POLITECNICO DI TORINO

Collegio di Ingegneria Gestionale e della Produzione
Corso di Laurea Magistrale in Management Engineering

Simulation based method to compare AGV and AMR systems in layouts with increasing level of complexity

Supervisor
Giulia Bruno
Co-supervisor
Antonio Carlin

Candidate
Stephanie Suarez Evangelio

A.A. 2025/2026

Index

Abstract

In recent years, the adoption of Industry 4.0 principles within intralogistics has grown significantly due to the increasing need of companies for flexible and smart factories. This shift is particularly evident in material handling, where automation has become the most common trend. Specifically, Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) represent the most important examples of automated technologies and have been widely analysed. However, there is a gap in the current literature on how the topological complexity of a factory layout influences the performances of an AGV and AMR system. Given this context, the thesis aims to bridge the gap by proposing a simulation based method to support companies in the selection of the most ideal automated material handling systems, between AGV and AMR, by taking in considerations the impact of the layout complexity on their performance. The proposed method is composed of the following phases: definition of 3 layout configurations with increasing layout complexity (low, medium, high); assessment of the layout complexity through the calculation of the Layout complexity Index (LCI); implementation of the simulation models within FlexSim environment; and, lastly, performing a comparative analysis based on a series of Key Performance Indicators (KPIs).

Chapter 1

Introduction

In recent years, the adoption of Industry 4.0 principles within intralogistics has grown significantly, driven by the increasing need of companies for flexible and smart factories. This shift is particularly evident in material handling, where automation has become the most common trend. Within this context, Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) represent the most important examples of automated technologies for the transportation of goods. These systems have been widely implemented and analyzed, but choosing between a traditional AGV system, which travels on predefined paths, and an AMR system, which instead navigates autonomously and dynamically, remains a complex challenge for companies. Often this decision relies on standard parameters, ignoring an important aspect: the topological complexity of a factory layout. In the current literature there is a significant gap regarding how this complexity influences the operational performances of an AGV and AMR system.

Given this context, the main goal of this thesis is to bridge this gap by proposing a simulation based method to support companies in selecting the most ideal automated material handling system. Specifically, the objective is to compare the performances of AGV and AMR systems across layouts characterized by increasing levels of complexity (low, medium, and high). This comparative analysis aims to understand which technology is better suited for each specific case based on the physical characteristics of the layout.

To ensure a complete evaluation, the comparative analysis is structured into three main levels. First, the two systems are tested in both static and dynamic environments to determine which navigation logic is more flexible when temporary obstacles are introduced, and how this impacts performance as the layout complexity increases. Second, the scenarios are tested by increasing the number of vehicles to observe how the fleet size affects the overall efficiency and highlights the differences between AGVs and AMRs. Finally, the systems are evaluated in deterministic versus variable scenarios to recreate more realistic conditions by introducing variability. By measuring KPIs (system output, vehicle utilization, and vehicle states), across these three levels, this work evaluates how spatial constraints, operational variability, and dynamic obstacles impact the navigation logic and overall efficiency of both technologies.

The thesis is divided into five chapters. The first chapter provides the theoretical background necessary to understand the topics discussed. It explores production intralogistics, material handling

automation, and the hardware, software, and control architectures of AGV and AMR technologies. At the end, it introduces the concepts of simulation and the FlexSim software used in this study.

The second chapter presents the state of the art of the current literature. It analyzes previous studies regarding the use of simulation to compare mobile robots, the impact of layout design, dynamic obstacles, and models to compute the complexity layout index.

The third chapter describes the proposed simulation based method, starting with the Layout Complexity Assessment (LCA) model used to calculate the complexity indices. Then it explains the simulation modeling strategy developed in FlexSim for both AGV and AMR systems, and the experimental design defined for the comparative analysis.

The fourth chapter illustrates the case study application, which is based on an electric bicycle assembly plant. It defines the three layout configurations, their implementation in the FlexSim environment, and the logic for the generation of obstacles. The chapter concludes by presenting the results obtained from both the static and dynamic simulation scenarios.

Finally, the thesis concludes by summarizing the final conclusions of the research, highlighting which system is more suitable depending on layout complexity and operating conditions, and suggests potential directions for future work.

Chapter 2

Theoretical Background

2.1 Production intralogistics

The concept of intralogistics is a relatively recent addition to industrial literature. The term combines the prefix "*intra*", which means "*within*", with the broader definition of logistics. While general logistics aims at ensuring the availability of the right goods in the correct quantity, quality, and cost at the right time and place, intralogistics specifically narrows this scope to those processes occurring within the company's boundaries.

Before 2004, the industry relied on fragmented terminology such as "materials handling" or "warehouse technology" to describe these operations. A unified terminology was introduced for the first time in 2005 at the CeMAT (*Centrum für Materialfluss und Fördertechnik*) exhibition in Germany by the International Network of Machinery Manufacturers [1]. However, the term was officially coined and industrialized by the German Engineering Federetion called VDMA (*Verband Deutscher Maschinen- und Anlagenbau*). The introduction of the term was revolutionary within the industry, as it established a clear distinction between internal flow management and external logistics. This formulation was further solidified in 2008 when Michael ten Hompel, a leading professor and director of the Fraunhofer IML, and Volker Heidenblut provided the scientific foundation and definition of intralogistics, that remains the industry standard today. According to them, intralogistics "*encompasses the organization, management, control and optimization of the internal flow of products from inbound to outbound goods, linking the flow of information and services in order to increase the value of the basic offering in industry, commerce and public facilities*"[1].

Given this definition, intralogistics can be described as the set of activities and processes performed within a company to manage the flow of goods and information from the moment of receipt up to the final preparation of the deliveries to the end customers. For this reason, it has become a vital component of any modern supply chain. Its importance comes from the ability to transform internal operations into a competitive advantage by reducing operational costs, improving resource productivity, and optimizing inventory management. In addition, the benefits of intralogistics also reach the end users through faster and better service, leading to greater satisfaction and brand loyalty.

To better understand how intralogistics operates, it is necessary to distinguish between the two main flows that characterize it: material flow and information flow. Their synchronization allows a modern facility to function effectively.

The **material flow** represents the tanglible aspect of intralogistics. It includes the physical movement and handling of goods within a facility, from the purchase of raw materials to the final preparation of finished products for dispatch. By optimizing it, a company can significantly eliminate bottlenecks and reduce non-value-added activities.

This process is composed of several key operational stages, for example:

- **Transport**. It is the movement of material between different functional areas, such as from the warehouse to the production line.
- **Storage and buffering**. It refers to the organized placement of goods in storage or temporary buffer zones to ensure inventory availability for production.
- **Handling**. It involves the physical manipulation of items, including loading, unloading, and the connection point with various transport systems.
- **Dispatching**. It is the final stage where the products are prepared and moved to the shipping area.

The physical movement of goods cannot be executed effectively without a synchronized system to coordinate every operation. It is for this reason that the **information flow** comes into play. Serving as the brain that directs physical actions, it ensures the systematic transfer of data and instructions that guide the materials through every stage of the process. In the context of intralogistics, managing information involves designing processes that facilitate the real-time exchange of data between humans, machines, and management software. This flow is typically structured through a hierarchy of systems, where each is responsible for a different level of the operation. Some examples of software are the following:

- **Enterprise Resource Planning** (ERP). It operates at the highest level, managing high-level data like customer orders, procurement, and overall production planning.
- **Warehouse Management System** (WMS). It is considered the core system for intralogistics since it helps a company to manage and control daily warehouse operations, optimizing storage locations and picking sequences.
- **Manufacturing Execution System** (MES). It is a software designed to manage and monitor manufacturing processes on the shop floor. It bridges the gap between ERP's high-level planning and the actual production operations, ensuring that the right materials are delivered to the right machines.
- **Material Handling Planning** (MHP). It is a computer-based system used to calculate the components needed based on the production schedule of finished products. By analyzing sales forecasts, open orders, and current stock, MHP determines exactly what to order and when.
- **Fleet management**. It is the operational layer responsible for assigning tasks to Automated Guided Vehicles (AGV) or Autonomous Mobile Robots (AMR). It manages their traffic, prevents collisions, and monitors their status and performance in real-time.

The integration of an efficient information flow offers significant strategic advantages by ensuring real-time visibility across the entire facility. This transparency allows for the precise tracking of goods and vehicles, which minimizes human error and increases operational accuracy. Finally, it allows to react instantly to production changes or bottlenecks.

Given the importance of the relationship between the two flows, intralogistics has always been the subject of research and development to find innovative solutions to improve and optimize their integration and management. In this context, the current trend involves the adoption of Industry 4.0 principles [2], since there is an increasing need to transform facilities into smart and flexible factories capable of adapting rapidly to fluctuating and unpredictable volumes, without requiring drastic changes to the facility's physical layout. For this reason, intralogistics has been considered as the ideal starting point for introducing the Industry 4.0 model within a factory [3].

Industry 4.0 is a term used to refer to the fourth industrial revolution that is currently reshaping the manufacturing landscape. It represents a new model of organizing the entire value chain of a product's life cycle. Specifically, the fulfillment of the individual customer requirements has become a central priority due to their great impact on every phase from order management, R&D, manufacturing commissioning, delivery, up to the recycling of products. Industry 4.0 aims, also, at creating open, and smart factory, where regular machines are transformed into self-aware and self-learning systems capable of real-time data monitoring and product tracking [3].

The paradigm of the fourth revolution can be described through its **four main drivers**: Internet of Things (IoT), Industrial Internet of Things (IIoT), Cloud-Based Manufacturing, and Smart Manufacturing. These drivers can be translated into practical applications through a series of core technologies, widely known in the literature as the **nine pillars** of Industry 4.0 [2]:

1. Big data and analytics.
2. Autonomous robots.
3. Simulation.
4. System integration.
5. Industrial Internet of Things (IIoT).
6. Cyber security.
7. Cloud.
8. Additive manufacturing.
9. Augmented reality.

The application of these drivers and pillars in intralogisitcs has led to two main trends: the digitalization and automation of equipment and operations. **Digitalization** in logistics involves connecting four aspects to the digital system: people, logistical objects, logistical process, and logistics facilities [4]. This connection creates transparency, allowing companies to monitor operations in real-time and make decisions based on actual data rather than assumptions.

Automation, on the other hand, focuses on the physical handling of materials. It aims at improving the equipment by using smart technologies, such as autonomous robots, to perform

tasks without the human intervention, resulting in a faster material flow management and reducing the risks of errors.

2.2 Mobile robots: Automation of material handling operations

As discussed in the previous chapter, the adoption of Industry 4.0 principles in intralogistics is essential for creating flexible and more efficient factories, and one of the current trends driving this change is automation. This thesis will focus on mobile robots, specifically Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMR), which are well-known technologies used to automate the material handling of goods.

2.2.1 Material handling

Material handling is the field concerned with solving problems related to the movement, storage, control and protection of materials throughout their entire value chain. In other words, material handling is the area of intralogistics that ensures the availability of the right amount of the right material at the right place at the right time for the right cost [5]. This has a significant impact on the end users level of satisfaction, since, despite the lack of direct interaction, the way raw materials and semi-products are managed determines the availability of finished products to consumers [6].

The main objectives of material handling are the following [6]:

- Reduce the unit cost of production.
- Reduce manufacturing cycle time.
- Promote safety and improve working conditions.
- Maintain or improve product quality.
- Promote productivity by optimizing material paths.
- Develop a preventive maintenance program.
- Control inventory.

Among these, the most important is the **reduction of production cost**. The cost of material handling contributes significantly to the total cost of manufacturing, ranging from 30% up to 70% [6]. Therefore, properly designing and integrating the material handling system became a crucial task, since it provides high cost savings and service improvements. In fact, poor material management can create operational issues from losses due to damages to, excessive movement, and shortage of supplies. To avoid these issues and achieve the mentioned objectives, material handling is composed of **four key operational activities** [6]:

1. Predicting the material movement.
2. Determining the material sources.

3. Controlling the delivery of the materials into the facility.

4. Monitoring the state of the materials.

By effectively managing these activities while strictly optimizing time, quantity, and space, an organization can secure a high profitability level and a favorable competitive position in the market.

To achieve the mentioned goals, organizations rely on three types of material handling systems, which are summarized in the Tab.1. This table shows a classification proposed by Professor Goetschalckx [7], who has categorized material handling based on 2 variables: the source of labor and the source of control. Besides this classification, Goetschalckx [7] provides a further comparison of the three levels based on their operational characteristics, as presented in the Tab. 2.

	Capability	
	Labor	Control
Type		
Manual	Human	Human
Mechanized	Machine	Human
Automated	Machine	Machine

Table 1: Labor and Control Providers for Material Handling Methods from [7]

Characteristic	**Type**		
	Manual	Mechanized	Automated
Weight	Low	High	High
Volume	Low	High	High
Speed	Low	Medium	High
Frequency	Low	Medium	High
Capacity	Low	Medium	High
Flexibility	High	Medium	Low
Acquisition cost	Low	Medium	High
Operating cost	High	Medium	Low

Table 2: Material Handling Methods Characteristics from [7]

On the first and second levels, there are manual material handling and mechanized material handling. **Manual material handling** refers to the physical processes involved in moving, lifting, and transporting materials by hand, while the **mechanized system** includes the use of devices such as cranes, conveyors, forklifts, and lifters to handle heavier loads more rapidly, where the human worker continues to have control over navigation and decision-making. According to the Tab. 2, the manual system is considered to offer the highest level of flexibility, as human workers can immediately adapt in case of unexpected changes without the need for replanning. However, their performance is restricted in terms of, for example, throughput and speed, due to human physical limitations. In fact, this level is highly prone to operator fatigue and ergonomic risks caused by awkward posture, repetitive motions, and heavy lifting. On the other hand, the mechanized approach significantly increases speed, capacity, and workspace safety, but it remains dependent on

the operator's skills and availability. Therefore, it is still sensitive to human error and subject to variable performance.

Lastly, the third level is **automated material handling**, which involves the use of automated systems and equipment to transport, sort, organize, and store materials. This category aligns most closely with the Industry 4.0 principles, as the need for human intervention is eliminated. In fact, the equipment manages both the physical transport and the decision-making. Based on the Tab. 2, these systems achieve better levels of performance compared to the previous two categories, for example, in terms of speed, capacity, and flexibility. Besides these performance metrics, automated material handling offers relevant operational benefits. First, it leads to a significant reduction in operational costs, since it requires fewer manual interventions, lower dependency on labor, and associated costs. Additionally, the precision and accuracy of these systems reduce waste and damage to materials. Second, it ensures a safer workspace by reducing risks of injuries related to manual handling, fatigue, and working at heights. However, they require greater capital investment, making the implementation decision of these systems a critical strategic choice.

The evolution of automated material handling has been further accelerated by the introduction of sophisticated technologies, such as robotics, Artificial Intelligence (AI), and IoT. In this context, robotics provides the "muscle" power, performing tasks ranging from lifting heavy loads to precision picking and placing, all with absolute consistency. Software and AI, on the other hand, bring intelligence to these operations, enabling systems to make data-driven decisions, predict maintenance needs, and optimize workflows in real-time. Finally, IoT connectivity ensures seamless communication between machines and management systems, offering greater visibility into every aspect of the material handling process.

Automated material handling systems come in various forms, each designed to optimize specific aspects of warehouse and manufacturing operations. For example, Automated Storage and Retrieval Systems (AS/RS), which maximize vertical space utilization and retrieval efficiency; and robotic picking systems, which are equipped with vision systems to execute tasks with high precision. Among all the technologies available, this thesis will focus on mobile robots, which play an important role in the transportation of materials.

2.2.2 Mobile robots

Mobile robots are machines equipped with sensors and navigation systems that allow them to navigate freely and move around their environment to perform various tasks without the need for constant human intervention. These systems have emerged as a strategic solution to the current market trends, in particular when it comes to the significant rise in personalized products, which has introduced considerable challenges in terms of quality, cost, and delivery for manufacturing enterprises [8].

They can be classified into two categories: **guided** and **autonomous mobile robots**. The first category includes mobile robots that require the presence of a guidance system to follow predefined paths within a controlled environment. These paths are defined by magnetic tape, bar codes, wires, or sensors installed on the facility's floor. A perfect example of this type of technology is the **Automated Guided Vehicles** (AGVs), which are widely used for automated material handling across various industries. Because of the presence of fixed infrastructure, AGVs

ensure high predictability in movement and consistent cycle times, which is ideal for repetitive and standardized workflows. But, at the same time, this rigidity represents a significant limitation as changing a route requires physical modification to the facility's layout, which costs time and money. Another issue of AGVs is their lack of obstacle avoidance capabilities, which means that if a path is blocked, the vehicle stops and waits until the obstacle is removed, leading to potential bottlenecks.

On the other hand, **Autonomous Mobile Robots** (AMR) represent a more advanced solution for the automation of material handling, since they are capable of independent decision-making. Unlike AGVs, AMRs can perceive and map their environment, and based on this information, they can navigate dynamically without relying on any fixed infrastructure. Their route is calculated in real-time, optimizing the travel distance to ensure the shortest and most efficient path to the destination. For this reason, an AMR is more flexible compared to an AGV since it can dynamically reroute its path in case of an obstacle without modifying the facility's layout while maintaining high productivity. But a fleet of AMRs struggles in case of extremely crowded or highly dynamic environments since they have to keep adjusting their paths to avoid obstacles.

2.2.2.1 Automated Guided Vehicles

AGV systems emerged in the 1950s, and they were initially used only in assembly line operations. However, in recent years, they have become well-known and have been widely introduced across various industries, such as manufacturing, warehousing, medicine, and logistics. This widespread adoption was possible because of their high degree of adaptability to any kind of operational environment [9].

As previously mentioned, AGVs are an example of guided mobile robots. They are programmed to carry out tasks and navigate predefined paths, within indoor or outdoor workspaces, to reach their destination, without human intervention, but with the support of guidance systems, sensors, and control systems. These vehicles are capable of transporting a wide range of items in terms of load type, weight capacity, or based on the complexity of the required logistic tasks [8]. Depending on their application, AGVs can be classified into different types, which are summarized in the Tab. 3 [8]. However, at the executive level, the core tasks are quite similar and can be performed using the same tools. The main task consists of automating the transportation of goods between two points of the operational environment: the vehicles secure the load at the pick-up station, transport it, and deposit it at the drop-off point. Furthermore, the execution of the tasks is always event-driven, and the trigger can originate from various sources, for example, from a sensor detecting a new load to transport, from a user requesting the transportation service via a GUI interface, or a command from an ERP system. [10].

Although the implementation of AGV systems requires a significant initial fixed capital investment, they have led to a series of **benefits** across economic, environmental, and social sustainability dimensions [8, 11, 12, 13]:

1. Increase productivity.
2. Labor cost saving. Despite the high initial capital investment, AGVs offer a greater economic potential due to their lower maintenance expenditure compared to traditional vehicles, and their capability to operate on a 24/7 basis with minimum labor cost and human intervention.

Additional savings derive from the associated improved safety and the consequent reduction in accidents involving both vehicle drivers and pedestrian workers.

3. Reduction of pollution and energy consumption given by the optimization of the routes.
4. Enhanced safety. The risks of collision are minimized since all vehicles are equipped with lights and horns to warn workers of their presence.
5. Optimization of space utilization. Unlike conventional fixed equipment, AGVs occupy workspace only temporarily while working and parking. This allows them to share space with other machinery and workers.
6. Improvement in the tracking of goods and reduction of their rate of loss.

In practical industrial applications, AGVs are rarely deployed as standalone units, but instead, they operate as a coordinated fleet. However, managing and coordinating multiple vehicles simultaneously introduces a series of complications compared to single-vehicle operations, like the avoidance of collisions and deadlocks, the optimization of global performance, and task allocation. Therefore, an AGV system has to rely on a robust control architecture capable of managing the fleet's core operations efficiently and effectively.

2.2.2.1.1 Fleet management: control architecture and core tasks

In a fleet of AGVs, the control architecture can be organized in two ways, as it is shown in Fig. 2.1 [9]: centralized and decentralized systems. The **centralized architecture** is deeply rooted within the industry and is typically characterized by a simple system architecture. It relies on a central server that has complete access to and visibility of all the information regarding the entire fleet and the facility environment, which allows to calculate a global optimum for scheduling and routing, ensuring the most efficient theoretical performance. However, this configuration is generally suited for small-scale systems, where there is a low degree of dynamic variability, since it is not robust against central failures.

On the other hand, **decentralized architectures** is defined as "*the distribution of the total intelligence of a system to its components*" [9], which means that each AGV acts as an independent and intelligent entity, that has access to local information rather than the full global picture, but it can communicate with other vehicles to retrieve more data. Decentralization is hardly implemented in the industry, but it is gaining a lot of attention over the last years since it guarantees higher flexibility, robustness, and scalability, which are necessary in order to manage larger and more complex systems. For these reasons, decentralization is the most compatible strategy to address the modern requirements, and is considered the way to achieve the so-called "*Factory of the Future*", which is used to denote a factory reflecting the Industry 4.0 principles. In fact, with a decentralized system, AGVs become smarter and capable of gathering information to make dynamic decisions. In addition, in this context, AGV can be used more for ad-hoc solutions, besides transport. However, there are also limitations with this architecture. The main drawback is the increased effort needed to coordinate the numerous independent entities, as each of them tries to reach its goals. This will not necessarily lead to a global optimum of the overall system. In fact, the implementation of a decentralized system always implies a trade-off between optimality and flexibility.

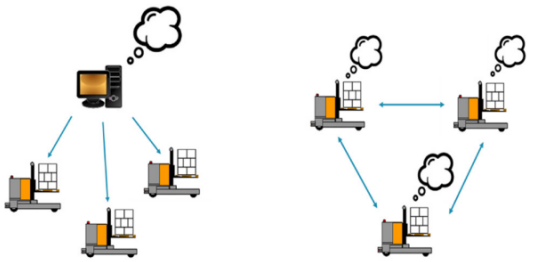


Figure 2.1: AGV central and decentral control architectures [9]

Regardless of the chosen control architecture, the operational efficiency of a fleet depends on the successful execution of a series of core tasks. De Ryck et al. [9] have decomposed the AGV control into 5 distinct core tasks, illustrated in the Fig. 2.2.

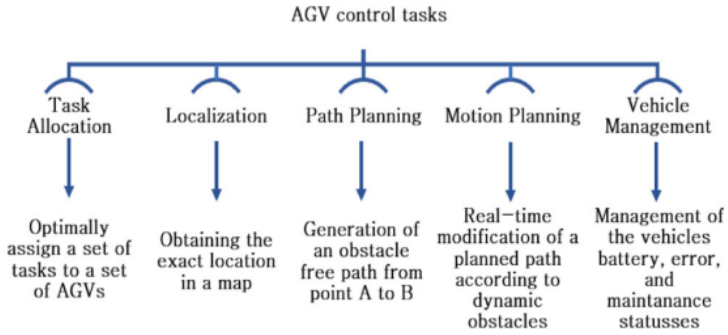


Figure 2.2: AGV core tasks [9]

1. Task allocation

Task allocation, also called task scheduling or dispatching, is one of the most challenging AGV tasks. It consists of distributing a set of tasks (or orders) to a fleet of AGVs, so that each vehicle is dispatched according to a schedule, as illustrated in the Fig. 2.3. An example of a task could be an item that needs to be loaded at a certain location and then dropped off at another point of the factory.

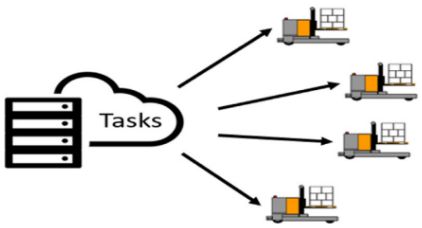


Figure 2.3: AGV task allocation [9]

Task allocation is considered a constrained optimization NP-hard problem (Non-deterministic Polynomial-time hard), where if the number of tasks and robots increases significantly, the solution space becomes too large, and, in this context, there does not exist an efficient algorithm capable of producing an exact solution to the problem in a finite amount of time. The easiest way to solve the allocation of tasks is to identify the closest AGV to the position of the ordered item.

The most important properties that characterize this core task are the following:

1. **Divisibility**. The total amount of work must be divided efficiently to have an optimized utilization of all the resources.
2. **Fault tolerance**. Task allocation should be able to operate correctly even in case of failures.
3. **Scalability**. Task allocation should be able to keep working without computation limits in case of an enlargement of the fleet.
4. **Flexibility**. The algorithm should be, in any circumstance, capable of adapting easily and rapidly to any changes in the system.
5. **Responsiveness**. Task allocation must have high performance also in highly dynamic environment.

Another aspect needed to be taken into consideration is **task constraints**. In the simplest case, tasks are independent of each other, and, therefore, the only relevant information that the vehicle has to know is the starting and ending points. But in real-world application there are several other constraints to consider. The first one is **temporal constraints** like the task duration, minimum starting time, and maximum ending time. Second, there could be a situation where tasks are dependent on each other, for example, some tasks must be completed before or after another one, or two tasks must be executed at the same time. These are called **precedence constraints**. Some other constraints can be related to **mobility interferences**, for instance, in the case of a narrow aisle, only one robot can pass on the way to the task. The last type of constraints is **resource constraints**, which can prevent a task from being executed when resources are empty, for example, when the AGV has a low battery and needs to be recharged, it can not be used to complete a task.

In the last decade, there has been a lot of research done to solve the problem of task allocation. In the literature, it is possible to identify two distinct ways to model the solution algorithms: optimization-based and market-based solutions. Regardless of the method used, the assignment of tasks focuses on optimizing the following variables:

1. The cost that it takes for a vehicle to execute a task. It could be time, distance, or fuel consumption.
2. Fitness. It refers to how well a robot can perform an assignment.
3. The reward that an AGV gains when completing a task.
4. Priority. The urgency of completing a task.
5. Utility, which is calculated by subtracting the cost from the reward or fitness.

The **optimization-based algorithms** search for an optimal solution that maximizes or minimizes a variable using global information and considering all the mentioned constraints. There are three types of algorithms: **exact algorithms, heuristics, and meta-heuristics**. The first approach is used when the solution space is small and, therefore, it is possible to find an exact solution. The other two are used to solve the NP-hard problem. Heuristics typically operate as iterative algorithms: they explore the search step-by-step, generating potential solutions and comparing them to the best one found so far, allowing for a progressive improvement of the result without examining every single possibility. But using this technique can lead to getting stuck in a local optimum. Meta-heuristics overcome this limitation by implementing strategies that broaden the search scope, aiming to find the global optimum rather than settling for the first good solution found.

On the other hand, **market-based solution algorithms**, instead of using a central server that processes all the information to find the optimal allocation, they are based on economic principles, specifically the concept of auction. The core idea is that the AGVs act like independent traders. They calculate their offer based only on their local information, while being unaware of the decisions made by the other vehicles. The auctioneer, in a decentralized system, is one of the AGV, evaluates all the bids, including its own, and assigns the task to the robot with the highest bid.

2. Localization

Localization allows the AGV to determine its current position in the environment, which is an important information to properly navigate. In fact, it is used by the other core tasks, like path planning to identify the shortest path to reach the desired destination, and motion planning to detect and avoid unforeseen obstacles blocking the path. In other words, localization guarantees precise movement in the environment and facilitates the implementation of obstacle avoidance tasks and traffic management. In contrast to other tasks, localization is already decentralized since all its equipment and software are onboard the vehicle; therefore, its execution is independent of the type of control architecture.

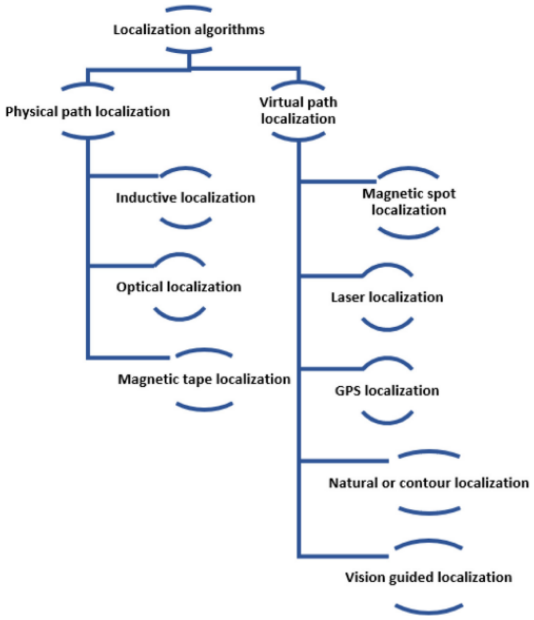


Figure 2.4: AGV types of localization methods [9]

Localization methods can be divided into two groups: **physical path localization** and **virtual path localization methods**, as shown in the Fig 2.4. With the first approach, the presence of physical guidelines on the floor is required. The AGV localizes itself using a sensor that detects these guidelines, which are typically defined by inductive wires, magnetic tapes, or optical lines embedded in the floor. Beyond navigation, these markers can also trigger specific vehicle actions, such as adjusting speed or modifying the steering angle during turns. In the Fig. 2.4 are illustrated the common traditional techniques based on this approach: **inductive, optical, and magnetic tape localization**. These methods are industry standards due to their simplicity and precision, particularly in narrow aisles. However, they are inherently rigid, as modifying the route requires physical alteration of the facility. Additionally, while surface-based methods, like optical and magnetic tape, reduce installation costs, they are prone to reliability issues if the guideline becomes dirty or damaged.

On the other hand, with virtual path localization methods, the predefined circuit can be maintained virtually, either within the AGV's local map or on a global map hosted by a central unit. Given the requirements of modern industry, fixed physical paths are often inefficient due to their lack of adaptability in case of frequent configuration changes. Virtual localization overcomes this limitation and offers superior flexibility, as routes can be easily modified online in the graphical design software without the need for physical alterations to the facility. However, this approach is more difficult to implement. Unlike physical localization, where the vehicle follows a linear trajectory, a virtual localization method requires the AGV to know its exact coordinates within a two-dimensional space.

Currently, the most accepted method for AGV virtual localization is **laser localization**, where the path is defined by reflective strips located in the operating area on known coordinates. The detection of these reference points is performed by a rotating laser mounted onto the vehicle, which

scans the environment by emitting a laser beam. The AGV can calculate its current position and update the global map if it can scan at least three reflective landmarks. This method is very accurate, secure, and reliable, but it is characterized by a high cost and effort due to the placement of all the reflectors in the factory area. Another well-known method used in the industry is **GPS localization**, which uses trilateration with satellites to determine the AGV position. However, this method requires a clear line of sight to the sky, making it impractical in an indoor industrial environment. An indoor alternative to overcome this problem is using **Local Positioning Radar** (LPR), which replicates the GPS principle within the factory, but its standard precision, typically around 10 cm, is a significant limitation when it comes to performing highly accurate AGV applications. This issue can be improved by using sensor fusion.

On the other hand, to overcome the need for installed infrastructure like reflectors, **natural localization** has been used as a flexible alternative. The AGV uses a Light Detection and Ranging (LiDAR) sensor to scan the environment and identify the existing structural features, like walls and pillars. Through a Simultaneous Localization and Mapping (SLAM), the robot creates a local two-dimensional map of its surroundings and compares it dynamically to the factory map to locate its position. This approach increases the flexibility of the AGV system in a dynamic environment, but the LiDAR sensors are very expensive, susceptible to errors caused by reflections, and they are not capable of detecting transparent materials. These limitations can be addressed by incorporating alternative sensors, such as sonar. Similarly, **vision-guided localization** operates without markers, using stereo cameras to generate three-dimensional point clouds that are then projected into a two-dimensional feature maps. The vehicle interprets its local environment using an Occupancy Grid System, where the area is divided into cells marked as either occupied or free. By continuously projecting observed features onto this grid, the vehicle builds and updates its map, often relying on an initial map created via SLAM. Its main disadvantage is its sensitivity to varying light conditions, which are common in industrial environments.

3. Path planning

Path planning consists of finding the shortest obstacle-free path to the desired destination while considering various constraints, such as vehicle capacity and the plant layout. This core task is typically categorized as the **static planning task** of an AGV since it computes a basic collision-free path using known information, like the location of walls and fixed machinery, while ignoring dynamic time-dependent elements. However, this static approach represents a significant approximation. In reality, industrial environments are not strictly controlled, but they can be dynamic and filled with unpredictable obstacles that a static planner cannot anticipate. This is where motion planning steps in. It modifies the vehicle's trajectory in real-time to deal with moving objects, ensuring that the AGV avoids situations of collisions and deadlocks.

Path planning is composed of two distinct steps, as illustrated in Fig. 2.5: **environmental representation** and **graph search algorithms**. The first step involves modeling the AGV's reachable states within the environment. This is typically achieved by defining a topological graph, which is a network of nodes and segments that corresponds to the predefined paths where the AGV has to move. The second step, instead, consists of using a graph search algorithm to identify the optimal route that connects the starting point and the goal. It aims to minimize a specific objective function, like travel time, distance, fuel consumption, or a combination of them.

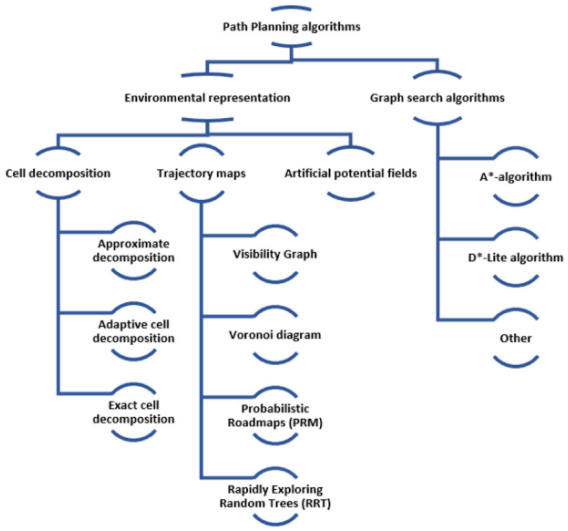


Figure 2.5: AGV path planning algorithms [9]

Currently, most of the AGV systems rely on a **centralized architecture** where the environment is static and graph-based. In this configuration, the search algorithm computes the shortest path strictly constrained to the fixed network. However, an emerging trend focuses on **decentralized path planning** to enhance system flexibility. While this implies allowing AGVs to move freely without rigid paths, such unstructured movement introduces a significant drawback: the vehicle's behavior becomes unpredictable for human employees, leading to safety and efficiency concerns. To address this trade-off, the optimal solution is a **hybrid approach**. This method combines the predictability of predefined paths with the flexibility of dynamic planning. In this way, the vehicle mainly move along predetermined fixed paths but can also deviate from it and move freely when, for example, an obstacle blocks the way.

Despite these improvements, a **gap** remains between theoretical research and practical industrial application [14]. The traditional algorithm typically simplifies the process by assuming fixed and constant kinematic constraints of AGVs, like acceleration, deceleration, and turning. These approximations result in theoretical paths that are difficult to execute in real-world scenarios. Furthermore, many studies assume a static task environment where the vehicles either remain at or disappear from their destination after delivery. In reality, modern factories operate with continuous order streams, requiring the AGVs to be constantly redirected without blocking critical paths [15]. To address these challenges, recent research has explored **Learning Reinforcement** (LR) and **Deep Reinforcement Learning** (DRL). These techniques aim to balance performance and adaptability, allowing AGVs to learn optimal behaviors in complex, dynamic environments where traditional rule-based or static algorithms may fail [14].

4. Motion planning

Motion planning, as it has already been mentioned, is the core task that modifies in real-time the basic static path, computed by path planning algorithms, to deal with unforeseen obstacles,

both static and moving, avoiding situations of collisions and deadlocks.

There are two ways of managing motion planning: a centralized and a decentralized approach. Currently, **centralized motion planning** is the most common approach implemented in AGV systems. In this configuration, the coordination of the entire fleet is delegated to a single central controller. This unit is responsible for computing collision-free and deadlock-free paths for all the vehicles simultaneously. By using global information, like the real-time position of every AGV, their goals, and the static paths they have to execute, it performs an optimization process to determine the optimal execution strategy. However, despite its power, this approach faces two significant limitations. First of all, it is limited by computational scalability, which means that the processing time to calculate the paths increases drastically with the size of an AGV fleet. Second, having only one central server creates a single point of failure, which decreases the overall robustness and reliability of the system. In other words, the failure of the central controller means failure of the whole system.

On the other hand, with **decentralized motion planning**, each AGV is responsible for avoiding collisions and deadlocks by using its own local information, as well as the one exchanged with the other vehicles, to react to unforeseen circumstances while traveling its predefined path. This method of motion coordination is more responsive, robust, and scalable than the previous one, as the information and intelligence are distributed among various entities, which eliminates the single point of failure. However, one of its drawbacks is solving deadlocks and finding primal paths.

A possible solution could be to apply a **hybrid method** that combines the above two and avoids the disadvantages of both. The central controller could be used to plan global paths and goals, but coordination motions are left to the specific AGV [16].

The execution of motion planning relies on three core functionalities: collision avoidance, deadlock avoidance, and zone control. **Collision avoidance** is the part of motion planning that deals with avoiding collisions with static or dynamic obstacles that cross the path of an AGV. A collision is any unwanted physical contact between the AGV and an obstacle, such as a wall, a worker, or another vehicle. The most basic and widely adopted implementation consists of using a safety scanner that monitors a specific safety zone around the vehicle. If an object enters this zone, the AGV automatically slows down and stops. Once the obstacle is removed or the path is clear, the vehicle resumes its operation. On the other hand, more advanced methods allow the AGV to move around the object or plan an alternative route. Collision avoidance strategies can be categorized into two main architectures: centralized and decentralized. In a **centralized** way, as described above, a central unit uses global information to perform an optimization process to determine the optimal collision-free paths for each AGV, ensuring that two vehicles do not occupy the same space at the same time. If a potential collision is detected, the central unit recalculates the paths until it finds a safe solution. This approach has several drawbacks. The first one is a scalability issue since the optimization process becomes computationally expensive as the number of AGVs increases. Second, the central unit is not capable of overseeing dynamic obstacles like workers or items that are not in the fixed map, making it ineffective for real-time safety against unexpected events. If the central controller also has to consider all dynamic features, it will probably be overloaded. For these reasons, modern systems prefer **decentralized collision avoidance**. In this configuration, each AGV acts independently, using onboard sensors to perceive its surroundings. If it detects an obstacle, it reacts locally by slowing down, stopping, or driving around it. This approach is much faster and handles dynamic environments better. However, its problem is related to predictability and safety. Since

the AGV can move freely, it means that its behavior is unpredictable for human workers, creating safety concerns. Therefore, many industries still prefer AGVs to move on predefined paths so that workers always know where the vehicles are going.

Deadlock avoidance, on the other hand, prevents deadlocks, which occur when an AGV is stuck in its current position and cannot move forward or backward. They are one of the main sources of bottlenecks and failures in an AGV system, causing significant time loss. To prevent this, engineers typically use three strategies: designing smart layouts, limiting the number of robots allowed in specific "control zones," or applying specific traffic rules. In the case of **centralized control**, the central unit constantly checks for potential deadlock and intervenes by permitting one AGV to pass while ordering others to stop or reroute. However, these algorithms are time-consuming and complex. They often detect the problem only moments before it occurs, making the system fragile and prone to delays. In contrast, in a **decentralized system**, instead of waiting for a complex calculation from a central unit, the AGV autonomously decides how to act to avoid getting stuck.

Lastly, **zone control** is a traffic management tool designed to prevent collisions and deadlocks by limiting the number of AGVs in a specific area. It consists of dividing the factory layout into non-overlapping zones, each with a maximum capacity. If a zone reaches the full capacity, any new vehicle trying to enter has to wait outside or calculate a different route. These zones can be either fixed or dynamic, where the size and arrangement can change in real-time based on the current traffic situation.

5. Vehicle Management

Vehicle management controls and monitors the status of an AGV, including battery lifetime, maintenance requirements, and error status handling. It is the simplest core task and, therefore, it does not require complex algorithms.

This task can be managed in two ways: centralized and decentralized control. In the **centralized approach**, there is a central controller that oversees all the vehicle management statuses. If the central unit detects that an AGV lacks battery life for the execution of a certain task, it will either assign a less energy-demanding task or direct the vehicle to a charging station. Similarly, when the central system receives error statuses from an AGV, the vehicle will be excluded from the task allocation optimization, and it will be sent to maintenance.

On the other hand, in a **decentralized control**, each AGV autonomously monitors their own parameters and makes independent decisions. For instance, when an AGV enters an error state, it can decide to reassign its local task to neighboring vehicles. This autonomy is particularly evident in a decentralized auction process for task allocation, where internal status directly influences bidding behavior. In this context, a vehicle with low battery capacity will not bid on an energy-intensive task, but it will directly head to a charging station.

Error statuses or **maintenance signals** cannot be fully controlled because they occur unexpectedly. Therefore, the priority is responsiveness, achieved by implementing robust control algorithms to address these issues effectively when they appear, rather than relying solely on the ability to predict them. **Battery management**, on the other hand, can be controlled, and it can have a great impact on the total performance. Regarding the strategy of when to recharge, three distinct types of schemes are identified. The first is **opportunity charging**, where the AGV uses idle time during operations to recharge its battery. The second is **automatic charging**, where

the vehicle recharges the battery when its level drops below a certain limit. Finally, **combination charging** combines the previous two. From a technological perspective, the ways batteries can be recharged differ. The most common methods used are battery swapping and automatic charging. In battery swapping, the AGV physically swaps its empty battery for a fully charged one. While in automatic charging, the battery remains onboard while the vehicle is connected to a charging station.

However, despite its critical impact, it has received limited attention in the existing literature, and battery constraints are often oversimplified or neglected in theoretical models, representing a significant gap between research and practical application [9].

2.2.2.1.2 Design of an AGV system

The previous section analyzed the control architecture and the core tasks, which are responsible for managing and coordinating a fleet of AGVs. However, the overall efficiency of an AGV system does not depend only on the robustness of these two elements, but it is also significantly influenced by the physical environment in which the vehicles operate. In fact, there is a direct dependency between control and design: core tasks like path and motion planning operate within predefined structural constraints. If the facility layout is designed inefficiently or poorly dimensioned, the optimization of the fleet execution will be compromised, regardless of how advanced the control software is. Therefore, the design of an AGV system is an important phase since it defines the physical limits for the fleet's operations. According to Ganesharajah et al. [12], the design of an AGV system involves the following aspects: designing the flowpath layout, determining the location of the pick-up and drop-off points (P/D points), and dimensioning the fleet size [12].

The **flowpath design** and **location of the P/D points** are crucial decisions that affect the installation costs and operating expense of the system. P/D points are the physical interface between the AGV fleet and the manufacturing or storage environment. They define the locations where the vehicle stops to perform the loading (pick-up point) or unloading (drop-off point) of the items. The flowpath design problem focuses on determining which aisles to include in the AGV system, and the direction of travel for each segment, to minimize the total travel distance. This total travel distance considers two types of travel distance: **loaded**, which corresponds to the distance covered while carrying an item, and **empty**, which refers to the distance to reach a load or charging station. The loaded travel distance is predictable and easier to estimate, while the estimation for the second type is more complex since it depends on real-time dispatching logic.

Another aspect to consider for the optimization of the total travel time is the direction of the path. Paths can be unidirectional, where vehicles move in only one direction, or bidirectional, allowing movement in both ways. Unidirectional paths simplify the control system and minimize congestion risks but may increase the total distance traveled, as vehicles cannot turn back. On the other hand, bidirectional paths optimize travel time by allowing more direct routes but require sophisticated collision avoidance mechanisms to manage potential traffic.

The paths are organized in standard structural configurations, ranging from basic linear paths to complex networks. The simplest form is the **single-line layout**, which is typically used for low-volume applications where vehicles move back and forth along a single path between two P/D points. Although it is a low-cost solution, it is not flexible and leads to low performance. Another common configuration is the **single-loop**, where the paths form a closed circuit. The traffic is usually

unidirectional, which simplifies the control logic and minimizes the congestion risks. However, if a vehicle misses a station, it must travel the entire loop again, which significantly increases travel time. For more complex operational environment, the **ladder-type layout** is often adopted. It consists of parallel lines connected by perpendicular cross-aisles, allowing AGVs to access specific storage rows without navigating the whole system. Finally, the most advanced configuration is the **complex network**, where the paths form a grid that offers multiple routing options to reach the same destination. This layout optimizes flexibility and fault tolerance, as in case a path is blocked, the AGV can take an alternative route. However, it requires a highly robust control system to manage the traffic and prevent deadlocks.

In recent years, the AGV design has shifted toward **virtual flowpaths**, where the trajectory of the vehicles is defined digitally. This allows AGVs to operate as free-ranging units without fixed infrastructure constraints.

Once the flowpath has been defined, the next step is **determining the fleet size**, defined as the minimum number of AGVs required to achieve a predefined throughput rate. This is also a crucial economic decision to avoid both production bottlenecks and unnecessary capital investment.

The unit load is an aspect that influences the dimension of a fleet. It refers to the quantity of components of items that an AGV has to carry per trip. This introduces a trade-off: larger unit loads allow for a smaller fleet but require heavier, more sophisticated handling equipment; while in the case of smaller unit loads, the cost per vehicle is reduced, but it implies having a larger fleet to move the same volume of material.

The number of AGVs can be computed using **mathematical methods**, like comparing the total work time with the total time available [12]. The total work time is the sum of the load travel, empty travel, and the handling time, while the total available time corresponds to the total operational time the fleet has to complete the required workload. This mathematical approach assumes ideal conditions where vehicles always take the shortest path without stopping, but, in real-case scenarios, vehicles may face traffic congestion, waiting time at intersections, and potential blockages, which delay the actual total work time. This assumption leads to an underestimation of the fleet size. To overcome this issue, **simulation** has been considered the most reliable tool for the final decision, as it can replicate the actual complexity of the environment and provide a more precise estimation.

2.2.2.2 Autonomous Mobile Robots

As discussed in the previous section, traditional AGVs present significant limitations regarding flexibility, particularly when facing unexpected moving obstacles or structural changes in the layout. Therefore, to address the challenge of highly dynamic environments, a different category of mobile robots has emerged: Autonomous Mobile Robots (AMRs). In recent years, the adoption of AMRs has drastically increased compared to traditional AGVs, driven by the need for systems capable of adapting easily and quickly to changing environemt and providing a wider range of services beyond simple transport, such as patrolling and collaborating with operators [17].

The first generic AMR patent was issued in 1987. Since then, it has been used mainly in the fields of robotics and information technology, but it has recently emerged in logistics applications, and its importance is expected to increase significantly in the near future. AMRs can now be found in industrial, healthcare, hotel, security, and domestic settings, performing a wide range of tasks. The activities performed characterize and divide AMRs into three main groups: material handling

operations, collaborative and interactive activities, and full service activities, as it is illustrated in the Fig. 2.6 [17].

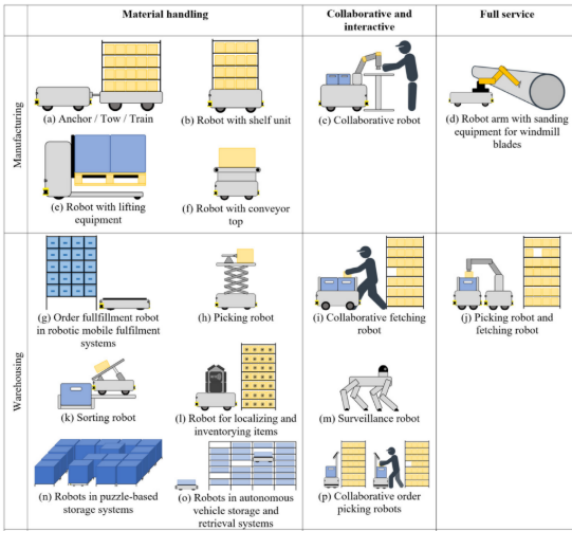


Figure 2.6: Difference between AGV and AMR [17]

AGVs and AMRs are often confused, but each system operates with fundamentally different technologies, from perception and navigation software to onboard sensors. While AGVs are automated and computer-controlled machines that can perform based on a set of defined tasks by following specific instructions with minimal or no human intervention, as it is shown in Fig. 2.7. AMRs are often considered an evolution of the AGV since they have the intelligence to make decisions by themselves, without direct human input or pre-configured scripts, when faced with new or unpredictable situations, like moving obstacles, offering flexible solutions. They are designed to move autonomously within their designated environment with the support of advanced laser-based perception and navigation algorithms.

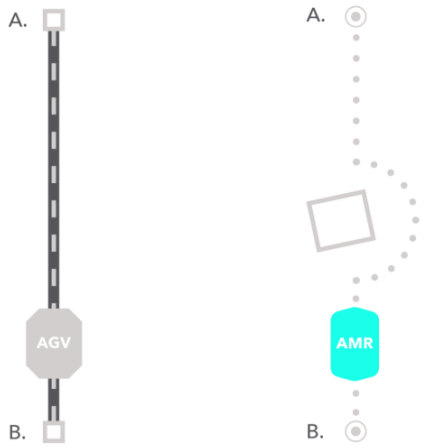


Figure 2.7: Difference between AGV and AMR [18]

AMRs overcome the limitations of conventional AGV technologies, offering five distinct advantages [18]. The first benefit is **flexibility**. AGVs follow predefined paths using lasers, beacons, barcodes or magnetic tape, which implies that it is required ongoing infrastructure maintenance and high precision when it comes to the definition of the paths. In this context, even minor foreign objects positioned in the laser's field of view prevent the vehicle from moving until the obstacle is manually removed by human personnel. On the other hand, AMRs do not require external infrastructure for navigation, making installation simple and flexible. In addition, most importantly, they are capable of detecting, avoiding, and dynamically moving around obstacles without human interaction. AMRs are also very versatile since a single unit can be reconfigured for diverse environments, including narrow corridors and shared workspaces. Another key difference lies in the flexibility to reassign the vehicles across different locations to optimize **resource utilization** and manage variable operational demands. Moving an existing AGV system is highly inefficient since it is essentially the same as installing a new AGV system for the first time. This rigidity means companies often end up buying separate, dedicated systems for every location, resulting in a waste of capital. In contrast, AMRs are much more flexible because they can be easily redeployed from one plant to another, or quickly reassigned to a different operational zone within the same facility. Additionally, the setup time after a move is short since AMR fleets operate with a centrally controlled fleet manager. This mobility has relevant economic benefits, for example, it allows organizations to treat AMRs as a shared resource, therefore units can be shared among multiple facilities to effectively manage seasonal demand peaks without permanent capital investment in every location or to improve the throughput in busy areas by redeploying underused assets from other facilities. Finally, this ability to balance resources reduces the total number of machines a company needs to buy, leading to significant cost savings.

The second benefit is their **scalability**, which is a critical differentiator in material handling systems. AGV systems have limited scalability because adding new vehicles depends heavily on the physical infrastructure. Expanding an AGV fleet is a demanding process that requires significant time and money for planning, maintenance, and installing new equipment. This approach also forces the facility layout to remain rigid to accommodate the guiding systems, limiting how the space can be used. While AMRs offer much faster scalability. New units can typically start working in less

than a day because they navigate using a shared digital map instead of physical guides. This means that increasing the fleet size does not require building renovations or extensive new training for the staff. Furthermore, a major advantage is that companies can change routes or add new vehicles internally without relying on external vendors, giving them greater independence and control over their operations.

Another advantage is **on-board intelligence**. AGVs lack this intelligence, which is required for the modern concept of a smart factory. Since they rely on fixed infrastructure, they do not need advanced processing capabilities, but this limits their utility, for example, they can collect only a small amount of data, leaving operators without real-time information. AGVs, as has been mentioned several times, are also not able to detect and avoid moving obstacles. Lastly, integrating traditional AGVs with external software like Enterprise Resource Planning (ERP) systems is often difficult. On the other hand, AMRs are characterized by onboard intelligence that allows them to adapt quickly to a changing environment and can easily be integrated with advanced ERP systems. With the support of machine learning, AMRs can collect data to update their maps, enabling them to learn from experience and identify the fastest routes even in complex areas. In these conditions, plant operators can safely interact and collaborate with the vehicles. Additionally, AMRs often use lighting and audio signals to communicate their intentions, making their movements predictable and easy to understand for plant operators.

The last benefit is related to **usability**, which covers the ease, cost, and complexity associated with setting up, operating, and making changes to the automated system. While AGVs are designed to perform simple tasks, their installation and daily operation are very complex and expensive. They rely on maintenance by certified experts or specialized engineers, which creates a significant operational burden. Additionally, implementing changes, like modifying a route, requires physical updates to the facility infrastructure and additional staff training, making any modification expensive. In contrast, implementing AMRs is significantly easier and cheaper. The setup involves driving the robot through the facility once to generate a map, then defining zones and key locations via software. The system is designed to be managed by the operator using a simple interface. This allows the employee to update routes or zones directly without needing specialized engineering support. Lastly, if the layout of production lines changes, these modifications are managed entirely within the digital map, requiring no physical changes to the building.

The evolution of AGVs into AMR has become possible due to new hardware and software technologies, which have allowed for operations to be autonomous in dynamic environments, not only for navigation and detection but also for object manipulation [17].

2.2.2.2.1 Hardware technologies

To operate effectively in dynamic environments, an AMR requires a physical structure that integrates three key aspects:

- **Locomotion**, which is the ability to move with agility.
- **Computation**, that ensures the processing of data in real-time.
- **Perception**, which allows the AMR to interpret its surrounding environment.

These functions are physically realized within the **Mobile Robotic Platform**, which acts as the core body of the AMR. It contains all the key components required for operation, including the chassis, the battery, the drive train, and the central onboard computer for processing sensor data and navigation. For emergency purposes, emergency stop buttons are always mounted on the exterior of the chassis. [19]. The platform's design, like shape, size, and steering mechanism, depends on the specific operation.

The **manipulating equipment** mounted on top of the platform is a technically independent system, making the robot highly modular. It is possible to perform new services and material handling operations by combining AMRs with different types of manipulating equipment. [17]. The manipulating equipment can be divided into two categories based on their power requirements [19]: passive and active applications. In passive applications, the modules do not require a power supply from the vehicle, while active applications rely on the platform's battery to operate. Therefore, the energy source is a critical component of the hardware architecture. Unlike traditional AGVs that often relied on bulky lead-acid batteries, AMRs often use high-capacity solutions such as **lithium-ion batteries** [17]. This technological shift enables a much more compact vehicle design, allowing robots to navigate narrow aisles or slide directly underneath load carriers, and provide the substantial energy required to sustain the complex onboard calculations needed for autonomous navigation.

Regarding the movement of an AMR, the **locomotion mechanism** is the subsystem that translates navigation algorithms into physical movement. It has a strong impact on the vehicle stability, manoeuvrability, and kinematics. Indoor mobile robots are designed to move, run, or walk to complete their specified assignments, so they can be equipped with legs, wheels, or wings, sometimes [20]. Since many intralogistics activities require a high level of stability, **wheeled locomotion** is the most common mechanism. It has achieved a high level of efficiency due to its easy controllability and a relatively simpler mechanical design. There are four different types of wheeled locomotion, which are illustrated in the Fig. 2.8 [20]:

1. The Standard Wheel (a), which is characterized by two degrees of freedom, and rotation occurs around the wheel axle and point of contact.

2. The Castor wheel (b), which has two degrees of freedom, and the turning occurs around the offset steering joint.

3. The Swedish (or Mecanum) wheel (c), characterized by three degrees of freedom, and the rotation occurs around the rollers, wheel axle, and contact point.

4. The Ball or Spherical wheel (d), which has more surface area than the previous cylindrical castor wheels.

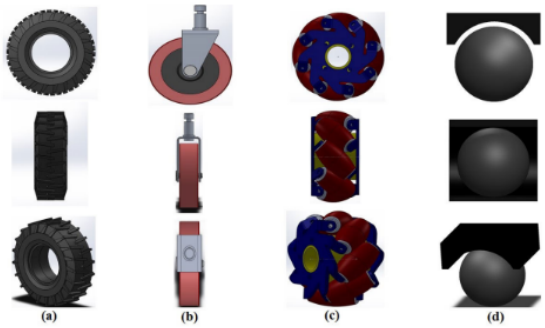


Figure 2.8: Types of wheels for AMR [20]

On the other hand, the **legged locomotion** is usually employed for movement in rough terrain since it allows the vehicle to be highly mobile and overcome significant obstacles such as several steps or stairs [17, 20].

The ability of AMRs to navigate and operate in a dynamic environment results not only from the types of locomotion mechanisms but also from their capacity to make real-time decisions, which has become possible with the introduction of **ultra-low-power AI processors**. The most widely recognized AI-focused processor architectures for vision recognition of face, body, gesture, object, or scene are the Intel Nervana, NVIDIA Xavier and Kneron AI SoC. This technological shift allows the introduction of new ways for calculating complex decisions related to dynamic routing, scheduling, navigating, and reacting to obstacles [19].

The last important aspect of an AMR hardware is related to **perception**, which is the ability to see, hear, or become aware of the surrounding environment. In the context of an autonomous mobile robot, perception means the ability to retrieve information about itself and the external environment within which it is located. For a vehicle to work autonomously, perception is vital and is achieved with the help of high-resolution sensors and algorithms to extract information from them [20]. The Fig 2.9 shows the basic flow diagram of the perception process, and its main components are processing data from sensors, representation of data, and AI algorithms. All together helps an AMR to operate numerous functions such as detecting moving objects and environmental changes, recognizing gestures and voices. All of these examples are achieved with the support of some specific sensors.

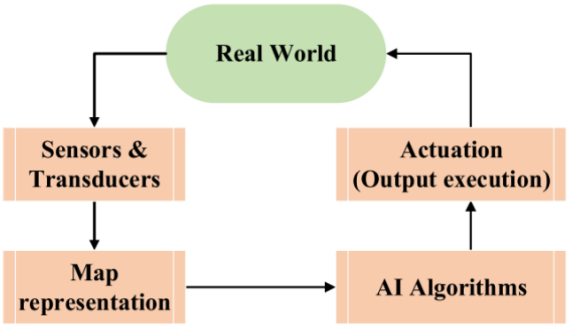


Figure 2.9: Main components of a perception process for AMR [20]

Sensors are devices capable of detecting any physical changes in the environment and converting them into electrical signals that can be used in other procedures. In the context of AMR systems, their main objective is to guarantee that the vehicles navigate autonomously and collision-free within their operating area while safeguarding danger zones. Sensors can be categorized into two categories [20]:

1. Proprioceptive or exteroceptive kinds of sensors.
2. Active or passive sensors.

A **proprioceptive sensor** can read the internal changes of the vehicle, such as the internal battery voltage and temperature, speed of motors used, acceleration, and net loads on the wheels. While **exteroceptive sensors** measure aspects of the external environment.

An **active sensor** has its own source of energy, while a **passive sensor** can detect energy in response to inputs from the internal or external environment, like gyroscope motion.

The number and types of sensors to equip depend on the degree of motion freedom of an AMR [19]: the greater the freedom of movement, the higher the probability of interacting with dynamic obstacles. Therefore, it implies the need of a larger number of sensors with superior precision. The most common safety device is the optical laser scanner, which can detect obstacles that block the path of the AMR. The installation of these scanners is strictly regulated, with specific requirements for their height and position to guarantee effective coverage. Besides safety-relevant sensors, AMRs can have additional built-in sensors, like for navigation and locomotion. Due to their high versatility and ability to provide real-time feedback, the most common sensors utilized in indoor industrial settings are the following [19, 17, 20]:

- 2D and 3D lasers sensors.
- 3D Depth Cameras sensors. They are the most used type of sensor since they provide high-resolution RGB pictures and a 3D point cloud. However, it is required an higher computational onboard power, limiting their practical operation to specific use cases.
- Gyroscope sensors. It is designed to measure or maintain orientation and angular velocity. In the context of AMRs, it detects how fast the robot is turning around its axes. This data is essential for the navigation system to calculate the robot's heading and stabilize its movement.

- Encoders sensors. It is a a kind of electro-mechanical device that can provide motion control with position, velocity, and change in direction.
- Ultrasonic sensors. They measure the distance by emitting high-frequency sound waves and calculate the "time of flight", which is the duration between emitting the signal and receiving the echo. These sensors are particularly valuable in industrial settings because, unlike optical sensors, they can reliably detect transparent surfaces like glass or clear plastic.
- Infrared sensors. They use infrared light to detect the presence of objects or measure short distances. They typically operate by emitting a beam of light and measuring the intensity of the reflection from a surface.
- Light Detection and Ranging (LiDAR) sensors. It is considered the primary sensor for AMR navigation. It operates by emitting rapid pulses of laser light toward the environment and capturing the reflected signal with a sensitive detector. By calculating the precise time it takes for the light to return (time of flight), the system generates a detailed, high-resolution map (point cloud) of the surroundings, allowing the robot to localize itself and detect obstacles in real-time.

2.2.2.2.2 Software technologies

According to Niloy et al. [20], a mobile robot consists of two vital sections: the mechanical structure, which has been described in the previous section, and the control section. While the hardware technologies provide the physical capabilities for movement and perception, the software architecture is responsible for transforming an AMR into an intelligent and autonomous vehicle, since it acts as the brain, taking care of monitoring the mechanical structure's ability to achieve the required objectives. The presence of this onboard intelligence leads to a **decentralized control architecture** [19]. Unlike traditional AGV systems, where one central unit decides the routing and dispatching for the entire fleet, AMRs possess the computing power to make decisions independently, which allows the vehicle to communicate and negotiate directly with external resources, like ERP systems or material handling software, and to react dynamically to any demand or changes. This decision-making process consists of three steps: **perception**, **processing**, and **cognition**. The raw data gathered by the perception system is refined and processed by the cognition function, which identifies the objective and plans an optimal collision-free path from the starting point to the destination [20].

This entire decision-making process is crucial for **navigation**, the most challenging task for AMRs, especially for indoor navigation. Since AMRs do not travel on fixed paths, they must constantly figure out their location while monitoring the environment. Therefore, navigation is typically divided into four main activities [20]: mapping, localization, path planning, and obstacle avoidance.

Mapping addresses the critical issue related to modelling the surrounding environment using a map-based approach. It is defined as the process of creating a representation of the environment, and it allows an AMR to interpret its surroundings, make informed decisions, and navigate towards its goal. However, this process presents a dual challenge often described as a "chicken-and-egg" problem: constructing a map requires knowledge of the location, but determining the location requires an existing map. This paradox is resolved by a supportive technology for real-time navigation

called **Simultaneous Localization and Mapping** (SLAM). It can create a detailed map of the area and calculate the instant position of the vehicle by converting raw 3D point clouds retrieved from scanning sensors into a stable reference map, filtering out dynamic obstacles to ensure that this reference map remains accurate and usable over time.

The choice of the appropriate map representation depends on three factors:

1. The map's precision must match the AMR's operational needs.
2. The features represented by the map must match the features that are extracted by the sensors.
3. The map's complexity must be balanced against the available computational power.

The two most popular types of traditional map representation are metric and topological maps, which are generally associated with **two-dimensional maps**. A **metric map** shows the geometric attributes of a surrounding environment, where the location of the objects is presented as precise coordinates in a two-dimensional space. While this approach offers high accuracy and is easily readable by humans, it is very sensitive to cumulative errors over time. The most popular metric map approaches are **feature-based**, which represents the environment using specific geometric landmarks (lines, points) to define static objects like walls in a continuous space, and **free-space representations**, which divides the environment into a discrete grid of cells, classifying each as "occupied" or "free" based on probabilistic sensor data.

On the other hand, the **topological approach** doesn't rely on exact geometric coordinates. Instead, it represents the environment as a graph composed of nodes and arcs. In this model, nodes refers distinct locations, like intersections, rooms, or workstations, while arcs represent the paths that connect these adjacent nodes. The main advantage of this representation is its computational efficiency since it doesn't require the transformation of the sensor data into a two-dimensional reference frame, but only requires the storage of location information given a sensor reading. However, a significant drawback is its high dependency on prior exploration. To define the nodes effectively, the robot must physically visit and record the characteristics of each location beforehand, meaning that it requires a more expensive exploration of the environment in order to estimate position with higher precision.

But the last decade has witnessed the emergence of **three-dimensional map** representations. An example is **sparse map** representation, which recreates the environment using only a selected set of key visual features. However, this approach lacks the geometric detail necessary for complex interaction. This problem is addressed in the **dense maps**, which reconstructs the full structural nature of objects by capturing all available visual data, providing rich information for decision-making. But this representation requires powerful GPUs. An alternative that balances the previous drawbacks is the **semi-dense map**, which generates depth maps using a higher density of key-points, offering enough geometric information for effective navigation while remaining efficient enough to operate on standard GPUs. **Semantic maps**, on the other hand, go beyond simple geometry by adding meaning to the environment. Unlike traditional maps that tell the vehicle where an obstacle is, semantic maps identify what it is, distinguishing, for instance, between a wall and a shelf. This method allows an AMR to better understand its surroundings and perform its tasks more efficiently, but it requires more computational power and advanced software to recognize objects.

Another issue that navigation has to address is determining the vehicle's precise location at a given time, and it is done through the process of **localization**. This activity is very challenging since raw

data coming from the sensors are very sensitive to minor errors, leading to significant inaccuracy. To overcome this problem, AMRs have to implement sensor fusion methods that combine data from multiple sources to estimate the vehicle's state.

According to Niloy et al. [20], there are three levels of localization problems based on the vehicle's prior knowledge. The first is **local pose tracking**, where the system assumes the vehicle is still near its last known coordinates and uses probability models to refine the position and correct small sensor inaccuracies. A more complex scenario is **global localization**, where, unlike the previous case, the vehicle does not know where it starts. Therefore, it is not capable of narrowing down its search to a specific area, and it must initially assume that it could be located anywhere on the map until it gathers enough sensor data to recognize its surroundings and identify its position. Finally, the most challenging problem is the **kidnapped robot problem**, which occurs when the AMR is physically moved to a new location without its knowledge, but the vehicle believes that it is still in the old location. The vehicle overcomes this problem when it realizes that its current sensors' reading contradicts its expectations and, therefore, restarts the search process.

Once the AMR has located its current position within the map, the next step is **path planning**, where the vehicle determines how to reach its desired goal. Unlike traditional AGVs that have to follow a predetermined path, an AMR calculates a new, unique, and colliision-free route from its start point to the destination. This process involves three steps, as illustrated in the Fig. 2.10: environmental modelling, defining optimization criteria, and executing a path searching algorithm.

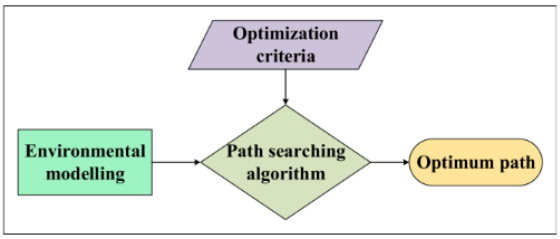


Figure 2.10: Difference between AGV and AMR [20]

Path searching algorithms are used to find a collision-free route to reach a destination point while satisfying a set of optimization measures, from route length to safety standards. They can be classified as

1. **Classical algorithms**. They focus on determining if an optimal path exists within a specific area. The main drawbacks are the high computational power required and the lack of flexibility in case of changing conditions. The most common examples of classic algorithms are the Cell Decomposition and Potential Fields.

2. **Heuristic graph-based algorithms**. They are designed for a static environment where the vehicle has easy access to a complete map, which is usually represented as a graph of connected nodes. Instead of exploring every possible path blindly, they use mathematical formulas to estimate the most efficient route. The most famous examples are Dijkstra's algorithm, which finds the absolute shortest path, and A*, which balances speed and accuracy to find the best path quickly.

3. **Reactive algorithms**. These methods are designed for dynamic environments where a complete map is not available. Instead of following a pre-calculated plan, these algorithms make immediate decisions based on live sensor data and it allow the AMR to adjust its speed and direction instantly to avoid unexpected obstacles. The most common algorithms are Fuzzy Logic or Neural Networks

Path planning techniques can be divided into two main categories: global and local path planning. **Global path planning** works on the static and pre-defined global map, which often includes annotations for changing stations, workstations, and no-go zones. Since this process assumes the map is known and fixed, it uses heuristic graph-based algorithms to calculate the optimal route. However, the environment is characterized by moving obstacles that are not usually present on the static map. This issue is addressed by **local planning**, which relies on data required in real-time from onboard sensors and it is responsible for the immediate interaction with the environment, allowing the vehicle to react to dynamic obstacles like human workers. In these unpredictable scenarios, the reactive algorithms are implemented.

To react to dynamic obstacles, path planning needs to be supported by **obstacle avoidance**, which refers to the change of the trajectory of the vehicle to prevent real-time collisions within dynamic environments. If an AMR is not capable of detecting and avoiding unexpected obstacles, besides risking physical damage, it fails to reach its destination. The most common algorithm implemented is **Artificial Potential Fields**, which balances two opposing forces: an attractive force pulling the robot toward its goal and a repulsive force pushing it away from obstacles. In more advanced architectures, this algorithm is integrated with a global map, which is updated as the AMR detects and avoids an object.

2.2.2.2.3 Design of an AMR system

Another important aspect that needs to be addressed is the design of an AMR system. This phase focuses on defining the architecture that governs how the entire fleet interacts with the factory layout and production workflows. The design process includes three core activities: fleet sizing, dynamic zoning, and resource management [19].

Regarding **fleet sizing**, the traditional methods, that are based on fixed paths and commonly used for AGV systems, are not suitable for AMR fleets. Due to the dynamic nature of AMR navigation, travel distances and times are highly variable and uncertain, making standard approaches ineffective. In fact, in the context of AMR systems, the number of possible paths to connect two points is effectively infinite. In addition, in many current applications of AMR systems, the vehicles can also operate in shared environments with humans, leading to congestion and high traffic that affect the performance of the system and increase the travel times. Given this context, the calculation of the fleet sizes requires new methods that also take into consideration the possibility of heterogeneous fleets, for example, a fleet of AMRs with different equipment, sizes, or functions. To solve the problem of determining the optimal number and configuration of vehicles, the main methods implemented are mathematical modeling and simulations.

Similarly, the spatial organization also goes through a significant shift. Since AMR vehicles are no longer tied to a fixed guide path, they can plan their path themselves and can move freely

in predefined travel zones. Therefore, the flowpath design and location of P/D points are not crucial decisions anymore, and they are replaced by the concept of **dynamic zoning**. It consists of dividing the operational environment into distinct zones where the vehicles move freely and operate autonomously. These zones can be modified on a daily or weekly basis, or even dynamically in a decentralized manner by the AMR to adapt to changes in the production needs. Limiting the operating area for each vehicle allows for improving the cost, productivity, vehicle availability, and overall system responsiveness.

The primary objective of this phase is to minimize travel distances, traffic congestion, and throughput time, while balancing the workload across the system. All of them are achieved through four key activities: analyzing the service area requirements, determining fixed or dynamic service points, configuring the zones (including defining boundaries and flow directions), and allocating the appropriate number of vehicles to each zone. However, this flexibility makes the system much harder to model compared to traditional AGV systems since service points can move and, therefore, the workload distribution shifts constantly. This creates a huge number of variables, making standard mathematical calculations too slow or difficult to apply effectively. So, simulation and evolutionary algorithms are considered the best tools to manage this complexity, as they can adapt to these changing conditions better than static models.

The third and last aspect of AMR system design is **resource management**, where the difference between AGV and AMR technologies becomes more apparent. While traditional AGV systems are usually limited to perform a single specific task, like lifting a pallet, AMRs are designed to be more versatile, meaning that the AMR's platform allows a wide range of resources to be used and shared. For example, they can load different types of items, exchange physical equipment, and manage their own battery swapping. This versatility required an efficient management of resources, where the system must decide how the resources are shared among the vehicles. This efficiency can not be achieved through a fully decentralized system, where every vehicle decides for itself, but, instead, it is necessary to have the presence of a coordination mechanism, where the AMRs share their operational data to make smarter decisions and reach the global optimum.

Currently, new modelling approaches are being developed to solve this coordination issue. The most common technologies implemented are **predictive analytics**, since they allow the system to forecast the perfect moment to change a battery or tool, minimizing the risk of conflicts and keeping productivity high.

2.2.2.2.4 Limitations and open research gap

Despite the technological advancements, the adoption of AMRs still presents significant limitations and open research gaps [19]. The first challenge is related to the **complexity of the manufacturing environment**. Modern factories are highly dynamic, particularly due to the shift towards small batch productions, which necessitate frequent layout changes, such as relocating workstations. When this happens, the AMR's map is updated manually, a process that is time-consuming and inefficient. Furthermore, current sensors are good at detecting obstacles, but they are not capable of distinguishing between a human, a forklift, or a cable duct on the floor. This implies that the vehicle can not make intelligent decisions based on the obstacles, like stopping for a person but driving over a cable cover, rather than treating everything as a generic wall to avoid. This limitation leads directly to critical **safety issues**. For example, most AMRs still rely on two-dimensional laser scanners that can perceive only at a specific height, creating dangerous

blind spots. For example, an AMR might detect the legs of a table but fail to see the tabletop protruding above. Therefore, the vehicle might wrongly calculate that there is enough space to pass between the legs, leading to a collision with the table surface. This gap in two-dimensional perception remains a significant risk for certification and safe operations alongside humans.

Another area of AMR systems that needs more development is **resource and energy efficiency**. As the fleet sizes grow and processing demands increase, the onboard energy consumption rises, rapidly draining batteries and reducing the robot's operating time. Research is currently exploring ways, like Edge Cloud, to transfer these heavy calculations to a central system via high-speed networks like 5G. This would allow AMRs to consume less power and focus on movement. Similarly, while AI algorithms offer smarter path planning than traditional methods, they are often too computationally heavy to be practical for large fleets right now.

Finally, the industry is struggling with **interoperability**. Currently, most fleet management software is proprietary, meaning vehicles that are designed by different manufacturers cannot communicate or coordinate with each other. New standards like VDA 5050 are being developed to create a common language for all vehicles, but they are not yet fully mature or universally adopted. Solving this problem would allow companies to use and integrate robots from different vendors into a single and cohesive system, breaking the dependency on a single supplier.

2.3 Simulation

In an increasingly competitive world, simulation has become a powerful analysis tool for the planning, design, and control of complex systems. According to Shannon [21], simulation is defined as "*the process of designing a model for a real system and conducting experiments with this model for the purpose of either understanding the behavior of the system and/or evaluating various strategies for the operation of the system*". In other words, simulation modelling is an experimental and applied methodology that aims at describing the behavior of a system through the construction of theories or hypotheses to create a model that predicts future behavior in case of any changes in the system or its method of operation.

The use of simulation brings many benefits, including its ability to explore "what if" scenarios and test new policies, layouts, or hardware without disrupting ongoing operations or before purchasing resources. The virtual environment created through the simulation enables engineers to manipulate time, allowing them to quickly examine long-term horizons or slow down a phenomenon for study, and gain insight into which variables are most important to performance and how these variables interact. In addition, the knowledge gained about a system while designing a simulation study can be used to understand how a system really functions. The most important benefits are its **versatility** and **flexibility**. In fact, simulation can be applied in almost any type of application, from computer system manufacturing to social and behavioral.

However, simulation has its own drawbacks. It is a time-consuming and expensive process that requires specialized training, as the quality of the analysis is strictly dependent on the skill of the modeler. Furthermore, interpreting results can be challenging, as it is often difficult to distinguish whether an observation made during a run is due to a significant relationship in the system or the randomness built into the model.

Simulation models are typically categorized based on three dimensions: **timing of change**, **randomness**, and **organization**. Specifically, based on the temporal aspect, it can be divided into static and dynamic [22]. A **static** simulation is independent of time, while **dynamic** simulation evolves over time. Dynamic simulation can be further categorised into continuous and discrete. In **discrete** simulation, changes occur at discrete points in time, while in **continuous** simulation, the variable of time is continuous. In addition, discrete simulation is divided into time-stepped and event-driven. **Time-stepped** consists of regular time intervals, and alterations take place after the passing of a specific amount of time. On the other hand, in **event-driven** simulation, updates are linked to scheduled events, and time intervals are irregular.

Regarding the role of randomness, a simulation can be classified as deterministic or stochastic. **Deterministic** means that the repetition of the same simulation will result in the same output, whereas **stochastic** simulation means that it will not always produce the same output.

Lastly, based on the third dimension, data organization, a simulation can be divided into grid-based and mesh-free. **Grid-based** means that data are associated with discrete cells at specific locations in a grid, and updates take place to each cell according to its previous state and those of its neighbours. On the other hand, **mesh-free** relates to the data of individual particles and updates look at each pair of particles.

2.3.1 Phases in a simulation process

Since the main purpose of simulation modelling is to provide support to the decision maker in solving complex problems, it is important to have a robust simulation process. Therefore, the following key phases for a simulation study have been identified [22, 23]: **definition of the simulation model**, **validation of the model**, and **generation of simulation results**. The first phase, which is considered the most important, consists of describing correctly the simulation goals, the inputs and outputs desired, and the assumptions made. Mistakes during this stage are expensive and difficult to fix later on. It starts with the **problem definition**, where usually stakeholders describe the situation in vague terms, focusing on operational "symptoms" such as profit losses, rather than identifying the underlying root cause, and it ends with the **definition of the goals**, which is important since it influences the required level of model accuracy.

Once the objectives are defined, the process shifts to **model formulation**, which is essentially the art of abstraction and simplification. Typically, a simulation model tends to be too detailed and include elements that contribute little or nothing to the understanding of the problem. This approach is unsatisfactory not only because of the increased difficulty of programming the model and the additional cost of longer experimental runs, but also because the truly significant aspects and relationships may get lost in all the trivial details. Therefore, the model must include only relevant aspects of the system. The most common approach used to have an effective modelling is to follow Pareto's Law, which suggests that 80% of a system's behavior is governed by only 20% of its components.

Once the conceptual model is defined in the form of a logic flow diagram, the next phase is validation of the model. It traslates the the logical design into a computer simulation using a selected IT tool. This stage is characterized by two distinct processes: verification and validation. **Verification** ensures that the computer program performs exactly as the modeler wanted. This involves rigorous debugging to ensure that the code logically reflects the conceptual model defined in the previous stage. The goal is to confirm that all parts of the simulation function correctly

without programming errors. On the other hand, **validation** focuses on establishing whether the model is an accurate representation of the real-world system. It involves comparing the model's behavioral data with historical data from the real system to ensure consistency. Validation is not a binary decision variable where the model is either valid or invalid, but rather a matter of degree. In other words, confidence in the usefulness of a model must gradually accumulate as the model passes more tests and as new points of correspondence between the model and empirical reality are found. This process is continuous and should involve the ultimate users, typically middle or upper management, throughout the study.

The final phase aims at conducting simulation experiments and developing various scenarios, depending on the variable parameters of the model. The simulation is executed to generate behavioral data, which is then analyzed to select the optimal solution based on the objectives defined in the first stage. However, ensuring the validity of these results requires careful execution based on the system's nature. A system can be defined as terminating, where the simulation has a natural endpoint, or non-terminating, where it operates indefinitely. In this last case, the analyst must focus on the steady-state behavior.

The last activities to perform are **documentation** and **implementation**. A simulation study is only considered successful if its results are effectively communicated and applied. Therefore, the analyst must translate complex technical results into clear reports that stakeholders can understand and use, making sure that the model is not only accurate but also practically useful for decision-making.

2.3.2 Simulation in manufacturing: impact on AGV and AMR systems

Simulation modelling is an essential tool for managing the complexity of today's manufacturing systems. This industry evolves toward globalization and mass customization, that made production processes increasingly intricate. Simulation allows to analyze these systems deeply and validate new strategies without interrupting ongoing operations. Specifically, the simulation of material flow is a critical area of focus. Since companies aim at responding quickly to market dynamics, adopting **Material Flow Simulation** allows for evaluating different scenarios, such as changes in production paths or the allocation of workstations, before any physical implementation happens. This preventive analysis allows the identification of bottlenecks and ensures that the planned system will operate effectively [22].

The application of simulation becomes particularly relevant in the context of automating internal logistics through the implementation of AGV or AMR systems. In fact, simulation has been extensively applied in the design of autonomous transport systems at two different levels: the **lowest level** focuses on the navigation capabilities of the single vehicle (the on-board control), while the **highest level** focuses on the global control to optimize routing, allocation, and scheduling policies [10]. In complex intralogistics environments, where large fleets are involved and must execute numerous tasks simultaneously, the high-level approach is usually prioritized. The main goal is to find a set of parameters to meet user requirements and maximize system performance. Those parameters include the fleet size, the task dispatching rules that set the criteria to assign the transportation requests to the mobile robots, the charging policy that decides when to recharge the batteries to make sure the vehicle does not run out of buttery in the middle of a task, the idle policy to choose what to do when a robot finishes a task and there is no new task, and, finally, the route planning and traffic management policy.

Given this context, the simulation techniques are widely preferred for the design over analytical ones, which use mathematical equations to find the optimal solution. The main reason is that analytical models become extremely difficult to construct when applied to complex systems characterized by a multitude of variables and dynamic parameters. In addition, analytical methods often require significant simplifications and assumptions, which increases the risk of deviating too far from reality, while simulation modelling can handle this complexity without compromising accuracy, and without the need for a detailed simulation for each vehicle, either AGV or AMR, when analyzing the global control system [10].

When it comes to the choice of simulation tools, there is no universal software solution. The selection of the appropriate software depends on the specific purpose of the simulation and the nature of the process. The most popular simulation packages include Arena, AnyLogic, FlexSim, Simio, Plant Simulation, Enterprise Dynamics, and Simul8. All these platforms are classified as Discrete Event Simulators (DES) and are specifically designed to study complex intralogistics processes [23]. For this reason, particularly when analyzing global autonomous systems, researchers generally prefer using these high-level simulation tools rather than modular control frameworks like the Robotic Operating System (ROS) [10].

2.3.3 FlexSim

FlexSim [24], developed by FlexSim Software Product, Inc. is one of the most popular, powerful and user-friendly 3D simulation software solutions available in the market. It is capable of modeling, simulating, predicting, and visualizing business systems in a variety of industries, including manufacturing and material handling. FlexSim supports companies in decision-making processes, like the optimization of systems and the analysis of alternative investment ideas. It is also used to successfully reduce costs since it can test methods for allocating resources more efficiently, minimize the negative effects of breakdowns and optimize prioritization and dispatching logic for goods and services.

The software operates on an **object-oriented architecture**, where the user can create a 3D model by dragging and dropping pre-defined **3D objects** into the simulation space. These objects are classified into three main classes: flow items, fixed resources, and task executers. **Flow items** are the items that flow through the simulation model, and they represent many different things, from materials moving through an assembly line to customers walking through a service center. By default, flow items look like brown boxes, as shown in the Fig.2.11, but their visual appearance can be easily changed and customized to satisfy specific requirements.

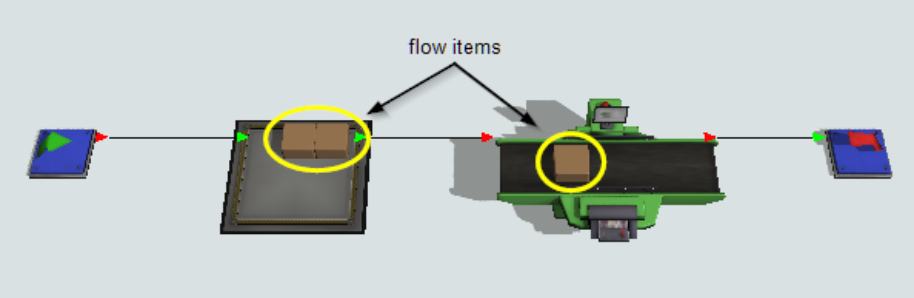


Figure 2.11: FlexSim - Flow items

Fixed resources, instead, are objects that remain fixed or stationary in the simulation model. They interact with flow items in the business system in various ways, and they can represent various steps or processes. Common examples include sources, which create items, processors, conveyors, queues, and sinks. All fixed resources operate in similar ways: they receive flow items through input ports, perform a specific operation on them, and then release the items through output ports to the next stage of the system.

Finally, **task executers** include mobile resources that perform operations like transporting items. Some examples are operators, AGVs, and forklifts. Unlike fixed resources, task executers have specific motion parameters (acceleration, max speed, deceleration) and can travel across the network grid or free space.

The logic of the simulation that dictates the behavior of each system component can be defined in two ways. The traditional method consists of configuring the parameters directly within the object's properties. However, for more complex systems, the most robust approach is using a visual programming tool that runs parallel to the 3D model called **Process Flow**. It consists of building the model's logic by connecting pre-built activity blocks with a flowchart approach. This method allows for the centralization of logic in one place, facilitating scalability and debugging as the model evolves and changes.

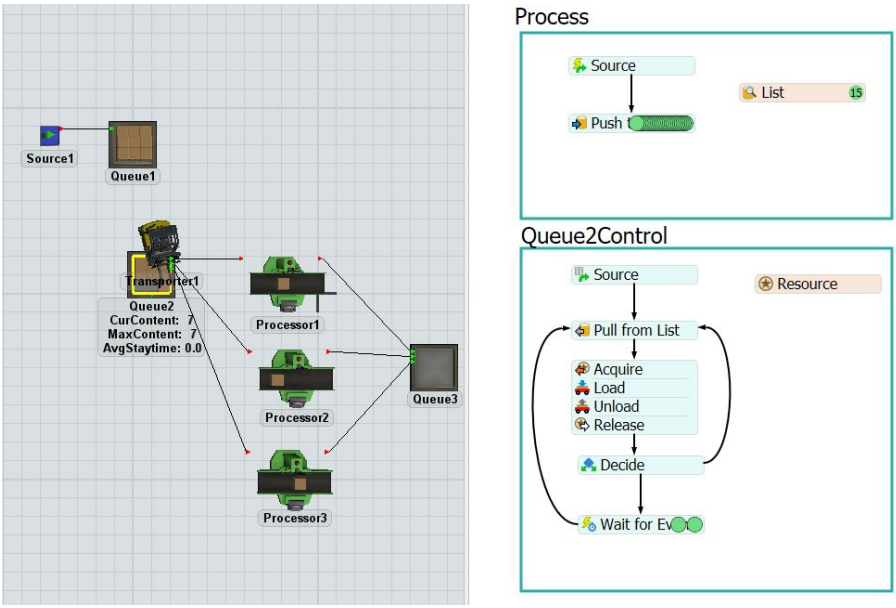


Figure 2.12: FlexSim - Process Flow

While the Process Flow tool can handle almost any kind of simulation logic, more complex modelling scenarios often require a higher degree of customization. In this case, **FlexScript** is the most appropriate method to exploit. FlexScript is the internal scripting language of FlexSim and it allows to have a better understanding of the model's internal mechanics and to implement custom logics that are too complicated to be efficiently represented by visual blocks. This language presents functions made specifically for the FlexSim simulation environment, including concepts such as variables, loops, conditionals, functions and data types; but it also has some similarities with other programming languages like C.

FlexSim can be used not just for visualization, but also for statistical validation. In fact, the software automatically collects data on every object in the model, which is accessible through FlexSim's **Dashboard** tool. This is a graphical user interface that displays real-time data, like TH or resource utilization, via charts and graphs. In the context of intralogistics, it is a very useful tool since it allows companies to monitor the simulation and calculate in real-time KPIs.

2.3.3.1 Transportation systems in FlexSim

Another reason why FlexSim is one of the most popular simulation software tools is the possibility of modelling in detail complex transportation systems, like AGVs and AMRs.

By default, in FlexSim, a task executer moves following a straight line, since it is the shortest way to connect two points. This default travel logic is often insufficient for realistic industrial environments, as it ignores physical constraints. For instance, task executers travel through walls, machinery, or other types of barriers. To overcome this limitation of the default travel system,

FlexSim offers more advanced travel tools to create more accurate paths, ensuring that the movement appears visually correct and the results are representative of the actual business system. Depending on the type of navigation required, FlexSim provides two main navigational tools: AGV network and A* navigation.

AGV network module is designed to simulate systems that implement AGVs to transport goods from one destination to another. This tool allows users to define specific paths (representing magnetic tapes or inductive wires) and control points. In addition, FlexSim offers pre-built **AGV Process Flow template** to accelerate the simulation design and set up. These templates contain the basic logic to simulate any common AGV systems, but they can be easily customized baased on the system's needs.

Since AGV systems are costly to implement, simulating them beforehand is a critical activity. In fact, beyond statistical analysis, an AGV simulation model can act as a communication tool for the implementation phase, helping engineers understand the logic and routing rules required for programming the physical vehicles.

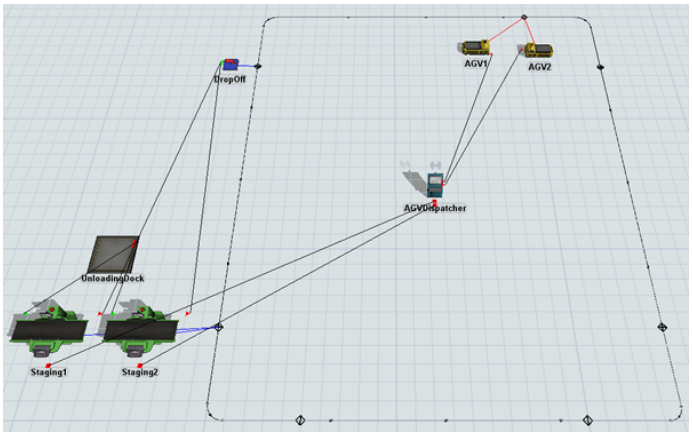


Figure 2.13: AGV network

On the other hand, **A * Navigation system** is usually implemented for systems that require some degree of movement, like in the case of AMR systems. This tool divides the model into a grid of nodes and any fixed resource connected to the A* system is considered as a barrier. In this context, a task executer travels following a path calculated by the A* search algorithm, which finds the fastest path to reach the destination while avoiding any barriers.

To determine the most suitable tool for the simulation, the following strengths and weaknesses, summarized in the Tab. 4, must be taken into consideration.

	AGV Networks	A* Navigation
Advantages	• Gives the user more control over task executer travel paths. • Models might run faster (fewer calculations required). • Can restrict travel direction (one-way vs two-way). • Can set specific speed limits on paths. • Allows creation of virtual distances (manipulating node distance vs actual distance).	• Fairly easy to set up; handles most logic automatically. • Easier to configure for models with high numbers of destinations and complex routing options.
Disadvantages	• Takes a slightly longer time to set up. • Creating paths between every possible destination can be cumbersome. • Troubleshooting connectivity issues can be time-consuming.	• In large/complex models, the search algorithm can slow down simulation speed. • Heavy calculation loads can sometimes result in strange movement visuals (lag).

Tab. 4: Comparison between AGV Networks and A* Navigation

Chapter 3

State of Arts

AGV and AMR systems, as it has been mentioned in the previous chapter, represent a technological revolution for the automation of material handling operations in intralogistics. However, given their high costs and risks, simulation has been considered in the scientific literature as a fundamental tool for analyzing these systems. In this context, this chapter presents a critical review of the current state of the art. The primary objective is to investigate the existing literature to identify research gaps regarding the correlation between layout complexity and the strategic choice of the transportation system. Specifically, the analysis focuses on how simulation is used to compare AGV and AMR fleets and to evaluate their adaptability across environments with increasing layout complexity.

The research and analysis process of the scientific material can be divided into two phases, as shown in Fig. 3.1: **paper identification** and **critical analysis**.

The first phase consists of searching and selecting relevant scientific papers to write both the *Theoretical Background* and the *State of the art* chapters. Specifically, for the first chapter, the research focuses on technical papers that define intralogistics concepts and describe the technical architecture of mobile robots in detail, including their navigation technologies, control strategies, and core fleet management tasks. On the other hand, for the *State of Arts*, the initial research objective was to identify papers specifically dedicated to the comparative analysis between AGV and AMR systems within layouts of increasing complexity, potentially using FlexSim as the simulation software. However, the literature search revealed a significant lack of articles addressing this specific topic. Therefore, the focus of the research was extended to cover related sub-topics, like the general application of simulation on AGV and AMR systems and studies on the impact of layout on their performance. In addition, the research papers was not limited to static environments but also included those analyzing AGV and AMR in dynamic circumstances characterized by the presence of moving obstacles. Regarding the simulation tools, due to the lack of papers using FlexSim, the research was extended to include all types of software. Finally, an important part of the research was dedicated to finding models that compute the complexity of a layout through a measurable index in order to objectively compare different layouts.

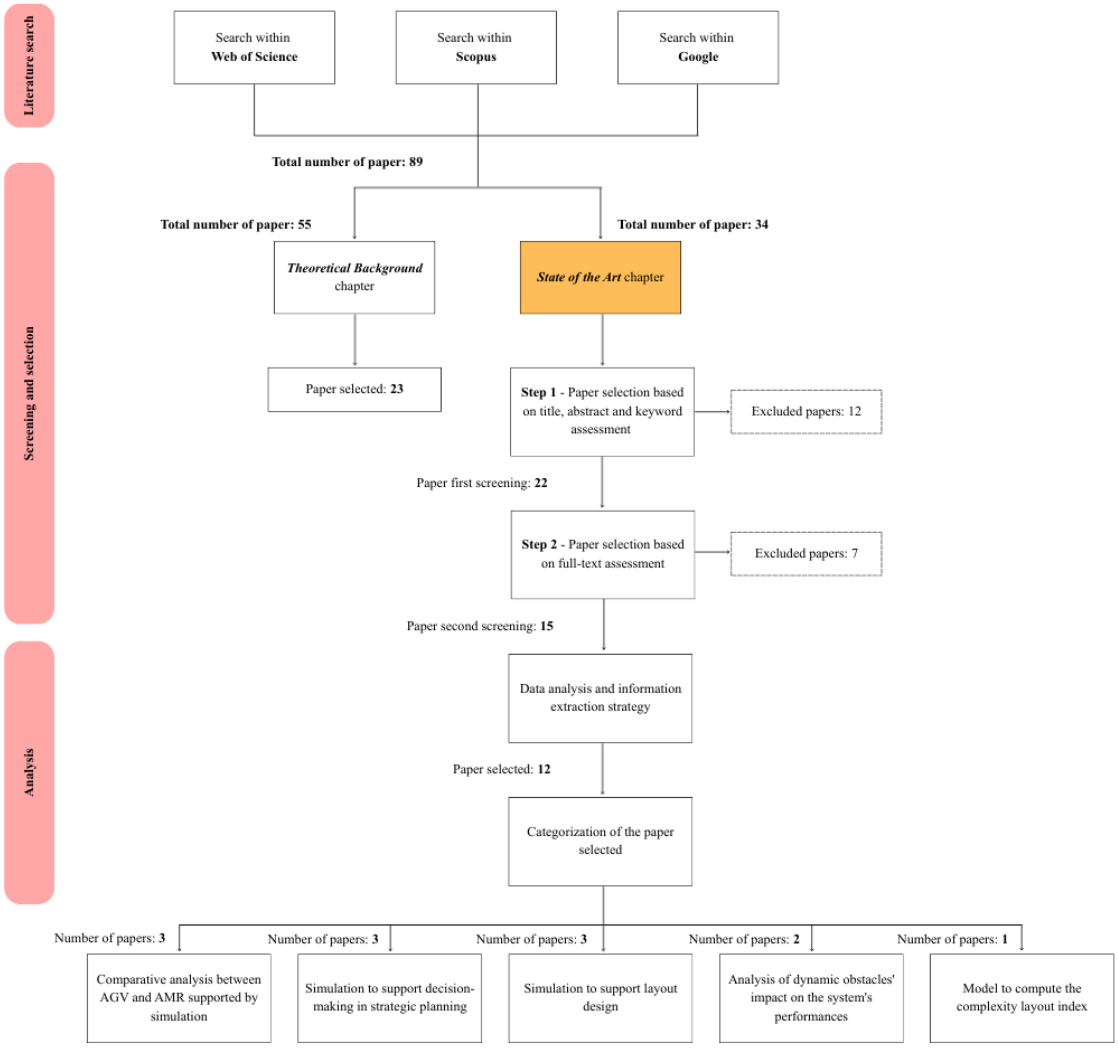


Figure 3.1: Research and selection process

These articles were primarily found in two major databases, Web of Science and Scopus, and the keywords used were the following:

- AGV.
- AMR.

- Simulation in AGV/AMR system.
- Comparative analysis of AGV/AMR using simulation.
- Simulation for facility layout design of AGV/AMR system.
- Moving obstacles in AGV/AMR system.
- Complexity index of a layout.

3.1 Paper Analysis

The second phase of the research process is the critical analysis of the papers selected for the *State of Arts*. To make the analysis more organized, the selected scientific materials were divided into five categories :

1. Comparative analysis between AGV and AMR supported by simulation.
2. Simulation to support decision-making in strategic planning.
3. Simulation to support layout design.
4. Analysis of dynamic obstacles' impact on the system's performances.
5. Model to compute the complexity layout index.

The Tab. 5 presents the papers divided by category and reports the essential information for each paper, such as the title, bibliographic reference number, year of publication, main topic addressed, and the conclusions reached by the authors.

1. Comparative analysis between AGV and AMR supported by simulation

The main goal of the research for this category was to identify studies that perform direct comparative analysis between AGV and AMR systems, specifically in scenarios where the layout complexity increases. Therefore, the objective is to find existing models that connect the type of layout to the system's performance. However, the research revealed a significant gap in the state of the art. Numerous studies are comparing the general performance of the two systems in different layouts, but none of them have analyzed how the results are influenced as the layout complexity changes. Therefore, the focus has shifted towards articles that use simulation to compare the operational performance of the two types of fleet, without considering the layout complexity metrics.

Silva et al. [25] explore the feasibility of the two systems by comparing their navigation performance within the same environment, through four KPIs (time, success rate, collision rate, and average number of collisions), using RViz and Stage simulation tools. The two path planning algorithms used are the TEA* ((Time Enhanced A*) for AGV and ROS Navigation with TEB (Time Elastic Band) planner for AMRs. The results of the analysis show that AMRs have superior time efficiency in dynamic environments, especially as the fleet size increases, while AGVs have a better collision rate due to the predictability of their fixed paths. On the other hand, Bang et al. [26]

addresses the challenge of managing a fleet in limited workspaces. The model proposed, built using FlexSim, is a Virtual Grid Layout with Direction Constraints (VGL-DC). The study involves comparing three scenarios: a conventional rail-based AGV system, AMR without restrictions, and AMR with directional restrictions. The results reveal that the VGL-DC model is the most effective, since it successfully eliminates deadlocks and achieves a 39.5% reduction in the average delivery time compared to the other systems.

Maas Genannt Bermpohl et al. [27], instead, focus on the problem of task allocation. They argue that traditional asynchronous, event-based dispatching methods, typically implemented in AGV systems, are not suitable for AMR systems as they are not capable of supporting high mobility and continuous path changes. The article proposes a dispatching method based on synchronous, timer-based evaluation strategies. Utilizing Agent-Based Simulation rather than the typical Discrete Event Simulation, the study reveals that advanced algorithms like the Stable Marriage Problem (SMP) and Linear Sum Assignment Problem (LSAP) significantly improve vehicle travel efficiency by reducing total travel distance and operational costs. However, the throughput rate did not increase.

Despite their different scopes, they have a series of common points. First, all studies rely on the use of simulation to validate the proposed models and to compare the performance of the two systems. Second, except [25], the studies propose new models or algorithms designed to improve the AMR capabilities, validating their effectiveness by comparing them with traditional AGV systems.

2. Simulation to support decision-making in strategic planning

Within this category, the focus of the papers is the use of simulation as a strategic tool for both management and system design. Simulation is not used only for technical validation, but it can also support decision-making processes in the context of automated material handling, ranging from determining the feasibility of an investment to optimizing fleet size. Unlike the previous category, there is a lack of comparative analysis, since these studies focus on optimizing one specific type of system, either AGV or AMR.

Gebennini et al. [28] propose a Discrete Event Simulation (DES) methodology to support the design and investment decisions for an AGV transport system in an end-of-line (EOL) logistics. Applied to a real-world case study, the study determines the optimal fleet size, the number of bottlenecks, and buffer effectiveness. The performance of different AGV fleet sizes was evaluated through 2 KPIs: mean service time and mean EOL queue size. A similar approach was used by Melo and Corneal [29] to automate the finished goods material flow using AMRs in an automotive supplier plant. The goal was to increase productivity, optimize floor space, enhance safety, and reallocate human labor to higher value-added tasks. Simulation was used to determine the optimal layout among single-loop, conventional, and tandem layouts. The authors prove that the tandem layout is the most beneficial approach in this context with an optimal fleet size of three mobile robots. Rema et al. [30] also addresses the problem of AMR fleet sizing, but in the context of a manufacturing warehouse. This approach combines real-world fleet management software, using the TEA* algorithm to control the robots, with FlexSim software simulation. The simulation modelled 5 scenarios with varying fleet size, and, for each of them, the KPIs measured are task completion capacity and robot utilization efficiency. The optimal fleet size obtained corresponds to 3 robots.

3. Simulation to support layout design

This category explores the application of simulation in the design and optimization of the physical layout of an operating environment. Unlike strategic planning, the focus is on the correlation between the vehicle's performance and the facility layout. Through simulation, the authors demonstrate how modifying the layout is often effective in reducing traffic congestion and the navigability of the system.

Stączek et al. [31] use Digital Twin (DT) technology to support decision-making for the implementation of an AMR system within a production hall. The system is modelled within the Gazebo simulation software, while RViz is used to monitor and visualize the real AMR's behaviour. The study demonstrates that DT technology is a crucial tool for making data-driven decisions, as it helps determine the modifications needed to the physical layout to reduce travel time and bottlenecks, as well as optimize the storage buffer size. On the other hand, Verma et al. [32] propose a model based on Control Zone systems to manage multi-AGV traffic. The core idea is to divide the layout into zones, where the number of vehicles that have access to it is limited. The analysis, made using FlexSim as simulation software, specifically focuses on the performance of two representative traffic management strategies: the Dynamic Resource Reservation (DRR) approach, which relies purely on conflict prevention by restricting shared route access, and the Improved Dynamic Resource Reservation (IDRR) algorithm, which further includes logic for conflict resolution. The study demonstrates that the layout discretization has a great impact on the efficiency of the entire fleet, specifically, they depend on the size and number of the control zone. When there is low discretization, the system TH and overall performance drastically decrease due to high delays, while in the case of high discretization, the system achieves higher performance in terms of costs, but, at the same time, it requires a higher computational burden to manage the increased number of zones in real-time.

Clarion et al. [33] address the problem of determining the optimal physical layout for a Reconfigurable Manufacturing System (RMS), which are inherently flexible system designed to rapidly adapt production capacity and functionality in response to changing market demands. Therefore, this type of system requires a layout that is easy and cost-effective to reconfigure. An interesting aspect touched in their study is the impact of the properties of the workstations, like their dimensions and operational constraints, on material flow efficiency and the overall cost of configuration. To address the challenge of RMS, the authors propose to use the Simulated Annealing (SA) metaheuristic, which allows exploring a vast solution space and avoiding being trapped in a sub-optimal local solution.

4. Analysis of dynamic obstacles' impact on the system's performances

The previous categories addressed static scenarios where the obstacles were fixed and known. On the other hand, this section presents the papers that examine the behavior of AGVs and AMRs in dynamic environments. The presence of moving obstacles, like human workers, is one of the most critical challenges in automated material handling. In this context, the two types of vehicles react in different ways. For traditional AGVs, which are characterized by fixed and predetermined paths, the sudden appearance of a dynamic obstacle on the path constitutes a significant problem, since they are unable to calculate an alternative route to avoid the obstacle. AGVs can only stop and resume traveling once the path is clear. While AMRs are equipped with software that ensure them to detect obstacles and calculate alternative trajectories in real-time.

Most of the papers in the literature regarding this topic address the issue of moving obstacles by proposing new models to improve the existing path and moving planning algorithms. It can be observed that in the context of AGV systems, researchers often analyze modern or hybrid types of AGV, which exhibit certain degrees of movement freedom, rather than traditional fixed-path ones. These systems bridge the gap between rigid AGVs and flexible AMRs, featuring local obstacle avoidance capabilities while adhering to global path constraints.

In this context, Li et al. [34] propose a two-stage hybrid strategy that uses a three-dimensional A* algorithm search to generate a collision-free path, and applies the QCQP (Quadratically Constrained Quadratic Program) optimization to refine and smooth this trajectory for dynamic moving obstacle avoidance in AGV systems that have a degree of free movement. Similarly, Teso-Fz-Betoño et al. [35] suggest a free navigation system for AGVs that allows tracking a non-static target while avoiding obstacles. It consists of adapting the proportional control logic of the "Moving to a Point" technique by integrating a real-time obstacle avoidance function, preventing the AGV from simply stopping. Meanwhile, Yao et al. [36] present a map-based Deep Reinforcement Learning approach to address the issue of AGVs moving in the presence of crowds.

Many of the reviewed papers concerning dynamic AGV systems have developed methods to improve the A* algorithm since it is the most used for path planning, but it is not powerful enough in a dynamic environemt. Therefore, many authors propose coupling it with dynamic obstacle avoidance algorithms, such as the Dynamic Window Algorithm (DWA). Wang et al. [37] demonstrates that, by fusing the two algorithms, it was possible to achieve global path planning and dynamic obstacle avoidance in a dynamic AGV system. The travel time is improved by 26.3%, the path length is 7.2% shorter, and the number of turning points is 71.4% less. Similarly, Li et al. [13] have developed a model that couple A* algorithm with DWA.

On the other hand, in the context of AMRs, Liu et al. [38] propose an obstacle-avoiding controller that combines Fuzzy Logic to make real-time decisions based on sensor data (such as proximity) and Genetic Algorithms (GA) to optimize the Fuzzy Logic control parameters. This generates more robust, safe, and efficient trajectories in multi-obstacle environments by maximizing performance in terms of speed, security, and path fluidity.

However, the objective of the thesis is neither to evaluate the suitability of specific algorithms nor to propose a new mathematical model to address dynamic obstacle avoidance. Instead, the goal is to use simulation to support industrial decision-making in the selection of the mobile robot system best suited to a given facility layout. Among the reviewed papers, there is a noticeable lack of research addressing dynamic obstacles from this perspective. In addition, this thesis concentrates on the performance and limitations of traditional AGVs rather than modern AGVs. However, there is a lack of studies addressing the impact of dynamic obstacles on traditional systems. In fact, it can be observed that the papers that analyze this issue from a managerial perspective are predominantly restricted to the context of AMRs. For example, Cadete et al. [39] addresses the challenge of ensuring safe AMR navigation in environments populated by moving obstacles, such as humans and other vehicles. To solve this, the authors propose a hybrid strategy fusing an enhanced A* algorithm, which was modified to incorporate obstacle trajectory prediction for global planning, with the Artificial Potential Field (APF) method for local collision avoidance. The system was validated within a Gazebo-ROS2 simulation environment through various scenarios involving traffic rules and pedestrian interactions. The performance was measured using specific KPIs: average time to goal, average velocity, number of stops, and the minimum distance to moving obstacles. The authors demonstrate that integrating trajectory prediction significantly improves

both safety and navigational efficiency.

Similarly, Hall et al. [40] propose a coordinated method based on independent optimization to address navigation in a dynamic environment, like a high-capacity warehouse. It consists of separating the decision-making process into two layers: global planning and local avoidance. The global avoidance focuses on global path planning, and it uses a variant of the A* algorithm to find an initial and global shortest path based on the static map of the environment. On the other hand, local avoidance is a function that is executed in real-time, and it focuses on dynamic obstacle avoidance. It allows the mobile robots to not recalculate the global path, but instead it uses a Model Predictive Control (MPC) to modify the immediate trajectory and optimize velocity commands, ensuring local safety. To validate the proposed method, the authors have constructed a virtual environment within Gazebo and RViz simulations to test the system's performance. The results confirm that separating the two optimization decisions leads to safer obstacle avoidance, faster computation time, and more responsive behavior compared to existing coupled methods, which tend to recalculate the path repeatedly when detecting a new obstacle.

5. Model to compute the complexity layout index

Another important aspect addressed in this thesis is the definition of layout complexity. It is crucial because the comparative analysis between AGV and AMR technologies is conducted across various scenarios, each characterized by a different level of spatial difficulty. Therefore, it is necessary to define layout complexity quantitatively through a specific index to ensure an objective comparison between the various scenarios and to determine more effectively the existence of a correlation between the performance of an automated material handling system and the structural complexity of the layout in which it operates.

However, the literature research revealed a lack of relevant studies that address this challenge. Only one paper [41] explicitly proposes a model to calculate the structural complexity of a physical layout, while the vast majority of existing research focuses on process complexity, such as production scheduling, product variety, or assembly sequence difficulty, rather than the physical constraints of the facility itself.

Given the importance of the structural complexity of the layout in the efficiency of a material handling system, ElMaraghy et al. [41] propose the **Layout Complexity Assessment (LCA) model**. It evaluates the layout complexity by measuring the effort required to manage the material flow of a system. The model is structured into three steps:

1. System layout analysis. The physical layout is mapped into a diagram composed of nodes and arrows.

2. Adjacency Matrix Creation. To mathematically define the relationships between these decision points, a square $n \times n$ adjacency matrix is generated (where n is the number of nodes). If a direct material flow exists between two workstations, the corresponding matrix value is "1" (or the specific number of parallel connections); otherwise, it is "0".

3. Complexity Indices Generation. Based on the adjacency matrix, the model calculates six distinct complexity indices: density, path, cycle, decision points, redundancy distribution, and redundancy magnitude. These indices measure the information content of the system, which directly correlates to the difficulty of making flow-related decisions. All indices are normalized on a scale from 0 to 1.

Finally, the six indices are uniformly distributed and plotted in a radar plot to calculate the Layout Complexity Index (LCI).

Chapter 4

Proposed method

The increasing adoption of automated transportation systems for material handling within manufacturing companies is a highly relevant issue due to its significant economic implications. The analysis of the *State of the Art* highlights that the selection of such a system often depends on a series of qualitative considerations, as well as cost-benefit analyses. However, a significant gap has been identified in the literature regarding the study of the correlation between the topological complexity of a facility layout and the performance of AGV and AMR systems.

Given this context, the objective of this thesis is to bridge the gap by proposing a **simulation based method**, designed to support companies in selecting the optimal automated transportation system, between AGV and AMR fleets, based on a comparative performance evaluation as the layout complexity increases. Specifically, the proposed method analysed two specific scenarios: static and dynamic environments. In the **static scenario**, the two systems are analysed under ideal conditions, where it is assumed the absence of unforeseen obstacles. This scenario represents the baseline to analyse the real impact of the layout complexity on the performance of the fleets. While in the case of a **dynamic environment**, the systems are subject to the presence of obstacles that can block the path of the vehicles. The objective here is to test the flexibility of both systems under unpredictable conditions, and to identify which technology's performance is more impacted as the layout complexity increases.

The proposed method is composed of the following **four sequential phases**, as illustrated in the Fig 4.1.

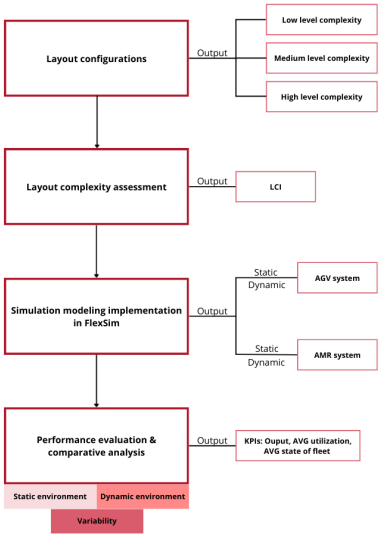


Figure 4.1: Sequential steps of the proposed method

The first phase, **layout configurations**, involves the design of three distinct layout configurations of the same manufacturing environment characterized by increasing levels of complexity: low, medium, and high. The **lowest level** is characterized by simple flows of material with minimal interactions. The **medium**, instead, features mixed flows with moderate alternative routing options. Lastly, in the **highest level** of complexity, the layout presents a highly connected network with redundant paths and intersections.

The second phase, **layout complexity assessment**, aims at providing a quantitative measurement of the topological complexity characterizing the three configurations designed in the previous step. This phase ensures that the complexity classification assigned to the layouts is validated through objective, quantitative, and mathematical metrics. An adaptation of the **Layout Complexity Assessment** (LCA) **model** proposed by ElMaraghy et al. [41] is applied. In this model, the physical layouts are converted into graph-based representations to calculate the **Layout Complexity Index** (LCI), which results from the combinations of 6 other indices (density, path, cycle, decision point, redundancy distribution, and redundancy magnitude).

The next phase is **simulation modeling implementation**, which consists of converting the conceptual layout configurations within the FlexSim simulation environment. The main objective is to model the operational behavior of both AGV and AMR fleets, ensuring that it represents a real manufacturing environment. This step involves designing the respective navigation control logic for each system under both static and dynamic conditions. Simulating before the real implementation makes this model a key tool for decision-making, allowing companies to test the system's behavior in a safe, virtual environment.

Following the simulation implementation is the **performance evaluation** and **comparative analysis** between the two systems. Since several parameters are assumed to be non-deterministic, this phase involves processing data through the FlexSim **Experimenter** tool, which allows the execution of multiple simulation runs to take in considerationthe system variability. The goal is to collect and compare a set of **Key Performance Indicators** (KPIs) to identify a correlation between the variation in the system performance and the increase in layout complexity, in both static and dynamic scenarios, while also considering the stochastic nature of the system.

4.1 Layout complexity assessment

For the computation of the layout complexity index, a modified version of the **LCA model** proposed by ElMaraghy et al. [41] is adopted. While the original model was designed for manufacturing flows, the proposed model of this thesis introduces specific adaptations to address the case of automated material handling systems.

The objective of the LCA model is to introduce metrics for assessing the structural complexity of manufacturing systems' layout by defining the characteristics and flow patterns that contribute to the complexity of decision-making during system operations [41].

The LCA model develops a graphical representation of the physical manufacturing system layout and generates measurable complexity indices based on the number, locations, and connectivity of decision points within the system layout, and they are used to evaluate and compare the structural complexity of various layout alternatives [41].

The proposed methodology is composed of three steps, as shown in the following Fig. 4.2.

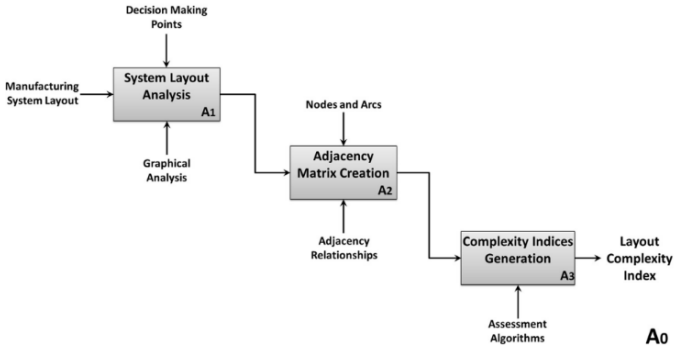


Figure 4.2: IDEF0 representation of the LCA model by ElMaraghy et al. [41]

In the first step, **system layout analysis**, the physical layout is analyzed and converted into an oriented, unweighted graph diagram with nodes and arcs. The objective is to identify where the decisions are made based on the connections between different manufacturing system components, areas, and departments. In this graph representation, for ElMaraghy et al. [41], the **nodes** represent the points where decisions regarding material flow direction and destinations are made [41], while **arcs** represent the connection and direction of material flow existing between nodes. The definitions of these elements have been adapted specifically for the context of this thesis. The nodes represent

any point where a vehicle stops to load or unload an item, or intersections where the path splits, requiring the vehicle to select a specific route among the alternatives. On the other hand, arcs represent the vehicle navigation network, defining the specific paths where AGVs and AMRs can travel within the layout to reach their destinations.

In addition to standard nodes, the LCA model adds two specific types of extra nodes to the graph [41]: system input and output nodes. These nodes allow identifying all the distinct paths that connect an input node to an output node. In the proposed model, they represent the parking spot, where the vehicles return to charge their batteries, or once they have completed all their orders. Within one system layout, it is possible to have several input and output nodes [41].

The second stage, **adjacency matrix creation**, consists of creating an n x n adjacency matrix, where n is the number of nodes, based on the graph representation created in the previous step. The purpose of this matrix is to mathematically generate the various complexity indices in the LCA model. The values within the adjacency matrix cells correspond to the actual number of connections between two nodes of the graph. For example, a value of "1" indicates that a single arc is connecting two nodes, while a value of "0" means the absence of connections. In cases where multiple parallel connections exist between two nodes, the exact number of arcs is recorded in the respective matrix cell [41].

The final step is the **generation of the layout complexity indices**. During this phase, six distinct complexity indices are calculated and then integrated into an overall **Layout Complexity Index** (LCI). This index allows the comparison between various layout design alternatives. These six indices are calculated based on graph theory, and they quantify the information content that impacts the difficulty of decision-making within the system layout. In addition, they are normalized, meaning that their values range from 0 to 1 [41].

The aggregation into a single metric is obtained by plotting the six normalized indices on a radar plot, assuming equal weights for each index [41].

The LCI formulation is given in the following way (Eq. (4.1)):

$$LCI = \left(\sum_{i=1}^{n} C_i \right)^2 - \sum_{i=1}^{n} C_i^2 \tag{4.1}$$

where C_i represents the individual complexity index value, and n is the number of indices considered. This formulation overcomes the sequence dependence drawback, ensuring that the value of LCI is not influenced by the order in which the indicators are distributed in the radar plot [41].

4.1.1 Calculation of complexity indices

The six complexity indices computed in the LCA model [41] are the following: density index, path index, cycle index, decision points index, redundancy distribution index, and redundancy magnitude index. Among these, only the computation of the cycle index has been modified.

Since the manual computation of all the indices' parameters is not feasible due to the scale and interconnectivity of the layout configurations, they were computed using a custom MATLAB script.

1. Density index

The density of a layout graph is defined as the ratio of the actual number of unique connections to the theoretical maximum number of unique connections given n nodes (Eq. (4.2)), obtained by assuming a fully connected graph where all pairs of nodes are linked [41]. This index considers unique connections, meaning that multiple arcs between the same pair of nodes are counted as a single connection, since redundancy is specifically measured by other two indices, redundancy distribution and magnitude indices.

$$D = \frac{k}{n(n-1)} \tag{4.2}$$

The parameter k represents the actual existing number of unique connections, and n is the actual number of nodes present in the graph representation. A high value of density index indicates a more complex system because of the relative increase in the number of unique connections compared to the number of nodes in the system layout [41].

The values of the parameters k and n were calculated using the following MATLAB code:

Listing 4.1: Calculation of k and n

```
% Adjacency matrix
% Definition of source (s) and target (t) vectors
s = [node_source_1, node_source_2, ..., node_source_m];
t = [node_destination_1, node_destination_2, ..., node_destination_m];
%vector with name of the nodes
name_nodes = {'name_1', 'name_2', ..., 'name_m',    }
% number of nodes
n = length(nomi_nodi);          % Number of nodes
%Creation of the weighted adjacency matrix A
A = accumarray([s(:), t(:)], 1, [n, n]);
%Creation of the binary version of the adjacency matrix A
A_bin = double(A > 0);
%Number of unique arcs
k = nnz(A);
```

As shown in the code, the number of arcs and nodes is extracted starting from the adjacency matrix, which is created in three steps. First, the connection between the nodes is manually defined using two vectors: the source vector (s) and the target vector (t). Each pair (s_i, t_i) represents a directed arc in the graph. To generate an $n \times n$ square matrix, the `accumarray` function is used. This command is relevant to manage system redundancies automatically. Whenever a specific pair of nodes appears multiple times, which indicates the presence of parallel physical connections, the function sums these occurrences. It assigns a weight greater than 1 to the corresponding matrix cell. In parallel, a binary version of this matrix, called `A_bin`, is created to calculate unique paths without the influence of redundant connections. In this matrix, the cells take the values 0 or 1 depending on whether a connection exists between the nodes.

Once the adjacency matrix is defined, the number of arcs is calculated using the `nnz(A)` function, which provides the number of non-zero elements within the matrix A, which corresponds to the total number of unique connections. While the number of nodes is calculated using the `length(name_nodes)` function, which measures the size of the `name_nodes` vector.

2. Path index

A path is defined as the sequence of nodes that connects the input nodes to the output nodes, without crossing the same node twice. As the number of alternative paths increases, the number of decisions required to control the flow within the system increases, thereby influencing layout complexity [41].

The path index compares the minimum theoretical number of unique paths to connect the input and output nodes to the actual number of unique paths present in the graph, as illustrated in Eq. (4.3).

$$P = 1 - \frac{p}{N} \tag{4.3}$$

In this equation, p is the minimum theoretical number of unique paths, which is calculated as the product of the number of input nodes and the number of output nodes (*input nodes x output nodes*), and N represents the actual number of existing unique paths [41].

The parameter N has been calculated starting from the binary matrix `A_bin`, as illustrated in in the code snippet 4.2. The total number of unique paths is obtained thorugh a recursive search function (`find_all_paths(A_bin, input_node, output_node)`), based on the Depth-First Search (DFS) algorithm, which explores the graph starting from the `input_node` and follows every possible branch until it reaches the `output_node`. Since, by definition, a path is a sequence of nodes where a node is not repeated, the function `find_all_path` keeps track of the nodes that have already been visited within each branch. The final output of this function is a cell array of vectors (`unique_paths`), where each of them represents a unique sequence of nodes. From this list, the total number of unique paths is calculated using the function `length(unique_path)`.

Listing 4.2: Calculation of the number of N

```
input_node= node_n;
output_node= node_n;
%Binary matrix
A_bin = double(A > 0);
% recursive search function
unique_path = find_all_paths(A_bin, input_node, output_node);
% Total number of unique paths (N)
num_unique_path= length(unique_path);
```

3. Cycle index

A cycle is defined as a loop of nodes that starts and ends at the same nodes. It contributes to the structural complexity of a facility layout since the flow does not follow a linear sequence, and it increases the difficulty of following the vehicle navigation network across the system layout [41].

In the LCA model proposed by ElMaraghy et al. [41], the cycle index is calculated as shown in Eq. (4.4).

$$CL = \frac{C}{MC} \tag{4.4}$$

where C is the actual number of cycles and MC is the theoretical maximum number of cycles, calculated as illustrated in Eq. (4.5) [41].

$$MC = \sum_{i=2}^{n} C_i^n \tag{4.5}$$

C_i^n is the combination of n nodes starting with two nodes, since at least a pair of nodes is needed to have a cycle.

However, this formulation has been considered not suitable for calculating the cycle complexity of a physical facility layout for two main reasons. First, the denominator MC takes into consideration all possible combinations of nodes that can exist within a graph, but, in a real industrial environment, certain flow combinations cannot exist due to physical constraints of a system, or they are practically superfluous. Therefore, MC leads to a significant overstimation of the theoretical number of cycles.

The second reason is that MC grows exponentially as the number of nodes n increases, causing the cycle index value to assume a value close to zero. In this way, the information regarding the impact of cycles on the final layout complexity index would be lost.

To overcome these critical issues, a different formulation was adopted, and it is called the **Alpha index** (α) [42], which is specifically designed for planar graphs, which are networks drawn on a 2D plane where edges do not cross each other. With this formulation, the maximum theoretical number of cycles is no longer given by a combination, but it is limited by the physical geometry of the graph. It measures the network connectivity independently of the number of nodes, reflecting the true complexity of the system and solving the issue of the other formulation.

The Alpha index is defined as the ratio between the actual number of independent cycles and the maximum possible number of independent cycles in a planar graph, as shown in Eq. (4.6).

$$\alpha = \frac{u}{2v - 5} \tag{4.6}$$

The numerator u represents the actual number of independent cycles. An independent cycle is a fundamental topological loop that can describe all possible circular routes in the system without geometric redundancy. It is calculated in the following way (Eq. (4.7)):

$$u = e - v + p \tag{4.7}$$

In this formulation, the parameters follow the original nomenclature of the Alpha index, but they correspond to the parameters previously defined and calculated in the density analysis. Specifically, v is the number of nodes, previously called n, and e is the number of unique arcs, also known as k. The parameter p represents the number of isolated parts of the network, known as sub-graphs. In case the network is fully connected, p is equal to 1.

On the other hand, the denominator, $2v - 5$, represents the theoretical maximum number of independent cycles that would be formed if the highest possible number of arcs were inserted among the v nodes while avoiding any arc intersections.

4. Decision points index

The decision point index represents the cumulative complexity of decision-making, which increases with the number of nodes in a path [41]. The index is calculated using Eq. (4.8).

$$DS = 1 - \frac{SP}{LP} \tag{4.8}$$

In this equation SP is the number of nodes on the shortest path, representing the theoretical minimum number of decision points in the layout graph, while LP is the number of nodes on the longest path, representing the actual number of decisions made per path.

These two parameters can also be obtained using a MATLAB implementation shown in the code snippet 4.3. Once all the unique paths have been found using the function `find_all_paths()`, the algorithm processes each path to determine its length by counting the number of unique nodes it contains. To identify the shortest and longest path, these lengths are iteratively compared, and each time a new minimum or maximum value is found, it is stored in the variables `min_len` (SP) and `max_len` (LP), respectively.

Listing 4.3: Calculation of SP and LP

```
% Initialize length variables
min_len = inf;
max_len = 0;

for i = 1:length(unique_path)
    p = unique_path{i};
    current_length = length(p);

    % Adjust length in case of cycle
    if p(1) == p(end), current_length = current_length - 1; end

    % Update minimum and maximum length
    if current_length < min_len, min_len = current_length; end
    if current_length > max_len, max_len = current_length; end
end
```

5. Redundancy distribution index

Redundancy implies the existence of alternative paths between two locations, which increases the complexity of decision-making. Specifically, the redundancy distribution index compares the number of adjacent node pairs connected by more than one arc to the total number of connected pairs in the graph, as shown in Eq. (4.9).

$$RD = 1 - \frac{r}{a} \tag{4.9}$$

where r is the actual number of adjacent node pairs that are connected with more than one arc. This parameter is calculated using the following MATLAB code snippet 4.4. The function `find(A)` extracts all existing connections from the weighted adjacency matrix. r corresponds to the length of the vector containing all connections with a weight greater than 1.

Listing 4.4: Calculation of r and a

```
a = nnz(A); %calculation of a
%calculation of r
```

```
[r_idx, c_idx, v] = find(A);
idx_multiple = find(v > 1); %connection with weight greater thatn 1
r = length(idx_multiple);
```

On the other hand, a is the existing total number of adjacent nodes connected. It is calculated using the MATLAB function `nnz(A)`, which counts all elements in the adjacency matrix A with a value different from zero. These elements correspond to the pairs that are connected by at least one arc.

6. Redundancy magnitude index

The redundancy magnitude measures the number of redundant parallel forward arrows in the system layout, and it is calculated using Eq. (4.10).

$$RM = \frac{pr}{w} = \frac{w - a}{w} \tag{4.10}$$

where pr is the total number of redundant forward parallel arrows, w represents the total number of forward arrows, and a is the number of adjacent nodes connected.

The parameter w is obtained using the MATLAB function `sum(A(:))`, which sums all the weights within the adjacency matrix. This is done because the weights correspond to the actual number of arcs connecting each pair of adjacent nodes.

4.2 Simulation modeling strategy

Once the three layout configurations have been designed and their complexity evaluated, the next step of the proposed method is to model the AGV and AMR systems within these layouts. This phase is implemented using FlexSim, a simulation software equipped with specific features to manage the dynamics of automated material handling vehicles. Therefore, this section illustrates how FlexSim can be used to simulate the behavior of AGVs and AMRs within the layouts designed in the previous step.

4.2.1 AGV system modeling strategy

FlexSim provides several pre-built process flow templates to simulate the behavior of AGV fleets. Specifically, there are five progressive types of process flow templates: Basic AGV, AGV with Work Forwarding, AGV with Basic Parking, AGV with Heuristic Parking, and Advanced AGV [43]. For the purpose of this study, the **Advanced template** was selected because it is the most complete version, since it includes all the features of the other four. It can be easily added to the model by selecting it from the "Process Flow" toolbar. Once implemented, this pre-built template allows the user to manage the AGV logic across four fundamental levels, as summarized in the Table 6.

These pre-built process flows operate exclusively on top of a predefined AGV network built within the 3D model. The physical paths are created using the following objects from the library [43]:

- `Straight` and `curved paths` are used to define the directions and the lengths of the paths.
- `Control points` are considered as the operational nodes of the network. They represent the specific locations where an AGV can stop, make routing decisions, and physically interact

Level	Key Question
Transportation or travel	How do task executers move around?
Task sequence generation	How are task sequences defined?
Job dispatching	Who or what will perform what jobs?
Item flows and routing	Where do the items need to go?

Tab. 6 - AGV - four level of logic of the advanced process flow [43].

with the surrounding 3D environment to load or unload items. There four main types of control points: next work point, pick up point, drop off point, and park points.

- `Control area` are used to manage complex layouts and prevent collisions at intersections by limiting the number of vehicles that can occupy a specific area of the layout.

To make the process flow work, the control points are connected using `Next Look For Work` connections to form a continuous loop. If there are no items waiting in a control point, the AGV simply moves to the next one in the loop. This is how the vehicle searches for work: it continuously travels around the network until it finds an available task. In other words, jobs are assigned to the vehicles based on their current location in the layout [43].

Regarding the **job dispatching**, it is handled by a combination of control points, the AGV process flow template, and a global list named `AGVWork`. In FlexSim, a global list acts as a centralized database accessible from anywhere in the model, used to queue and manage tasks. When a fixed resource, such as a queue, needs to request an AGV transport for a flow item, it pushes the item to the `AGVWork` global list. Here, the item pushed onto the list represents the "job." The process flow template immediately pulls this job, identifies the item's location, and pushes it back onto the `AGVWork` list, but this time partitioned by the specific work point associated with that item. When AGVs arrive at these work points, they pull from that specific partition to check for available work. This logic can be easily customized by modifying the pull query activities within the process flow [43].

An advantage of the job selection implemented in this process flow is that when a new job arrives, it is not instantly assigned to a specific vehicle. Instead, it waits for an AGV to arrive at its designated work point. In this way resource selection is determined by the first AGV to reach that location, generally ensuring the job is picked up by the closest vehicle. This delay allows the system to assign jobs to the vehicles more efficiently [43].

The final level, item flows and routing, is not handled directly by the process flow template and can be personalized based on system requirements. In this proposed method, the routing of the items corresponds to the specific material flow being studied. To effectively manage this routing, a label named `Input_type` was created in the queues to differentiate the various items. This label tells the AGV which processor that requires that specific input to begin its operations.

4.2.1.1 AGV advanced process flow

The Advanced process flow template is structured into six main functional sections, as illustred in the Fig. 4.3 [43]:

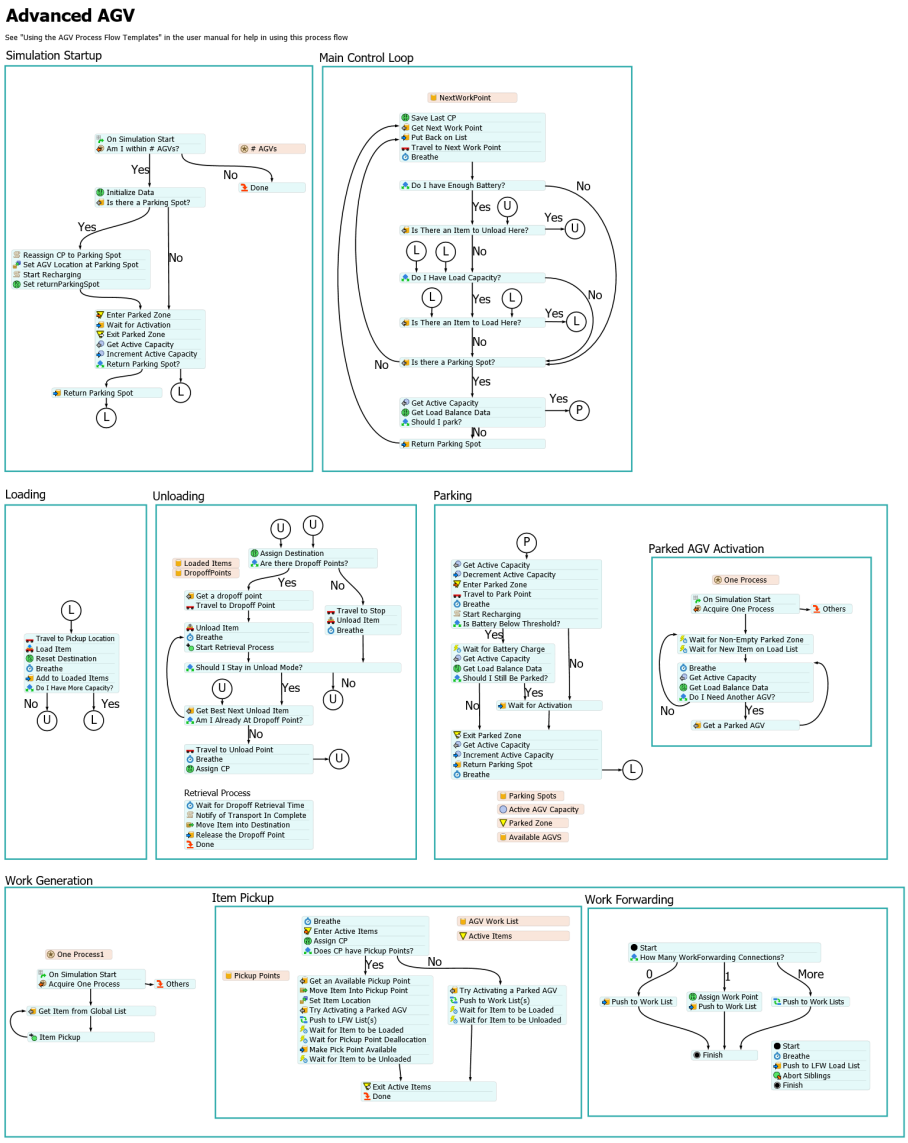


Figure 4.3: AGV advanced provess flow template

1. **Simulation Startup**. It initializes the variables and the lists required for the logic to function.

2. **Main Control Loop**. It represents the core body of the entire process flow, where the logic evaluates a series of conditions depending on the AGV's position in the network, such as battery levels, remaining vehicle capacity, and the presence of items to load or unload. If any of these conditions are met, the process enters the corresponding section to complete the activity. To improve standard routing, this template introduces a smart parking logic and an automated

battery management system compared to the other templates. The system continuously compares the number of transport requests with the total capacity of the currently active AGVs. When a vehicle finishes its job, it evaluates whether the remaining active fleet is sufficient to handle the remaining tasks; if so, the idle AGV will park. On the other hand, if the demand exceeds the active capacity, the system automatically reactivates a parked vehicle. Furthermore, the process flow monitors energy levels: if an AGV's battery drops below a predefined recharge threshold, it is immediately sent to a parking location to recharge until it reaches the resume threshold [43].

3. **Loading**. The process enters this section when there is an item to load at the current point, allowing the AGV to load items up to its maximum capacity.

4. **Unloading**. The process enters this section once the AGV has arrived at its final destination to drop off the items.

5. **Parking**. The process enters this section when the AGV is empty and cannot find any new tasks. Therefore, the system directs the vehicle to an available parking spot to wait for future jobs.

6. **Work Generation**. This section provides a sophisticated method for AGVs to find jobs. The logic uses the WorkForwarding control point connection to "forward" work from an item's location to one or more distant work points. This allows the AGV to see whether there is available work at a location without having to physically travel there first [43].

4.2.1.2 Dynamic scenario: Agent system implementation

In this study, the AGV system is analyzed under both static and dynamic conditions. In dynamic environments, the AGV is not capable of avoiding unforeseen obstacles. Instead, it is forced to stop if its path is blocked, and it resumes its activity only once the obstacle is removed. This behavior is modeled in FlexSim using the **agent system** tool.

The agent system gives the vehicles, defined as agents, the ability to sense their surroundings and interact with objects that are not connected to the regular AGV network. It allows the user to set specific behavior rules based on distance, which are reffered in FlexSim as `proximity behavior`. In the proposed method, two detection phases were implemented: a broad phase and a narrow phase. These phases define two different safety zones around the vehicle, each triggering a specific reaction. The main difference between them is the activation radius. In fact, the broad phase uses a larger radius. When an obstacle enters this zone, the vehicle decreases its speed, but it does not completely stop yet. This recreates real industrial scenarios, where AGVs decrease their speed and emits acoustic warnings when approaching an obstacle, which is often a human operator, to signal their presence. On the other hand, the narrow phase uses a smaller radius. In this context if an obstacle enters this delimited safety zone the vehicle stops completely, since at this distance the probability of a collision is much higher.

Both behavior rules rely on the concept of *preemption*. This means that when an obstacle is detected, the vehicle interrupts its current activity to give priority to the safety behavior (slowing down or stopping). Once the obstacle moves away, the preemption ends, and the vehicle automatically resumes its path to complete the original task. Technically, this proximity detection mechanism calculates the distance by measuring the space from the center of the AGV agent to the closest point on the bounding box of a neighboring obstacle agent [44].

To implement these proximity behavior within the FlexSim environment, specific scripts were created for both the broad and narrow phases. These scripts define exactly how an AGV should behave when an obstacle enters its safety zone, and how it should resume its activities when the path is clear again.

For the implementation of the **broad phase**, two set of rules were defined: `Active narrow phase` and `Deactivate narrow phase`. The `Activate narrow phase` rule, shown in the Fig. 4.4 corresponds to the trigger `On enter proximity`, which means it is activated when an obstacles enter the larger safety radius. When this happens, the exact position relative to the AGV of the obstacles, referred to as *neighbor agent*, is calculated. This calculation is important to prevent the vehicle from stopping for obstacles that are behind or beside it. Therefore, the rule checks if the obstacle is inside a specific frontal vision cone, whose angle can be adjusted based on the operational circumstances.

An important detail of the script is the condition `Type == 1`, which allows the AGV to react only if it detects neighbur agent with this specific label. If it is equal to 1, it means that they are valid obstacles; therefore the vehicle slows down and changes its color to yellow for visual feedback, and the narrow phase detection is activated. In other cases, for example, if another AGV enters the radius, the proximity rule ignore it, since its label is equal to zero. In fact, the collisions avoidance between vehicles is directly managed by the AGV network.

```
/**Activate Narrow Phase*/
Agent agent = param(1);
Object obj = agent.object;
Vec3 size = obj.size;
Agent neighbor = param(2);

Vec3 loc = neighbor.object.getLocation(0.5, 0.5, 0.0).project(neighbor.object.up, obj);
loc += Vec3(0.0, 0.5 * size.y, -0.5 * size.z);
double direction = radianstodegrees(atan2(loc.y, loc.x));
if (direction > 50 || direction < -50 || time() == 0)
    return 1;

obj.slow = 1;
if (neighbor.object.Type == 1) {
    obj.color = Color.yellow;
    agent.activateBehavior("Narrow Phase");
    TaskSequence ts = TaskSequence.create(obj, 0, PREEMPT_ONLY);
    ts.addTask(TASKTYPE_DELAY, 0, 0, 0);
    ts.dispatch();
}

```

Figure 4.4: Agent system - Activate narrow phase rule

On the other hand, the `Deactivate narrow phase` rule (Fig. 4.5)corresponds to the `On exit proximity` trigger, which is activated when the obstacles leaves the delimited zone. In this situation, the vehicle restores its original speed, turns green, and resumes its transportation activity and the rule deactivates the narrow phase.

```
/**Deactivate Narrow Phase*/
Agent agent = param(1);
Object obj = agent.object;
Agent neighbor = param(2);
int numCollisions = param(3);
//
if (obj.slow? == 1 && numCollisions == 0) {
    obj.slow = 0;
    obj.color = Color.green;
    agent.deactivateBehavior("Narrow Phase");
    TaskSequence ts = TaskSequence.create(obj, 0, PREEMPT_ONLY);
    ts.addTask(TASKTYPE_DELAY, 0, 0, 0);
    //ts.addTask(TASKTYPE_TRAVEL, 0,0,0);
    ts.dispatch();
}
```

Figure 4.5: Agent system - Deactivate narrow phase rule

The implementation of the **narrow phase** applies a similar logic but it enforces stricter rules to prevent imminent collisions. For the `On enter proximity` trigger, the active rule is called `Preempt AGV`. In this case, the frontal vision cone is reduced, and, if an obstacle is detected within this tighter radius and angle, the rule forces the vehicle to stop completely. This is achieved by creating a preempting task sequence with a high priority (`PREEMPT_ONLY`), using the `TASKTYPE_UTILIZE` command to block all AGV movements until the obstacle is removed from its path. In this state, the vehicle color is also changed to red, as showns in the Fig. 4.6.

```
/**Preempt AGV*/
Agent agent = param(1);
Object obj = agent.object;
Vec3 size = obj.size;
Agent neighbor = param(2);


Vec3 loc = neighbor.object.getLocation(0.5, 0.5, 0.0).project(neighbor.object.up, obj);
loc += Vec3(0.0, 0.5 * size.y, -0.5 * size.z);
double direction = radianstodegrees(atan2(loc.y, loc.x));
if (direction > 30 || direction < -30 || time() == 0)
    return 1;

if (neighbor.object.Type == 1){
    obj.color = Color.red;
    TaskSequence ts = TaskSequence.create(obj, 1, PREEMPT_ONLY);
    ts.addTask(TASKTYPE_UTILIZE, NULL, NULL, STATE_BLOCKED);
    ts.addTask(TASKTYPE_DELAY, NULL, NULL, 5.0);
    ts.dispatch();
}
```

Figure 4.6: Agent system - Agent preempt AGV rule

When the obstacle is removed, the *On exit proximity* trigger is activated. In this case, the rule, called ***Free AGV*** (Fig. 4.7), executes a `freeoperators` command, which releases the AGV from the preempted state and allows the vehicle to safely resume its original transportation task. But the AGV does not resume immediately its activity, but wait for 5 seconds in case a new obstacle appears again.

```
/**Free AGV*/
Agent agent = param(1);
Agent neighbor = param(3);

agent.object.color = Color.orange;
freeoperators(agent.object, 0);
```

Figure 4.7: Agent system - Agent free AGV rule

4.2.1.3 AGV modeling assumptions

After establishing how the vehicles navigate and interact with the environment through the FlexSim logic, it is necessary to define the physical and operational constraints of the models. Therefore, this section illustrates the fundamental modeling assumptions applied in this study to ensure a realistic and standardized simulation of an AGV system.

To accurately build the simulation model, a specific type of AGV was selected as the reference vehicle. Specifically, the model is based on **lightweight AGV with floor-guidance navigation**. This type of vehicle loads and unloads the transported goods in just a few seconds. This high efficiency is possible because the transfer can be performed using motorized coupling systems. Based on this reference technology, several operational hypotheses were established.

First, the carrying **capacity** of the vehicles is assumed to be unitary. Regarding the **routing logic**, the physical network was designed to be strictly unidirectional, with a constant distance between control points. The kinematic behavior of the fleet, instead, is characterized by **constant acceleration and deceleration rates**. The travel speed is also assumed to be constant, but it can be dynamically reduced when the agent system detects an obstacle along the path. When this happens, the speed reduction is immediate, rather than dropping gradually over time.

However, the FlexSim AGV network requires a detailed categorization of the speeds based on the vehicle's direction and the geometry of the path. Specifically, the model distinguishes between forward speed and reverse speed. For both directions, specific limits are defined depending on the track type:

- **Straight speed** is applied on main linear paths.
- **Curved speed** is used on curved segments, typically used for deviations or reaching workstations.
- **Spur speed** limits the vehicle when traveling on secondary tracks or branch lines.

In addition to linear movements, there are specific **rotation thresholds** that define how the vehicle should move at intersections. The model defines a rotation speed for when the vehicle turns on its own vertical axis. It also includes a stop and rotate threshold, representing the minimum path angle that forces the vehicle to stop completely and rotate in place, rather than curving while moving. All of these parameters are assumed to be constant through the entire simulations.

Regarding the physical interactions with the layout stations, the loading and unloading times were modeled as probabilistic functions to introduce a natural variability into the system. This choice aims to recreate a more realistic industrial scenario, where these values are usually not deterministic and can change due to minor inconveniences, thereby introducing a natural variability into the system.

Finally, the simulation runs for a standard 8-hour working shift. During this time, FlexSim allows monitoring the battery levels by calculating two different types of energy consumption: active use (when the vehicle is moving) and idle use (when the vehicle is parked and waiting). To keep the simulation simple, it is assumed that the complete battery recharge happens only at the end of the day. This is a realistic assumption, as industrial AGVs are typically equipped with large-capacity batteries capable of lasting the entire shift without the need for intermediate charging.

4.2.2 AMR system modeling strategy

Unlike for AGV systems, FlexSim does not provide a pre-built process flow to model AMR systems. However, the capability of an AMR to move freely in the environment without relying on any fixed paths can be recreated using the **A* navigation systems**.

This tool allows any connected task executer to dynamically calculate the shortest path between two locations while avoiding obstacles. To achieve this, the algorithm divides the model into a customizable grid of nodes, where each node specifies the allowed direction. The algorithm evaluates the nodes in the direction of the destination to determine the fastest route, which can also include diagonal movements [45].

The following Fig. 4.8 shows an example of a grid that has several barriers in place [45].

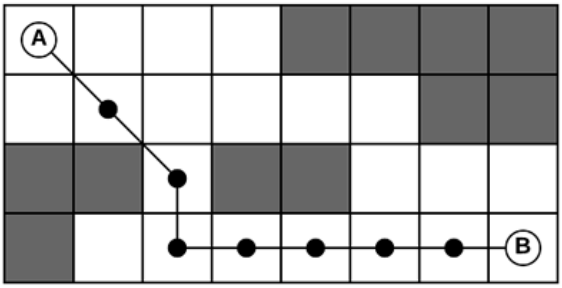


Figure 4.8: Example of path in A* navigation system

In this study, the AMRs cannot move completely freely within the entire layout. Therefore, to adapt the A* algorithm to the specific facility, the default node grid was modified by introducing custom barriers. The vehicles can travel only in designated corridors and do not have access to restricted areas where the machineries are located.

In addition, to manage the physical interactions with the layout stations, the A* system uses specific travel thresholds. These thresholds define the exact entry points a vehicle must reach to interact with a fixed resource. As shown in the Fig. 4.9 below, an object's travel threshold consists of two main zones [45]:

- Calculated path zone (red zone). It represents the physical boundary of the object within the grid. The algorithm uses this area to calculate the final steps of the route toward the destination.
- Travel arrival zone (blue zone). It defines the exact area where the vehicle is considered to have successfully reached the object. Once the AMR enters this zone, the travel task is complete.

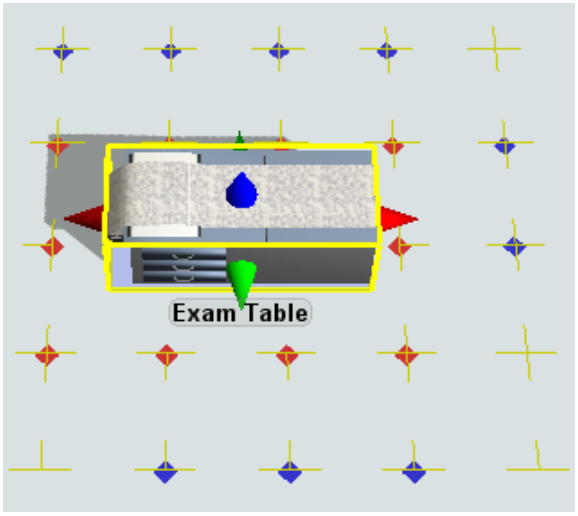


Figure 4.9: FlexSim A* navigation - object's travel threshold

This travel threshold, therefore, allows the AMR to avoid obstacles easily, without the need to implement an agent system, like in the AGV implementation. The most important thing to do is connecting correctly all the objects and task executer that are part of model to ensure that the AMR is capable of sensing their presence within the A* navigation system. This is done by simply going in the `Member panel` of the A* navigator section and click on the `Sampler` button which allows the user to select all the 3D objects to connect to the navigator.

The A* navigation systems allows also avoiding collisions between other vehicles. This is possible by checking the option `collision avoidance`. When checked, travelers will allocate nodes in their path and deallocate them once passed, preventing two travelers from traveling through the same nodes at the same time.

While the A* module efficiently handles spatial navigation, it is not sufficient on its own to fully manage an AMR fleet. To control advanced traveling patterns and the job assignment logic, it is required a custom process flow.

4.2.2.1 AMR control logic and custom process flow

The modeling logic applied to the AMR system is conceptually similar to the AGV system. Therefore it is structured across four control levels: item flow, task sequence, job dispatching, and

transportation. However, unlike the AGV system which relied mainly on a single global list, the AMR logic is more articulated and relies on multiple global lists to track the exact state of the orders:

- `Component_list` contains all the items ready to be transported to the next machine or to the shipping area.
- `AMRWork_Request` contains the transport requests generated by the departments.
- `AMRWork_Transported` tracks the orders that are currently being transported by an AMR.
- `AMR_Released` tracks the orders that have been successfully delivered.

To efficiently manage these lists, the control architecture is divided into three separate process flows: Input Generation, Task assignment, and Fleet management.

The **Input Generation flow** (Fig. 4.10 is responsible for creating the 3D items entering the facility, and define which are the items ready to be delivered. To each of them, it assigns specific labels, such as its current location (`Load`), the ID of the queue (`Load_Queue`), and specific category of the item (*Input_ type*), and then pushes them into the `Component_list`.

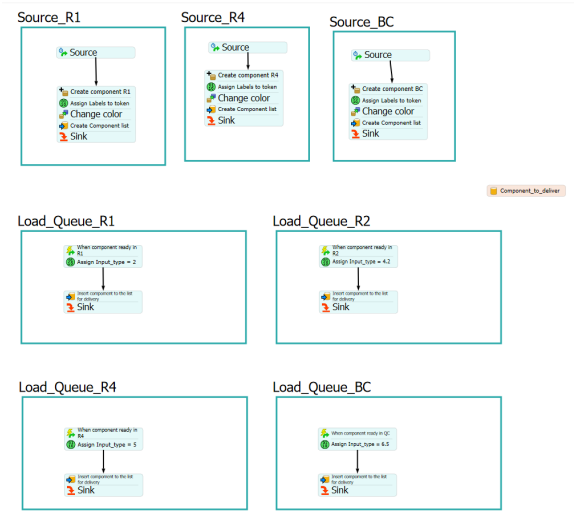


Figure 4.10: AMR process flow - Generation of input

The **Task Assignment** (Fig. 4.11) process flow distributes these items among the departments. Each department have a pick-up queue, which contains the output ready for transport, and a drop-off queue, which, instead, receives the necessary inputs. The drop-off queues actively manage the material requests. Using a `pull from list` activity, they run an SQL query to search the `Component_list` for items whose `Input_type` matches their specific needs. If a match is found, the queue pushes a transport request to the `AMRWork_Request` list.

While the AMR moves the item, the order is moved to the `AMRWork_Transported` list. Finally, the AMR arrived to the designated `unload queue`, the item is pushed to the `AMR_Released` list to inform the vehicle that the task is complete and it can proceed to the next one.

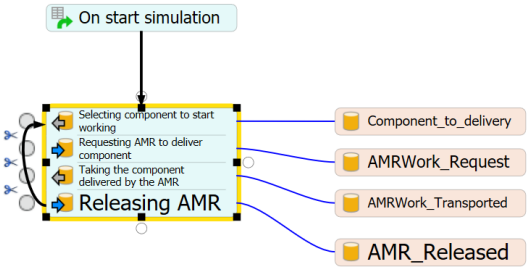


Figure 4.11: AMR process flow - Task assignment

Regarding the **transportation level**, as mentioned in the previous section, the physical movements of the vehicles are calculated by the A* Navigation System. However, their destinations and tasks are controlled by the **fleet management process flow**, which is illustrated in teh Fig. 4.12.

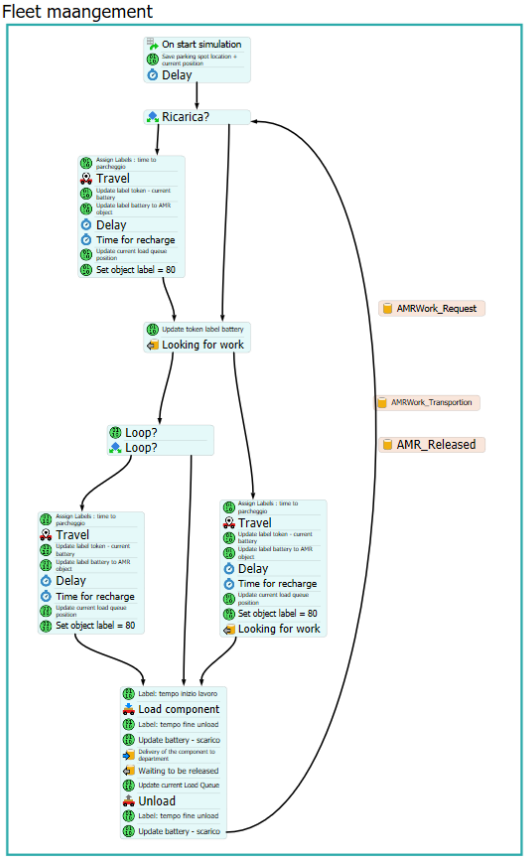


Figure 4.12: AMR process flow - Fleet management

Vehicles leave the parking area only when there is a mission available in the `AMRWork_Request` list. The **job dispatching logic replicates** the AGV system: the AMR always selects the nearest available task. This is achieved using a label called `Load_Queue`, which numbers the layout queues based on their logical position along the path. When pulling a new task, an SQL query extracts the pickup and drop-off information and sorts the list in ascending order based on the `Load_Queue` label, ensuring the closest task is assigned first. This is the SQL query used: `SELECT component, Input_type, Queue_Department, Load, ID_Load_Queue, ID_Unload_Queue ORDER BY ((ID_Load_Queue - puller.current_Unload_Queue+ 100)%100) ASC`.

A major difference compared to the AGV system is the **battery management**, which has a strong influence on the job dispatching. While AGVs recharge only at the end of the shift, AMRs are forced to recharge their batteries during working hours in the charging station that are located inside the parking area.

The recharging process occurs in three specific situations:

1. When the battery level is below the minimum threshold. The Process Flow constantly checks

the battery level before assigning a new task. If it is too low, the vehicle have to recharge the battery (Fig. 4.13).

2. After a full loop. Before moving to a new pickup point, the logic checks if the AMR has completed a full round of the facility layout. If a loop is completed, the AMR is forced to return to the parking area to recharge. Once fully charged, it resumes its interrupted task. It is important to note that, while parked, the AMR does not update its assigned job, even if a new task closer to the parking spot appears (Fig. 4.14).

3. During idle state. If the `AMRWork_Request` list is empty, the vehicle returns to the parking area to wait for new orders and recharges its battery in the meantime.

To determine if the AMR has completed a full loop, the following query is evaluated within a `Decide` activity: `token.next_department >= token.current_Unload_Queue ? 2 : 1`. This process checks the current position of the AMR (which corresponds to its current drop-off point) and compares it with the location of the next available task. This comparison uses the `token.next_department` label, which is an integer representing the specific step of the job within the material flow. If the next job is logically positioned behind the AMR's current location, meaning the vehicle must complete a full loop to reach it, the process flow enters the section on the left to cahrge the battery of the vehicle. Otherwise, the AMR proceeds directly to complete the task (Fig. 4.15).

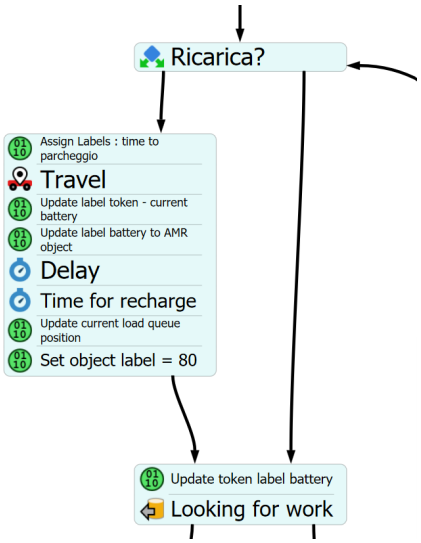


Figure 4.13: AMR process flow - Fleet management - Battery level check

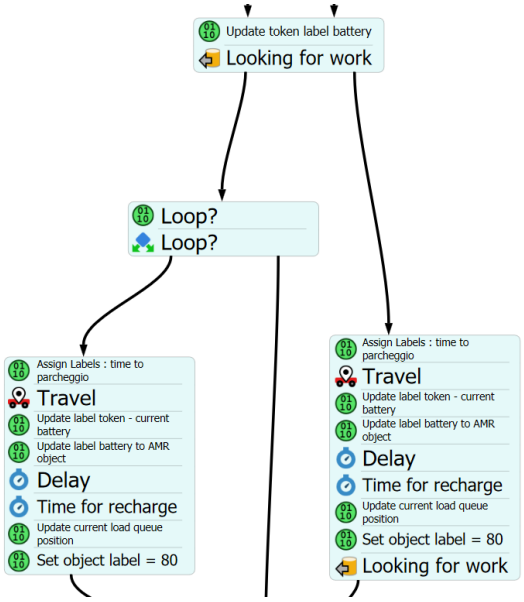


Figure 4.14: AMR process flow - Fleet management - Check of full loop

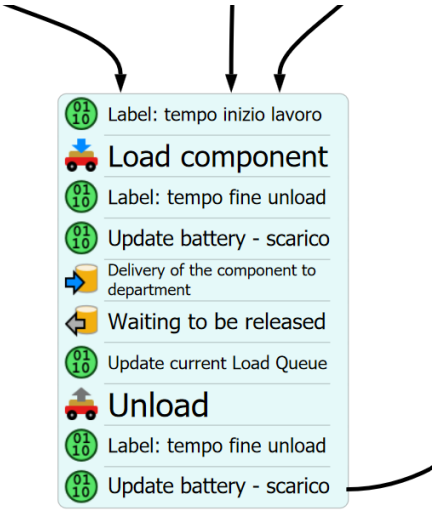


Figure 4.15: AMR process flow - Fleet management - Loading and unloading section

To track the energy levels, the battery status is manually updated using the `Assign Label` activity within the fleet management process flow. The battery consumption is modeled as a linear function of the active travel time. Specifically, every time the vehicle reaches a pickup or drop-off point, its battery level decreases proportionally to the time it spent moving. It is assumed that the battery decreases only during active travel, while the battery consumption during loading and

unloading operations is minimal and considered negligible. The query used to update the battery level are the following:

- Update of the battery after reaching the loading and unloading point: `token.instance.battery - ((token.t_fine_work - token.t_inizio_work)*rate of consumption)`
- Update of the abttery after reaching the parking spot: `token.instance.battery - ((token.t_at_parcheggio - token.time_start)*rate of consumption)`

Similarly, the recharging process follows a linear logic. When the AMR enters the charging station, it does not stay for a fixed predefined amount of time. Instead, it remains parked for the exact time required to fully restore the battery to its maximum capacity, based on a constant charging rate. Once the maximum level is reached, the AMR is immediately released and ready to accept new tasks. The query to update the battery level is `(max level of battery - token.instance.battery)/rate of charging`.

4.2.2.2 Modeling assumptions

Similar to the AGV systems, after defining the control logic, it is necessary to establish the physical and operational constraints of the AMR fleet.

The reference vehicle selected for this analysis is an **AMR capable of towing carts**. While this system configuration offers higher flexibility, this advantage is balanced by a longer positioning phase in the loading and unloading areas, with estimated times ranging between 15 and 30 seconds.

This increased duration is due to the complex recognition and alignment checks the AMR must perform to accurately identify the specific area (cell) where it needs to position itself. Similar to the AGV case, the actual picking operations of bins or products from the cart shelves are performed directly by the human operator. Finally, in the simulation model, the vehicle's carrying capacity is assumed to be unitary.

Regarding the kinematic behavior, an AMR is characterized by constant acceleration and deceleration rates. However, unlike the AGV network, it operates with a single, constant forward speed, which does not decreases to avoid the A* navigation system dynamically recalculates the path to steer around obstacles.

On the other hand, the physical interactions with the departments is similar to the AGV systems. Therefore, the loading and unloading time were modeled as probabilistic functions to introduce natural variability and recreate a realistic industrial scenario. However, compared to the AGV system, the AMR loading and unloading times are longer. This is because, in a real-world industrial environment, these operations are not instantaneous since the vehicle requires additional time to perfectly align and center itself under the cart before performing the loading or unloading operation.

As for the routing logic, in this model the AMR's movements within the facility has been limited. The vehicle cannot freely choose any desired path across the layout since the default A* grid nodes is modified and physical dividers were introduced to prevent AMR from entering restricted zones. This modeling choice was made because the simulation assumes the use of AMRs equipped with **QR code navigation**, which forces the vehicles to not travel diagonally across open spaces to reach their destination, but, instead, they are have to follow a precise path defined by QR codes. These QR codes are strategically positioned on the facility floor, and they define the allowed routes

and exactly where the vehicle is permitted to turn. Therefore, in this context, the AMR must navigate in a strictly orthogonal, rectangular manner.

To accurately recreate this QR code system within the 3D model, the FlexSim `barrier` object was utilized. `Barriers` are typically used to define solid areas where vehicles cannot enter, but they also possess an advanced feature: they can modify the allowed travel directions of the A* grid nodes. As shown in the figure below, the barrier tool provides interactive directional arrows that allow the user to manually select which specific movements are possible to the vehicle. By using these modified barriers, a custom grid was created. In this study, the grid was designed with a constant distance between the nodes to accurately reflect real-world industrial standards. It is also assumed that the allowed paths are strictly unidirectional, and turns are granted only at specific predefined intersections. In this way,whenever the A algorithm is triggered to find a route, it is forced to calculate the shortest path strictly within the boundaries of this customized grid and all the imposed directional constraints.

Finally, the simulation operates over a standard 8-hour working shift. As previously detailed in the control logic section, the battery management is strictly monitored throughout the day. The battery consumption is modeled as a linear function of the active travel time. The battery is recharged during the shift based on three specific triggers: when the energy level drops below the minimum threshold, when the AMR completes a full layout loop, or when the vehicle is in an idle state.

In addition, it is assumed that the AMR does not attach to the charging station instantaneously. The docking time required to physically connect to the charger is not a fixed value; instead, it is modeled using a probabilistic function. Once successfully connected, the recharging process is also linear, keeping the vehicle at the station only for the exact time required to fully restore its capacity.

4.2.3 Generation of obstacles

In this study, obstacles represent human operators generated at specific times and locations within the layout. They appear at key operational zones, like pick-up and drop-off points, because the loading and unloading operations are performed by human workers rather than the vehicles themselves. Therefore, it can happen that an operator may occasionally block a vehicle's path while performing their tasks. In the FlexSim model, the exact position of an obstacle is designated by a `plane` object, with the operator generated directly above it. It is assumed that only one human operator is present in each area at any given time.

To recreate a realistic industrial situation, the frequency of obstacle appearance is proportional to the activity level of the zone: higher in busy areas and lower in less active ones. This appearance time is not constant but follows a lognormal distribution, allowing for a detailed analysis of how varying obstacle frequencies impact overall system performance.

The generation of these obstacles is managed in FlexSim through a dedicated **process flow**, which is applied to both the AGV and AMR systems. The physical creation of the obstacle is handled by the `create object` activity. Immediately after the object is created, a custom code is executed. This script differs depending on the navigation system in use:

- for the **AGV system**, the code adds the obstacle into the agent System. This allows the AGV to sense the operator's presence and react based on predefined proximity logic (Fig. 4.16).
- For the **AMR system**, the code adds the obstacle to the A* navigation system, enabling the AMR to dynamically calculate a new path and avoid the operator upon appearance (Fig. 4.17). The specific script for this action is detailed in Code Snippet 4.18.

The second section of this process flow defines the movement of the operator. To simplify the simulation, the operator is programmed to remain stationary for a designated amount of time before moving away. This logic was implemented to prevent the system from being permanently blocked, ensuring that vehicles can eventually resume their transportation activities. The process flow proposed in this study is highly adaptable and it can be easily modified to introduce more sophisticated behaviors, uch as dynamic movement around the facility, and study more complex situations.

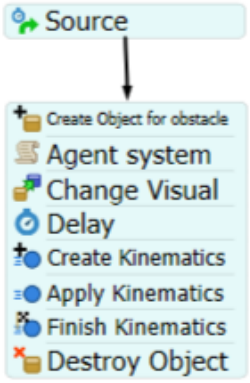


Figure 4.16: AGV - Generation of obstacles

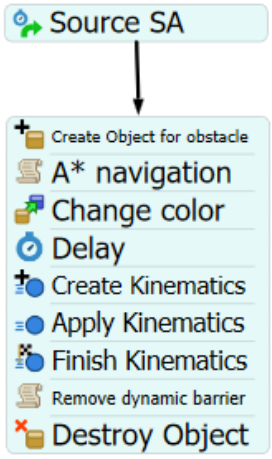


Figure 4.17: AMR - Generation of obstacles

```
/**Custom Code*/
Object current = param(1);
treenode activity = param(2);
Token token = param(3);
treenode processFlow = ownerobject(activity);

Group("Obstacles").addMember(token.obstacle);

// Dico ad A* di creare un muro rosso grande quanto l'ostacolo
AStar.navigator.addDynamicBarrier(token.obstacle, 0);

// Trovo la rete e aggiungo l'ostacolo come se fosse un "pedone" fermo
treenode navigator = Model.find("AStarNavigator");
function s(navigator, "addMember", token.obstacle);
```

Figure 4.18: AMR process flow - Fleet management - Loading and unloading section

4.3 Experimental design and key performance indicators

The core of this study is the final phase: the comparative analysis between the AGV and AMR systems. The main objective is to evaluate and compare their performances, specifically observing how they react and adapt as the layout complexity increases under various conditions: static vs dynamics and deterministic vs variable parameters.

In fact, this study is structured in three main levels.

1. The systems are tested in both **static and dynamic environments**. The goal is to see which navigation system is more flexible when obstacles are introduced, and how this impacts the performance as the layout complexity increases.

2. The scenarios are then tested by **increasing the number of vehicles**. This allows observing how the fleet size affects the overall performance and the differences between AGVs and AMRs.

3. The systems are evaluated in **deterministic versus variable scenarios**. The aim is not only to study the deterministic case with fixed parameters, but to recreate more realistic conditions by introducing variability. The types of variability analyzed differ between the static and dynamic studies. The objective is to observe how each navigation system adapts to increasing levels of variability across the three layout complexities, while also comparing the performance between AGVs and AMRs.

Regarding the last level, the variability was introduced in a limited number of parameters. For the **static** case, these parameters are:

- **Inter-arrival time** of the items entering the system.
- **Loading** and **unloading times**.
- **Docking time** at the charging station only for the AMR system.

All these parameters follow a **log-normal distribution**. Specifically, the mean value is kept fixed, while the standard deviation is modified to create three levels of variability(low, medium, and high) based on the value of the coefficient of variation which is a statistical measure defined as the ratio of the standard deviation (σ) to the mean (μ), as shown in the (4.11).

$$c = \frac{\sigma}{\mu} \tag{4.11}$$

To study the impact of these parameters in the **static** case, a specific set of scenarios was designed. Each scenario represents a different combination of the variability levels for the parameters mentioned above. Rather than simulating every possible combination, the study was intentionally limited to a carefully selected set of scenarios. This approach ensures a comprehensive analysis while maintaining a reasonable simulation time and avoiding an excessive computational load.

The methodology follows a progressive approach to systematically analyze how each factor affects the overall performance. The scenarios were grouped in the following way:

- **Deterministic scenario**, where all standard deviations were set to zero. This represents the fundamental baseline to compare all the variable scenarios.
- **Isolated variability**. It consists on focusing on the impact of single parameters on the performance. Variability was introduced to only one parameter at a time, for example only the inter-arrival time of a specific department, or only the loading time. This isolates the sensitivity of the system to a single source of variation.
- **Partial combinations**. In this case, the variability was applied simultaneously to pairs or groups of related parameters. This step helps to observe the interaction effects when multiple localized variations occur together.
- **Total system variability:**. Finally, the study examines a combination of all the variables. These represent the most realistic scenarios, applying variability to all selected parameters simultaneously across the different predefined levels.

On the other hand, the **dynamic** case introduces a new parameter, which is used to modify the behavior and the variability of the temporary obstacles. This paramter is the **time of appearence** of the obstacles in the system. It follows a lognormal distribution where the mean value is fixed while the standard deviation changes based in the level of variability being studied. The goal is to observe how the previously defined scenarios change when the systems are forced to operate under these new and unpredictable circumstances.

Also in this case, only a limited number of scenarios was analyzed. Since the main objective of the dynamic analysis is to evaluate the impact of unexpected obstacles and compare the results with the static situation, the obstacles were introduced only in the most significant static scenarios. These selected scenarios are the deterministic baseline and the total system variability scenario. For each of these, temporary obstacles were introduced and tested across all three levels of variability for their time of appearance. In addition, to understand if the average frequency also has an impact, the study was not limited to a single mean time of appearance. Instead, a second configuration was tested where this mean time is halved (meaning obstacles appear twice as often).

Finally, to compare the two systems (AGV and AMR), the same **Key Performance Indicators** (KPIs) are measured in all the analysis levels:

- **System output**, which calculates the total number of items exiting the system in 8 hours.
- **Vehicle utilization**. It measures the average utilization of the fleet, and it is calculated as the sum of the active travel time and the loading/unloading time.
- **State of the fleet**. The overall utilization was broken down to observe the single states: travel loaded, travel empty, loading, and unloading, blocked and idle.

  The **blocked time** is important to compare the systems in the dynamic case, while the **idle time** is very useful for the AMRs to see how much time they spend in the parking area and if this has a strong negative impact compared to the AGVs.

To manage the simulation efficiently, all these variables were created using the `Parameters` tool in FlexSim. This tool allows defining the minimum and maximum values that each parameter can assume, without having to select and modify the individual 3D objects in the model every time. To configure and run the scenarios, the `Experimenter` tool was used. This tool allows the user to define easily the specific parameter values directly in its interface, and it automatically runs all the scenarios in a single execution, performing he desired number of replications. It also calculates the KPIs using specific `Performance Measures`, and provides several charts, like **boxplots** and **frequency histograms**, which are used for the final comparative analysis.

Chapter 5

Case study application and results: Electric bicycle assembly implant

The case application selected for this study is an **electric bicycle assembly plant**. This facility is an example of a manufacturing environment that presents the typical characteristics found in real industrial systems. The choice to model an electric bicycle plant was made because this type of production is highly widespread today, making this study more coherent with the current industrial trends. The overall dimensions of the facility are 160x130 meters.

It is also assumes that the production is completely **in-house**, meaning that all the main processes, such as welding, painting, assembly, quality control, and packaging, are performed internally without relying on ready-made frame from external suppliers. It is assumed that each department have only one machine.

Another assumption is that the facility operates based on a **Just-In-Time** (JIT) production strategy. Therefore, the storage areas for the finished and packaged products are reduced to a minimum, and, instead of having a warehouse equipped with traditional racks, the facility uses a simple floor area for the pallet storage. Additionally, it is assumed that the plant operates in batches, meaning that each machine is configured to process a batch of ten components at a time.

Regarding the layout design,the facility adopts a classic **U-shaped layout** where the main input point and output point are on the same side of the building. It is assumed that there is only one of each. To effectively organize the internal logistics, the various departments are separated by secondary corridors that are 4 meters wide. While, the main corridor, which corresponds to the most heavily trafficked area where the vehicles travel, has a width of 6 meters.

For the purpose of this study, three different versions of this facility were created, each characterized with a different levels of layout complexity. The dimensions of the various departments differ in each layout to adapt to their specific features. However, despite these structural differences, the core **material flow** remains identical across all three configurations. It starts at the supermarket area (SA), where the incoming raw materials are stored. Specifically, three types of items enter the system:

- Raw tubes for the welding department to build the frames.

- Various component parts for the assembly department.
- The materials necessary for the packaging.

The raw tubes processed in the welding department are transformed into frames, which are then transported to the painting department. In the medium and high-complexity layouts, this department is divided into standard and customized lines. It is assumed that, in addition to painting, this phase also involves the preparation of accessory components, such as wheels, tires, gears, and batteries.

After the painting process, the painted frames are moved to the assembly department, where they are assembled with the remaining components. Once this operation is completed, the resulting electric bicycle undergoes a quality control check to detect any potential issues. For the purpose of this simulation model, it is assumed that the quality check is always successful. Finally, the last step of the material flow is the packaging of the finished batch before it reaches the output point.

The Fig. 5.1 represents the material flow for the low level configurations, while the Fig. 5.2 is for the medium and high level, where there is also the customized painting department.

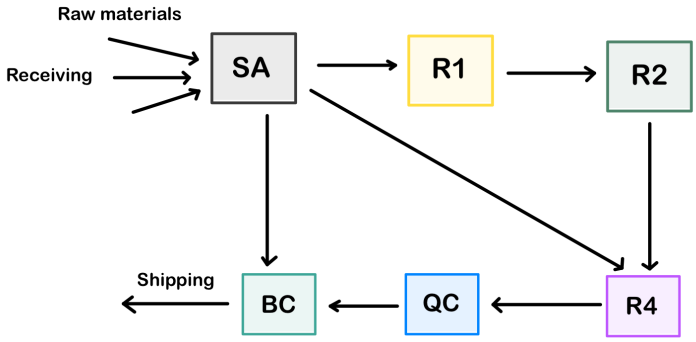


Figure 5.1: Material flow for low level layout

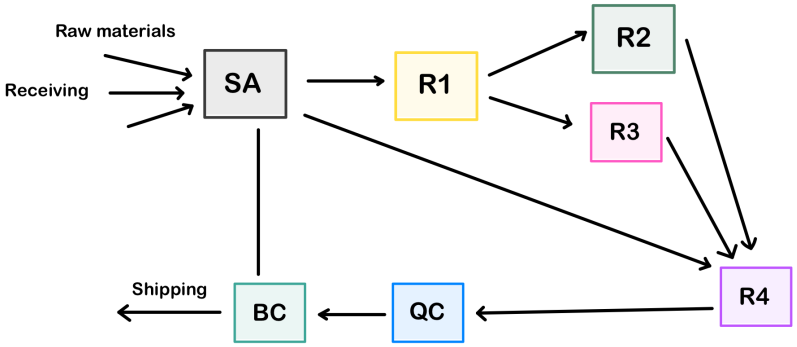


Figure 5.2: Material flow for medium and high level layout

5.1 Layout configurations and complexity assessment

5.1.1 Low level of complexity

The Fig. 5.3 shows the layout configurations with the lowest level of complexity. The areas present in this facility are the follwoing:

- Supermarket Area (SA).
- Welding department (R1).
- Standard painting department (R2).
- Assembly department (R4).
- Quality check department (QC).
- Packaging department (BC).

Since this scenario represents the lowest level of complexity, only the standard painting line is included. The departments are positioned in two main sections of the plant (top and bottom) and are separated by the main and secondary corridors. The specific dimensions of each department are summarized in Table 7. For each department, the pickup and drop-off points coincide in a single location, represented in the figure by pink rectangles.

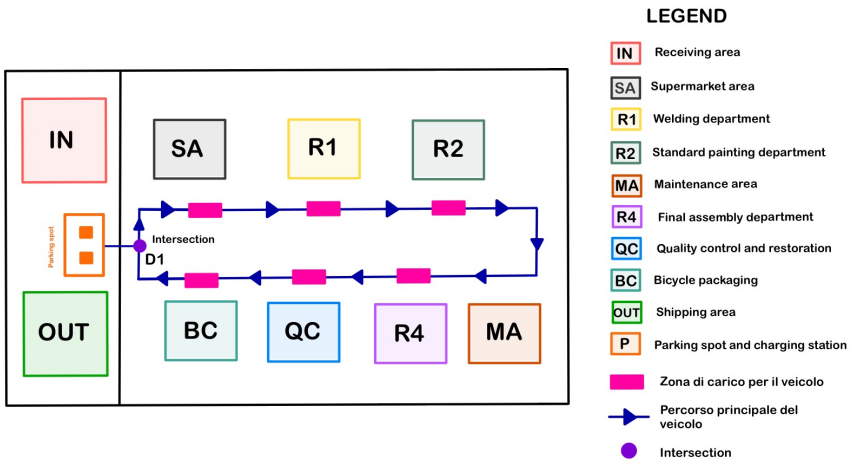


Figure 5.3: Layout configurations: low level of complexity

It is assumed that there is only one parking spot for the vehicles, located on the same side of the layout as the receiving and shipping zones. In addition, the purple dots indicate the intersection points, where the path bifurcates and the vehicle must choose between two possible directions. In this specific layout, there is only one intersection point, located near the parking area. Here, a vehicle can either decide to return to the parking spot or, if there are new tasks, to go back in the main loop.

The material flow corresponds to the one shown in the Fig. 5.1. To respect this material flow, the vehicles must follow the specific path indicated by the blue arrows in the Fig. 5.3, which is a a simple unidirectional loop. However, a major drawback of this configuration is that vehicles are forced to travel along the entire loop to reach farther destinations, leading a reduction in overall system efficiency and a considerable loss of time during material transport operations.

To calculate the **Layout Complexity Index** (LCI), this layout configurations is converted into a graph diagram, as shown in Fig. 5.4. In this graph, the arcs represent the navigation system, while the nodes represent the departments and the intersection points. Finally, the input and output nodes coincide at the parking area.

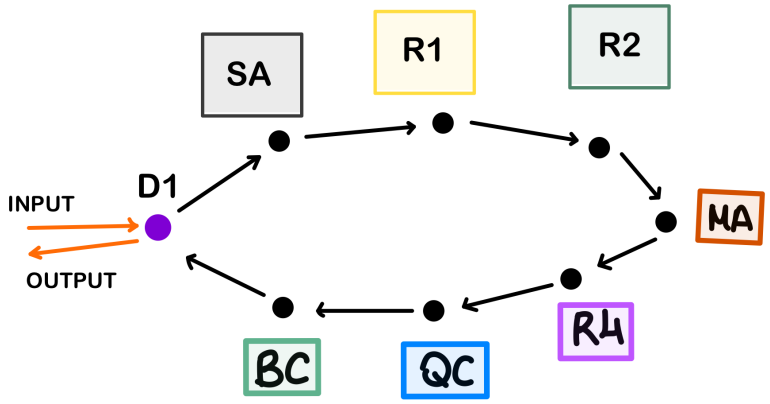


Figure 5.4: Layout configurations: low level of complexity

The parameter values for each sub-index and their final values, expressed as percentages to provide a clear understanding of their specific contribution to the total complexity, are shown in the Fig. 5.5. It can be observed that the only indices involved in this scenario are the density and the cycle indices. The other indicators have a null value due to the extreme simplicity of the layout, which lacks alternative paths and multiple decision points. The LCI of this layout is equal to 0.2857.

DENSITY INDEX	
k (# unique arcs)	8
n (# nodes)	8
D = k/n*(n-1)	14,29%

$$D = \frac{k}{n(n-1)}$$

$$P = 1 - \frac{p}{N}$$

PATH INDEX	
input nodes	1
output nodes	1
p	1
N	1
P = 1- (p/N)	0,00%

$$u = e - v + p$$

$$\alpha = \frac{u}{2v-5}$$

CYCLE INDEX (alpha)	
e (# arcs)	8
p (#sub graphs)	1
v (# nodes)	8
u (# acual number of indipendent cycles	1
alpha = u / (2v - 5)	9,09%

DECISIONS INDEX	
SP (#nodes shortest path)	8
LP (#nodes longest path)	8
DS = 1 - SP/LP	0,00%

$$DS = 1 - \frac{SP}{LP}$$

$$RD = \frac{r}{a}$$

REDUNDANCY DISTRIBUTION INDEX	
r (# pair of adjacent nodes with multiple connections)	0
a (# pairs of adjacent nodes)	8
RD = r/a	0,00%

$$RM = \frac{pr}{w} = \frac{w-a}{w}$$

REDUNDANCY MAGNITUDINE INDEX	
w (#forward arcs)	8
a (# pairs of adjacent nodes)	8
RM = (w - a)/w	0,00%

LCI	0,2857

$$LCI = \left(\sum_{i=1}^{n} C_i\right)^2 - \sum_{i=1}^{n} C_i^2$$

Figure 5.5: Low layout indices

The implementation of this layout in **FlexSim** is illustrated in the Fig. 5.6. The specific dimensions of each department have been accurately reproduced. Specifically, each department is equipped with the following elements:

- One machine. Specifically, the object `Processor` was used in all the department, except in R4 and BC, where the machine are represented with the object `Combiner`, a fixed resource used to group multiple items together.

- Queues to store the incoming items, called `Load_Queue`. SA has three of these queues, while R4 and BC two, since the input comes from different sources.
- One queue which collects the ouput of the process, called `Unload_Queue`.

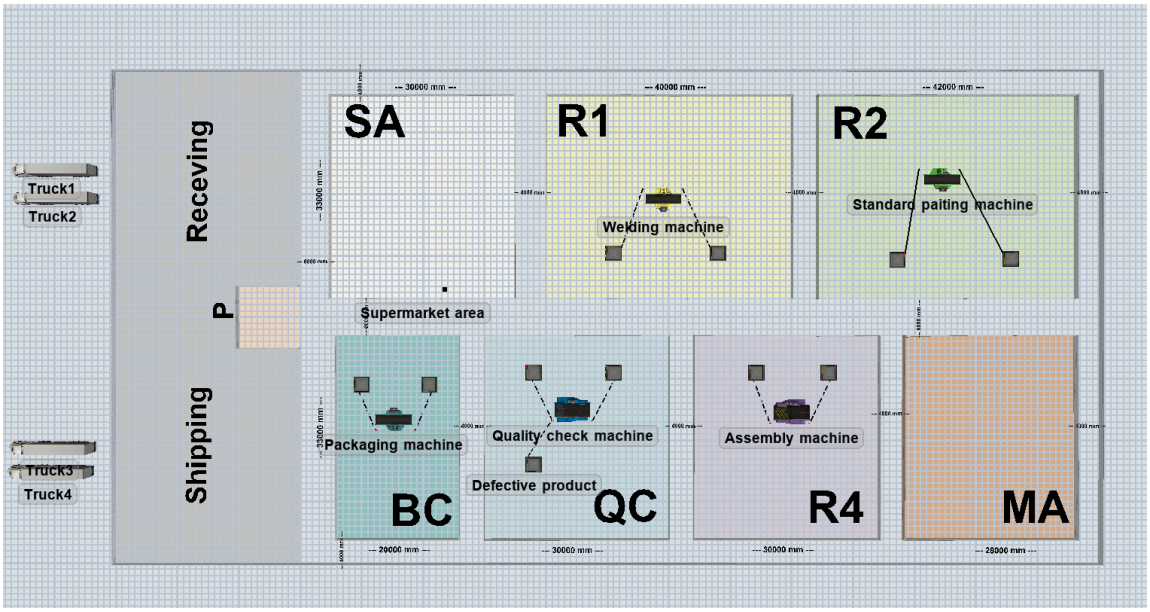


Figure 5.6: Implementation in FlexSim of the low level layout

5.1.2 Medium level of complexity

The Fig.5.7 shows the layout configurations with a medium level of complexity. The operational areas are the same as in the previous configuration, with the addition of the customized painting department (R3). Their location within the layout is identical to the lowest complexity level, while their dimensions have been modified, as illustrated in Table 7. The main difference involves the Supermarket Area (SA), which has become larger due to the addition of R3. It is assumed that this expansion is necessary to store the additional and specific components required to feed the customized painting process. As in the previous configuration, the pickup and drop-off points for each department coincide in a single location.

Compared to the low-complexity layout, this configuration introduces alternative paths to reach departments R3, R4, and BC faster. This structural upgrade allows the system to satisfy the production demand more quickly and achieve a higher total output within a working shift. This improves the overall efficiency of the navigation system, since the vehicles are no longer forced to travel along the entire loop to reach their destinations. Particularly for the AMR system, taking these shorter routes results in significantly lower battery consumption. In fact, as shown in the Fig 5.7, this layout features four intersection points.

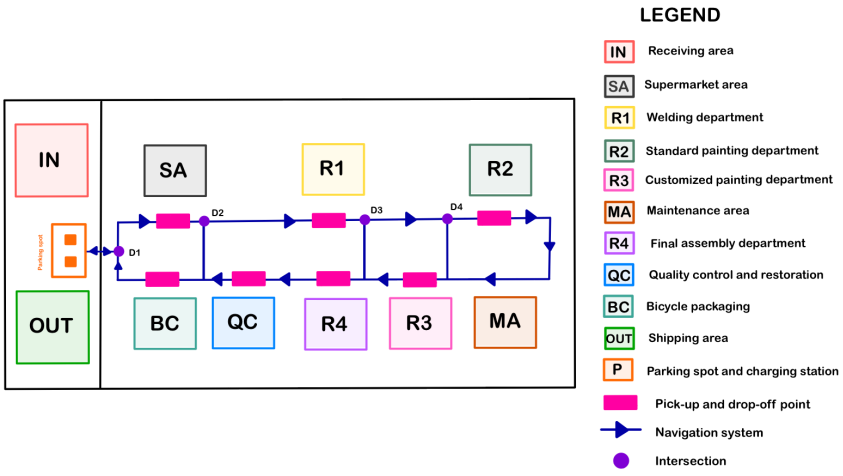


Figure 5.7: Layout configurations: medium level of complexity

To calculate the **LCI**, the corresponding diagram graph is shown in the Fig. 5.8, while the paramters values for each each sub-index are illustrated in the Fig. 5.9. In this configuration, because of the presence of the shortcuts, the indices involved are density, cycle, decision points, and path. As expected, the overall LCI value of the medium level is higher compared to the lowest level, due to the introduction of new indices and the increase in the existing ones. Specifically, the density increases because there are more nodes in the graph, and the cycle index increases due to the new arcs that create additional independent cycles. Therefore, the path index also increases, as there are now different ways to connect the input and output nodes, which also influences the decision point index. The shortest path corresponds to the path for the BC department, while the longest path remains the one moving along the entire external loop, similar to the low-complexity level. Finally, the redundancy indices are still equal to zero. The final value of LCI is equal to 1.97.

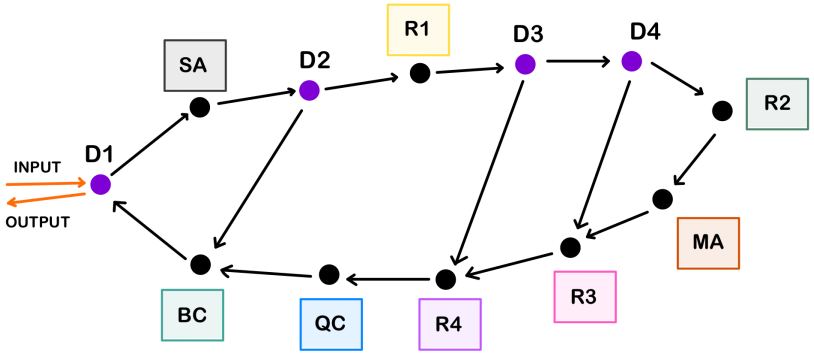


Figure 5.8: Layout configurations: low level of complexity

DENSITY INDEX		PATH INDEX		CYCLE INDEX (alpha)	
k (# unique arcs)	15	input nodes	1	e (# arcs)	15
n (# nodes)	12	output nodes	1	p (#sub graphs)	1
D = k/n*(n-1)	11,36%	p = input x output nodes	1	v (# nodes)	12
$D = \frac{k}{n(n-1)}$		N	4	u (# acual number of indipendent cycles)	4
		P = 1- (p/N)	75,00%	alpha = u / (2v - 5)	21,05%
DECISIONS INDEX		REDUNDANCY DISTRIBUTION INDEX		REDUNDANCY MAGNITUDINE INDEX	
SP (#nodes shortest path)	4	r (# pair of adjacent nodes with multiple connections)	0	w (#forward arcs)	15
LP (#nodes longest path)	12	a (# pairs of adjacent nodes)	15	a (# pairs of adjacent nodes)	15
DS= 1 - SP/LP	66,67%	RD = r/a	0,00%	RM = (w - a)/w	0,00%
$DS = 1 - \frac{SP}{LP}$					
		LCI	1,97		

$P = 1 - \frac{p}{N}$ $u = e - v + p$ $\alpha = \frac{u}{2v-5}$ $RD = \frac{r}{a}$ $RM = \frac{pr}{w} = \frac{w-a}{w}$ $LCI = \left(\sum_{i=1}^{n} C_i\right)^2 - \sum_{i=1}^{n} C_i^2$

Figure 5.9: Medium layout indices

The implementation of this layout in **FlexSim** is illustrated in the Fig. 5.10. Similarly to the low level, in this representation each department is equipped with one machine, one `Load_Queue`, and `Unload_Queue`. The main difference is R1, which has two `Unload_Queue`, one to store the customized frames and the other the standard frames.

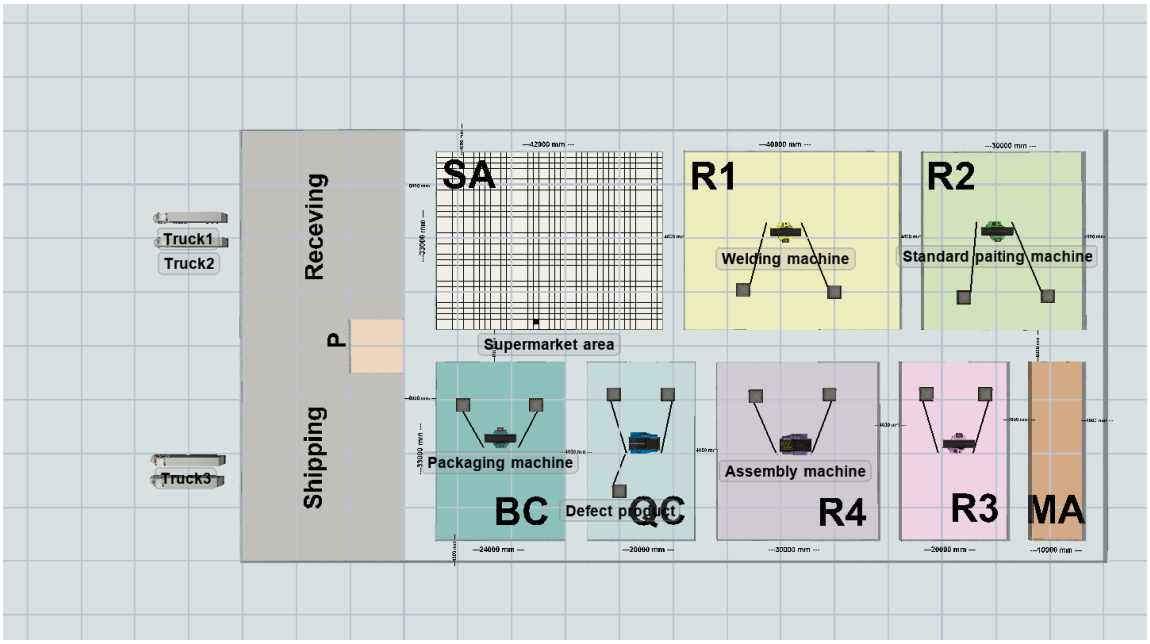


Figure 5.10: Implementation in FlexSim of the medium level layout

5.1.3 High level of complexity

The Fig. 5.11 shows the layout with a high level of complexity. The position and dimensions of the department are identical to those in the medium level configuration. The first main difference between the two layouts is that pick-up and drop-off point are now separated. Specifically, the pick-up point is situated in the upper part of each department, while the drop-off point is located at the bottom. This separation makes the navigation system much more complicated. The paths remain unidirectional, and the allowed directions are illustrated by the blue arrows in the Fig. 5.11. Due to the introduction of these new paths, there is now a of total seven intersection points.

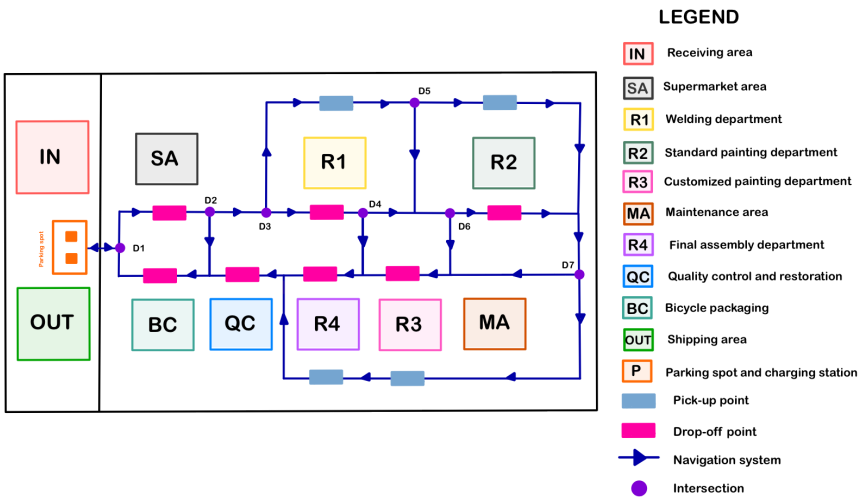


Figure 5.11: Layout configurations: high level of complexity

The last difference is in the packaging department (BC), which now presents two machine instead of one. This addition was implemented to create redundancy in the path that connects the QC and BC department, as shown in the **graph** 5.12 created to calculate the LCI. Therefore, the diagram graph as become significantly complicated due to the separation of the pick-up and drop-off points. In this specific graph, the nodes indicate both the pickup and drop-off locations, in addition to the intersections.

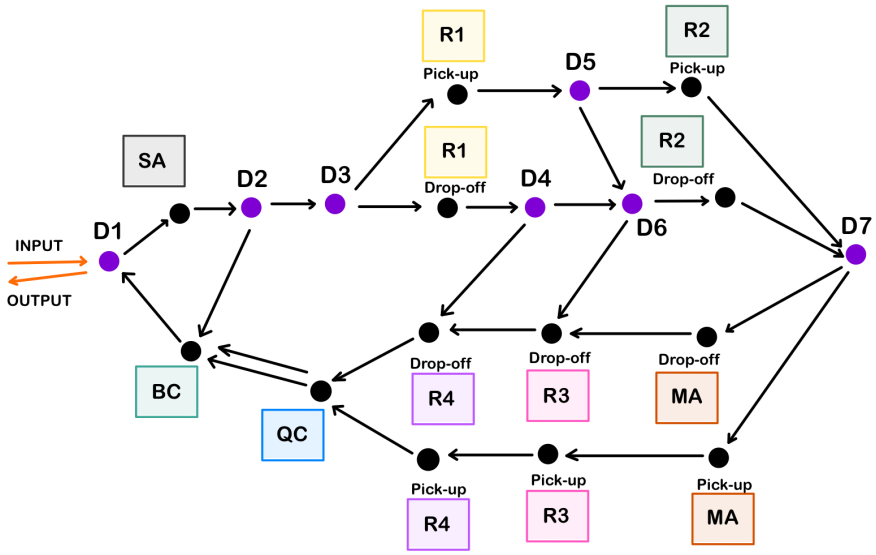


Figure 5.12: Layout configurations: low level of complexity

The value of each parameters to calculate **LCI** are shown in the Fig. 5.13. It can be observed that the LCI has increased up to 2.47, since all the indices are now involved, and their respective values have grown accordingly.

DENSITY INDEX	
k (# unique arcs)	26
n (# nodes)	20
D = k/n*(n-1)	6,84%

$$D = \frac{k}{n(n-1)}$$

$$P = 1 - \frac{p}{N}$$

PATH INDEX	
input nodes	1
output nodes	1
p = input x output nodes	1
N	10
P = 1- (p/N)	90,00%

$$u = e - v + p$$

CYCLE INDEX (alpha)	
e (# arcs)	26
p (#sub graphs)	1
v (# nodes)	20
u (# acual number of indipendent cycles)	7
alpha = u / (2v - 5)	20,00%

$$\alpha = \frac{u}{2v - 5}$$

DECISIONS INDEX	
SP (#nodes shortest path)	4
LP (#nodes longest path)	14
DS = 1 - SP/LP	71,43%

$$DS = 1 - \frac{SP}{LP}$$

$$RD = \frac{r}{a}$$

REDUNDANCY DISTRIBUTION INDEX	
r (# pair of adjacent nodes with multiple connections)	1
a (# pairs of adjacent nodes)	26
RD = r/a	3,85%

$$RM = \frac{pr}{w} = \frac{w - a}{w}$$

REDUNDANCY MAGNITUDINE INDEX	
w (#forward arcs)	27
a (# pairs of adjacent nodes)	26
RM = (w - a)/a	3,70%

LCI	2,47

$$LCI = \left(\sum_{i=1}^{n} c_i\right)^2 - \sum_{i=1}^{n} c_i^2$$

Figure 5.13: High layout indices

The implementation of this layout in **FlexSim** is illustrated in the Fig. 5.14. The implementation is similar to the medium level. The only difference is the number of processing machines in BC.

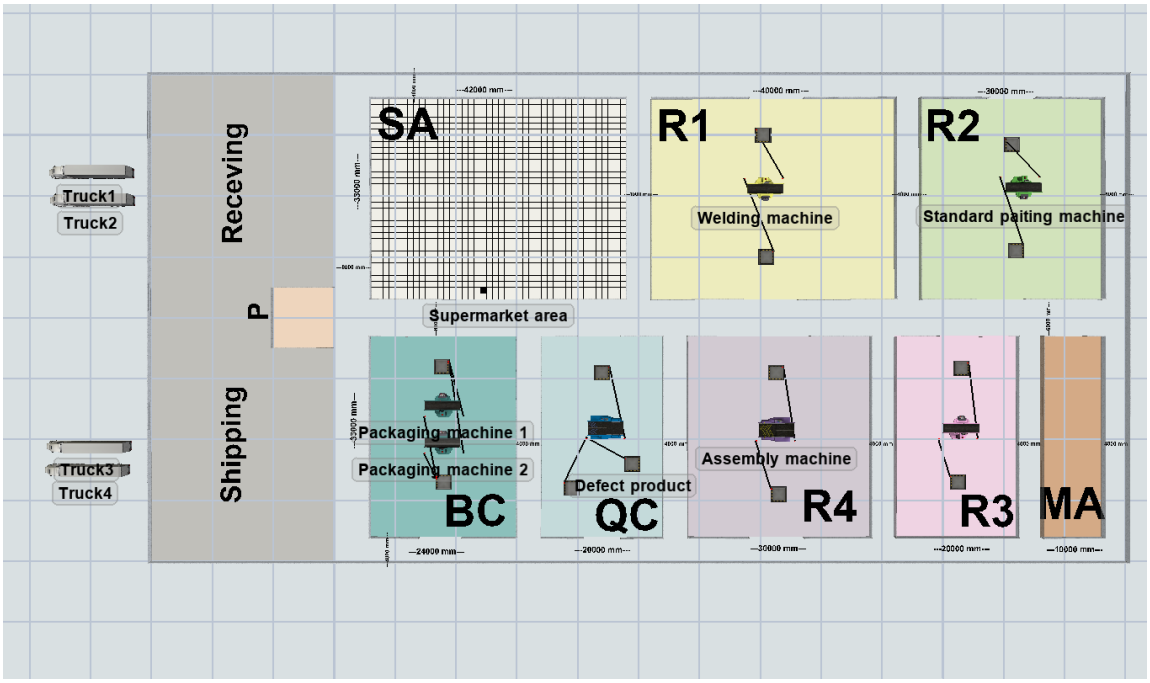


Figure 5.14: Implementation in FlexSim of the high level layout

5.2 Simulation model implementation in FlexSim

After the creation of the layout configurations and the layout complexity assessment, these layout were implemented into FlexSim to evaluate the performance of the automated material handling systems (AGVs and AMRs).

The common characteristics between the two systems are the logic behind input generation in SA, the material flow management, and the implementation of dynamic obstacles.

Regarding the first feature, the raw materials stored in SA are generated with an inter-arrival time that follows a lognormal distribution with a mean (μ) of 600 seconds. The standard deviation (σ) is not a fixed value but it changes depending on the scenarios being analyzed. This approach allows for the analysis of the system under different levels of variability. Its value ranges from 0 to 900 seconds, as it is shown in the Tab. 8.

Inter-arrival time			
Level of variation	**Coefficient of variation (c)**	**Standard deviation σ (s)**	**Mean value μ (s)**
Low	0.5	300	600
Medium	1.0	600	600
High	1.5	900	600

Tab. 8 - Value of standard deviation of inter-arrival time in SA.

As mentioned in the previous chapter, the material flow in FlexSim is managed using labels created directly within the `Load_Queue` of each department. When an item enters the system, a specific destination label is assigned to it, telling the vehicle the next step in the production process. For most departments, the flow is linear. However, the assembly (R4) and packaging (BC) areas represent critical nodes because they receive inputs from multiple sources. Specifically, R4 receives raw materials from the SA and processed components from the painting department (R2/R3), while BC receives materials from both SA and QC. To manage these multiple flows, two separate `Unload_Queue` objects were created for these departments, each associated with a specific label to ensure the transport system (AGV/AMR) correctly identifies drop-off points. The specific label assignment strategy is detailed in Table 9.

Department	**Input_type label**
R1	1
R2	2
R3	3
R4	4 and 4.2
QC	5
BC	6 and 6.5

Tab. 9 - Label assignment for each department.

5.2.1 AGV system implementation

The layout configurations described in the previous chapter were adapted to implement an **AGV system** by adding the physical travel paths. The path were positioned in the layout following the scheme shown in the Fig. 5.3, 5.7, 5.11. In the low level configuration, the control point were placed at a constant distance of 9 meters, while for the other two configurations the distance was reduced to 6 meters, since the total lenght to cover is significantly higher. Additionally, bending radius for the curved sections of the paths was set to 1 meter.

Regarding the **agent system**, the coverage radius for the proximity behavior during the broad phase is equal to 4 meters, while the radius for the narrow phase is 1.5 meters. Furthermore, the Field of View (FOV) in the broad phase is set to 50 degrees on each side, creating a total frontal vision cone of 100°. Instead, the narrow phase has an angle of 30 degrees, resulting in a 60° total cone. The Tab. 10 summarizes all the Agent system values used for the obstacle detection.

Tab. 10 - AGV: Agent system parameters for obstacle detection phases.

Detection Phase	**Coverage Radius (m)**	**Total Frontal FOV**
Broad Phase	4.0	100° (± 50°)
Narrow Phase	1.5	60° (± 30°)

Finally, the **system parameters** and **assumptions** described in the chapter 4.2.1.3, such as vehicle dimensions, capacities, and speeds, are summarized with their corresponding values in Table 11. It can be observed that some kinematic characteristics present two distinct values: one for the standard speed and one for the reduced speed. This difference is due to the proximity behavior rules set by the agent system, which forces the AGV to slow down as it approaches an obstacle. Specifically, the speeds are reduced by half.

Another aspect to noticed is that the loading and unloading time follows a lognornmal distribution with mean value μ equals to 5 seconds and standard deviation σ that changes depending on the level of variability, as it is shown in the Tab. 12

Loading and unloading time			
Level of variation	**Coefficient of variation (c)**	**Standard deviation σ (s)**	**Mean value μ (s)**
Low	0.5	2.5	5
Medium	1.0	5	5
High	1.5	7.5	5

Tab. 12 - Standard deviation of loading and unloading time of an AGV.

5.2.2 AMR system implementation

Similarly to the previous system, the navigation system for the **AMR** was designed following the schemes illustrated in the Fig. 5.3, 5.7, 5.11. As it has been mentioned in the section 4.2.2, the

AMR's movements are restricted rather than allowing it to move freely around the facility, in order to simulate the behavior of an AMR equipped with QR code navigation. To recreate this system, a custom grid was created using the `barrier` object. It was assumed that the distance between the QR codes was around 1.5 meters. The grid was designed as a checkerboard pattern: at every node, the vehicle can move forward (never backward, since the paths are unidirectional), but lateral movements (left or right) are only allowed at specific nodes. These points, where lane changes or turns are permitted, are positioned every two nodes.

Another main difference between AGV and AMR system is the **battery management**. As it has been described in details in the previous chapter 4.2.2, three different scenarios trigger the charging process. The first occurs when the battery level falls below the minimum threshold, which was set at 25% for this specific application. Typically, the minimum usage limit is 20%, however, the threshold was raised to 25% as a safety margin to further minimize the risk of battery degradation. The second instance where the vehicle must return to the parking spot to charge is after completing a full loop. The battery consumption is calculated based on the time spent in active travel, excluding the time spent during loading and unloading operations. The consumption percentage was determined as follows: it is assumed that during an 8-hour shift, the AMR consumes an average of 80% of its battery (dropping from 80% to 20% if not recharged throughout the 8-hour working shift). Therefore, knowing the hourly consumption rate, along with the total active travel time, the total consumed battery can be easily calculated. In this case the hourly consumption rate is equal to 7.5% per hour (0.002083% per second).

Regarding the charging time, it is calculated based on the amount of battery required to reach an 80% charge level. In fact, it is assumed that the battery is never charged to its maximum capacity, to preserve its health over time. The charging rate follows this logic: it usually takes 90 minutes to charge from 20% to 80%, the hourly charging percentage is 40% per hour (0.011% per second).

Another assumption made during the charging process at the station involves the docking time. This time is not a fixed value butit is represented by a lognormal distribution with a mean (μ) of 25 seconds, while the standard deviation (σ) varies depending on the variability level, as shown in Tab. 13.

Level of variation	**Coefficient of variation** (cv)	**Standard deviation** (σ) [s]	**Mean value** (μ) [s]
Low	0.5	12.5	25
Medium	1.0	25.0	25
High	1.5	37.5	25

Tab. 13 - AMR: Docking time at the charging station

The remaining system parameters are summarized in the Tab. 14. Similarly to the AGV system, the loading and unloading times are not fixed values but follow a lognormal distribution. However, the mean value (μ) is higher in this case, up to 15 seconds. The standard deviation (σ) values are reported in the Tab. 15.

Level of variation	Coefficient of variation (cv)	Standard deviation (σ) [s]	Mean value (μ) [s]
Low	0.5	7.5	15
Medium	1.0	15.0	15
High	1.5	22.5	15

Tab. 15 - AMR: Loading and unloading time

5.2.3 Generation of obstacles

Following the methodology described in the previous section 4.2.3, the obstacles were positioned near the loading and unloading points of each department across all three complexity levels. Specifically, they were placed at a distance of three meters from these points. To accurately model the industrial environment, these locations were divided into three main groups based on the probability of an operator blocking the path, which strongly correlates with the volume of items handled:

- **Group 1 - high probability**. This group includes only SA, since there are three different types of items to be loaded on the vehicles, meaning that the activity level is extremely high. Therefore, this area is assumed to have the highest frequency of obstacle appearance.
- **Group 2 - medium probability**. This group includes departments R1, R4, and BC, because they directly connected to the SA and receive items more frequently and quickly compared to other areas. In addition, R4 and BC receive inputs from multiple different sources.
- **Group 3 - low probability**. This group includes the remaining departments (R2, R3, and QC), which it is assumed that are characterized by the lowest frequency of obstacle appearance.

In the high-complexity layout, these three groups are not sufficient because the pick-up and drop-off points are physically separated. Therefore, in this specific configuration, the three groups mentioned above represent only the unloading points. Therefore, two additional groups were introduced: **group 4 - loading points with medium probability** (R1, R4, BC) and **group 5 - loading points with low probability** (R2, R3, QC).

For these new loading groups, the appearance frequency is lower compared to their corresponding unloading points. Specifically, it was assumed that the appearance time is doubled, since, given the characteristics of layout, the vehicles frequent these separated zones less often.

The appearance times (μ) and standard deviations (σ) assigned to each group are detailed in Tab. 16 and Tab. 17. Finally, the last assumption is that the human operator remains stationary as an obstacle for a fixed duration of **20 seconds** before moving away toward the department's machinery.

Group	Appearance Time (μ)	Standard Deviation (low/med/high)
1 - SA	600	300 / 600 / 900
2 - R1/R4/BC	1200	600 / 1200 / 1800
3 - R2/R3/QC	2100	1050 / 2100 / 3150

Tab. 16 - Obstacle appearance times and standard deviations for low and medium level configurations.

Group	Appearance Time (μ)	Standard Deviation (low/med/high)
1 - SA	600	300 / 600 / 900
2 - Unload R1/R4/BC	1200	600 / 1200 / 1800
3 - Unload R2/R3/QC	2100	1050 / 2100 / 3150
4 - Load R1/R4/BC	1200	600 / 1200 / 1800
5 - Load R2/R3/QC	4200	2100 / 4200 / 6300

Tab. 17 - Obstacle appearance times and standard deviations for high level configuration.

5.3 Evaluation of the performance and comparative analysis

5.3.1 Scenario configurations

Following the methodology described in chapter 4.3, the following tables summarize the standard deviation (σ) values assigned to the parameters inter-arrival time, loading time, unloading time, and docking time (only for AMR). First, the **Tab. 18** presents the **scenarios** for the **static AGV system**, while **Tab. 19** shows the scenarios for the **static AMR system**. The only difference between the two tables is the addition of the docking time parameter in the AMR case. Therefore, there are in total 40 scenarios for the AGV and 49 for the AMR. For each scenario, **30 simulations** were performed. This number was chosen to ensure reliable results while maintaining a reasonable computational time.

The logic used to define these scenarios includes scenarios with:

- Deterministic values, where all the σ are equal to zero.
- Variability in a single parameter.
- Pairs of inter-arrival times.
- The combination of loading and unloading times.
- Combinations of all inter-arrival times with a single operation (load, unload, or docking).
- Full variability, where all parameters vary simultaneously.

On the other hand, for the **dynamic case**, a total of 32 scenarios were analyzed for both systems, as shown in **Tab. 20** and **21**. As mentioned in the previous chapter, the introduction of the obstacles was studied only on the deterministic and the full variability scenarios.

5.3.2 Static environment results

This section analyzes the **static scenarios**, where both systems are tested under ideal conditions without the presence of unforeseen human obstacles. The analysis focus on evaluating how the KPI, specifcally output, average utilization, and average vehicle states, respond to the progressive introduction of variability across the three levels of layout complexity, while increasing the size of the fleet. The maximum size of the fleet is set at three, since higher number does not provide significant results in terms of output. Rather than presenting the results of every single scenario, this section highlights the most significant trends, and outlier values for the comparison betwee AGV and AMR systems.

Output

First of all, it can be observed that both systems **output** level tends to **decreases** as the variability of the inter-arrival time increases across all the level of layout complexity, as shown in the Fig. 5.15. The figures show the boxplot graphs for the AGV (top) and AMR (bottom) systems working in the medium level configurations in the presence of two vehicles. AS it can be seen, both systems output levels have have a decreasing trend as the variability increases.

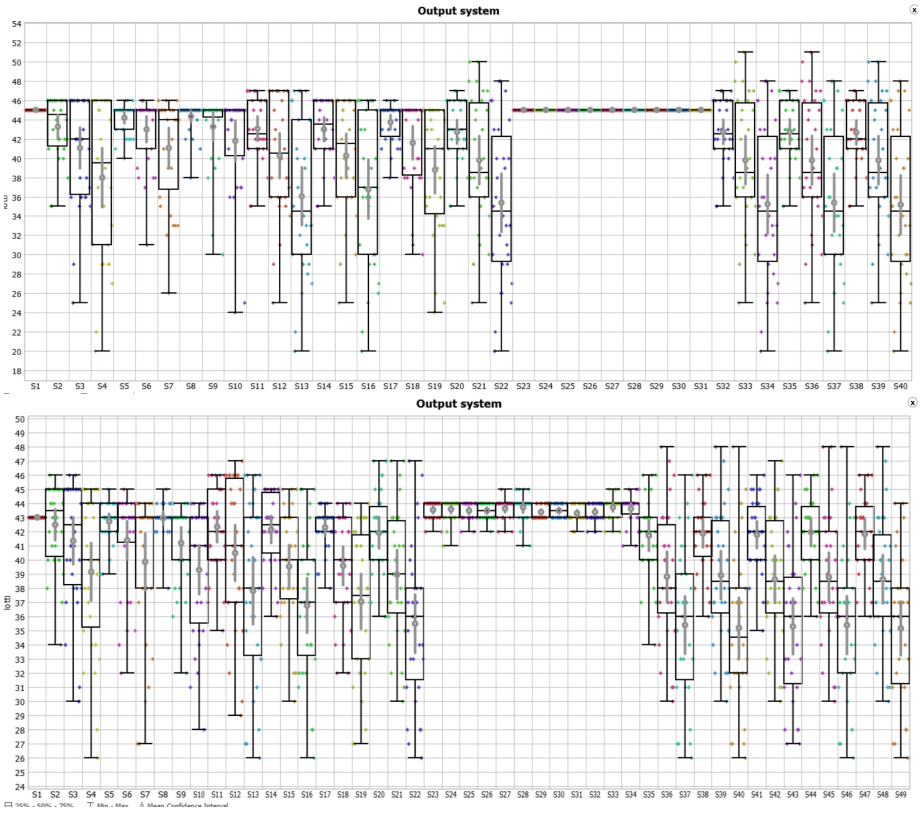


Figure 5.15: Boxplot: Comparison between AGV (top) and AMR (bottom) level of output with 2 vehicles in medium level configuration.

This trend happens because high level of variability implies that the batches arrive at irregular moments, resulting in fewer completed jobs within the 8-hours working shift. This downward trend remains valid even when increasing the number of vehicles. In fact addind a new vehicle to the fleet results in the increase of the mean value, but the negative impact of the variability remains consistent. The only exception can be found in the AGV layout with a high level of complexity when there is only one vehicle. Looking at the boxplot (Fig. 5.16), the mean value of each scenario appears to trend upward as the level of variability increases, but this is a statistical artifact. In this layout configuration, a single AGV cannot satisfy the demand and the AGV, in an 8-hour shift, can completes only 2 batches on average. The graph does not show the lower tail of the distribution since negative output levels are impossible. Because the data is bounded at zero, the higher variance artificially pulls the mean upward, masking what would otherwise be a continued flat or downward trend.

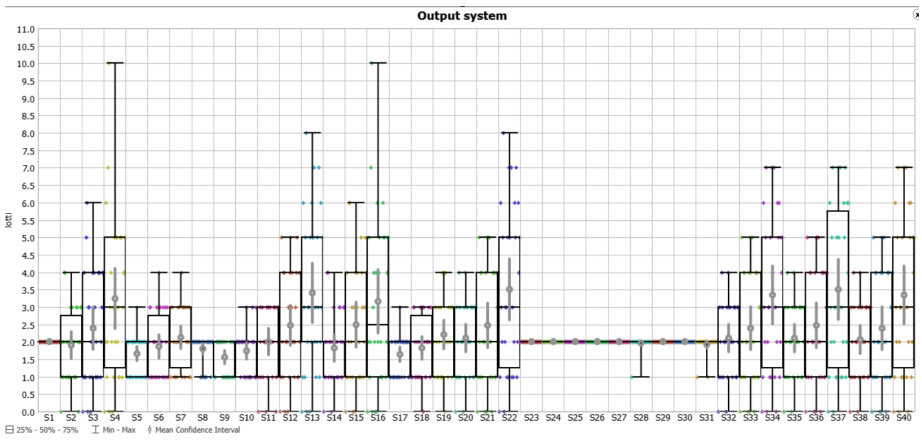


Figure 5.16: Boxplot: AGV level of ouput - High complexity layout with one vehicles

On the other hand, the impact of the loading and unloading operations (scenarios S23-S28), instead, is relevant only in the case of the AMR system, while the AGV fleet remains unaffected, as illustrated in Fig. 5.15. The reason for this behavior is that the AMR is characterized by loading and operating times higher than those of the AGVs, making it more sensitive to varaibility. The effect of the loading and unloading time can both increases or decreases the level of output, as illustrated in the Fig. 5.17. It manly depends on the number of vehicles present in the system: with two vehicle having longer loading and unloading time creates collision situations among the vehicles.

Overall, the inter-arrival times have a more significant impact on the performance compared to the loading and unloading times, as it can be seen in the three scenario (S47- S49) of the boxplot, where all the parameters are not deterministic.

Overall, the output values between the two systems are distinctly different and strictly dependent on layout complexity. The AGV system performs better in the low and medium configurations, while the AMR is superior in the more complex layout. In the **low-complexity layout**, characterized by a simple linear loop, the AGV fleet has a significant advantage since it is not required having a complex routing logic and the vehicles simply travel along their designated path. Given the

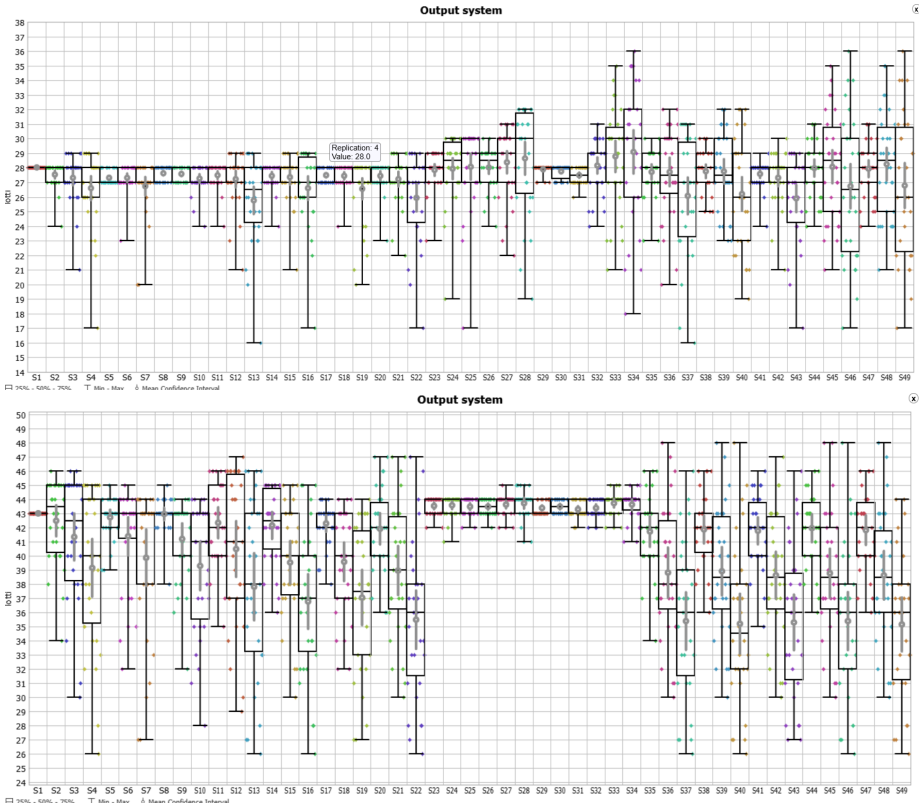


Figure 5.17: Boxplot: Impact of loading and unloading time on the output in AMR system in medium configuration. Top - one vehicle, bottom - two vehicles

simplicity of the navigation system, a single AGV is capable of completing a very high number of batches (40 in the deterministic scenario S1, ranging from 18 to 41 with variability, as shown in Fig. 5.18). On the other hand, the AMR is limited by its battery recharging constraints and the restrictive grid, and therefore it can completes on average only half that amount. It can also be observed that the AGV is more affected by variability, showing wider 95% confidence intervals compared to the AMR.

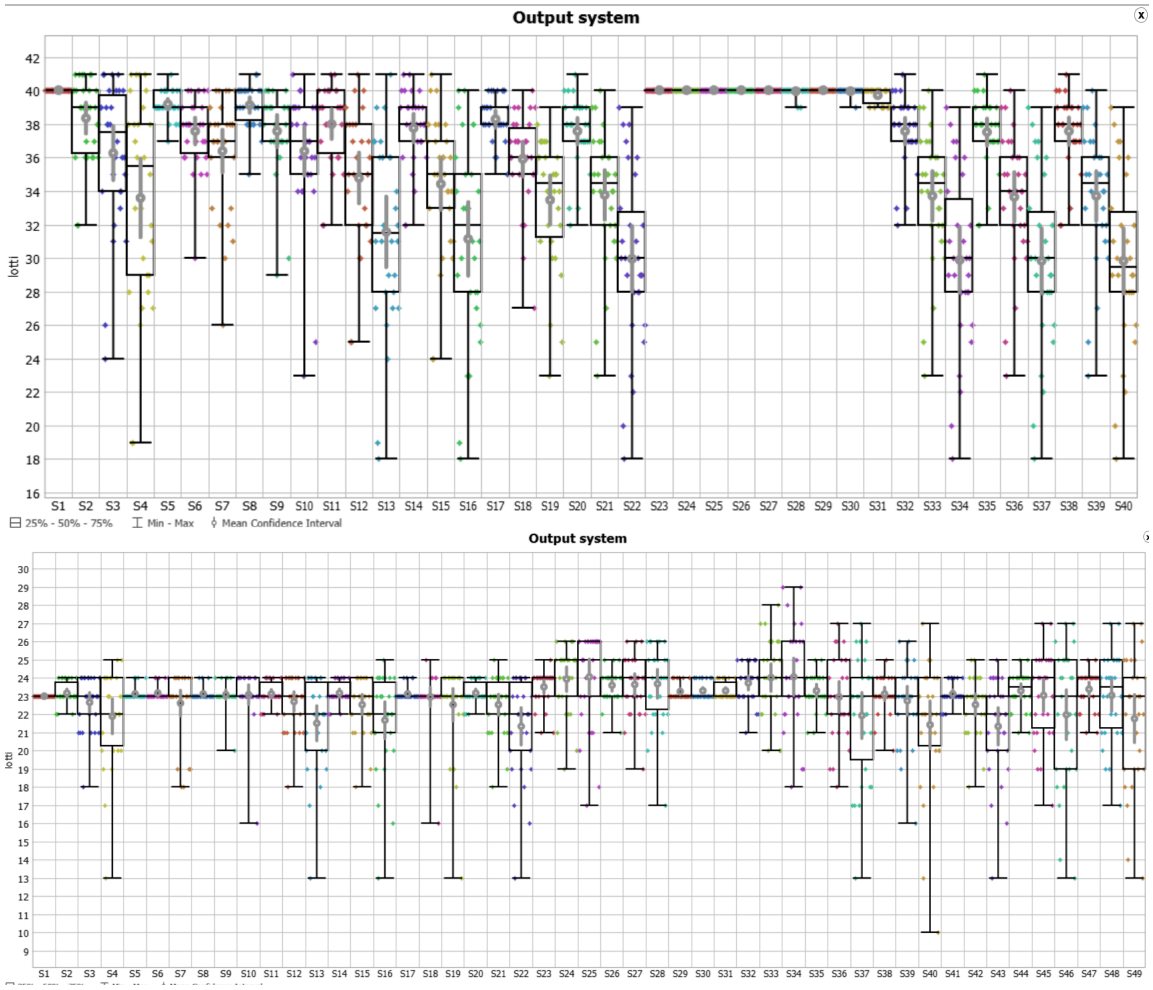


Figure 5.18: Boxplot: Comparison between AGV (top) and AMR (bottom) output in low configuration with one vehicle.

In the **medium-complexity layout**, the introduction of short-cuts improves the output of both systems. These short-cuts allow the AMR to travel shorter distances and save battery, making vehicles available faster. However, the AGV system still remains superior, with a value of output

equal to 44 in the determinitic scenario S1, while the AMR complete only 28, as shown in the Fig. 5.19.

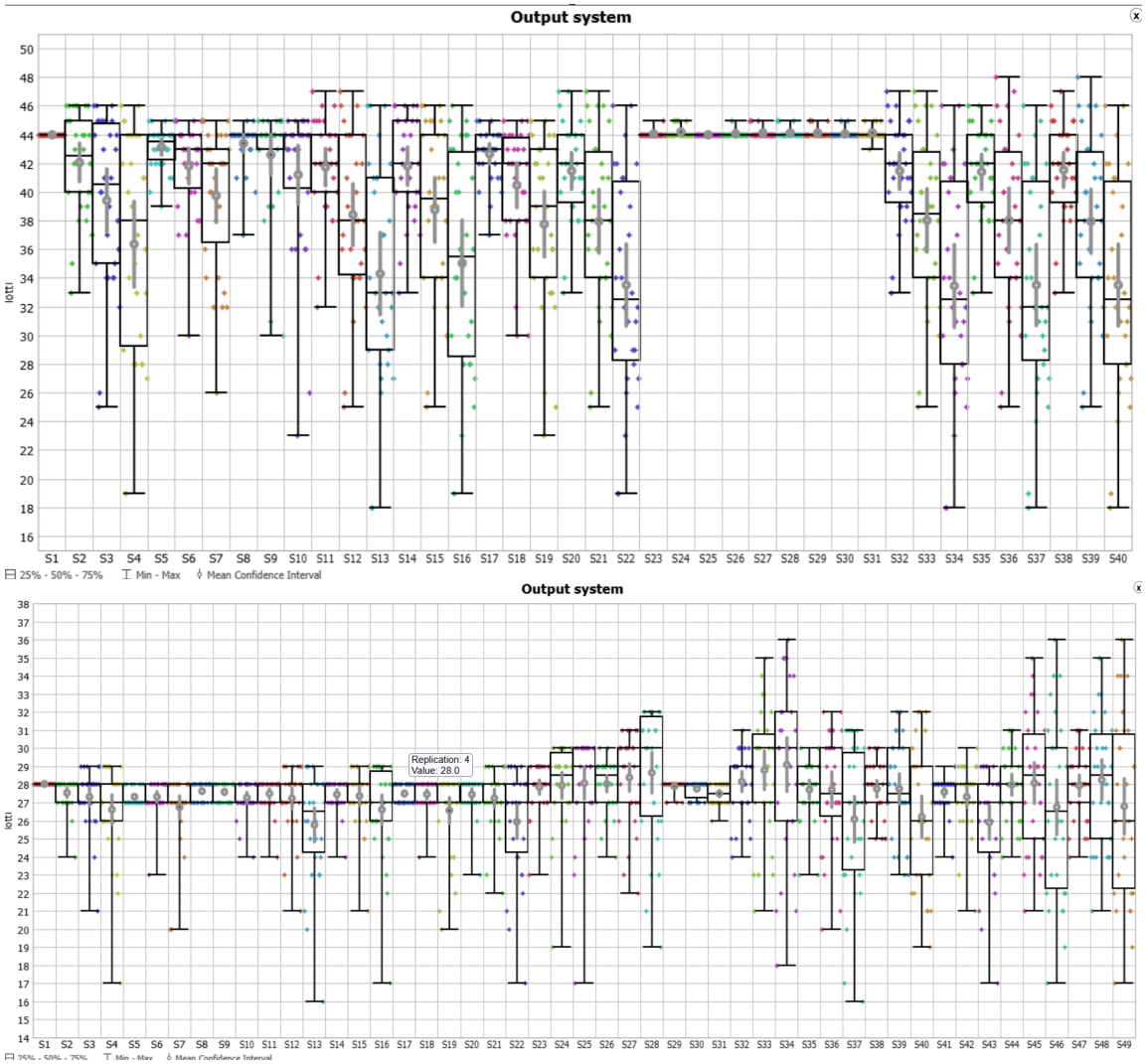


Figure 5.19: Boxplot: Comparison between AGV (top) and AMR (bottom) output in medium configuration with one vehicle.

Finally, in the **high-complexity layout**, the AMR system outperforms the AGV, especially with a single-vehicle fleet. The AGV's rigid job-search logic is tied to its immediate position on the track, making it highly inefficient over large distances since it must physically circulate the entire layout to find a job. The AMR, knowing the exact position of the request, calculates the most direct route immediately (Fig. 5.20). In the deterministic scenario (S1), AGV completes 2 batches, while the AMR 11.

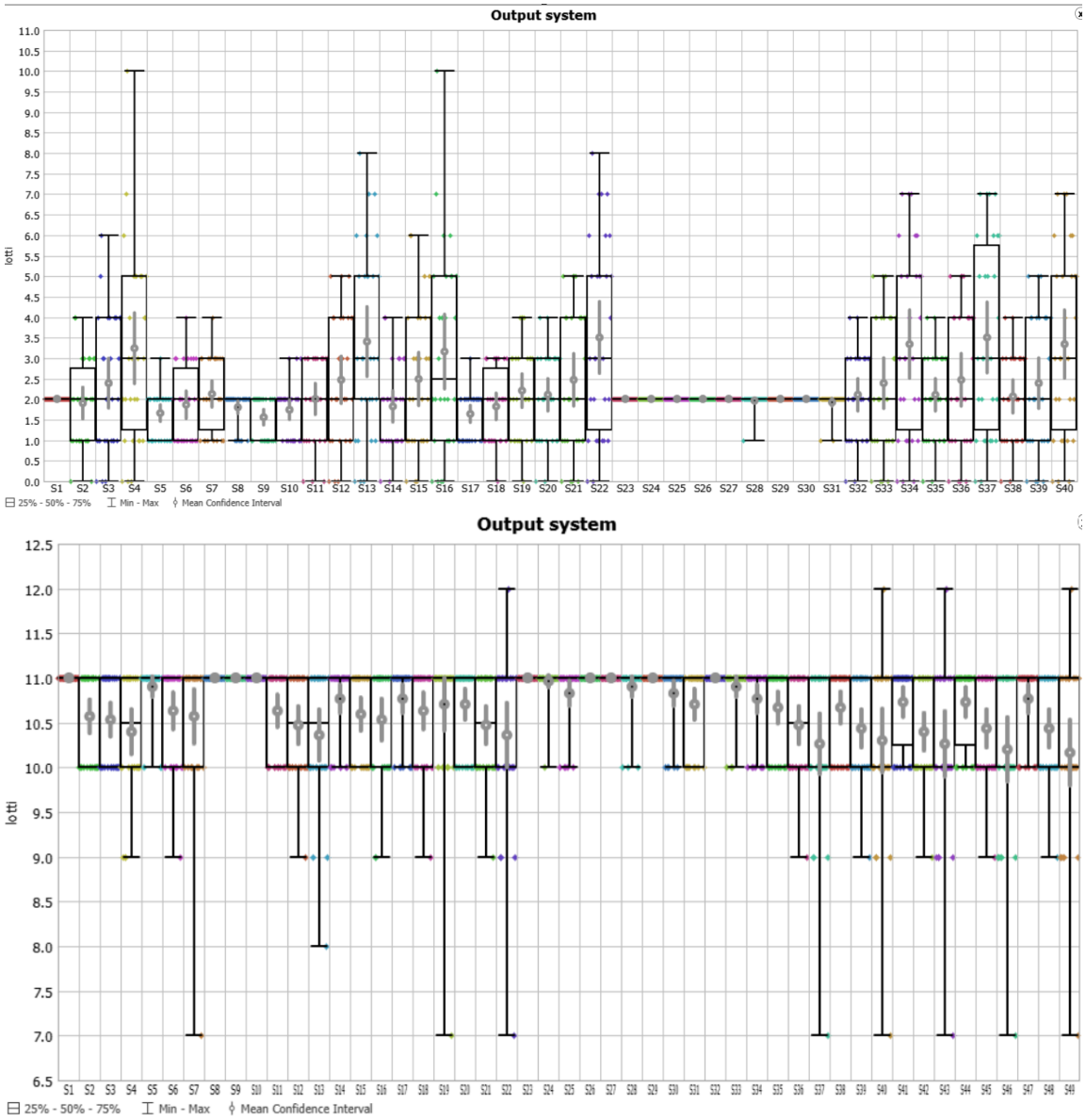


Figure 5.20: Boxplot: Comparison between AGV (top) and AMR (bottom) output in high configuration with one vehicle.

Utilization and fleet states

The limitation imposed by battery charging on the performance of the AMR vehicles is significantly reflected in the utilization values. The AMR system generally shows lower levels of utilization, moving around 0.7 and never exceeding 0.85. This is unlike the AGV system, which is capable of reaching 100% utilization, as seen in Fig. 5.21 for the medium layout complexity.

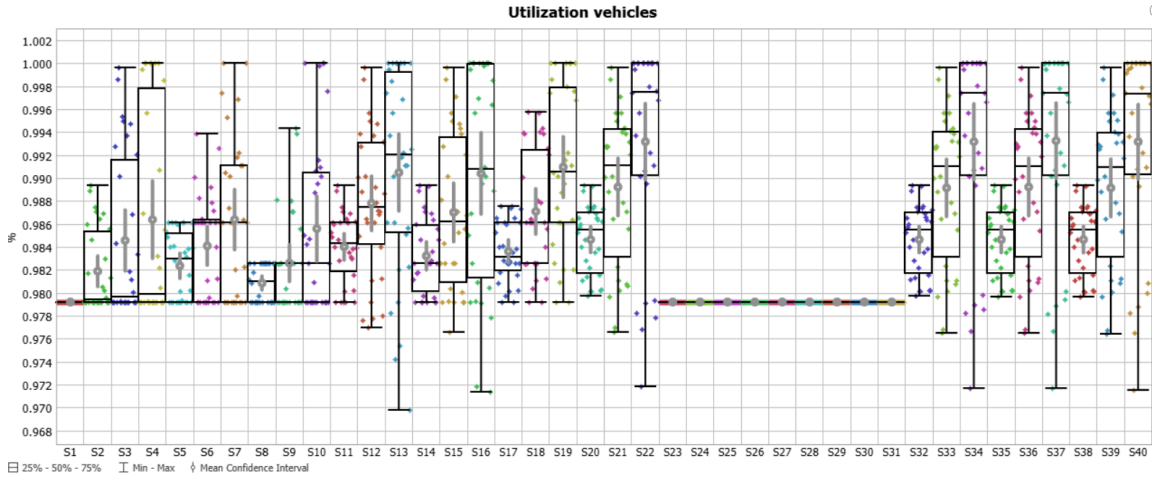


Figure 5.21: Boxplot: AGV utilization - Medium complexity layout with one vehicles

The trend of utilization in response to variability follows a specific behavior: the variability actually causes utilization to increase when the fleet is under capacity (typically with only one vehicle), as irregular arrivals force the vehicle to work continuously during rush periods. However, as the fleet size increases, utilization depends heavily on how evenly the work is distributed. Typically, utilization decreases as the fleet grows because vehicles spend more time blocked in traffic or idling.

This work distribution is highly dependent on the layout configuration. In the **low-complexity layout**, the single unidirectional loop prevents the system from efficiently splitting tasks. Since vehicles travel sequentially and cannot pass each other, the vehicle in fron of the line performs most of the work. As it can be seen in the Fig. 5.22, the AGV utilization mean value drops significantly from 0.98 to 0.68 (in S1) when moving from one vehicle to two.

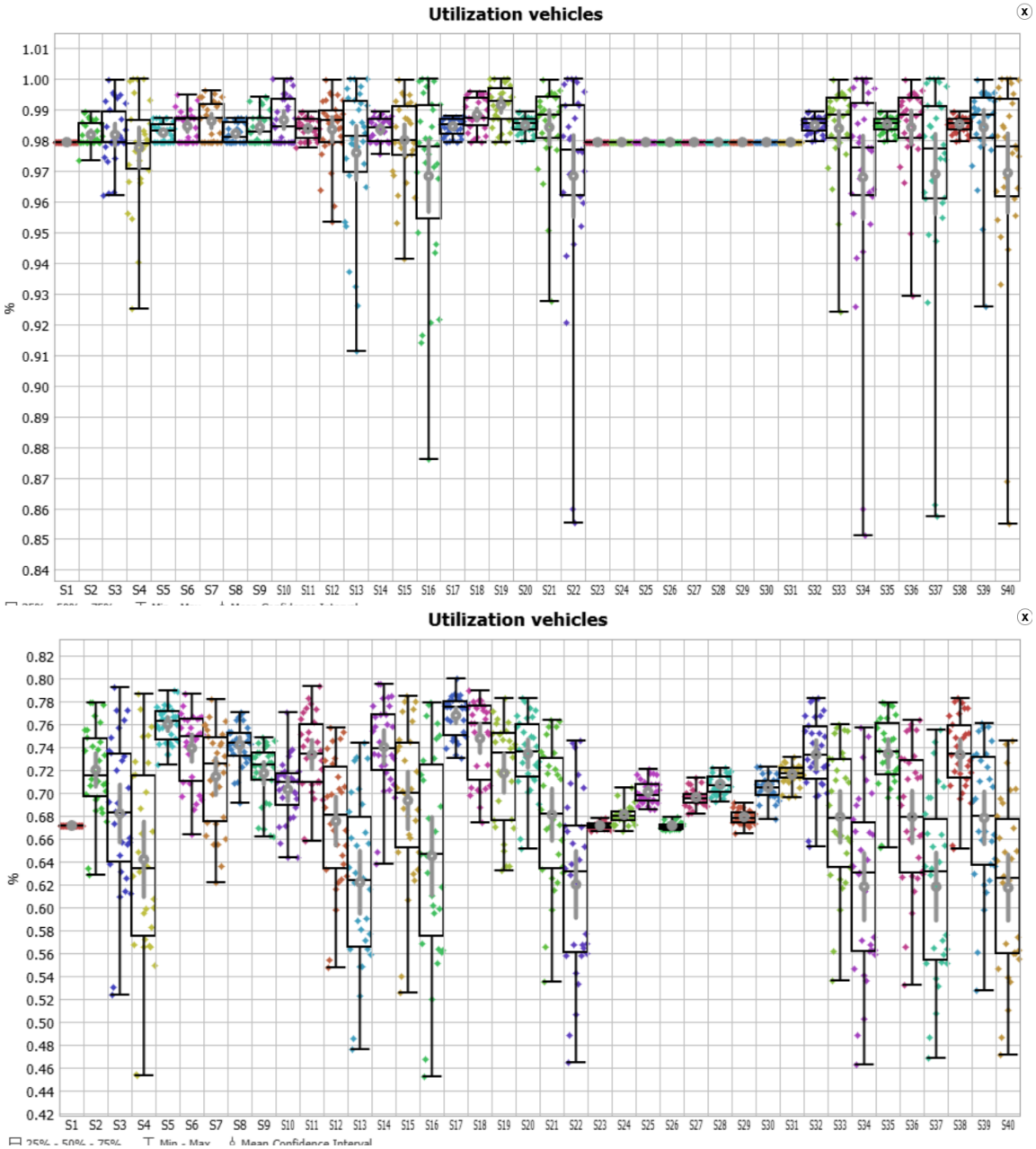


Figure 5.22: Boxplot: Comparison of utilization between AGV with one (top) and two (bottom) vehicles in low configuration

In the **medium layout**, adding a second vehicle *increases* overall utilization in the AGV system. The short-cuts allow the vehicles to distribute more evenly across the facility, reducing bottlenecks.

This is an example where adding a second vehicles lead to an improvement of the level of utilization, as shown in the Fig. 5.23. In all other configurations, introducing a second vehicle decreases the fleet's average utilization, because of an overestimation of the required fleet size. This same logic applies to the AMR system. However, the drop in utilization is mitigated by the vehicle's constant need to return to the parking station to recharge.

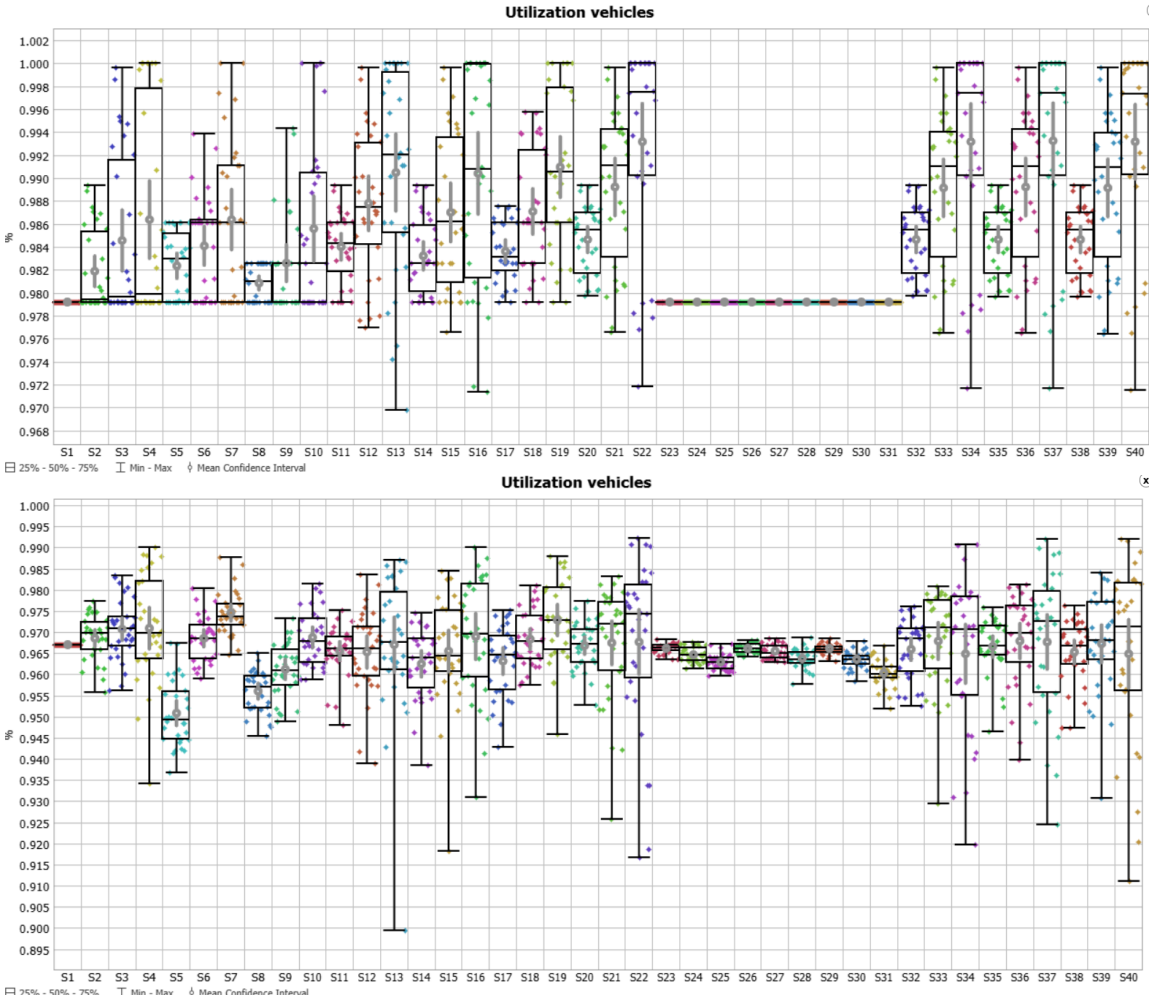


Figure 5.23: Boxplot: Comparison of utilization between AGV with one (top) and two (bottom) vehicles in low configuration

Utilization values are strictly correlated with the fleet state. In fact, utilization is the sum of the travel loaded, travel empty, and loading/unloading states, making its value complementary to the blocked and idle states.

The time spent **traveling loaded** or **empty** depends heavily on the layout configuration. In the **high-complexity layout**, since pick-up and drop-off points are separated and positioned far

apart, the vehicles must travel greater distances to reach their destinations, leading to a higher percentage of empty travel compared to loaded travel.

Similarly, the **blocked state** depends on the number of intersections and the distance between them. In the high-complexity layout, given the large dimensions of the navigation area, vehicles are sent to different zones and encounter each other less frequently, resulting in much lower blocked times for both systems compared to the configurations with lower complexity lever. In this context, for the AMR system, the blocked state remains higher than the AGV because the parking area becomes a highly congested zone due to frequent returns for charging. As fleet size increases across any layout, the probability of collisions rises, generally decreasing the active travel states.

The **idle** state highlights the fundamental operational difference between the two systemss. The AMR's physical constraint of returning to the parking spot to recharge undermines its output but keeps its idle time relatively steady across all three layout complexities. It generally stays within the 22% to 26% range, trending slightly upward with increased variability (with a few outliers in the medium layout caused by the short-cuts). On the other hand, the AGV typically presents very low idle times (in the 0% to 2% range). However, due to variability and clustered job arrivals, this idle time can spike to nearly 30% when the fleet size is maximized under high-variability conditions.

5.3.3 Dynamic environment results

The addition of unforeseen obstacles causes different behaviors in the AGV and AMR systems. It is known that **AGV vehicles** cannot avoid obstacles. If an obstacle blocks their path, they must wait until it moves away. This limitation has several negative impacts on the system's performance. Across the three layout levels, the negative changes in the KPIs are quite constant. Specifically, a high level of variability in the obstacle appearance time drastically reduces performance, while low and medium variability cause almost negligible variation. For example, the output can drop to zero in the worst-case scenario across all three levels. The same thing happens when the frequency of appearance is doubled (AT2). In these scenarios with AT2, the ranges of the confidence intervals are narrower because the block level is extremely high and the vehicles are constantly stopped. Therefore, the AGV system can tolerate low-variability, however, highly unpredictable and frequent obstacle appearances inevitably leads the fleet into a state of total deadlock. Fig. 5.24 illustrates the AGV system in a low configuration in the presence of obstacles. The top boxplot shows the level of output, while bottom graph shows the blocked state.

On the other hand, the **AMR system** is not affected this strongly by the presence of obstacles. By looking at the output, it can be seen that the results in the three levels of variability do not change much compared to the static scenarios. More importantly, there are no significant differences between the two times of appearance, as can be seen in the graphs below (5.26). In this configuration, the blocked state is null, unlike the AGV system (Fig. 5.24)

Actually, in some cases like the medium layout, the AMRs tend to collide less than in the static scenario, as illustrated in the Fig. 5.26. This happens because the obstacles force the vehicles to take different paths. In normal conditions, the AMRs are free to calculate their paths, but, given the simplicity of the layout, they often calculate the same routes and therefore collide. With the presence of random obstacles, the AMRs are forced to change paths every time. However, with a higher frequency (AT2), the obstacles act almost like fixed barriers because they are always present. In that situation, the block levels return to be similar to the static scenario, as the vehicles are forced to travel the same few free paths.

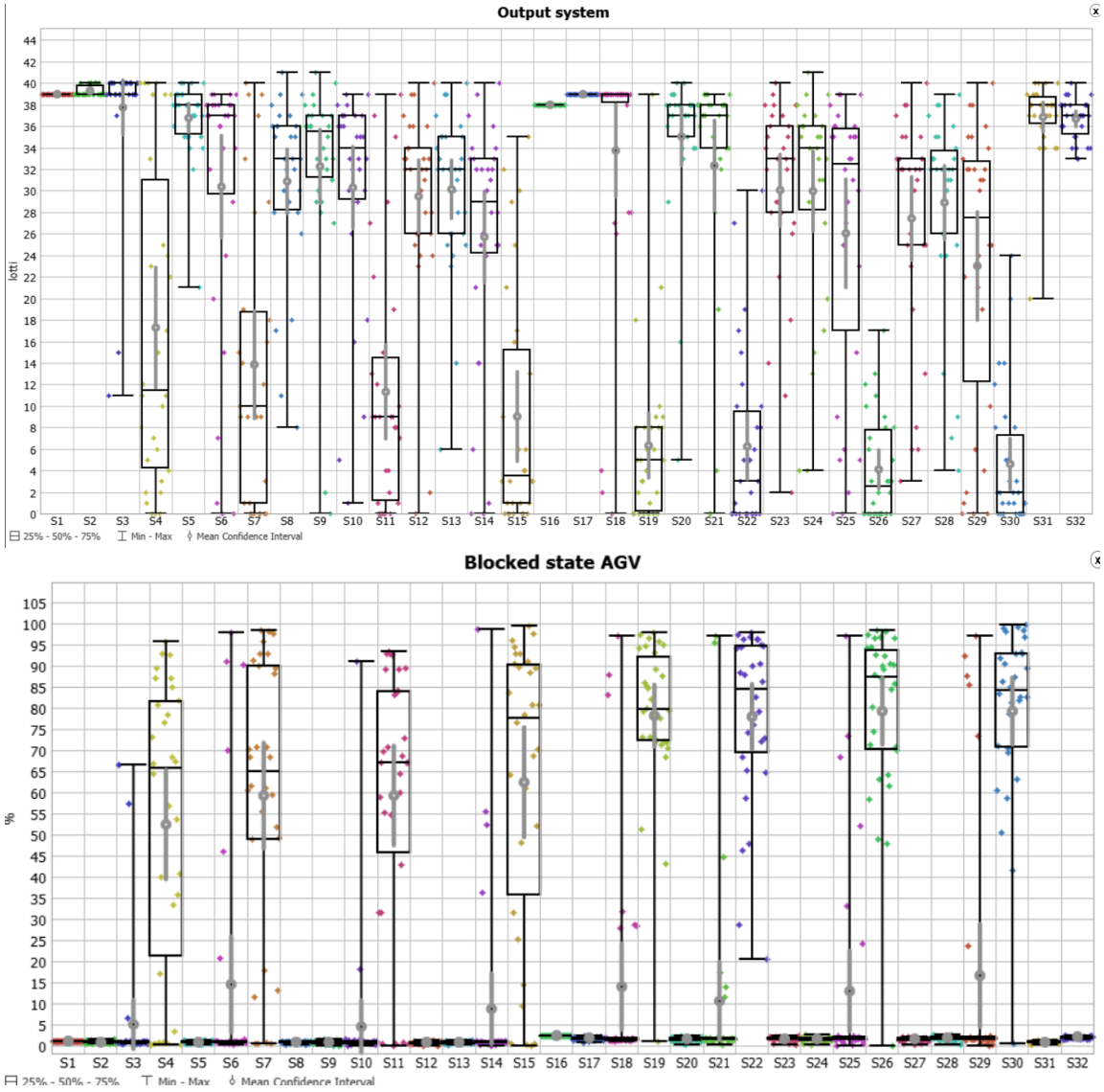


Figure 5.24: Boxplot: AGV low configuration in dynamic setting. Output (top) and blocked state (bottom).

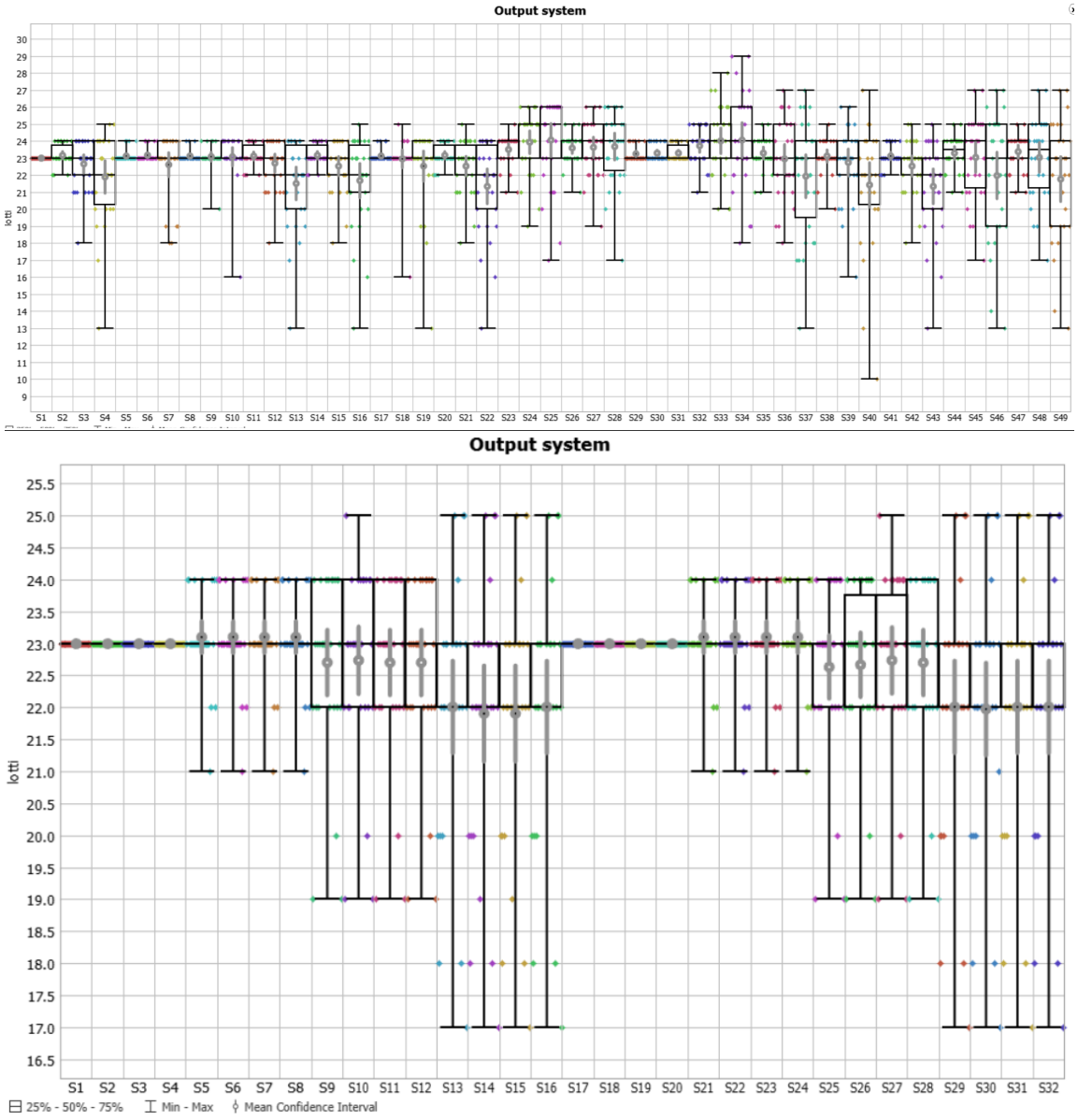


Figure 5.25: Boxplot: AMR low configuration in static (top) and dynamic (bottom) settings.

Overall, the AMRs maintain high performance levels, with some exceptions, but they never reach the significantly low values of the AGVs. In addition, changing the appearance times does not strongly impact the AMRs, which show similar average results even when changing the layout level. The real differences start to appear when the number of vehicles increases. Since the vehicles do not control or communicate with each other, the presence of obstacles makes their behavior much more unpredictable. This lack of coordination can lead to negative results when the fleet is larger.

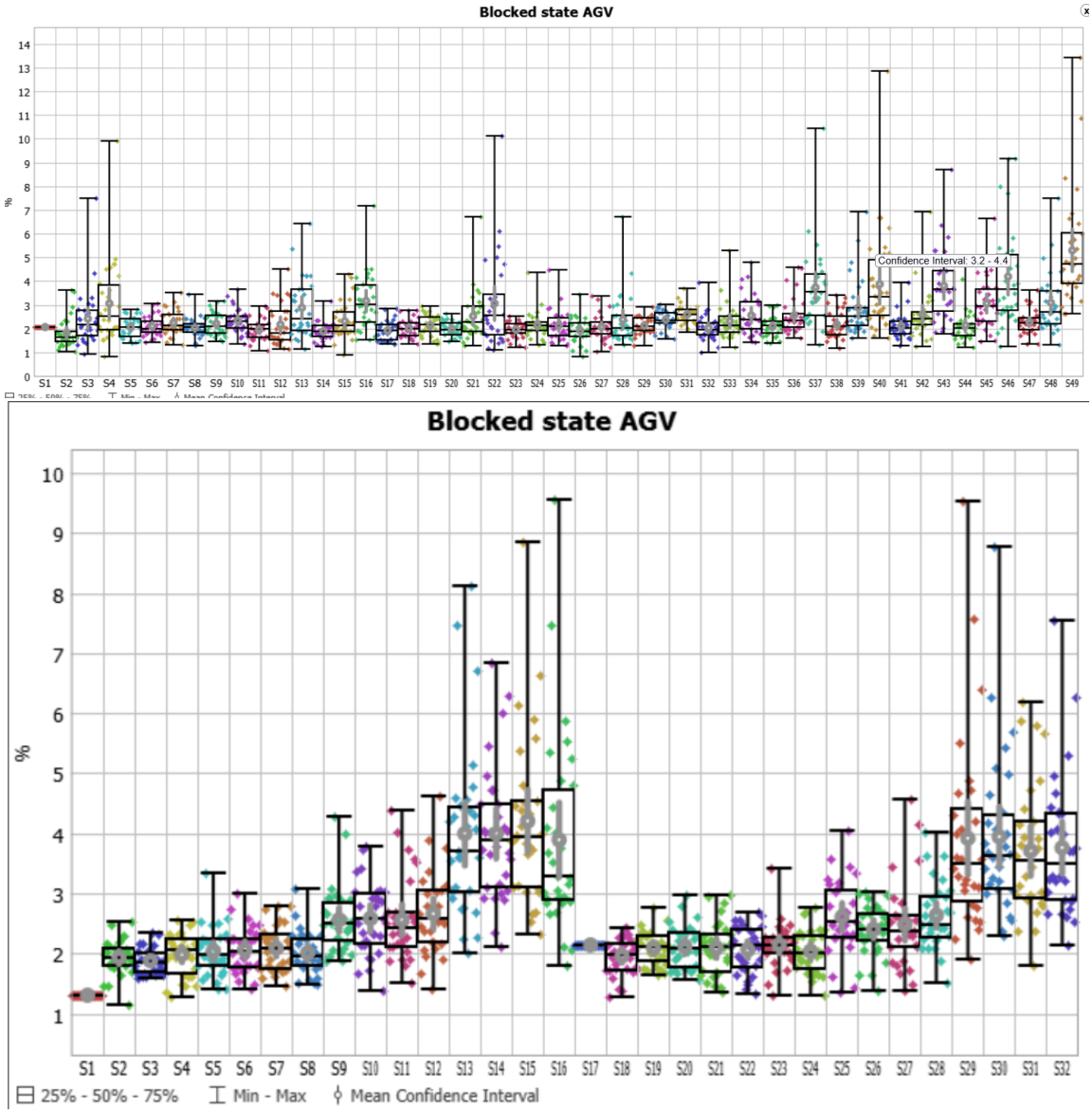


Figure 5.26: Boxplot: AMR blocked state in medium configuration in static (top) and dynamic (bottom) settings with 2 vehicles

Chapter 6

Conclusions

The simulation results highlight that the optimal choice between AGVs and AMRs depends on the layout complexity, the presence of obastacles, and the required fleet size.

In simple layouts that do not require complex navigation logic, and under static conditions or low obstacle variability, the AGV system proves to be the superior option. A single AGV can complete a significantly higher number of jobs (up to 40 batches in the deterministic scenario), which is exactly twice the output of a single AMR. While the AMR is heavily limited by its battery recharging constraints, it can eventually match the AGV's output in simple layouts, but only at the cost of increasing the fleet size.

On the other hand, in more complex layouts characterized by separated and distant pick-up and drop-off points, the AMR becomes the better solution, particularly for single-vehicle fleets. In this configuration, a single AMR completes an average of 11 batches per shift, compared to only 2 for the AGV in the deterministic scenario. For larger fleets, the two systems present comparable output levels, even though the AMR must constantly return to the parking station at the end of each loop.

However, the most significant advantage of the AMR over the AGV becomes evident in the presence of unpredictable obstacles. As demonstrated in the dynamic scenarios, the AMR's performance remains stable and similat to the static environment. For example the output does not get worse even when high variability is introduced across all parameters (from inter-arrival times to obstacle appearances). This is possible because of the AMR's dynamic routing capability, which allows it to actively avoid obstacles that appear in its path. On the other hand, AGVs are forced to stop and wait for their predefined paths to clear, leading to severe bottlenecks and blocked times.

In coclusion, the selection between these two material handling systems requires a comprehensive trade-off analysis. Companies must balance their layout complexity and the frequency of obstacles against their budget, since AMRs are much more expensive to buy than AGVs.

To improve this comparative analysis, there are several aspect that can be further developed . First, the layout complexity index could be improved by incorporating spatial metrics, specifically the physical distances between nodes. While the total number of intersections is a valid metric, the distance between them and the concentration of high-traffic zones shows the true performance of

the system, especially for larger fleets. Two layouts with the same complexity indices can perform very differently if their high-traffic areas are grouped together in one spot instead of being spread out.

Second, the analysis of external obstacles can be developed into a more sophisticated model. The current study focuses primarily on the frequency of obstacle appearance but the true severity of an obstacle also depends on its location within the system. Future research could evaluate scenarios where the location of the obstacles is changed. Lastly, introducing moving obstacles could also provide a more realistic representation of an industrial environment.

Table

Table 1: Labor and Control Providers for Material Handling Methods from [7]

	Capability	
	Labor	Control
Type		
Manual	Human	Human
Mechanized	Machine	Human
Automated	Machine	Machine

Table 2: Material Handling Methods Characteristics from [7]

Characteristic	**Type**		
	Manual	Mechanized	Automated
Weight	Low	High	High
Volume	Low	High	High
Speed	Low	Medium	High
Frequency	Low	Medium	High
Capacity	Low	Medium	High
Flexibility	High	Medium	Low
Acquisition cost	Low	Medium	High
Operating cost	High	Medium	Low

Table 3: Classification of AGV types and their characteristics [8].

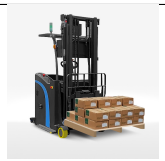
Name	Description	Benefits	Image
Automated Guided Carts (AGC)	The oldest, simplest, and lower-cost kind of AGVs and they are used for light-duty, frequent, and repetitive tasks.	Their simplicity results in a lower initial investment and easier implementation.	
Forklift AGV	Vehicles equipped with a fork and are capable of lifting and depositing loads at varying vertical heights. They are used to perform standard pallet handling tasks, including placing and retrieving pallets from racks or transporting them between conveyor systems and storage areas.	They offer maximum vertical flexibility without requiring major changes to the facility's existing storage infrastructure. They enhance safety by automating difficult stacking tasks and improve inventory accuracy by reducing human error in vertical replacement.	
Towing AGV	Specialized tractors equipped with a hitch or coupling mechanism designed to pull multiple unpowered carts or trailers, forming a "logistics train". They are used to move many items at once along a long, set route, like a train that makes a stop.	High efficiency in mass transport, as they consolidate multiple individual tripos in one. This ability reduces labor costs and ensures synchronized, scheduled delivery, which is excellent for kitting and large-scale assembly line feeding	
Heavy burden carriers	Robust and specialized AGV platforms engineered for the automated handling of exceptionally heavy loads, often exceeding capacities manageable by standard industrial vehicles. These units operate by positioning themselves beneath the load, using integrated lifting systems to engage and transport the material.	Enhance operational safety by automating the handling of high-mass items, and ensure unparalleled positioning accuracy for materials that are too large or heavy for conventional manual or light-duty automated systems	

Table 4: Comparison between AGV Networks and A* Navigation

	AGV Networks	**A* Navigation**
Advantages	• Gives the user more control over task executer travel paths. • Models might run faster (fewer calculations required). • Can restrict travel direction (one-way vs two-way). • Can set specific speed limits on paths. • Allows creation of virtual distances (manipulating node distance vs actual distance).	• Fairly easy to set up; handles most logic automatically. • Easier to configure for models with high numbers of destinations and complex routing options.
Disadvantages	• Takes a slightly longer time to set up. • Creating paths between every possible destination can be cumbersome. • Troubleshooting connectivity issues can be time-consuming.	• In large/complex models, the search algorithm can slow down simulation speed. • Heavy calculation loads can sometimes result in strange movement visuals (lag).

Table 5: **Summary of analyzed papers by category - 1**

Category	Title paper	Ref. Num.	Year	Topic	Results
1	A Comparative Study of Fleet Performance and Flexibility.	[25]	2024	This study, utilizing Rviz and Stage simulations, contrasts the TEA* algorithm for AGVs against the ROS Navigation Stack with a TEB planner for AMRs to evaluate efficiency in dynamic settings.	AGVs demonstrated better collision rates in large fleets due to fixed paths, while AMRs proved more time-efficient in highly dynamic and variable environments.
1, 2	Virtual grid layout with direction constraints for autonomous mobile robot routing performance improvement.	[26]	2023	Addressing the issue of congestion in limited workspaces, this paper uses FlexSim to evaluate and propose "Virtual Grid Layout with Direction Constraints" (VGL-DC). The study compares this constrained model against conventional AGVs and unrestricted AMRs to assess its effectiveness in mitigating collision risks and deadlocks.	The VGL-DC model proved to be the most effective, as it successfully eliminated all deadlocks while achieving a remarkable 39.5% reduction in average delivery time compared to the conventional AGV system.
1	Experimental Evaluation of AGV Dispatching Methods in an Agent-Based Simulation Environment and a Digital Twin.	[27]	2023	A task allocation and dispatching method (synchronous, timer-based) is proposed that is suitable for AMR systems by performing an Agent-Based simulation to compare 3 dispatching methods.	The results were that SMP and LSAP significantly reduce the movement and cost. However, the TH rate did not increase as well.
2	Simulation based approach for supporting automated guided vehicles (AGVs) system design.	[28]	2008	Development of a conceptual and operational methodology utilizing Discrete Event Simulation (DES) to support design and investment decisions for AGV transport systems in EOL logistics.	The DES methodology proppsed proves that the AGV system is the optimal solution to improve and ensure accuracy and reliability of performance assessment.
2	Case study - evaluation of the automation of material handling with mobile robots.	[29]	2020	Introduction of autonomous mobile robots in an automotive parts supplier plant. The strategic objective is to analyze the performance of automating its finished goods material flow and its impact on productivity, safety, and human labor.	Identified the Tandem Layout and a minimum fleet of three robots as the optimal configuration for the specific plant to maximize productivity and safety metrics.
2	Decision-Making Framework For AMR Fleet Size In Manufacturing Environments.	[30]	2025	It is highlighted the key role of simulation in supporting manufacturing and logistics managers in determining the optimal fleet size for AMR. The proposed solution has been developed in 5 scenarios with varying fleet size by combining the real-world fleet management software, which controls the actual robots using TEA* algorithm, directly with FlexSim software simulation.	The results obtained showed that the optimal fleet size is equal to 3 AMRs.
3	A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR's) Operating Environment - A Case Study.	[31]	2021	It highlighted the key role of Digital Twin (DT) technology in supporting companies by testing the operating environment and validating design assumptions for the implementation of an AMR system in a production hall. The methodology involves modeling the system within the Gazebo simulation software, while Rviz is used for the monitoring and visualization of the real AMR's behavior.	It is demonstrated that the DT improved the overall performance, like reducing the travel time, identifying bottlenecks, determining how to modify the physical layout, and optimizing the size of the storage buffers.

Summary of analyzed papers by category - 2

Category	Title paper	Ref. Num.	Year	Topic	Results
3	Effect of Layout Discretization on the Performance of Zone Control-Based Multi-AGV Traffic Management Systems.	[32]	2024	The study proposed a model, using FlexSim, based on Control zone system to manage multi-AGV traffic. The analysis specifically focuses on the performance of two representative traffic management strategies: Dynamic Resource Reservation (DRR) and Improved Dynamic Resource Reservation (IDRR) algorithm.	It demonstrates that the performance of the overall system is influenced by the density of discretization (number and size of control zones): low discretization decreases the TH, while high discretization decrease the cost but it requires higher computational power.
3	A simulated annealing approach for optimizing layout design of reconfigurable manufacturing system based on the workstations' properties.	[33]	2022	The study addresses the challenge of determining the optimal physical layout for a Reconfigurable Manufacturing System (RMS), while considering the properties of the workstations. The authors propose utilizing the Simulated Annealing (SA) metaheuristic.	The SA approach proposed successfully generates near-optimal RMS layouts that offer a notable reduction in total costs compared to conventional search methods.
4	Dynamic AMR Navigation: Simulation with Trajectory Prediction of Moving Obstacles	[39]	2024	It is proposed a fusion between an enhancement of A* algorithm and an artificial potential field (APF) method for local collision avoidance, with the support of advanced LiDAR and stereo camera for real-time environmental perception.	The AMR achieves significantly improved safety and navigational efficiency compared to other systems
4	Independent Optimization for Robot Path Planning and Dynamic Obstacle Avoidance	[40]	2022	It is suggested a coordinated method that separates the deicision on where to go from the decision regarding how to avoid obstacles, so that in this way the robot does not have to recalculate each time an obstacle is moving. Therefore, the approach can be divided in two layers: global planning and local avoidance.	Separating the two optimization decisions leads to safer obstacle avoidance, faster computation time, and more responsive behavior compared to existing coupled methods, which tend to recalculate the path repeatedly upon detecting a new obstacle.
5	A model for assessing the layout structural complexity of manufacturing systems	[41]	2014	It proposed the Layout Complexity Assessment (LCA) model to quantify structural layout complexity. The method utilizes graph theory and adjacency matrices to generate six normalized indices, measuring the decision-making effort required for material flow.	The study validates that the physical structure of a facility can be translated into numerical indices, enabling the objective comparison of different scenarios based on the required control effort.

Table 6: AGV - four level of logic of the advanced process flow.

Level	**Key Question**
Transportation or travel	How do task executers move around?
Task sequence generation	How are task sequences defined?
Job dispatching	Who or what will perform what jobs?
Item flows and routing	Where do the items need to go?

Table 7: Department dimensions across the three levels of layout complexity

Department	**Width (m)**	**Length Low (m)**	**Length Medium (m)**	**Length High (m)**
Supermarket Area (SA)	33	30	42	42
Welding (R1)	33	40	40	40
Standard Painting (R2)	33	42	30	30
Custom Painting (R3)	33	–	20	20
Assembly (R4)	33	30	30	30
Quality Control (QC)	33	30	20	20
Packaging (BC)	33	20	24	24
Manufacturing (MA)	33	28	10	10

Table 8: Value of standard deviation of inter-arrival time in SA.

	Inter-arrival time		
Level of variation	**Coefficient of variation (c)**	**Standard deviation σ (s)**	**Mean value μ (s)**
Low	0.5	300	600
Medium	1.0	600	600
High	1.5	900	600

Table 9: Label assignment based on the input type for each department.

Department	**Input_type label**
R1	1
R2	2
R3	3
R4	4 and 4.2
QC	5
BC	6 and 6.5

Table 10: AGV: Agent system parameters for obstacle detection phases.

Detection Phase	**Coverage Radius (m)**	**Total Frontal FOV**
Broad Phase	4.0	100° (± 50°)
Narrow Phase	1.5	60° (± 30°)

Table 11: Assumptions and main parameters of the AGV system.

Parameter	**Fast AGV**	**Slow AGV**	**Unit**
Physical Parameters			
Vehicle dimensions	1		m^3
Travel Parameters			
Max speed	2		m/s
Acceleration	1		m/s^2
Deceleration	1		m/s^2
Rotation speed	30		°/s
Forward speed			
Straight	2	1	m/s
Curved	1	0.5	m/s
Spur	1	0.5	m/s
Reverse speed			
Straight	1	0.5	m/s
Curved	1	0.5	m/s
Spur	1	0.5	m/s
Rotation thresholds			
Stop and rotate	180		°
Switch to reverse	91		°
Switch to forward	91		°
Battery			
Capacity	100		Ah
Battery use	5		A
Idle use	1		A
Recharge rate	60		A
Loading / Unloading			
Capacity	1		unit
Load time	5		s
Unload time	5		s

Table 12: Standard deviation of loading and unloading time of an AGV.

Loading and unloading time			
Level of variation	**Coefficient of variation (c)**	**Standard deviation σ (s)**	**Mean value μ (s)**
Low	0.5	2.5	5
Medium	1.0	5	5
High	1.5	7.5	5

Table 13: AMR: Docking time at the charging station

Level of variation	**Coefficient of variation (cv)**	**Standard deviation (σ) [s]**	**Mean value (μ) [s]**
Low	0.5	12.5	25
Medium	1.0	25.0	25
High	1.5	37.5	25

Table 14: AMR default parameters

Category	**Input**	**Values**	**Unit**
Travel	Max speed	1	m/s
	Turn speed	90	deg/s
	Acceleration	1	m/s^2
	Deceleration	3	m/s^2
	Dimensions	1	m^3
Loading - Unloading	Capacity	1	unit
	Load time (mean μ)	15	s
	Unload time (mean μ)	15	s

Table 15: AMR: Loading and unloading time

Level of variation	**Coefficient of variation (cv)**	**Standard deviation (σ) [s]**	**Mean value (μ) [s]**
Low	0.5	7.5	15
Medium	1.0	15.0	15
High	1.5	22.5	15

Table 16: Obstacle appearance times and standard deviations for low and medium level configurations.

Group	**Appearance Time (μ)**	**Standard Deviation (low/med/high)**
1 - SA	600	300 / 600 / 900
2 - R1/R4/BC	1200	600 / 1200 / 1800
3 - R2/R3/QC	2100	1050 / 2100 / 3150

Table 17: Obstacle appearance times and standard deviations for high level configuration.

Group	**Appearance Time (μ)**	**Standard Deviation (low/med/high)**
1 - SA	600	300 / 600 / 900
2 - Unload R1/R4/BC	1200	600 / 1200 / 1800
3 - Unload R2/R3/QC	2100	1050 / 2100 / 3150
4 - Load R1/R4/BC	1200	600 / 1200 / 1800
5 - Load R2/R3/QC	4200	2100 / 4200 / 6300

Table 18: AGV system - Static scenarios.

ID	Scenario	σ_{R1}	σ_{R4}	σ_{BC}	σ_{Ld}	σ_{Un}
S1	Deterministic	0	0	0	0	0
S2	σ_{R1} - Low variability	300	0	0	0	0
S3	σ_{R1} - Medium variability	600	0	0	0	0
S4	σ_{R1} - High variability	900	0	0	0	0
S5	σ_{R4} - Low variability	0	300	0	0	0
S6	σ_{R4} - Medium variability	0	600	0	0	0
S7	σ_{R4} - High variability	0	900	0	0	0
S8	σ_{BC} - Low variability	0	0	300	0	0
S9	σ_{BC} - Medium variability	0	0	600	0	0
S10	σ_{BC} - High variability	0	0	900	0	0
S11	σ_{R1+R4} - Low variability	300	300	0	0	0
S12	σ_{R1+R4} - Medium variability	600	600	0	0	0
S13	σ_{R1+R4} - High variability	900	900	0	0	0
S14	σ_{R1+BC} - Low variability	300	0	300	0	0
S15	σ_{R1+BC} - Medium variability	600	0	600	0	0
S16	σ_{R1+BC} - High variability	900	0	900	0	0
S17	σ_{R4+BC} - Low variability	0	300	300	0	0
S18	σ_{R4+BC} - Medium variability	0	600	600	0	0
S19	σ_{R4+BC} - High variability	0	900	900	0	0
S20	σ_{All_Ta} - Low variability	300	300	300	0	0
S21	σ_{All_Ta} - Medium variability	600	600	600	0	0
S22	σ_{All_Ta} - High variability	900	900	900	0	0
S23	σ_{Load} - Low variability	0	0	0	2.5	0
S24	σ_{Load} - Medium variability	0	0	0	5	0
S25	σ_{Load} - High variability	0	0	0	7.5	0
S26	σ_{Unload} - Low variability	0	0	0	0	2.5
S27	σ_{Unload} - Medium variability	0	0	0	0	5
S28	σ_{Unload} - High variability	0	0	0	0	7.5
S29	σ_{Ld+Un} - Low variability	0	0	0	2.5	2.5
S30	σ_{Ld+Un} - Medium variability	0	0	0	5	5
S31	σ_{Ld+Un} - High variability	0	0	0	7.5	7.5
S32	σ_{All_Ta+Ld} - Low variability	300	300	300	2.5	0
S33	σ_{All_Ta+Ld} - Medium variability	600	600	600	5	0
S34	σ_{All_Ta+Ld} - High variability	900	900	900	7.5	0
S35	σ_{All_Ta+Un} - Low variability	300	300	300	0	2.5
S36	σ_{All_Ta+Un} - Medium variability	600	600	600	0	5
S37	σ_{All_Ta+Un} - High variability	900	900	900	0	7.5
S38	All - Low variability	300	300	300	2.5	2.5
S39	All - Medium variability	600	600	600	5	5
S40	All - High variability	900	900	900	7.5	7.5

Legend: *σ: standard deviation;* ***R1, R4, BC****: inter-arrival locations;* ***Ld****: loading;* ***Un****: unloading;* ***Ta****: inter-arrival time.*

Table 19: AMR system - Static scenarios

ID	Scenario	σ_{R1}	σ_{R4}	σ_{BC}	σ_{Ld}	σ_{Un}	σ_{Dk}
S1	Deterministic	0	0	0	0	0	0
S2	σ_{R1} - Low variability	300	0	0	0	0	0
S3	σ_{R1} - Medium variability	600	0	0	0	0	0
S4	σ_{R1} - High variability	900	0	0	0	0	0
S5	σ_{R4} - Low variability	0	300	0	0	0	0
S6	σ_{R4} - Medium variability	0	600	0	0	0	0
S7	σ_{R4} - High variability	0	900	0	0	0	0
S8	σ_{BC} - Low variability	0	0	300	0	0	0
S9	σ_{BC} - Medium variability	0	0	600	0	0	0
S10	σ_{BC} - High variability	0	0	900	0	0	0
S11	σ_{R1+R4} - Low variability	300	300	0	0	0	0
S12	σ_{R1+R4} - Medium variability	600	600	0	0	0	0
S13	σ_{R1+R4} - High variability	900	900	0	0	0	0
S14	σ_{R1+BC} - Low variability	300	0	300	0	0	0
S15	σ_{R1+BC} - Medium variability	600	0	600	0	0	0
S16	σ_{R1+BC} - High variability	900	0	900	0	0	0
S17	σ_{R4+BC} - Low variability	0	300	300	0	0	0
S18	σ_{R4+BC} - Medium variability	0	600	600	0	0	0
S19	σ_{R4+BC} - High variability	0	900	900	0	0	0
S20	σ_{All_Ta} - Low variability	300	300	300	0	0	0
S21	σ_{All_Ta} - Medium variability	600	600	600	0	0	0
S22	σ_{All_Ta} - High variability	900	900	900	0	0	0
S23	σ_{Load} - Low variability	0	0	0	7.5	0	0
S24	σ_{Load} - Medium variability	0	0	0	15	0	0
S25	σ_{Load} - High variability	0	0	0	22.5	0	0
S26	σ_{Unload} - Low variability	0	0	0	0	7.5	0
S27	σ_{Unload} - Medium variability	0	0	0	0	15	0
S28	σ_{Unload} - High variability	0	0	0	0	22.5	0
S29	$\sigma_{Docking}$ - Low variability	0	0	0	0	0	12.5
S30	$\sigma_{Docking}$ - Medium variability	0	0	0	0	0	25
S31	$\sigma_{Docking}$ - High variability	0	0	0	0	0	37.5
S32	σ_{Ld+Un} - Low variability	0	0	0	7.5	7.5	0
S33	σ_{Ld+Un} - Medium variability	0	0	0	15	15	0
S34	σ_{Ld+Un} - High variability	0	0	0	22.5	22.5	0
S35	σ_{All_Ta+Ld} - Low variability	300	300	300	7.5	0	0
S36	σ_{All_Ta+Ld} - Medium variability	600	600	600	15	0	0
S37	σ_{All_Ta+Ld} - High variability	900	900	900	22.5	0	0
S38	σ_{All_Ta+Un} - Low variability	300	300	300	0	7.5	0
S39	σ_{All_Ta+Un} - Medium variability	600	600	600	0	15	0
S40	σ_{All_Ta+Un} - High variability	900	900	900	0	22.5	0
S41	σ_{All_Ta+Dk} - Low variability	300	300	300	0	0	12.5
S42	σ_{All_Ta+Dk} - Medium variability	600	600	600	0	0	25
S43	σ_{All_Ta+Dk} - High variability	900	900	900	0	0	37.5
S44	$\sigma_{All\_Ta+Ld+Un}$ - Low variability	300	300	300	7.5	7.5	0
S45	$\sigma_{All\_Ta+Ld+Un}$ - Medium variability	600	600	600	15	15	0
S46	$\sigma_{All\_Ta+Ld+Un}$ - High variability	900	900	900	22.5	22.5	0
S47	All - Low variability	300	300	300	7.5	7.5	12.5
S48	All - Medium variability	600	600	600	15	15	25
S49	All - High variability	900	900	900	22.5	22.5	37.5

Legend: *σ: standard deviation;* ***Ld****: loading;* ***Un****: unloading;* ***Dk****: docking;* ***Ta****: inter-arrival time.*

Table 20: AGV system - Dynamic scenarios.

ID	**Scenario**	σ_{R1}	σ_{R4}	σ_{BC}	σ_{Ld}	σ_{Un}	μ_1	σ_1	μ_2	σ_2	μ_3	σ_3
D1	Det - AT1 det	0	0	0	0	0	600	0	1200	0	2100	0
D2	Det - AT1 low	0	0	0	0	0	600	300	1200	600	2100	1050
D3	Det - AT1 med	0	0	0	0	0	600	600	1200	1200	2100	2100
D4	Det - AT1 high	0	0	0	0	0	600	900	1200	1800	2100	3150
D5	All Low - AT1 det	300	300	300	2.5	2.5	600	0	1200	0	2100	0
D6	All Low - AT1 low	300	300	300	2.5	2.5	600	300	1200	600	2100	1050
D7	All Low - AT1 med	300	300	300	2.5	2.5	600	600	1200	1200	2100	2100
D8	All Low - AT1 high	300	300	300	2.5	2.5	600	900	1200	1800	2100	3150
D9	All Med - AT1 det	600	600	600	5	5	600	0	1200	0	2100	0
D10	All Med - AT1 low	600	600	600	5	5	600	300	1200	600	2100	1050
D11	All Med - AT1 med	600	600	600	5	5	600	600	1200	1200	2100	2100
D12	All Med - AT1 high	600	600	600	5	5	600	900	1200	1800	2100	3150
D13	All High - AT1 det	900	900	900	7.5	7.5	600	0	1200	0	2100	0
D14	All High - AT1 low	900	900	900	7.5	7.5	600	300	1200	600	2100	1050
D15	All High - AT1 med	900	900	900	7.5	7.5	600	600	1200	1200	2100	2100
D16	All High - AT1 high	900	900	900	7.5	7.5	600	900	1200	1800	2100	3150
D17	Det - AT2 det	0	0	0	0	0	300	0	600	0	1050	0
D18	Det - AT2 low	0	0	0	0	0	300	150	600	300	1050	525
D19	Det - AT2 med	0	0	0	0	0	300	300	600	600	1050	1050
D20	Det - AT2 high	0	0	0	0	0	300	450	600	900	1050	1575
D21	All Low - AT2 det	300	300	300	2.5	2.5	300	0	600	0	1050	0
D22	All Low - AT2 low	300	300	300	2.5	2.5	300	150	600	300	1050	525
D23	All Low - AT2 med	300	300	300	2.5	2.5	300	300	600	600	1050	1050
D24	All Low - AT2 high	300	300	300	2.5	2.5	300	450	600	900	1050	1575
D25	All Med - AT2 det	600	600	600	5	5	300	0	600	0	1050	0
D26	All Med - AT2 low	600	600	600	5	5	300	150	600	300	1050	525
D27	All Med - AT2 med	600	600	600	5	5	300	300	600	600	1050	1050
D28	All Med - AT2 high	600	600	600	5	5	300	450	600	900	1050	1575
D29	All High - AT2 det	900	900	900	7.5	7.5	300	0	600	0	1050	0
D30	All High - AT2 low	900	900	900	7.5	7.5	300	150	600	300	1050	525
D31	All High - AT2 med	900	900	900	7.5	7.5	300	300	600	600	1050	1050
D32	All High - AT2 high	900	900	900	7.5	7.5	300	450	600	900	1050	1575

Legend: *σ: standard deviation; μ: mean;* ***Ld****: loading;* ***Un****: unloading;* ***1****: Appearance SA;* ***2****: Appearance R1, R4, BC;* ***3****: Appearance R2, R3, QC;* ***AT****: appearance time (AT2 is half of AT1);* ***Det****: deterministic;* ***All****: all system parameters have variability.*

Table 21: AMR system - Dynamic scenarios.

ID	Scenario	σ_{R1}	σ_{R4}	σ_{BC}	σ_{Ld}	σ_{Un}	σ_{Dk}	μ_1	σ_1	μ_2	σ_2	μ_3	σ_3
D1	Det - AT1 det	0	0	0	0	0	0	600	0	1200	0	2100	0
D2	Det - AT1 low	0	0	0	0	0	0	600	300	1200	600	2100	1050
D3	Det - AT1 med	0	0	0	0	0	0	600	600	1200	1200	2100	2100
D4	Det - AT1 high	0	0	0	0	0	0	600	900	1200	1800	2100	3150
D5	All Low - AT1 det	300	300	300	2.5	2.5	12.5	600	0	1200	0	2100	0
D6	All Low - AT1 low	300	300	300	2.5	2.5	12.5	600	300	1200	600	2100	1050
D7	All Low - AT1 med	300	300	300	2.5	2.5	12.5	600	600	1200	1200	2100	2100
D8	All Low - AT1 high	300	300	300	2.5	2.5	12.5	600	900	1200	1800	2100	3150
D9	All Med - AT1 det	600	600	600	5	5	25	600	0	1200	0	2100	0
D10	All Med - AT1 low	600	600	600	5	5	25	600	300	1200	600	2100	1050
D11	All Med - AT1 med	600	600	600	5	5	25	600	600	1200	1200	2100	2100
D12	All Med - AT1 high	600	600	600	5	5	25	600	900	1200	1800	2100	3150
D13	All High - AT1 det	900	900	900	7.5	7.5	37.5	600	0	1200	0	2100	0
D14	All High - AT1 low	900	900	900	7.5	7.5	37.5	600	300	1200	600	2100	1050
D15	All High - AT1 med	900	900	900	7.5	7.5	37.5	600	600	1200	1200	2100	2100
D16	All High - AT1 high	900	900	900	7.5	7.5	37.5	600	900	1200	1800	2100	3150
D17	Det - AT2 det	0	0	0	0	0	0	300	0	600	0	1050	0
D18	Det - AT2 low	0	0	0	0	0	0	300	150	600	300	1050	525
D19	Det - AT2 med	0	0	0	0	0	0	300	300	600	600	1050	1050
D20	Det - AT2 high	0	0	0	0	0	0	300	450	600	900	1050	1575
D21	All Low - AT2 det	300	300	300	2.5	2.5	12.5	300	0	600	0	1050	0
D22	All Low - AT2 low	300	300	300	2.5	2.5	12.5	300	150	600	300	1050	525
D23	All Low - AT2 med	300	300	300	2.5	2.5	12.5	300	300	600	600	1050	1050
D24	All Low - AT2 high	300	300	300	2.5	2.5	12.5	300	450	600	900	1050	1575
D25	All Med - AT2 det	600	600	600	5	5	25	300	0	600	0	1050	0
D26	All Med - AT2 low	600	600	600	5	5	25	300	150	600	300	1050	525
D27	All Med - AT2 med	600	600	600	5	5	25	300	300	600	600	1050	1050
D28	All Med - AT2 high	600	600	600	5	5	25	300	450	600	900	1050	1575
D29	All High - AT2 det	900	900	900	7.5	7.5	37.5	300	0	600	0	1050	0
D30	All High - AT2 low	900	900	900	7.5	7.5	37.5	300	150	600	300	1050	525
D31	All High - AT2 med	900	900	900	7.5	7.5	37.5	300	300	600	600	1050	1050
D32	All High - AT2 high	900	900	900	7.5	7.5	37.5	300	450	600	900	1050	1575

Legend: *σ: standard deviation; μ: mean;* ***Ld/Un/Dk****: loading, unloading, docking;* ***1****: App. SA;* ***2****: App. R1-R4-BC;* ***3****: App. R2-R3-QC;* ***AT****: appearance time;* ***Det****: deterministic;* ***All****: all system parameters have variability.*

Bibliography

[1] Ewa Placzek and Kornelia Osieczko-Potoczna. "Current State of Knowledge and Research Needs of Intralogistics". In: *EUROPEAN RESEARCH STUDIES JOURNAL* XXVIII (July 2024), pp. 97–112. DOI: 10.35808/ersj/3425.

[2] Saurabh Vaidya, Prashant Ambad, and Santosh Bhosle. "Industry 4.0 – A Glimpse". In: *Procedia Manufacturing* 20 (2018). 2nd International Conference on Materials, Manufacturing and Design Engineering (iCMMD2017), 11-12 December 2017, MIT Aurangabad, Maharashtra, INDIA, pp. 233–238. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2018.02.034. URL: https://www.sciencedirect.com/science/article/pii/S2351978918300672.

[3] J. Fernandes et al. "Intralogistics and industry 4.0: designing a novel shuttle with picking system". In: *Procedia Manufacturing* 38 (2019). 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing, pp. 1801–1832. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2020.01.078. URL: https://www.sciencedirect.com/science/article/pii/S2351978920300792.

[4] Hannes Winkler and Lena Zinsmeister. "Trends in digitalization of intralogistics and the critical success factors of its implementation". In: *Brazilian Journal of Operations & Production Management* 16 (Aug. 2019), pp. 537–549. DOI: 10.14488/BJOPM.2019.v16.n3.a15.

[5] Philip Kosky et al. "Chapter 11 - Industrial Engineering". In: *Exploring Engineering (Fifth Edition)*. Ed. by Philip Kosky et al. Fifth Edition. Academic Press, 2021, pp. 229–257. ISBN: 978-0-12-815073-3. DOI: https://doi.org/10.1016/B978-0-12-815073-3.00011-9. URL: https://www.sciencedirect.com/science/article/pii/B9780128150733000119.

[6] Aneta Risteska Jankuloska et al. "The Importance Of Material Handling In Logistics System". In: (Oct. 2019).

[7] Marc Goetschalckx. *Logistics Systems Design: Material Handling Systems (Course Lecture Notes)*. Georgia Institute of Technology. Chapter 14, pp. 239-272. 2011.

[8] S. Kammal et al. "A Contemporary Assessment on the Development of Automated Guided Vehicle to the Current Trends and Requirements". In: *Automated Guided Vehicle Systems: Design, Control and Applications*. Springer, 2023, pp. 1–25. DOI: 10.1007/978-3-031-25358-4_1.

[9] M. De Ryck, M. Versteyhe, and F. Debrouwere. "Automated guided vehicle systems, state-of-the-art control algorithms and techniques". In: *Journal of Manufacturing Systems* 54 (2020), pp. 152–173. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2019.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0278612519301177.

[10] Joaquín López, Eduardo Zalama, and Jaime Gómez-García-Bermejo. "A simulation and control framework for AGV based transport systems". In: *Simulation Modelling Practice and Theory* 116 (2022), p. 102430. ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2021.102430. URL: https://www.sciencedirect.com/science/article/pii/S1569190X21001271.

[11] Dimitrios Bechtsis et al. "Sustainable supply chain management in the digitalisation era: The impact of Automated Guided Vehicles". In: *Journal of Cleaner Production* 142 (2017), pp. 3970–3984. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2016.10.057. URL: https://www.sciencedirect.com/science/article/pii/S0959652616316675.

[12] Tharma Ganesharajah, Nicholas G. Hall, and Chelliah Sriskandarajah. "Design and operational issues in AGV-served manufacturing systems". In: *Annals of Operations Research* 76.0 (1998). Accessed: 2025-12-27, pp. 109–154. DOI: 10.1023/A:1018936219150. URL: https://doi.org/10.1023/A:1018936219150.

[13] Kaiyu Li et al. "Towards Path Planning Algorithm Combining with A-Star Algorithm and Dynamic Window Approach Algorithm". In: *International Journal of Advanced Computer Science and Applications (IJACSA)* 14.6 (2023). DOI: 10.14569/IJACSA.2023.0140655. URL: http://dx.doi.org/10.14569/IJACSA.2023.0140655.

[14] Lixiang Zhang et al. "Automated guided vehicle dispatching and routing integration via digital twin with deep reinforcement learning". In: *Journal of Manufacturing Systems* 72 (2024), pp. 492–503. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2023.12.008. URL: https://www.sciencedirect.com/science/article/pii/S0278612523002558.

[15] Ruizhong Wu et al. "A Lifelong Conflict-Aware AGV Routing System". In: *Databases Theory and Applications*. Ed. by Tong Chen et al. Singapore: Springer Nature Singapore, 2025, pp. 447–462.

[16] Andor Bálint Viharos and István Németh. "Simulation and scheduling of AGV based robotic assembly systems". In: *IFAC-PapersOnLine* 51.11 (2018). 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018, pp. 1415–1420. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.08.317. URL: https://www.sciencedirect.com/science/article/pii/S2405896318314411.

[17] Giuseppe Fragapane et al. "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda". In: *European Journal of Operational Research* 294.2 (2021), pp. 405–426. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2021.01.019. URL: https://www.sciencedirect.com/science/article/pii/S0377221721000217.

[18] Otto Motors. "AGV vs AMR: A Comparison of Automated Material Transport". In: (2017). URL: https://ottomotors.com/blog/amr-vs-agv-a-comparison-of-automated-material-transport.

[19] Thorge Lackner et al. "Review of autonomous mobile robots in intralogistics: state-of-the-art, limitations and research gaps". In: *Procedia CIRP* 130 (2024). 57th CIRP Conference on Manufacturing Systems 2024 (CMS 2024), pp. 930–935. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2024.10.187. URL: https://www.sciencedirect.com/science/article/pii/S2212827124013441.

[20] Md. A. K. Niloy et al. "Critical Design and Control Issues of Indoor Autonomous Mobile Robots: A Review". In: *IEEE Access* 9 (2021), pp. 35338–35370. DOI: 10.1109/ACCESS.2021.3062557.

[21] Robert E. Shannon. "INTRODUCTION TO SIMULATION". In: *Proceedings of the 1992 Winter Simulation Conference*. New York, NY, USA: ACM, 1992, pp. 3–12.

[22] D. Mourtzis, M. Doukas, and D. Bernidaki. "Simulation in Manufacturing: Review and Challenges". In: *Procedia CIRP* 25 (2014). 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution, pp. 213–229. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2014.10.032. URL: https://www.sciencedirect.com/science/article/pii/S2212827114010634.

[23] Magdalena Dobrzańska and Paweł Dobrzański. "Simulation Model as an Element of Sustainable Autonomous Mobile Robot Fleet Management". In: *Energies* 18.8 (2025). ISSN: 1996-1073. DOI: 10.3390/en18081894. URL: https://www.mdpi.com/1996-1073/18/8/1894.

[24] FlexSim Software Products, Inc. *Welcome to FlexSim*. FlexSim 25.2 User Manual. Consultato: 2026-01-28. 2025. URL: https://docs.flexsim.com/en/25.2/Introduction/Welcome/Welcome.html.

[25] Rita Tomé Silva et al. "AGVs vs AMRs: A Comparative Study of Fleet Performance and Flexibility". In: *2024 7th Iberian Robotics Conference (ROBOT)*. 2024, pp. 1–6. DOI: 10.1109/ROBOT61475.2024.10797381.

[26] Inhye Bang, Byung-In Kim, and Yonggu Kim. "Virtual Grid Layout with Direction Constraints for Autonomous Mobile Robot Routing Performance Improvement". In: *The International Journal of Advanced Manufacturing Technology* 128.9–10 (2023), pp. 4653–4665. DOI: 10.1007/s00170-023-12219-w. URL: https://doi.org/10.1007/s00170-023-12219-w.

[27] Fabian Maas Genannt Bermpohl, Andreas Bresser, and Malte Langosz. "Experimental Evaluation of AGV Dispatching Methods in an Agent-Based Simulation Environment and a Digital Twin". In: *Applied Sciences* 13.10 (2023). ISSN: 2076-3417. DOI: 10.3390/app13106171. URL: https://www.mdpi.com/2076-3417/13/10/6171.

[28] Elisa Gebennini et al. "A simulation based approach for supporting Automated Guided Vehicles (AGVs) systems design". In: *2008 Winter Simulation Conference*. 2008, pp. 2156–2163. DOI: 10.1109/WSC.2008.4736314.

[29] Adriana Melo and Lindsay Corneal. "Case study: evaluation of the automation of material handling with mobile robots". In: *International Journal of Quality Innovation* 6 (June 2020), p. 3. DOI: 10.1186/s40887-020-00037-y.

[30] Catarina Rema et al. "Decision-Making Framework For AMR Fleet Size In Manufacturing Environments". In: *2025 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2025, pp. 70–77. DOI: 10.1109/ICARSC65809.2025.10970155.

[31] Paweł Stączek et al. "A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR's) Operating Environment—A Case Study". In: *Sensors* 21.23 (2021). ISSN: 1424-8220. DOI: 10.3390/s21237830. URL: https://www.mdpi.com/1424-8220/21/23/7830.

[32] Parikshit Verma, Josep M. Olm, and Raúl Suárez. "Effect of Layout Discretization on the Performance of Zone Control-Based Multi-AGV Traffic Management Systems". In: *Applied Sciences* 14.17 (2024). ISSN: 2076-3417. DOI: 10.3390/app14177817. URL: https://www.mdpi.com/2076-3417/14/17/7817.

[33] JB. Clarion et al. "A simulated annealing approach for optimizing layout design of reconfigurable manufacturing system based on the workstation properties". In: *IFAC-PapersOnLine* 55.10 (2022). 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022, pp. 1657–1662. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2022.09.635. URL: https://www.sciencedirect.com/science/article/pii/S2405896322019528.

[34] Bai Li et al. "Fast Trajectory Planning for AGV in the Presence of Moving Obstacles: A Combination of 3-dim A* Search and QCQP". In: *2021 33rd Chinese Control and Decision Conference (CCDC)*. 2021, pp. 7549–7554. DOI: 10.1109/CCDC52312.2021.9602686.

[35] Daniel Teso-Fz-Betoño et al. "A Free Navigation of an AGV to a Non-Static Target with Obstacle Avoidance". In: *Electronics* 8.2 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8020159. URL: https://www.mdpi.com/2079-9292/8/2/159.

[36] Shunyi Yao et al. "Crowd-Aware Robot Navigation for Pedestrians with Multiple Collision Avoidance Strategies via Map-based Deep Reinforcement Learning". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 8144–8150. DOI: 10.1109/IROS51168.2021.9636579.

[37] Te Wang et al. "Global Dynamic Path Planning of AGV Based on Fusion of Improved A* Algorithm and Dynamic Window Method". In: *Sensors* 24.6 (2024). ISSN: 1424-8220. DOI: 10.3390/s24062011. URL: https://www.mdpi.com/1424-8220/24/6/2011.

[38] Qiao Liu, Yong-gang Lu, and Cun-xi Xie. "Fuzzy Obstacle-avoiding Controller of Autonomous Mobile Robot Optimized by Genetic Algorithm under Multi-obstacles Environment". In: *2006 6th World Congress on Intelligent Control and Automation*. Vol. 1. 2006, pp. 3255–3259. DOI: 10.1109/WCICA.2006.1712969.

[39] Tomás Cadete et al. "Dynamic AMR Navigation: Simulation with Trajectory Prediction of Moving Obstacles". In: *2024 7th Iberian Robotics Conference (ROBOT)*. 2024, pp. 1–7. DOI: 10.1109/ROBOT61475.2024.10797420.

[40] Terrance Hall et al. "Independent Optimization for Robot Path Planning and Dynamic Obstacle Avoidance". In: *2022 IEEE 19th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*. 2022, pp. 198–201. DOI: 10.1109/HONET56683.2022.10019217.

[41] H. ElMaraghy et al. "A model for assessing the layout structural complexity of manufacturing systems". In: *Journal of Manufacturing Systems* 33.1 (2014), pp. 51–64. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2013.05.012. URL: https://www.sciencedirect.com/science/article/pii/S0278612513000794.

[42] Karel Joseph Kansky. *Structure of transportation networks: relationships between network geometry and regional characteristics*. Research Paper No. 84. University of Chicago, Department of Geography, 1963, pp. 62–67.

[43] FlexSim Software Products, Inc. *Using the AGV Process Flow Template - FlexSim 25.2 Documentation*. 2026. URL: https://docs.flexsim.com/en/25.2/WorkingWithTasks/AGVNetworks/UsingAGVProcessFlowTemplate/UsingAGVProcessFlowTemplate.html (visited on 03/08/2026).

[44] FlexSim Software Products, Inc. *Proximity Behavior - FlexSim 25.2 Documentation*. 2026. URL: https://docs.flexsim.com/en/25.2/Reference/Tools/AgentSystem/ProximityBehavior/ProximityBehavior.html (visited on 03/08/2026).

[45] FlexSim Software Products, Inc. *Key Concepts About Travel - FlexSim 24.1 Documentation*. 2024. URL: https://docs.flexsim.com/en/24.1/WorkingWithTasks/Travel/KeyConceptsTravel/KeyConceptsTravel.html (visited on 03/08/2026).