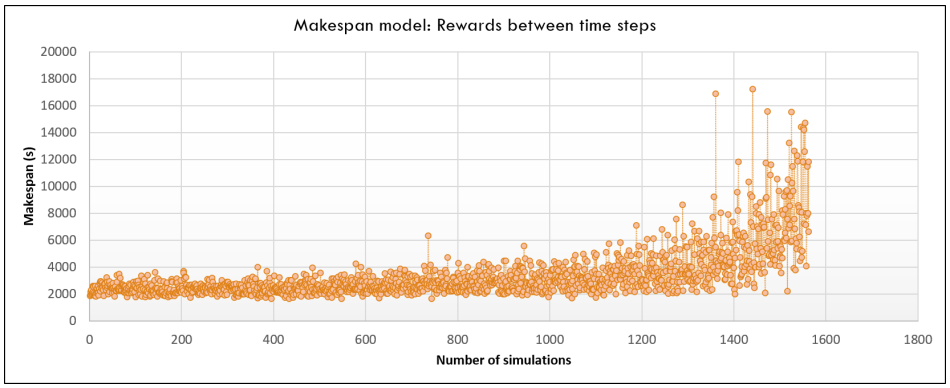

Figure 5.5: Makespan related to Scenario 5, for a total of 250k time steps

Table 5.8: Numerical interpretation of makespan referring to Scenario 5 (in seconds)

Minimum Value	Maximum Value	Final Value
1630	17202	6645

5.4.6 Scenario 6 – Type of reward: between time steps, with penalties and without normalization

Analysing the scenarios 2 and 6, both using penalties, it appears that the two have similar behaviours, with a greater number of simulations carried out in the second scenario. These questions can be observed in Tables 5.5 (scenario 2) and 5.9 (scenario 6).

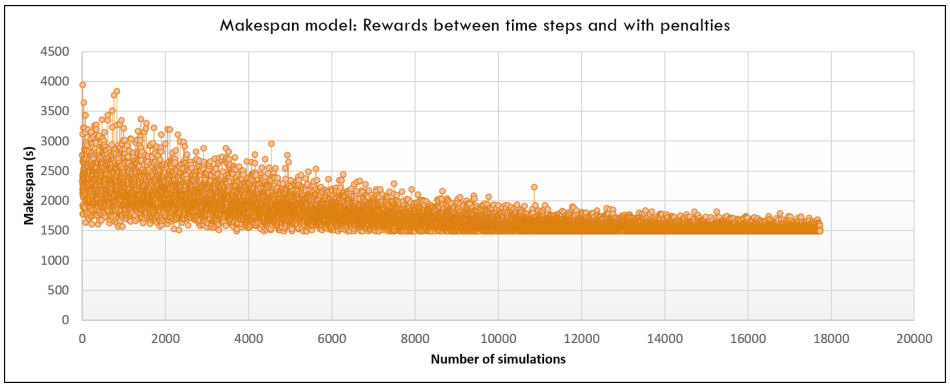

Figure 5.6: Makespan related to Scenario 6, for a total of 250k time steps

Table 5.9: Numerical interpretation of makespan referring to Scenario 6 (in seconds)

Minimum Value	Maximum Value	Final Value
1486	3945	1486

5.4.7 Scenario 7 – Type of reward: between time steps, without penalties and with normalization

As in the previous scenario, the optimal value is also achieved (Figure 5.7). Nevertheless, as evidenced in scenarios 2 and 3, the use of the normalization in a singular way allows obtaining the best solution earlier when compared to the isolated application of penalties.

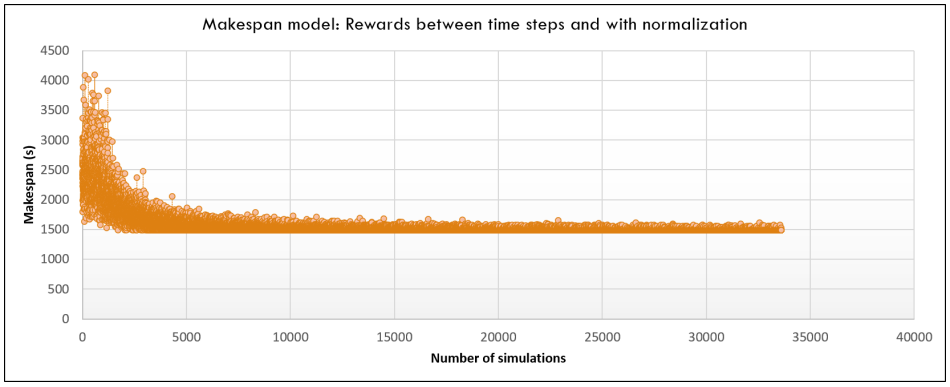

Figure 5.7: Makespan related to Scenario 7, for a total of 250k time steps

Table 5.10: Numerical interpretation of makespan referring to Scenario 7 (in seconds)

Minimum Value	Maximum Value	Final Value
1486	4096	1486

5.4.8 Scenario 8 – Type of reward: between time steps, with penalties and with normalization

As well as the fourth scenario, and as expected, when the normalizations and penalties are applied simultaneously, the optimal solution is obtained in advance in relation to the models in which the penalties or normalization are applied separately.

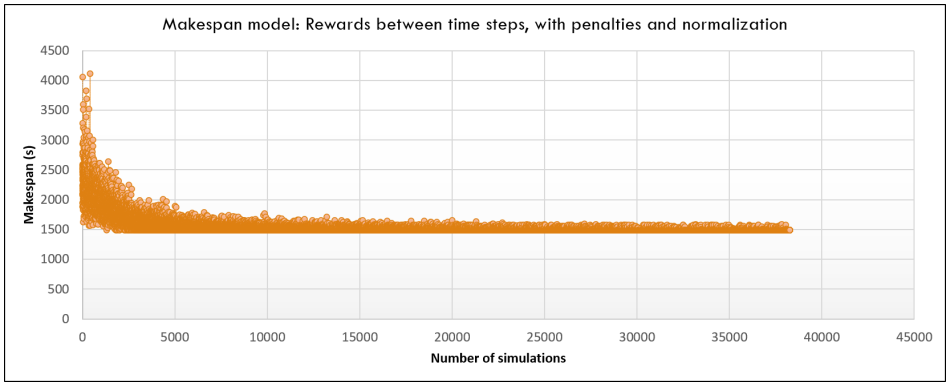

Figure 5.8: Makespan related to Scenario 8, for a total of 250k time steps

Table 5.11: Numerical interpretation of makespan referring to Scenario 8 (in seconds)

Minimum Value	Maximum Value	Final Value
1486	4116	1486

5.4.9 Synthesis

In short, regarding the models presented in Section 5.4 related to makespan, it is possible to state that the Reinforcement Learning PPO algorithm was correctly implemented, given that the models follow their characteristic curve, allowing to maximize (-) makespan.

The results also suggest that the normalization of rewards and the application of penalties improve the learning capacity of the model.

5.5 Base Model

Returning to the question of the productivity and taking into account the PPO algorithm, the results achieved for the time horizons of 36 and 52 hours were divided according to the existence or not of a pre-training of the neural network and are represented and indicated, respectively, in Figures 5.9, 5.10, 5.11, 5.12 and Tables 5.12, 5.13, 5.14, 5.15. With respect to the Agent's learning period, as referred to in Section 4.7, the number of time steps was set to 10M. Regarding the pre-training, it respects the NearestWS Rule, discussed in Section 3.6.

For a better understanding of the influence of the pre-training in the learning phase, some sets of observation-action pairs resulting from this same stage will also be presented. For this purpose, a probability function will be applied to the actions, after the learning phase is performed. The objective of this step is to verify in which WS the AGV will carry out the respective load.

Subsequently, similarly to the previous Section (5.4) referring to the makespan, the results derived from the application of a penalty factor (-0.5) are also presented.

5.5.1 Without pre-training, for a time horizon of 36 hours

Figure 5.9 allows us to infer that the model's behaviour, with regard to productivity values, follows the PPO optimization algorithm.

As can be seen, the model reaches the maximum productivity values in the first simulations. This phenomenon is explained by the weights which are arbitrarily assigned in the initial learning phase of the network. Subsequently, after a certain number of simulations, the model presents lower productivity values, also due to the weights defined by the Agent and because at the beginning of the information collection, the network has little knowledge yet. However, the productivity characteristic curve allows verifying that the productivity values tend to a final maximum solution, which approves that the learning phase was performed correctly.

The final value of productivity, which comes from the network's learning process, is 343 parts that enter at Sink.

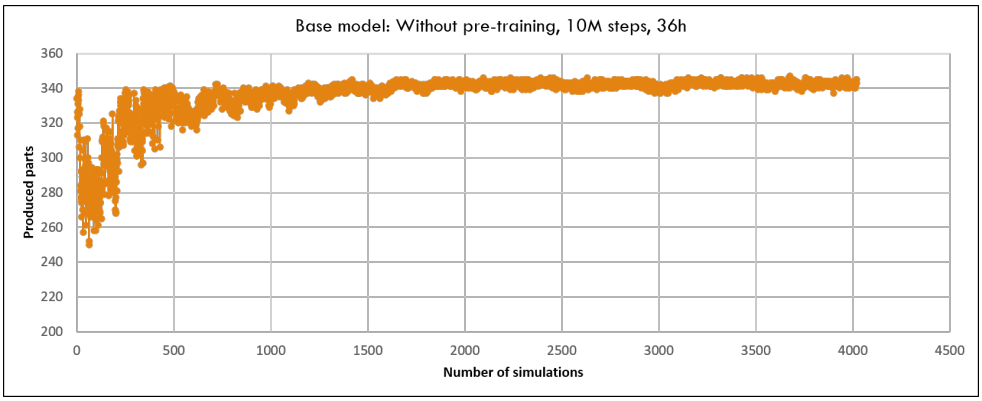

Figure 5.9: Productivity referring to the Base Model without pre-training, to a total of 10M time steps and a time horizon of 36h

Table 5.12: Numerical interpretations of the productivity (in parts) referring to the Base Model without pre-training, to a total of 10M time steps and a time horizon of 36h

Minimum Value	Maximum Value	Final Value
250	347	343

The set of observation-action pairs resulting from the learning phase, without the pre-training, are exemplified below:

- **Observation-Action pair nº 1:** Parts in WS1 | Current WS: 4
  Observation: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
  Action: [8.3619851e-01, 1.9839258e-04, 7.7551082e-02, 2.0058152e-04, 1.7890350e-04, 6.3502125e-02, 1.9391190e-02, 2.7792549e-03]

  In this particular situation, the behaviour of the network is as expected, given that the highest probability is associated with the WS1.

- **Observation-Action pair nº 2:** Parts in WS3, WS4 | Current WS: 4
  Observation: [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
  Action: [1.4122830e-03, 1.3475084e-05, 9.6138531e-01, 2.4895007e-03, 7.2637980e-05, 2.9261060e-02, 3.8612273e-03, 1.5044548e-03]

  With regard to this observation-action pair, the network's behaviour does not correspond to what would be done by the NearestWS Rule, given that, in this situation, the AGV would load a part in the WS3 and not in the WS4, as the high probability suggests.

5.5.2 With pre-training, for a time horizon of 36 hours

Like the model without pre-training, this one also respects the curve referring to the PPO optimization algorithm, reproduced in Figure 5.10.

However, in comparison with the model without pre-training, it appears that the results of this model before the learning stage are lower, in terms of productivity. Despite that, similarly to what happened in the optimized Milk-Run and NearestWS Rules, this episode can be explained by the fact that, sometimes, the empty travels or the transport of parts for distant WSs, more common in the model without pre-training, can bring advantages in terms of the final result of productivity, reducing possible bottlenecks and balancing the machine occupancy rates.

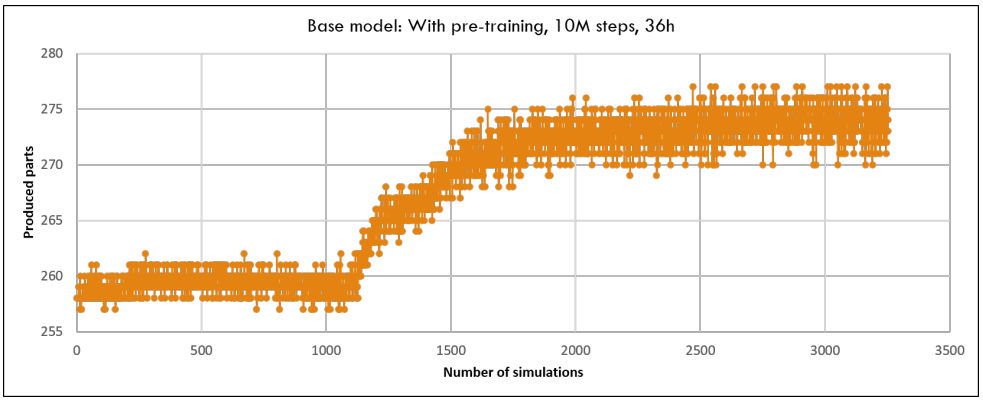

Figure 5.10: Productivity referring to the Base Model with pre-training, to a total of 10M time steps and a time horizon of 36h

Table 5.13: Numerical interpretations of the productivity (in parts) referring to the Base Model with pre-training, to a total of 10M time steps and a time horizon of 36h

Minimum Value	Maximum Value	Final Value
257	277	274

One more time, the examples of the observation-action pairs, resulting from the learning process, evidence the previous accomplishment of the learning phase:

- **Observation-Action pair nº 1:** Parts in WS1 | Current WS: 4
  Observation: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
  Action: [9.9999833e-01, 6.0297623e-07, 9.3485424e-22, 1.0542382e-06, 3.6167533e-10, 4.3565872e-16, 4.4415426e-18, 2.8845209e-15]

  The behaviour of the network is the one expected, since the WS1 has the highest probability.

- **Observation-Action pair nº 2:** Parts in WS3, WS4 | Current WS: 4
  Observation: [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
  Action: [9.00684897e-25, 1.09724185e-16, 1.59646764e-13, 1.00000000e+00, 1.18856031e-19, 6.28844218e-25, 2.55962060e-27, 2.96594020e-27]

  In contrast with what happened in the previous scenario in which the pre-training did not occur, the higher probability is now consistent with the expected (WS4), according to the NearestWS Rule.

5.5.3 Without pre-training, for a time horizon of 52 hours

According to the graph in Figure 5.11, the behaviour of the learning phase, like the Model without pre-training for the 36-hour time horizon, follows the PPO algorithm, despite the initial drop of the productivity values.

Regarding the final value of productivity, this is logically higher than the productivity referring to the period of 36 hours.

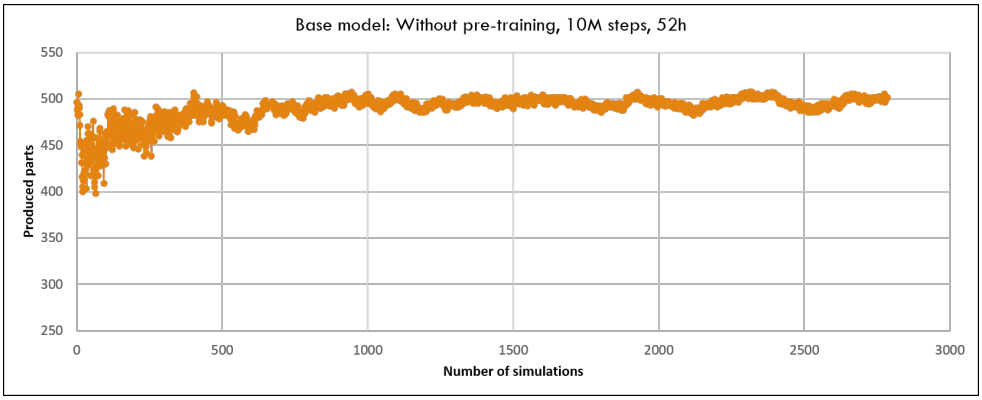

Figure 5.11: Productivity referring to the Base Model without pre-training, to a total of 10M time steps and a time horizon of 52h

Table 5.14: Numerical interpretations of the productivity (in parts) referring to the Base Model without pre-training, to a total of 10M time steps and a time horizon of 52h

Minimum Value	Maximum Value	Final Value
398	507	501

For the same observation-action pairs, the solutions are the following ones:

- **Observation-Action pair nº 1:** Parts in WS1 | Current WS: 4
  observation: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
  Action: [8.6369443e-01, 2.2193830e-04, 4.7343463e-02, 6.8649366e-05, 1.9357287e-04, 7.4317701e-02, 1.2128289e-02, 2.0318190e-03]

  For this case, the behaviour of the network is the one expected (WS1).

- **Observation-Action pair nº 2:** Parts in na WS3, WS4 | Current WS: 4
  observation: [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
  Action: [2.6672529e-03, 7.9401660e-05, 9.7478318e-01, 2.3247360e-03, 1.2263234e-04, 1.7269300e-02, 2.2482565e-03, 5.0515484e-04]

  In accordance with the results of the last model (36h), without pre-training, it is concluded that the learning of the network leads the AGV to do the load in the further station (WS3) instead of the nearest one (WS4).

5.5.4 With pre-training, for a time horizon of 52 hours

In this last scenario, Figure 5.12 proves the expected behaviour of the model. As Table 5.15 indicates, the final value of productivity is lower than the value obtained in the scenario in which the pre-training did not occur. The most admissible reasons are presented in Section 5.6 that compares the results of all models.

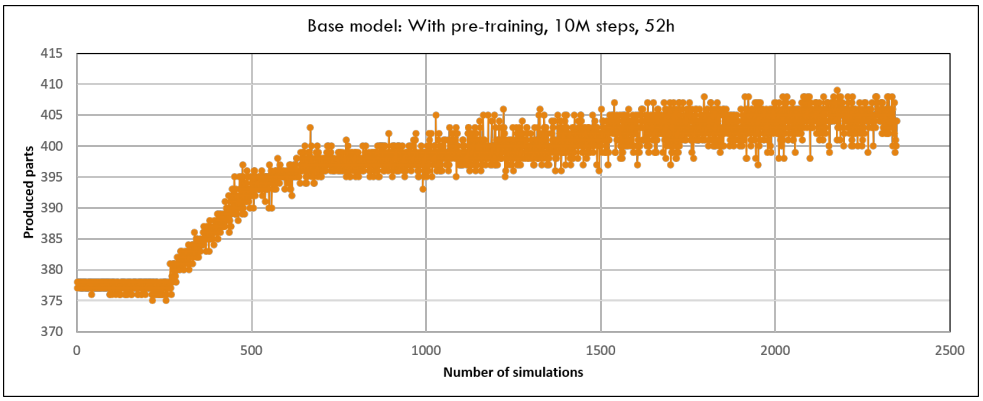

Figure 5.12: Productivity referring to the Base Model with pre-training, to a total of 10M time steps and a time horizon of 52h

Table 5.15: Numerical interpretations of the productivity (in parts) referring to the Base Model with pre-training, to a total of 10M time steps and a time horizon of 52h

Minimum Value	Maximum Value	Final Value
375	409	404

The following observation-action pairs follow the conjunctures performed previously, similarly with the results obtained in the model with pre-training, referring to the time horizon of 36 hours:

- **Observation-Action pair nº 1:** Parts in WS1 | Current WS: 4
  Observation: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
  Action: [9.9983370e-01, 1.7993931e-06, 4.0928912e-19, 1.6300578e-04, 1.4105979e-06, 4.8301262e-15, 1.4212710e-17, 2.4972540e-14]

- **Observation-Action pair nº 2:** Parts in WS3, WS4 | Current WS: 4
  Observation: [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
  Action: [8.7494940e-28, 5.5310151e-18, 3.6004825e-13, 1.0000000e+00, 3.2771040e-19, 4.0131407e-26, 4.2842155e-29, 7.9547190e-29]

5.5.5 Application of Penalties

After the penalties of -0.5 were applied whenever an empty movement is carried out, it was concluded that the four models follow the expected behaviour in relation to the PPO algorithm, as its curve presents. However, as Table 5.16 expresses, the results are inferior.

Since performing empty movements is not necessarily a bad practice, this is one of the reasons for which the productivity values are lower using penalties.

Table 5.16: Productivity (in parts) referring to the Base Model, with penalties

Base Model	Productivity (36h)	Productivity (52h)
Without pre-training	343	501
With pre-training	274	404
Without pre-training, with penalties	304	454
With pre-training, with penalties	271	403

5.6 Comparison of the FIFO, Optimized Milk-Run, NearestWS Rule and Base Models

Initially, comparing all the models previously studied in Sections 5.1, 5.2, 5.3 and 5.5, with regard to productivity, the Base Model was the one that presented the best results. That said, the use of Machine Learning techniques combined with simulation approaches has enabled us to achieve good solutions.

Regarding the Base Model, the difference between the usage or not of pre-training has to do with the fact that, at the beginning of the learning process, the resulting solutions are already or not close to a local optimum. If pre-training occurs, the solutions derived from the learning process will be in close to a local optimum, unlike what happens in the situation where there is no pre-training.

Since the solutions start near a local optimum, the Agent has difficulties to explore and find new better solutions than the ones settled outside this region of the solution space.

In this sense, it was found that the final productivity results obtained were higher in the case where the pre-training was not carried out (Table 5.17). This is explained by the fact that, in the absence of pre-training, the occurrence of transports to distant stations to the detriment of closer stations allows the reduction of bottlenecks and the balancing of the machine occupancy rates. The fact that there is no pre-training also allows the Agent not to be so easily stagnated in a local optimum given by the pre-training, so there is more freedom to explore new solutions, which will allow the Agent to achieve better solutions, contrarily to what happens in the model where there is pre-training.

Table 5.17: Productivity (in parts) referring to the models in study

Model	Productivity (36h)	Productivity (52h)
FIFO	177	262
Optimized Milk-Run	266	389
NearestWS Rule	258	377
Base, with pre-training	274	404
Base, without pre-training	343	501

5.7 Robustness of the Simulation Models

After analysing the models when applied to deterministic environments, it is pertinent to verify their behaviour when applied to other situations. In particular, it is important to study the influence of different production mixes and processing times, in order to examine the adaptability of the models, but only regarding productivity, which therefore excludes the model related to the study of makespan.

It is also important to mention that, according to the Base Model, there was no customized re-learning for these scenarios, because the weights obtained in the base scenario (original mix) were used.

5.7.1 Production Mixes

Five different mixes were defined, randomly generated, so that, as described in Section 4.8.1, the number of orders for each part type is balanced ($\simeq 20\%$).

The time horizons under study are 36 and 52 hours for comparison principles, and the AGV speed also remains constant at 1.5 m/s, since speed does not constitute a factor under consideration.

Tables 5.18 and 5.19 enable us to conclude that, for both temporal perspectives, the Agent when applied to different production mixes ensures that the productivity results remain, on average, close to each other. So, it is possible to validate the Base Model and its adaptability in different production contexts, just like the other models.

It should also be noted that the Base Model, without pre-training, remains the one that obtains the best results.

Table 5.18: Productivity (in parts) analysis referring to different production mixes, for a time horizon of 36 hours

Model	Initial Mix	Mix 1	Mix 2	Mix 3	Mix 4	Mix 5
FIFO	177	196	179	171	178	174
Optimized Milk-Run	266	261	251	257	261	242
NearestWS Rule	258	252	239	250	252	235
Base, with pre-training	274	273	259	266	278	252
Base, without pre-training	343	341	328	314	325	329

Table 5.19: Productivity (in parts) analysis referring to different production mixes, for a time horizon of 52 hours

Model	Initial Mix	Mix 1	Mix 2	Mix 3	Mix 4	Mix 5
FIFO	262	271	254	256	268	253
Optimized Milk-Run	389	385	359	379	386	370
NearestWS Rule	377	366	344	362	371	355
Base, with pre-training	404	409	376	388	394	382
Base, without pre-training	501	473	471	473	477	478

5.7.2 Processing Times

The second criterion to evaluate the adaptability of these models related to productivity includes the introduction of them in a stochastic environment in which, according to Section 4.8.2, the definition of processing times follows a triangular distribution.

The variability factor, α, applied to the processing times of each part was 0.4. It is also important for comparison purposes to maintain the time horizons in 36 and 52 hours and the AGV speed in 1.5 m/s.

The following Tables 5.20 and 5.21 provides the results of the 10 simulations generated and individually applied to each one of the implemented models. These results are important to analyse the average and standard deviations of the resulting productivity values.

The low standard deviation as well as the similarity between the average values of productivity obtained for each one of the models and the values associated with the original mix, also indicate that all the values are in agreement with each other. So, similarly to what happened with the production mixes, we can conclude that all models present good adaptability when subject to stochastic environments.

To conclude, the Base Model, without pre-training, is the one that allows the achievement of the higher productivity values.

Table 5.20: Productivity (in parts) in stochastic environments, for a time horizon of 36 hours

Index	Models				
	FIFO	Optimized Milk-Run	NearestWS Rule	Base with pre-training	Base without pre-training
Original Mix	177	266	258	274	343
1	174	266	257	273	340
2	176	273	259	273	343
3	176	269	261	273	341
4	172	270	257	276	346
5	180	271	259	276	341
6	179	266	252	270	338
7	174	269	256	270	340
8	177	267	260	269	340
9	178	269	256	277	345
10	177	269	260	273	341
Average (10 tests)	176.3	268.9	257.7	273.0	341.5
Standard deviation	2.452	2.183	2.669	2.749	2.461
Coef. of variation	1.391%	0.812%	1.036%	1.007%	0.721%

Table 5.21: Productivity (in parts) in stochastic environments, for a time horizon of 52 hours

Index	Models				
	FIFO	Optimized Milk-Run	NearestWS Rule	Base with pre-training	Base without pre-training
Original Mix	262	389	377	404	501
1	262	384	375	395	496
2	264	390	372	399	498
3	263	386	373	394	504
4	261	387	373	393	495
5	265	384	381	400	497
6	264	386	369	399	503
7	264	388	374	392	499
8	264	383	372	397	499
9	263	386	378	398	503
10	261	386	378	403	498
Average (10 tests)	263.1	386.0	374.5	397.0	499.2
Standard deviation	1.370	2.055	2.567	3.464	3.120
Coef. of variation	0.521%	0.532%	0.952%	0.873%	0.625%

Chapter 6

Conclusions and Future Work

This chapter presents the conclusions resulting from the dissertation project at issue and makes it possible to verify whether all the objectives initially proposed were fully accomplished. Finally, some future contributions of this project are also suggested.

6.1 Conclusions

The main purpose of the project was the application of hybrid simulation approaches with Machine Learning techniques in order to maximize the productivity of a factory characterized by a functional layout. To this end, it was considered as pre-determined aspects that the AGV speed would be 1.5 m/s and the time horizons in study would be 36 and 52 hours.

At the beginning, the construction of the factory layout was carried out in the simulation environment, using the Flexsim software, and, in order to study the productivity for a set of 2080 production orders, initial elementary decision rules to be applied to an AGV were implemented, in particular the FIFO and Milk-Run heuristics.

Subsequently, a new simulation model was implemented, characterized by an alternative decision rule, called NearestWS Rule, that allows minimizing the distances covered by the AGV. However, in this specific case, an external programming environment was used in order to define the workstations where the load would be carried out. Instead of what happened in the previous two models, this decision rule was not implemented in the simulation environment itself, which implied the implementation of a TCP communication protocol between the external program and the simulation environment. That said, we can prove that the RQ 1 was accomplished.

In a more advanced phase of the project, Machine Learning techniques were introduced in order to verify that the inclusion of a third entity, called Agent and equivalent to a neural network, provides benefits in the increase of the productivity values. For this, the PPO algorithm from *OpenAI Baselines* was used as the network learning model. For this purpose, in addition to the productivity analysis, the study of makespan was also carried out, corresponding to a part type number 4, in order to verify if the PPO algorithm was correctly implemented.

Relatively to this third entity, with regard to productivity, the Agent receives an observation, with the current status of each WS, and a reward, indicating whether a particular entity has entered at Sink or not. Subsequently, it sends the action that it considers to be the most appropriate and that contains the place where the AGV will carry out the load. It should also be noted that the neural network considered has 1 input layer and 1 output layer, composed, respectively, by 16 and 8 neurons. Finally, the Base model was the one which presented higher productivity values, in comparison to the FIFO, optimized Milk-Run and NearestWS Rule Models. This means that the introduction of this Agent has significantly increased the values of productivity, derived from the learning phase of the network.

Regarding the Makespan Model, it was proved that the normalization of the rewards is a fundamental factor to obtain better results since, without the normalization, the network learning would not have been so efficient.

Still referring to the Base Model, the influence of pre-training on final productivity results was studied and it was found that when pre-training is not carried out, the results are even higher, given that it is possible to better explore the space of solutions and reach a better solution than the one obtained with pre-training. Herewith, we are able to claim that the RQ 2 was answered.

Therefore, in order to verify and prove the adaptability and robustness of the models related to productivity, the original production mix was changed and, at a later stage, all the models were introduced in a stochastic environment, in relation to the processing times. In both situations, the Base Model, without pre-training, was always the one that showed the best results in terms of productivity and, answering the RQ 3, it is possible to state that both models studied have a certain level of robustness that allows them to be applied in a real context.

Finally, as a conclusion, it should be noted that all the initial objectives proposed were fully respected and fulfilled, and all the results and conclusions resulting from the project may constitute an analysis and study tool for further future work, within the scope of transport systems in job-shop environments.

6.2 Future Work

In order to achieve better and innovative solutions within the scope of logistics systems, it is important to study and understand the influence that diverse parameters would have on productivity performance.

That said, it would be interesting to analyse the impact that the change in the AGV speed value would have on the system in general, in terms of the final achieved productivity values.

Another pertinent aspect would be to apply both decision rules developed in a different manufacturing context, with regard to the workstations layout, in order to verify their flexibility of adaptation. Regarding the model that uses the PPO algorithm, it would be curious to see the behaviour of the Agent when applied to different circumstances.

Finally, a third and no less important aspect, regarding the models that use the RL techniques, would be the analysis of how the addition of more information to the observations coming from

the simulation environment (such as number of parts in each input buffer) would affect the network learning process in terms of its effectiveness and efficiency. In other words, it would be interesting to verify if the final solution would tend to even higher values of productivity.

References

[1] C. d. E. e. E. Da, C. Mendes, C. R. Da Silva, D. d. F. C. Costa, J. S. Sousa, M. L. S. Monteiro, and N. R. B. De Azevedo, "Jidoka: Pilar de sustentação do sistema toyota de produção nas organizações," 2013.

[2] J. K. Liker, *O modelo Toyota: 14 princípios de gestão do maior fabricante do mundo*. Bookman Editora, 2016.

[3] L. A. Risso *et al.*, "Procedimentos sistemáticos para projeto de" layout" para ambientes" job shop"= systematic procedures for layout design for job shop environments," 2016.

[4] D. M. B. . B. M. Kilic, H. S., "Classification and modeling for in-plant milk-run distribution systems," 2012.

[5] T. da Silva Santos, "Otimização e simulação de sistemas de logística interna-caso real de definição de rotas milk run numa empresa de semicondutores," 2018.

[6] A. M. Andrew, "Reinforcement learning: An introduction by richard s. sutton and andrew g. barto, adaptive computation and machine learning series, mit press (bradford book), cambridge, mass., 1998, xviii 322 pp, isbn 0-262-19398-1.," *Robotica*, vol. 17, no. 2, p. 229–235, 1999.

[7] P. Cortez and J. Neves, "Redes neuronais artificiais," 2000.

[8] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[9] J. Faria, "Continuous improvement-kaizen." Faria, José, Continuous Improvement-Kaizen, 2019. 80 slides. Support material for the SQFI course of MIEEC of FEUP. Accessed in 2019-11-19.

[10] J. Faria, "Costumer focus." Faria, José, Costumer Focus, 2019. 86 slides. Support material for the SQFI course of MIEEC of FEUP. Accessed in 2019-11-19.

[11] A. Azevedo, "Organização dos sistemas de transformação." Azevedo, Américo, Organização dos Sistemas de Transformação, 2018. 18 slides. Support material for the GOPE course of MIEEC of FEUP. Accessed in 2019-11-19.

[12] M. Bonneton and D. Janvier, "Automated selfpowered materal handling truck," 1986.

[13] M. Baudin, "Lean logistics: The nuts and bolts of delivering materials and goods," Productivity Press, 2005.

[14] R. Drießel and L. Mönch, "An integrated scheduling and material-handling approach for complex job shops: a computational study," *International Journal of Production Research*, vol. 50, no. 20, pp. 5966–5985, 2012.

[15] R. M. Logística, "Milk run: Conceitos, vantagens e ganhos para a operação logística." https://revistamundologistica.com.br/blog/achiles/milk-run-conceitos-vantagens-e-ganhos-para-a-operacao-logistica. Accessed in 2019-11-23.

[16] R. F. D. Santos, "Estratégias híbridas de machine learning e simulação para a resolução de problemas de escalonamento," 2018.

[17] S. Huang, "Optimization of job shop scheduling with material handling by automated guided vehicle," 2018.

[18] A. de Freitas Ribeiro, "Taylorismo, fordismo e toyotismo," *Lutas Sociais*, vol. 19, no. 35, pp. 65–79, 2015.

[19] A. Azevedo, "Lean thinking: princípios, fundamentos, principais ferramentas." Azevedo, Américo, Lean thinking: princípios, fundamentos, principais ferramentas. Sistema de Produção Toyota Parte I, 2013. 28 slides. Support material for the GOPE course of MIEEC of FEUP. Accessed in 2020-01-23.

[20] G. Zäpfel, R. Braune, and M. Bögl, *Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics*. Springer Berlin Heidelberg, 2010.

[21] A. Meyer, "Milk run design (definitions, concepts and solution approaches). dissertation, karlsruher institut für technologie (kit) fakultät für maschinenbau, kit scientific publishing, karlsruher (2015)," 2015.

[22] . S. G. Brar, G. S., "Milk run logistics: Literature review and directions, london, england," 2011.

[23] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[24] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[25] S. Marsland, *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2014.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[27] V. H. de Sousa Carneiro, "Abordagens híbridas de" machine learning"/simulação para sistemas logísticos dinâmicos," 2019.

[28] I. Grigoryev, *Anylogic in three days*. Fifth Edit.C, 2018.

[29] J. Banks, J. S. Carson II, L. Barry, *et al.*, "Discrete-event system simulationfourth edition," 2005.

[30] A. K. d. Silva, *Método para avaliação e seleção de softwares de simulação de eventos discretos aplicados à análise de sistemas logísticos.* PhD thesis, Universidade de São Paulo, 2006.

[31] V. H. S. Carneiro, *Hybrid Machine Learning/Simulation Approaches for Decentralized Production Scheduling*. 2018.

[32] "Socket programming in python (guide) by nathan jennings." Accessed in 2020-03-02.

[33] M. Almeida, Luís; Santos Pedro; Sousa, "Support information for lab assignment 1a tcp sockets in c." Support information for lab assignment 1a TCP sockets in C, Luís Almeida, Pedro Santos, Mario Sousa, 18 slides. Support material for the ACI course of MIEEC of FEUP. Accessed in 2020-04-15.

[34] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines." https://github.com/hill-a/stable-baselines, 2018.