applied sciences

MDPI

Article

Dynamic Reactive Assignment of Tasks in Real-Time Automated Guided Vehicle Environments with Potential Interruptions

Xabier A. Martin [1], Sara Hatami [2], Laura Calvet [3], Mohammad Peyman [4] and Angel A. Juan [1,5,*]

1 Department of Applied Statistics and Operations Research, Universitat Politècnica de València, 03801 Alcoy, Spain; xamarsol@upv.es
2 Department of Management, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain; sara.hatami@upc.edu
3 Department of Telecommunication and Systems Engineering, Autonomous University of Barcelona, 08202 Sabadell, Spain; laura.calvet.linan@uab.cat
4 Department of Computer Science, Multimedia and Telecommunication, Universitat Oberta de Catalunya, 08018 Barcelona, Spain; mpeyman@uoc.edu
5 Department of Management, Euncet Business School, 08225 Terrassa, Spain
* Correspondence: ajuanp@upv.es

Abstract: An efficient management of production plants has to consider several external and internal factors, such as potential interruptions of the ongoing processes. Automated guided vehicles (AGVs) are becoming a widespread technology that offers many advantages. These AGVs can perform complex tasks in an autonomous way. However, an inefficient schedule of the tasks assigned to an AGV can suffer from unwanted interruptions and idle times, which in turn will affect the total time required by the AGV to complete its assigned tasks. In order to avoid these issues, this paper proposes a heuristic-based approach that: (i) makes use of a delay matrix to estimate circuit delays for different daily times; (ii) employs these estimates to define an initial itinerary of tasks for an AGV; and (iii) dynamically adjusts the initial agenda as new information on actual delays is obtained by the system. The objective is to minimize the total time required for the AGV to complete all the assigned tasks, taking into account situations that generate unexpected disruptions along the circuits that the AGV follows. In order to test and validate the proposed approach, a series of computational experiments utilizing real-life data are carried out. These experiments allow us to measure the improvement gap with respect to the former policy used by the system managers.

Keywords: automated guided vehicles; dynamic task assignment; internal logistics; heuristic optimization; disruption management

check for updates

Citation: Martin, X.A.; Hatami, S.; Calvet, L.; Peyman, M.; Juan, A.A. Dynamic Reactive Assignment of Tasks in Real-Time Automated Guided Vehicle Environments with Potential Interruptions. *Appl. Sci.* **2023**, *13*, 3708. https://doi.org/10.3390/app13063708

Academic Editors: Paweł Sitek and Jarosław Wikarek

Received: 7 February 2023
Revised: 7 March 2023
Accepted: 13 March 2023
Published: 14 March 2023

1. Introduction

As pointed out by Zhang and Chen [1], the concept of Industry 4.0 brings the idea of automation to industrial systems by employing computerized, controlled, and monitored processes as well as industrial machinery and equipment. These improvements are favored by technological advances in different fields, such as robotics, artificial intelligence (AI), and autonomous vehicles, among many others. Manufacturers proceed to integrate new technologies and techniques, including Internet of Things (IoT), cloud computing, data analytics, and machine learning, into their production facilities and throughout their operations. There are applications in a wide range of fields such as transportation [2], wireless sensor networks [3], or integrated circuit design [4], just to name a few. These technologies and techniques collect data in an easy and agile way, which leads to the control of production plants and the forecasting of possible events that might have a negative impact on productivity.

Different factors, both internal and external, need to be considered in order to empower plant management at different levels. For instance, process automation is a key factor in this new digital era. An appropriate control of these factors facilitates decision-making in the most complex production processes. Smart factories are equipped with advanced sensors, embedded software, and robotics such as driverless mobile robots or AGVs. These are capable of transporting small loads (those oriented to the medical or picking sector) as well as large loads of several tons (those oriented to large-volume logistics, construction, or mining industries). AGVs perform as fully automatic devices, i.e., driverless and without manual activities being performed by human operators. AGVs have become an integral part of Industry 4.0 [5]. In recent years, the use of AGVs in the industrial sector has raised a great deal of interest. Although AGVs have been used for more than 25 years already [6], their popularity is still increasing day by day due to the significant reduction in production costs they allow and a noticeable diminishing in price. Among the main benefits and advantages that AGVs provide, we can cite: improving productivity, increasing safety and security for plant operators, reducing damage to structures and products during load handling, being able to work in unfavorable environments (cold chambers, ovens, etc.), improving the operational efficiency of production lines, and the enhancement of stocks and inventory systems management. AGVs offer a large economic potential due to their lower maintenance expenditure compared to conventional vehicles and their capability to function with minimum labor cost and human intervention, as well as a range of benefits across environmental and social sustainability dimensions [7].

The fundamental components of each AGV include the physical vehicle, a localization guidance system, a control system, and a communication system [8]. The two last systems enable the AGV to execute highly complex maneuvers at an individual level by recognizing the environment. The routes that the AGV follows within the production plants are affected by external factors, such as collaboration with other vehicles, machinery, or operators which may generate interruptions in the scheduled activities of the AGV. In this circumstance, the control and communication systems enable the AGV to communicate with the external factors to avoid crashes, and also allow it to communicate with other industrial robots, so that they can coordinate better. Wireless communication technologies used in AGVs allow them to react to different obstacles in a dynamic industrial environment, including facing other AGVs, operators, etc. Some types of AGVs stop when encountering a moving obstacle and wait for the obstacle to disappear, while other types change their path and perform a path re-planning to avoid collision with the obstacle. In any case, these interruptions lead to a delay in the estimated arrival of the AGV to its destination, which disrupts and affects the optimal operation of the production plant [9].

The idea of this research is inspired by a real case. With the goal of minimizing total time required in completing a series of tasks, production managers need to determine the order of tasks assigned to an AGV under dynamic conditions. Hence, the associated travel and processing times are dynamic in the sense that they vary with time. This is due to potential disruptions in some tasks that might occur at different times during the working hours. The problem under consideration can be viewed as a generalized (dynamic) version of the single-machine scheduling problem, with the AGV serving as a single machine. Because most single-machine scheduling problems are NP-hard [10], the dynamic version considered here might require a fast and reactive heuristic approach if real-time decisions are to be made using dynamic inputs. Hence, this paper proposes an ″agile″ optimization algorithm [11] for the mobile robotics AGV problem. The proposed algorithm is an extremely fast reactive heuristic procedure that aims at minimizing the total time required by an AGV to complete all the assigned tasks under a dynamic scenario with potential disruptions. Thus, our reactive approach makes use of a dynamic matrix which accounts for such delays. In order to build the aforementioned matrix, we employ historical data to estimate the delay associated with a given circuit at any point in time. Therefore, our reactive approach uses estimated delays and dynamically adjusts the initial agenda as new information on potential delays is obtained by the system. We study the creation of the permutation of tasks for the proposed reactive heuristic and compare it to other approaches. In a nutshell, our proposed reactive algorithm does the following: (i) it makes use of a delay matrix to estimate circuit delays for different daily times; (ii) it employs these estimates to define an initial itinerary of tasks for an AGV; and (iii) it dynamically adjusts the initial agenda as new information on actual delays is obtained by the system.

The optimization problem is depicted in Figure 1. Let us consider a warehouse layout where each product has to be processed in different workstations. Each product can visit these stations in any order. Initially, all products are located at a depot station. An AGV has to move each product (one at a time) from the depot station to each workstation and then, once it has been processed, bring the product back to the depot station. The expected time required to complete a circuit (i.e., the sequence depot station–specific working station–depot station) is given by the sum of the travel times in ideal conditions plus the sum of the expected delays throughout the circuit. Unfortunately, delay times are dynamic, i.e., they are a function of the system conditions at each moment in time. The objective is to assign and re-assign the itineraries of the tasks to be completed by the AGV in order to minimize the total time required for the AGV to complete all the assigned tasks.

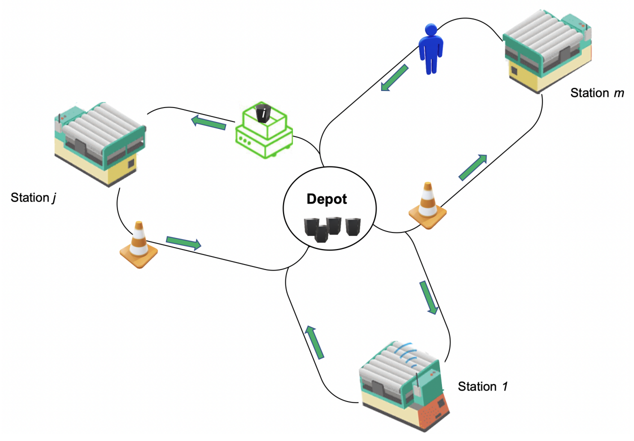

Figure 1. Visual representation of the AGV optimization problem being considered.

To the best of our knowledge, the scientific literature has not previously investigated minimizing the total time employed by an AGV in completing a series of tasks under a dynamic scenario in which disruptions in the form of unexpected or random delay times might occur while the AGV is moving along defined pathways. We propose a method for determining the order of tasks assigned to an AGV using estimated circuit delays and dynamically adjusting the initial agenda as new information on actual delays becomes available. Hence, the main contributions of our work can be described as follows: (i) the introduction of a single AGV task order determination with unexpected interruption events in order to minimize the total time or makespan; (ii) the use of data analytics techniques to estimate the AGV interruption and caused delay for each determined pathway during determined time-spans; and (iii) the development of a reactive heuristic [12,13], studying the resulting permutations of tasks generated using three different approaches. The remaining part of the paper is organised as follows. Section 2 reviews related work. Section 3 provides a formal description of the problem being tackled. Section 4 proposes a heuristic-based approach to solve the industrial problem. Section 5 illustrates the methodological concepts with an example. Section 6 includes the computational experiments, while Section 7 provides an analysis of the results obtained. Finally, the paper is completed in Section 8 with key conclusions.

2. Overview on AGVs and Related Problems

New manufacturing strategies must be oriented towards flexible, agile, and efficient production chains that allow for obtaining quality products. Internal logistics represent an important factor in terms of competitiveness and economy. This is mainly due to the dynamism of the current market that has caused globalization. The objective of today's manufacturing companies is focused on reducing idle times and zero reserve stock. This improves the company's reliability in the market, which in turn increases the availability of its products [14]. The application of advanced methods for planning and anticipating events in a production plant, as well as the level of interconnection of the different systems, is relatively low in practice [15].

This section first describes a database search identifying works related to AGVs, focusing on those referring to, at least, one of the following four types of popular and challenging decision-making processes: assignation, routing, scheduling, and sequencing. Afterwards, a representative sample of recent works on sequencing or scheduling processes related to AGVs is reviewed.

2.1. Database Search

The number of works on AGVs has been growing during the last three decades, but the increase has been higher since 2016. Figure 2 shows some results from Scopus using the following query: TITLE-ABS-KEY ("Automated Guided Vehicle") AND (LIMIT-TO (DOCTYPE, "ar")) AND (LIMIT-TO(LANGUAGE, "English")). This query searched Scopus-indexed articles in English, including the term "Automated Guided Vehicle" either in the title, abstract, or keywords. Subfigure (a) reveals the evolution of the number of documents from 1984 to 2022. This number was around 25 until 2016, when it started to grow until reaching its peak in 2021 with more than 125 articles. Subfigure (b) indicates the percentage of documents by subject area. The bigger area corresponds to ''Engineering" (38.7%, 1112), followed by ''Computer Science" (21.8%, 627), ''Decision Sciences" (9.8%, 283), ''Business, Management and Accounting" (8.0%, 229), and ''Mathematics" (7.4%, 214). These 5 areas account for almost 86% of the documents. Subfigure (c) identifies the 10 countries/territory with the highest number of documents. The United States and China have more than 200 documents each. In contrast, Japan achieves a number slightly lower than 100. South Korea, India, Germany, Iran, Taiwan, Canada, and the United Kingdom complete the list.

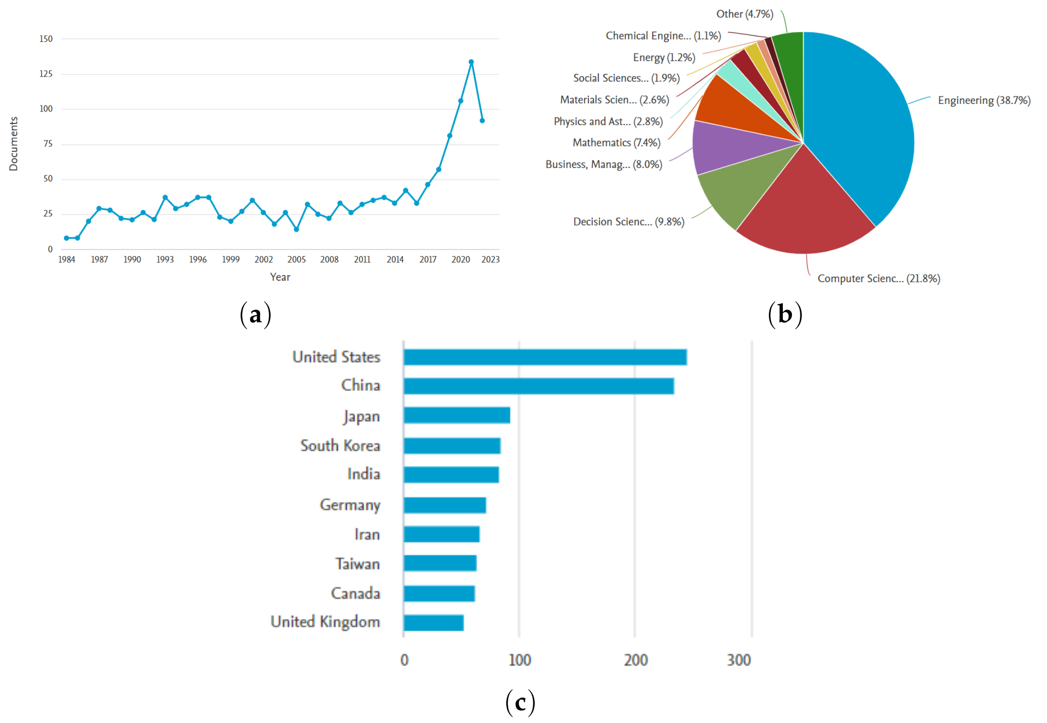

Figure 2. Analysis from articles based on AGVs. (**a**) Documents by year. (**b**) Documents by subject area. (**c**) Documents by country/territory.

A more specific search is carried out using this query: TITLE-ABS-KEY (“Automated Guided Vehicle”) AND (TITLE-ABS-KEY(assign*)) AND (LIMIT-TO(DOCTYPE, “ar”)) AND (LIMIT-TO(LANGUAGE, “English”)). Thus, this new query is more restrictive and requires the root word “assign*” (i.e., assign, assignation, assigning, etc.). Three more queries are created replacing “assign*” by “rout*”, “schedul*”, and “sequenc*”. The analysis of documents by year is displayed in Figure 3. The four series have a positive trend. The number of works by year on AGVs and sequencing is similar to the number of works on AGVs and assignation. The number of works on AGVs and routing is higher, but lower than those on AGVs and scheduling.

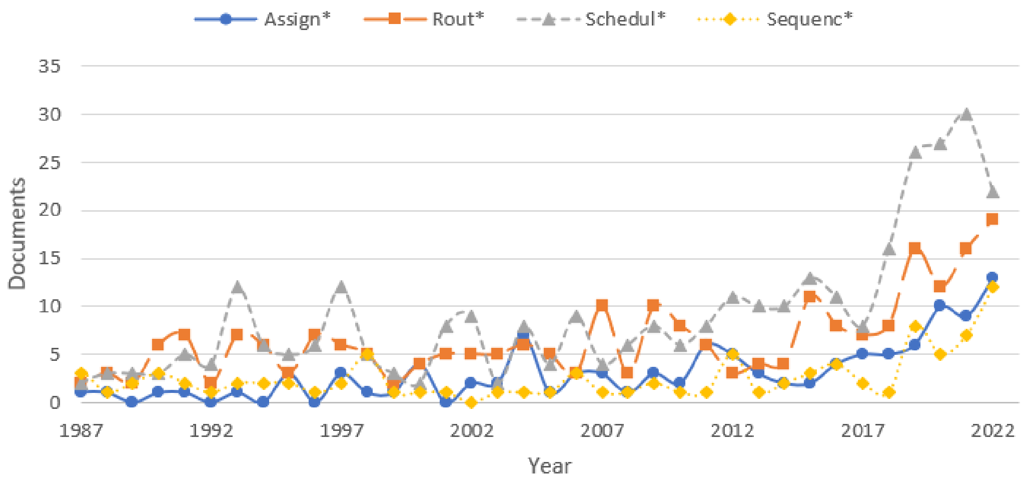

Figure 3. Analysis from articles based on AGVs and assignation/routing/scheduling/sequencing.

2.2. Related Work

Li et al. [16] describe the robotic task sequencing Problem as follows. Let there be an AGV initially located at its parking position P_0. It receives n tasks, indexed by $1, 2, \ldots, n$, where task i requires the AGV to move some goods from position B_i to position D_i. Starting from position P_0, the AGV should finish all the tasks and, then, return to that parking position. The main goal is to determine the task execution sequence $S = (P_0, s_1, s_2, \ldots, s_n, P_0)$, where s_i denotes the index of the task executed in the i-th place, so as to minimize the total distance (where any distance is considered deterministic and known in advance). The authors propose transforming this problem to an equivalent asymmetric traveling salesman problem and apply the 2-opt movements (which constitutes a basic operator used by local-search based heuristics), developing data structures able to reduce the above time complexity from $O(n^3)$ to $O(n^2)$. Experimental results based on simulations are shown to illustrate the approach. Li et al. [17] explore the tasks assigning and sequencing problem of several capacitated multiple-load AGVs. A mathematical model is formulated which describes three objective functions to minimize: the total travel distance, the standard deviation of workload, and the standard deviation of the difference between the latest delivery time and the predicted time of tasks. The solving approach relies on an improved harmony search algorithm, which adopts dynamic changing harmony memory considering rate parameters and implementing neighborhood search strategy. A case study based on a manufacturing enterprise is presented.

Zhong et al. [18] present the integrated scheduling of multi-AGV with conflict-free path planning. First, a mixed-integer programming model is formulated to analyze an AGV scheme that combines scheduling and path planning in an automated container terminal considering that the task allocation is known. The aim is to minimize AGVs delay time. A set of experiments are carried out to validate the solving approach, which relies on a hybrid genetic algorithm (GA) and a particle swarm optimization (PSO) algorithm. Zou et al. [19] address the multiple AGV dispatching problem, which involves two types of decisions: assigning tasks to AGVs and sequencing the tasks of each AGV. The main goal is to minimize the transportation cost (which adds travel cost, penalty cost for violating time and AGV cost). The problem is formulated as a mixed integer linear programming model. A discrete artificial bee colony algorithm is presented. Five neighborhood operators are proposed to obtain diverse neighboring solutions, as well as theorems to avoid unfeasible solutions, new control parameters, and an insertion-based local search method. The authors collected a number of instances with up to 50 tasks from Foxconn Technology Group (an advanced electronics equipment manufacturing enterprise in China). Interestingly, the instances are split into test instances and calibrating instances, and all are publicly available. The proposed algorithm is compared against a heuristic, the Gurobi solver and four existing metaheuristics. The Gurobi solver is only able to obtain solutions for small instances if the computational time employed is limited (5 s). The other methods are coded in C++ language. The results proof the efficiency of the proposed approach. Zou et al. [20] study the multi-compartment AGV scheduling problem, aiming to minimize the total cost, which includes the travel cost, the service cost, and the cost of vehicles. A mixed-integer linear programming model is formulated and a novel iterated greedy algorithm, which makes use of accelerations for evaluating solutions, is designed. Zou et al. [21] address the AGV scheduling problem with pickup and delivery and present a novel multi-objective evolutionary algorithm to solve it. The goal is to maximize customer satisfaction while minimizing distribution cost. A multi-objective mixed-integer linear programming model is formulated. A set of computational experiments, based on real-world instances, are described.

Mousavi et al. [22] focus on the multi-objective scheduling of AGVs, where the aim is to allocate AGVs to tasks, minimizing makespan and number of AGVs while considering the AGVs' battery charge. A mathematical model is presented and integrated with different evolutionary algorithms: GA, PSO, and hybrid GA-PSO. According to the results of the computational experiments, the hybrid GA-PSO outperforms the other algorithms. Evaluation and validation of the approach is carried out by simulation via the FlexSim simulation software [23]. More recently, Dang et al. [24] address the problem of scheduling transport requests on multi-load and multi-ability AGVs with battery management. Each request has a soft time window and a priority. The specific decisions involved are: assign transport and charging requests to AGVs, sequence them, and determine the arrival times and charging duration. The aim is to minimize the tardiness costs of requests and travel costs of AGVs. The authors present a mixed-integer linear programming model and, as a solving methodology, a hybrid adaptive large neighborhood search. This approach is illustrated by means of an industry case study using real-world data. Finally, Singh et al. [25] study the problem of scheduling AGVs with battery constraints. Each transport has an associated soft time window and each AGV has specific capabilities and travel costs. The AGV batteries can be recharged partially if required. The decisions involved in this problem are: to assign the transport and charging requests to AGVs, sequence the requests, and determine their starting times and recharging durations of the AGVs. The aim is to minimize a weighted sum of the tardiness costs of transport requests and travel costs of AGVs. The authors present a mixed-integer linear programming model and a novel matheuristic to address the problem. Finally, an industry case study is described.

While most works rely on heuristics/metaheuristics [26], there are also some relying on reinforcement learning [27]. For instance, Xue et al. [28] deal with a multi-AGV flow-shop scheduling problem, where each AGV moves along fixed tracks, transporting products between successive machines. As in many flow-shop scheduling problems, the usual goal is to minimize the average job delay and total makespan, for which many different types of algorithms can be employed [29,30]. The authors formulate this scheduling problem as a Markov problem by defining state features, actions space and a reward function. Numerical experiments are carried out to assess the performance of the proposed approach.

All in all, there is an increasing number of works studying the sequencing of tasks in AGV-related problems. Table 1 gathers the main features of the works related described before. Most of these works describe challenging restrictions related to the use of batteries or the capabilities of the AGVs (e.g., capacitated multiple-load AGVs), or different objective functions. Many authors formulate a mathematical model and present a solving approach for a case study based on a real-life manufacturing enterprise. While the need to consider unexpected events and dynamic routing times has been acknowledged in some works, there is a gap in the literature regarding the use of simple yet efficient heuristics that feed from historical data and are capable to react, in real-time, to unexpected disruptions in the AGV paths.

Table 1. Related work.

Work	Problem's Name	Goal	Restrictions	Methodology	Experiments
Li et al. [16]	Robotic Task Sequencing Problem	Min. total distance	-	Efficient 2-opt local search	Based on simulations
Li et al. [17]	Tasks Assigning and Sequencing Problem of Multiple AGVs	Min. total distance, standard deviation of workload, and standard deviation of the difference between latest delivery time and predicted time of tasks	Capacitated multiple-load AGVs	Improved harmony search algorithm	Case in a real-world manufacturing enterprise of producing back cover of smart phone in China
Zhong et al. [18]	Integrated Scheduling of Multi-AGV with Conflict-Free Path Planning	Min. AGVs' delay time	Multiple AGVs; Safe distance; Capacitated road; Task allocation known	Hybrid genetic algorithm-particle swarm optimization with fuzzy logic controller to adaptive auto tuning	Based on Xiamen Ocean Gate port in China
Zou et al. [19]	Multiple AGV Dispatching Problem	Min. transportation cost including travel cost, penalty cost for violating time and AGV cost	Multiple AGVs	Discrete artificial bee colony algorithm	Based on Foxconn Technology Group in China
Zou et al. [20]	Multi-Compartment AGV scheduling problem	Min. total cost including travel cost, service cost, and cost of vehicles involved	Multiple AGVs; Multiple compartments	Iterated greedy algorithm	Based on an electronic equipment manufacturing enterprise in China
midrulehline Zou et al. [21]	AGV Scheduling Problem with Pickup and Delivery	Max. customer satisfaction and min. distribution cost	Multiple AGVs; Goods handling process in a matrix manufacturing workshop with multi-variety and small-batch production	Evolutionary algorithm	Based on an electronic equipment manufacturing enterprise in China
Mousavi et al. [22]	Multi-objective AGV Scheduling	Min. makespan and number of AGVs while considering AGVs' battery charge	Multiple AGVs with unit-load capacity	Hybrid genetic algorithm and particle swarm optimization	Numerical examples
Dang et al. [24]	Scheduling Heterogeneous Multi-load AGVs	Min. tardiness costs of requests and travel costs	Multiple AGVs; Battery Constraints	Hybrid adaptive large neighborhood search	Based on Brainport Industries Campus in the Netherlands
Singh et al. [25]	Scheduling AGVs	Min. tardiness costs of requests and travel costs	Multiple AGVs; Battery Constraints; Soft Time Windows; Heterogeneous fleet	Matheuristic relying on an adaptive large neighborhood search algorithm and a linear program	Based on Brainport Industries Campus in the Netherlands
Xue et al. [28]	Multi-AGV Flow-shop Scheduling Problem	Min. average job delay and total makespan	Multiple AGVs	Reinforcement learning method	Based on simulations

3. Formal Description of the Problem

Let us consider a warehouse layout where each product $i \in I$ has to be processed by different workstations, $j \in J$. A typical working day has a final time $t_f > 0$. There is a depot station, 0, where all products are located initially. An AGV has to move each product $i \in I$ (one at the time) from depot 0 to each workstation $j \in J$ and then, once the product i has been processed in j, bring it back to depot 0. Given a product $i \in I$ and a workstation $j \in J$, the actual time required by i to complete a circuit $0 \rightarrow j \rightarrow 0$, $T_{ij} \geq 0$, is given by the sum of the travel and processing times in ideal conditions, $t^*_{ij} > 0$, plus the sum of the actual delays throughout the circuit, $D_{ij} \geq 0$. In other words: $T_{ij} = t^*_{ij} + D_{ij}$, where D_{ij} is a function of time $t \in [0, t_f)$, i.e., $D_{ij} = D_{ij}(t)$. Notice that delay times are dynamic, i.e., the delays are a function of the system conditions at each moment in time $t \in [0, t_f)$. It is precisely this dynamic behavior that makes the optimization problem a non-trivial one. In addition, completing any single circuit $0 \rightarrow j \rightarrow 0$ should take no more than a user-defined time threshold $t_0 > 0$. Under these circumstances, the objective is to minimize the total time required for the AGV to complete all the assigned tasks. The problem can mathematically be defined as follows:

$$\min \sum_{i \in I} \sum_{j \in J} T_{ij} \tag{1}$$

subject to:

$$T_{ij} = t^*_{ij} + D_{ij} \qquad \forall i \in I, \forall j \in J \tag{2}$$

$$D_{ij} = D_{ij}(t) \qquad \forall i \in I, \forall j \in J, \forall t \in [0, t_f] \tag{3}$$

$$T_{ij} \leq t_0 \qquad \forall i \in I, \forall j \in J \tag{4}$$

4. Solving Methodology

The solving methodology presents a heuristic-based approach to solve the problem described in Section 3. We consider two relatively simple yet effective and extremely fast heuristics. These heuristics assume we can estimate the expected delay associated with each circuit at any moment in time, being this delay a function of the current system conditions. This is achieved by creating a dynamic matrix of expected delays that uses historical data to estimate the delay associated with a given circuit under a specific configuration of the environmental conditions. In other words, the dynamic matrix provides, for the current time, the updated estimates for the delays associated with each possible circuit. Algorithm 1 presents the pseudo-code for our first heuristic, which is a non-reactive one.

Algorithm 1 Non-Reactive Heuristic for Dynamic Delays

```
1:  Data: List of tasks $T$, and matrix of expected delays $D$
2:  Result: Permutation of completed tasks $P$
3:  $P \leftarrow \emptyset$
4:  $t_c \leftarrow 0$
5:  while $T \neq \emptyset$ do
6:      $T \leftarrow \text{estimateDelays}(T, D, t_c)$
7:      $T \leftarrow \text{sort}(T)$
8:      $t \leftarrow \text{selectTask}(T)$
9:      $P \leftarrow P \cup \{t\}$
10:     $t_c \leftarrow \text{updateSystem}(t_c, t)$
11: end while
12: return $P$
```

The non-reactive heuristic algorithm receives the following input parameters: (i) the list of assigned tasks T a particular AGV needs to complete; and (ii) the dynamic matrix of expected delays D. More specifically, each task in the list contains a product-workstation pair, along with the time needed by the AGV to complete the associated circuit. The dynamic matrix contains the estimated delays associated with each circuit for each daily time. The non-reactive heuristic works as follows: first, an empty permutation of tasks P is constructed (line 3). The elements from the list of tasks will be added to this permutation in the order the assigned tasks are completed. In addition, the system conditions track the current time t_c, which is initialized to the starting time $t = 0$. Next, the list of tasks to be completed T is iterated until no more tasks are in the list (lines 5–11). In each iteration, the delays of each circuit are estimated based on the system conditions (line 6). More specifically, the delays for each circuit are extracted from the dynamic matrix of estimated delays using the current time t_c. In addition, the estimated delay of each circuit is added to the time each task in the list of tasks needs to complete the particular circuit it needs to traverse. Hence, we compute the estimated total time needed by the AGV to complete each assigned task. The list of tasks is then sorted from the lowest to the highest estimated delay (line 7). The task with the lowest delay t is extracted from the list (line 8). This task is then added to the end of the permutation of completed tasks (line 9). Once the completed task is added to the permutation, the system conditions are updated, which adds the estimated total time needed to complete the assigned task to the system variable that tracks the current time (line 10). Once the list of assigned tasks to be completed is empty, the constructed permutation of completed tasks is returned (line 12).

The non-reactive heuristic presented here selects the task with the lowest estimated delay when the circuit is traversed, which constitutes the delays estimated from historical data. However, the delays encountered when the circuit is actually traversed might be different than the estimated ones. The second heuristic, which is described next, solves this issue by employing a reactive approach: taking into account the current system conditions, which are updated at the end of each circuit. Algorithm 2 presents the pseudo-code of our reactive heuristic.

Algorithm 2 Reactive Heuristic for Dynamic Delays

```
1:  Data: List of tasks $T$, and matrix of expected delays $D$
2:  Result: Permutation of completed tasks $P$
3:  $P \leftarrow \emptyset$
4:  $t_c \leftarrow 0$
5:  while $C \neq \emptyset$ do
6:      $T \leftarrow \text{estimateDelays}(T, D, t_c)$
7:      $T \leftarrow \text{sort}(T)$
8:      $t \leftarrow \text{selectTask}(T)$
9:      $t \leftarrow \text{executeTask}(t)$
10:     $P \leftarrow P \cup \{t\}$
11:     $t_c \leftarrow \text{updateSystem}(t_c, t)$
12: end while
13: return $P$
```

The reactive heuristic approach, which receives the same input parameters as the non-reactive heuristic for dynamic delays, works as follows. First, an empty permutation of tasks P is constructed (line 3). Additionally, the system conditions track the current time t_c, which is initialized to the starting time $t = 0$. Next, the list of tasks to be completed T is iterated until no more tasks are in the list (lines 5–11). In each iteration, the delays of each circuit are estimated based on the system conditions, and the list of tasks is then sorted from the lowest to the highest estimated delay (lines 6 and 7). The first task from the new permutation of tasks t is extracted, and the task is executed by the AGV (lines 8 and 9). The expected total travel time is then updated with the actual time the AGV takes to travel throughout the circuit. In other words, we compute the real total time needed by the AGV to complete task t, taking into account the actual delays encountered along the way. Next, the task is added to the end of the permutation of completed tasks (line 10). Finally, the system conditions are updated, which adds the actual time needed to complete the task to the system variable that tracks the current time (line 10), not the estimated time needed to complete the task. Once the list of assigned tasks to be completed is empty, the constructed permutation of completed tasks is returned (line 12).

Notice that the reactive heuristic proposed here selects the task with the lowest estimated delay at each iteration, just as the non-reactive heuristic. However, the reactive heuristic updates the system conditions with the real total time the AGV needs to complete the circuit. Also, it recomputes the permutation of remaining tasks according to the current system conditions in subsequent iterations. This gives the reactive heuristic the ability to adapt to potential disruptions.

5. An Illustrative Example of Our Reactive Approach

This section describes a simple example that illustrates how the proposed reactive algorithm solves the problem, in an iterative way, as new data is updated in the system at the end of each round trip (i.e., each time the AGV returns from a station to the depot). Figure 4 shows a toy case with 2 products and 2 workstations. Let us denote by $f_{ij}(t)$ a function that provides an estimate of the real round trip time associated with assigning product i to station j at time $t \in [0, t_f)$, t_{ij}^t, where t_f refers to the end time of a working day. The main concept here is: (i) we will use $f_{ij}(t)$ to design an initial assignment plan based on estimated travel times; and (ii) every time the AGV returns to the depot, since the actual times might be different than the estimated ones, the remaining assignments will be recomputed using $f_{ij}(t)$ again. In other words, each time we pass through the depot (and while the list of pending assignments is not empty), the current time $t_c > 0$ will be registered and the previous assignment plan will be re-adjusted (from t_c to t_f), taking into account the estimates provided by $f_{ij}(t)$. Hence, the figure shows a first assignment plan (E1) where: (i) product 1 (P1) is first sent to station 2 (S2) and then returned to the depot (0) once processed; (ii) product 2 is taken to station 1 and then returned to the depot; (iii) product 2 is taken to station 2 and returned to the depot; and (iv) product 1 is taken to station 1 and returned to the depot. Would the estimated travel times be perfectly accurate (i.e., without delays), this plan would finish by time t_{E1}. Unfortunately, as we put this plan in action, we might experience changes in the estimated travel times (usually in the form of delays), as shown in the real execution plan R1, which might lead to a new estimated makespan t_{R1}. At this point in time, when product P1 has just returned to the depot, we cannot do anything to change the fact that product 1 has already been processed in station 2. However, we can re-adjust our initial plan taking into account the current time t_c and $f_{ij}(t)$. This is how plan E2 emerges. Notice that plan E2 is able, at least in theory, to enhance the makespan associated with R1. This process of comparing the estimated travel times with the real ones, and then re-adjusting future trips if convenient, is repeated until all products have been processed by all stations. In our case, this leads to a final makespan t_{R3}, which is not as good as t_{E1} but definitively better than t_{R1} thanks to our ability to reactively adjust the plan according to the current system conditions (current time every time the AGV visits the depot and the estimation function, which depends on the time t at which a trip is launched). One should notice that this is just a toy example, but in a situation in which the number of products n is large, and the number of stations m also grows, the size of the possible solutions space is vast and the problem suffers from combinatorial explosion. In addition, when several AGVs and depots are considered, the problem becomes even more challenging.

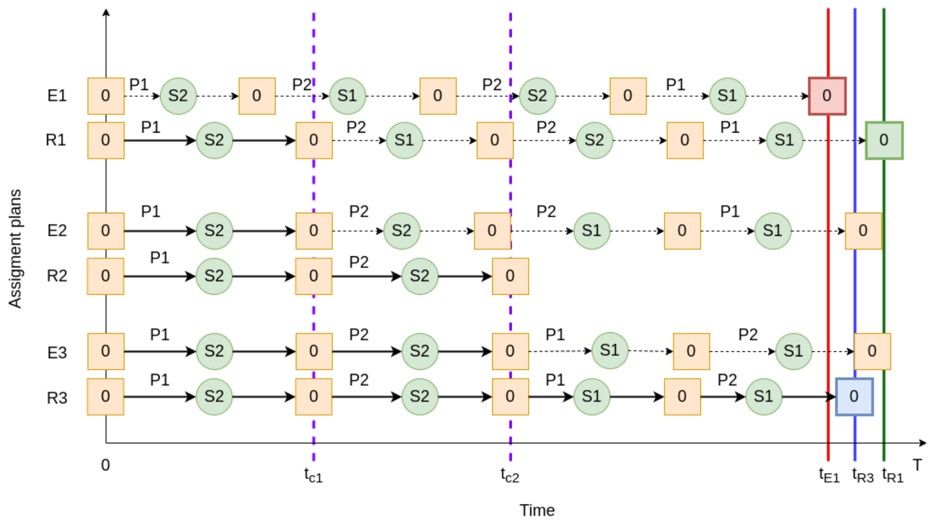

Figure 4. An illustrative description of the proposed algorithms.

6. Computational Experiments

The proposed heuristic approaches have been implemented using Python 3.10 [31], and tested on a workstation with a multi-core processor Intel Xeon E5-2630 v4 and 32 GB of RAM. The remainder of this section explains the computational experiments carried out to validate and illustrate our reactive heuristic. First, the test instances employed are described. Afterwards, we describe in detail the three different approaches used to test the instances, along with the method used to provide the actual delays of the circuits. For the computational experiments, each generated instance has been run 10 times for each approach. Each run has been executed with a different seed for the random number generator used for simulating the actual delays of the circuits (which are based on historical data), and the average are reported for different levels of dynamism. In particular, we have which introduced a low, medium, and high level of dynamism, respectively.

6.1. Test Instances

Since there are no benchmark instances publicly available for the problem we have presented, a set of instances have been created to assess the performance and the quality of the proposed algorithm. We used real historical data from a company, a single dataset gathered over several days by an AGV to conduct our testing. This dataset included various variables, including date, time, circuit, destination node, and location (X and Y coordinates). The date and time represent the starting time of the AGV and are updated every second from 5 a.m. to 11 p.m. over a span of seven days. The dataset consists of six circuits, each of which accomplished a specified task. The AGV contacts various nodes designated as the destination node during each circuit, and the X and Y coordinates denote the AGV's location at a specific time.

To facilitate our methodology, we extracted useful features such as distance and filtered the data where the distance was zero to record the times and locations where the AGV stopped due to specific circumstances. We then computed the delay time per hour for each circuit, and a dynamic matrix table was created to feed our methodology. This matrix includes the combination of the average delay time of each pair of circuits per hour that is intended to execute a particular task.

We defined four different instances, namely *small*, *medium*, *large*, and *very large*, which describe scenarios where three stations must process 30, 60, 120, and 240 products, respectively. Additionally, the travel time with no delay for each circuit is in seconds, while the starting time of the AGV is set at 5 a.m.

6.2. Test Approaches

The three approaches considered in our computational experiments are described as follows:

- FIFO: In this approach, the list of tasks to be completed is iterated in a first-in, first-out (FIFO) manner without taking into account the estimated delays for each of the circuits as well. Once the list of tasks is iterated, the actual delays associated with the FIFO permutation of tasks are computed.
- Non-Reactive: In this approach, the permutation of tasks is created using the non-reactive heuristic for dynamic delays presented in a previous section. Thus, the task with the lowest expected delay when traversing its circuit is added to the permutation of tasks at each iteration. Subsequently, the actual delays associated with the permutation of tasks are computed.
- Reactive: Regarding the last approach, the permutation of tasks is created using the reactive heuristic algorithm described in Section 4. Thus, a new permutation of remaining tasks along with their respective actual delays are computed, and the system is updated with the actual time it takes to traverse each of the circuits.

In our numerical experiments, we employ white noise [32] in order to emulate the behavior of the actual delay values that the AGV will encounter when traversing each circuit. In particular, each sample generated from white noise has a normal distribution with zero mean, which is added to the estimated delays to emulate the actual delay values. In other words, the actual delays are simulated by adding a white Gaussian noise sample to the estimated delays. During the experiments, the samples are generated with mean $\mu = 0$ and standard deviation σ. The standard deviation σ allows us to consider different levels of dynamism in our experiments, as lower values of σ indicate a lower dynamism and higher values of σ produce a high level of dynamism. In our experiments, we have set σ to the values 100, 200, and 300 for a low, medium, and high level of dynamism, respectively.

7. Analysis of the Results

The analysis of the results from the computational experiments has been carried out using R [33]. Tables 2–4 present the results for the different levels of dynamism, from the lowest to the highest. The first column identifies the instances, while the remaining columns show the total time (in seconds) required by an AGV to complete all the assigned tasks for the three considered approaches. First, we report the results obtained for the FIFO approach, where the tasks are completed in a FIFO manner. In the next section of the table, we report the obtained results for the non-reactive approach. The first column shows the total time when the tasks are completed in the order returned by the non-reactive heuristic for dynamic delays, while the second column compares the average FIFO solutions (OBF) with the average non-reactive solutions (OBN). The last section of the table exhibits the obtained results for the reactive approach. The first column shows the total time when the tasks are completed in the order returned by the reactive heuristic, while the second column compares the average FIFO solutions (OBF) with the average reactive solutions (OBR). Finally, the average values are shown in the last row.

Table 2. Computational results for the test instances with a low level of dynamism.

Instance	FIFO Approach	Non-Reactive Approach		Reactive Approach	
	OBF [1]	**OBN [2]**	**GAP(%) [1–2]**	**OBR [3]**	**GAP(%) [1–3]**
Small	86,735.7	82,403.1	−5.00	82,236.4	−5.19
Medium	173,105.4	166,014.1	−4.10	164,528.6	−4.95
Large	346,699.7	335,395.6	−3.26	328,056.4	−5.38
Very large	691,926.7	677,652.9	−2.06	655,090.7	−5.32
Average:	324,616.9	315,366.4	−3.6	307,478.0	−5.2

Table 3. Computational results for the test instances with a medium level of dynamism.

Instance	FIFO Approach	Non-Reactive Approach		Reactive Approach	
	OBF [1]	**OBN [2]**	**GAP(%) [1–2]**	**OBR [3]**	**GAP(%) [1–3]**
Small	87,921.7	83,161.8	−5.41	82,761.6	−5.87
Medium	175,289.6	168,075.4	−4.12	165,001.2	−5.87
Large	351,553.2	338,148.2	−3.81	329,533.5	−6.26
Very large	699,450.7	681,942.1	−2.50	656,655.8	−6.12
Average:	328,553.8	317,831.9	−4.0	308,488.0	−6.0

Table 4. Computational results for the test instances with a high level of dynamism.

	FIFO Approach	Non-Reactive Approach		Reactive Approach	
Instance	**OBF [1]**	**OBN [2]**	**GAP(%) [1–2]**	**OBR [3]**	**GAP(%) [1–3]**
Small	90,084.8	84,662.7	−6.02	83,959.6	−6.80
Medium	179,177.4	171,094.5	−4.51	166,810.5	−6.90
Large	360,197.0	344,949.7	−4.23	334,807.1	−7.05
Very large	716,942.5	690,075.2	−3.75	665,469.4	−7.18
Average:	336,600.4	322,695.5	−4.6	312,761.6	−7.0

As expected, all the percentage gaps are negative, which indicates that the non-reactive and reactive approaches outperform the FIFO approach. The objective function value increases with the size of the instance, as increasing the number of assigned tasks requires more time from the AGV to complete them. The percentage gaps of the solutions obtained with the non-reactive and reactive approaches with respect to the FIFO approach tend to be high in absolute terms. In particular, the percentage gaps of the solutions obtained with the reactive approach with respect to the FIFO approach are the highest. In other words, the reactive approach takes the least time to complete the assigned tasks, so the reactive approach can be considered the most effective approach to solve the described problem. The barplot in Figure 5 shows the average gaps of non-reactive, and reactive approaches with respect to the FIFO approach considering the different levels of dynamism. Notice that the mean gap is always higher, in absolute terms, for the reactive approach. In addition, the average gaps increase as the level of dynamism grows. Thus, it is particularly useful to apply a reactive approach in a dynamic environment subject to unexpected disruptions in the delay times. Specifically, the reactive approach allows for the AGV to react to unexpected disruptions when traversing the environment, and alter the assignment of the tasks to minimize the total time. This is achieved because the reactive approach considers the actual delays happening as the AGV completes the assigned tasks, not only estimating the delays that could happen to the AGV while completing the assigned tasks.

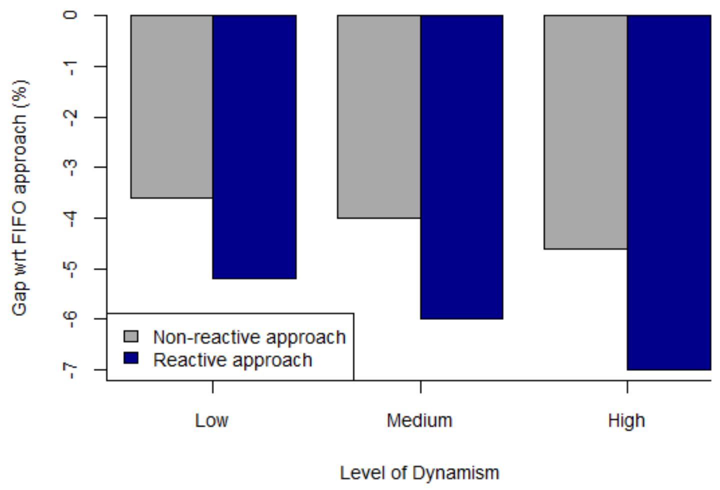

Figure 5. Barplot of the mean gaps (%) of non-reactive and reactive approaches with respect to the FIFO approach considering different levels of dynamism (from low to high).

Next, an analysis of variance (ANOVA) is used to analyze the differences among means. The gap is the dependent variable, and the approach, the dynamism level, and the instance are the independent variables. Table 5 displays the results obtained. The conclusion of this analysis is that there are statistically significant differences among the means of the different approaches, and dynamism levels, but not among the means of the different instances. Finally, the Tukey multiple comparisons of means among approaches (95% family-wise confidence level) are applied to find means that are significantly different from each other. Table 6 shows the results obtained. Notice that there are significant differences between the non-reactive and the reactive approaches, as the *p*-value is lower than 0.05.

Table 5. Results from the ANOVA analysis.

	Df	Sum Sq	Mean Sq	F Value	Pr (>F)
Approach	1	24.24	24.24	50.37	0.00000
Level of Dynamism	2	7.88	3.94	8.19	0.00324
Instance	3	4.56	1.52	3.16	0.05181
Residuals	17	8.18	0.48		

Table 6. Tukey multiple comparisons of means (95% family-wise confidence level).

Pair of Approaches	Difference	Lower	Upper	P Adjusted
Reactive—Non-Reactive	−2.01	−2.6075	−1.4125	0.00000

8. Conclusions

To the best of our knowledge, this paper is the first attempt to minimize the total time employed by an AGV in completing a series of tasks under a dynamic scenario in which disruptions in the form of unexpected or random delay times might occur while the AGV is moving along defined pathways. We have presented a reactive heuristic, which is combined with a dynamic matrix of expected delays. This matrix is built using historical data to estimate the delay associated with a given circuit at any point in time. Additionally, we study the creation of the permutation of tasks for the proposed reactive heuristic and compare it to three other different approaches.

Data analytics techniques are used to estimate the AGV interruption for each determined pathway during determined time spans. White Gaussian noise is also employed to artificially generate scenarios with circuit delays. Different levels of dynamism (low, medium, and high) are considered in creating the circuit delays. Computational evaluations were performed with four groups of small, medium, large and very large instances and ANOVA and Tukey tests were used to analyze results. These results show that the FIFO approach works worse than all the other ones, while the mean gap provided by the reactive approach is always is higher in absolute terms. The gaps increase as the level of dynamism grows, which confirms the usefulness of the proposed reactive approach in a dynamic environment as the one considered here for the AGVs.

For future works, more than one AGV can be considered. In addition, machine learning models can be used to predict delays from historical data. Finally, simulation-optimization methods such as simheuristic [34] may also be explored.

Author Contributions: Conceptualization, A.A.J. and S.H.; methodology, X.A.M.; software, X.A.M.; validation, L.C. and M.P.; writing—original draft preparation, X.A.M., L.C. and M.P.; writing—review and editing, S.H. and A.A.J.; project administration, S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by the Spanish Ministry of Industry, Commerce and Tourism (AEI-010500-2021b-54), the EU Comission (HORIZON-CL4-2021-TWIN-TRANSITION-01-07, 101057294 AIDEAS), and the Generalitat Valenciana (PROMETEO/2021/065).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, C.; Chen, Y. A review of research relevant to the emerging industry trends: Industry 4.0, IoT, blockchain, and business analytics. *J. Ind. Integr. Manag.* **2020**, *5*, 165–180. [CrossRef]
2. Barreto, L.; Amaral, A.; Pereira, T. Industry 4.0 implications in logistics: An overview. *Procedia Manuf.* **2017**, *13*, 1245–1252. [CrossRef]
3. Joby, P. An Extensive Research on Acoustic Underwater Wireless Sensor Networks (AUWSN). *IRO J. Sustain. Wirel. Syst.* **2022**, *4*, 121–129. [CrossRef]
4. Madhura, S. A Review on Low Power VLSI Design Models in Various Circuits. *J. Electron.* **2022**, *4*, 74–81. [CrossRef]
5. Javed, M.A.; Muram, F.U.; Hansson, H.; Punnekkat, S.; Thane, H. Towards dynamic safety assurance for Industry 4.0. *J. Syst. Archit.* **2021**, *114*, 101914. [CrossRef]
6. Vis, I.F. Survey of research in the design and control of automated guided vehicle systems. *Eur. J. Oper. Res.* **2006**, *170*, 677–709. [CrossRef]
7. Bechtsis, D.; Tsolakis, N.; Vlachos, D.; Iakovou, E. Sustainable supply chain management in the digitalisation era: The impact of Automated Guided Vehicles. *J. Clean. Prod.* **2017**, *142*, 3970–3984. [CrossRef]
8. Zhan, M.; Yu, K. Wireless communication technologies in automated guided vehicles: Survey and analysis. In Proceedings of the IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 21–23 October 2018; pp. 4155–4161.
9. Zhao, Y.; Liu, X.; Wang, G.; Wu, S.; Han, S. Dynamic resource reservation based collision and deadlock prevention for multi-AGVs. *IEEE Access* **2020**, *8*, 82120–82130. [CrossRef]
10. Lenstra, J.K.; Kan, A.R.; Brucker, P. Complexity of machine scheduling problems. In *Annals of Discrete Mathematics*; Elsevier: Amsterdam, The Netherlands, 1977; Volume 1, pp. 343–362.
11. Martins, L.d.C.; Tarchi, D.; Juan, A.A.; Fusco, A. Agile optimization for a real-time facility location problem in Internet of Vehicles networks. *Networks* **2022**, *79*, 501–514. [CrossRef]
12. Raba, D.; Estrada-Moreno, A.; Panadero, J.; Juan, A.A. A reactive simheuristic using online data for a real-life inventory routing problem with stochastic demands. *Int. Trans. Oper. Res.* **2020**, *27*, 2785–2816. [CrossRef]
13. Tordecilla, R.D.; Copado-Méndez, P.J.; Panadero, J.; Martins, L.d.C.; Juan, A.A. An agile and reactive biased-randomized heuristic for an agri-food rich vehicle routing problem. *Transp. Res. Procedia* **2021**, *58*, 385–392. [CrossRef]
14. Fusko, M.; Rakyta, M.; Manlig, F. Reducing of intralogistics costs of spare parts and material of implementation digitization in maintenance. *Procedia Eng.* **2017**, *192*, 213–218. [CrossRef]
15. Synakova, L. Production smoothing and cost performance in a production-inventory system. *J. Compet.* **2017**, *9*, 117–133.
16. Li, H.; Liu, S.Y.; Huang, Y.W.; Chen, Y.Q.; Fu, Z.H. An Efficient 2-opt Operator for the Robotic Task Sequencing Problem. In Proceedings of the 2019 International Conference on Real-time Computing and Robotics (RCAR), Irkutsk, Russia, 4–9 August 2019; pp. 124–129.
17. Li, G.; Li, X.; Gao, L.; Zeng, B. Tasks assigning and sequencing of multiple AGVs based on an improved harmony search algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 4533–4546. [CrossRef]
18. Zhong, M.; Yang, Y.; Dessouky, Y.; Postolache, O. Multi-AGV scheduling for conflict-free path planning in automated container terminals. *Comput. Ind. Eng.* **2020**, *142*, 106371. [CrossRef]
19. Zou, W.Q.; Pan, Q.K.; Meng, T.; Gao, L.; Wang, Y.L. An effective discrete artificial bee colony algorithm for multi-AGVs dispatching problem in a matrix manufacturing workshop. *Expert Syst. Appl.* **2020**, *161*, 113675. [CrossRef]
20. Zou, W.Q.; Pan, Q.K.; Tasgetiren, M.F. An effective iterated greedy algorithm for solving a multi-compartment AGV scheduling problem in a matrix manufacturing workshop. *Appl. Soft Comput.* **2021**, *99*, 106945. [CrossRef]
21. Zou, W.Q.; Pan, Q.K.; Wang, L. An effective multi-objective evolutionary algorithm for solving the AGV scheduling problem with pickup and delivery. *Knowl.-Based Syst.* **2021**, *218*, 106881. [CrossRef]
22. Mousavi, M.; Yap, H.J.; Musa, S.N.; Tahriri, F.; Md Dawal, S.Z. Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization. *PLoS ONE* **2017**, *12*, e0169817. [CrossRef]
23. Nordgren. FlexSim simulation environment. In Proceedings of the 2003 Winter Simulation Conference, New Orleans, LA, USA, 7–10 December 2003; Volume 1, pp. 197–200. [CrossRef]
24. Dang, Q.V.; Singh, N.; Adan, I.; Martagan, T.; van de Sande, D. Scheduling heterogeneous multi-load AGVs with battery constraints. *Comput. Oper. Res.* **2021**, *136*, 105517. [CrossRef]
25. Singh, N.; Dang, Q.V.; Akcay, A.; Adan, I.; Martagan, T. A matheuristic for AGV scheduling with battery constraints. *Eur. J. Oper. Res.* **2022**, *298*, 855–873. [CrossRef]
26. Juan, A.A.; Keenan, P.; Martí, R.; McGarraghy, S.; Panadero, J.; Carroll, P.; Oliva, D. A review of the role of heuristics in stochastic optimisation: From metaheuristics to learnheuristics. *Ann. Oper. Res.* **2023**, *320*, 831–861. [CrossRef]
27. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
28. Xue, T.; Zeng, P.; Yu, H. A reinforcement learning method for multi-AGV scheduling in manufacturing. In Proceedings of the 2018 IEEE International Conference on Industrial Technology (ICIT), Lyon, France, 20–22 February 2018; pp. 1557–1561.

29. Hatami, S.; Calvet, L.; Fernández-Viagas, V.; Framinan, J.M.; Juan, A.A. A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem. *Simul. Model. Pract. Theory* **2018**, *86*, 55–71. [CrossRef]
30. Ferone, D.; Hatami, S.; González-Neira, E.M.; Juan, A.A.; Festa, P. A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem. *Int. Trans. Oper. Res.* **2020**, *27*, 1368–1391. [CrossRef]
31. Van Rossum, G.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009.
32. Kuo, H.H. *White Noise Distribution Theory*; CRC Press: Boca Raton, FL, USA, 2018.
33. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2022.
34. Chica, M.; Juan, A.A.; Bayliss, C.; Cordón, O.; Kelton, W.D. Why simheuristics? Benefits, limitations, and best practices when combining metaheuristics with simulation. *SORT* **2020**, *44*, 311–334. [CrossRef]