applied sciences

MDPI

Article

Experimental Evaluation of AGV Dispatching Methods in an Agent-Based Simulation Environment and a Digital Twin

Fabian Maas genannt Bermpohl *, Andreas Bresser and Malte Langosz

German Research Center for Articial Intelligence—Robotics Innovation Center, 28359 Bremen, Germany; andreas.bresser@dfki.de (A.B.); malte.langosz@dfki.de (M.L.)
* Correspondence: fabian.maas_genannt_bermpohl@dfki.de; Tel.: +49-421-178-45-5145

Abstract: A critical part of Automated Material Handling Systems (AMHS) is the task allocation and dispatching strategy employed. In order to better understand and investigate this component, we here present an extensive experimental evaluation of three different approaches with randomly generated, as well as custom designed, environment configurations. While previous studies typically focused on use cases based on highly constrained navigation capabilities (e.g., overhead hoist transport systems), our evaluation is built around highly mobile, free-ranging vehicles, i.e., Autonomous Mobile Robots (AMR) that are gaining popularity in a broad range of applications. Consequently, our experiments are conducted using a microscopic agent-based simulation, instead of the more common discrete-event simulation model. Dispatching methods often are built around the assumption of the asynchronous evaluation of an event-based model, i.e., vehicles trigger a cascade of individual dispatching decisions, e.g., when reaching intersections. We find that this does not translate very well to a fleet of highly mobile systems that can change direction at any time. With this in mind, we present formulations of well known dispatching approaches that are better suited for a synchronous evaluation of the dispatching decisions. The formulations are based on the Stable Marriage Problem (SMP) and the Linear Sum Assignment Problem (LSAP). We use matching and assignment algorithms to compute the actual dispatching decisions. The selected algorithms are evaluated in a multi-agent simulation environment. To integrate a centralised fleet management, a digital twin concept is proposed and implemented. By this approach, the fleet management is independent of the implementation of the specific agents, allowing to quickly adapt to other simulation-based or real application scenarios. For the experimental evaluation, two new performance measures related to the efficiency of a material handling system are proposed, *Travel Efficiency* and *Throughput Effort*. The experimental evaluation indicates that reassignment mechanisms in the dispatching method can help to increase the overall efficiency of the fleet. We did not find significant differences in absolute performance in terms of throughput rate. Additionally, the difference in performance between SMP- and LSAP-based dispatching with reassignment seems negligible. We conclude with a discussion, where we consider potential confounding factors and relate the findings to previously reported results found in the literature.

Keywords: vehicle dispatching; agent-based simulation; digital twin; industry 4.0

Citation: Maas genannt Bermpohl, F.; Bresser, A.; Langosz, M. Experimental Evaluation of AGV Dispatching Methods in an Agent-Based Simulation Environment and a Digital Twin. *Appl. Sci.* **2023**, *13*, 6171. https://doi.org/10.3390/app13106171

Academic Editor: Dimitris Mourtzis

Received: 25 March 2023
Revised: 12 May 2023
Accepted: 13 May 2023
Published: 18 May 2023

1. Introduction

1.1. Motivation and Problem Statement

Automation and the digital transformation of workflows are becoming an increasingly important factor in today's work environments. The physical and digital integration of machines, material, data, and workforce is being pushed forth under the keyword Industry 4.0. The reduction in costs and increasing flexibility, speed, and quality are the leading promises. In order to deliver on these promises, Industry 4.0 applications involve interconnected devices, cloud services, advanced automation, and Artificial Intelligence (AI) technologies [1,2]. Thereby, any relevant detail of the involved physical entities in the

environment also has a Digital Twin (DT) representation. Changes to the physical object are tracked and synchronised with the DT. Changes on the DT will be considered during the processing of the physical object [3]. Zeb et al. [4] describe a *functional digital twin* that actually represents a simulation, which is initialised from the most recent state of the environment tracked in the DT. It is deployed to explore system behaviours and evaluate the effects of measures and to predict and control the connected risks. The availability of the actual state of the production environment in a machine readable form enables computer systems to support or take over responsibility in decision-making and robotic systems to execute physical tasks, especially in form of autonomous material handling.

An important role in the implementation of efficient Industry 4.0 production environments comes to the automation of the intralogistics. The digital integration enables Automated Material Handling Systems (AMHS), e.g., via fleets of Autonomous Mobile Robots (AMR), to provide for the manufacturing processes effectively, flexibly and with low coordination overhead [5–8]. Through the application of Artificial Intelligence methods, the system can automatically adapt to emerging situations and optimise its behaviour as required. Already today, those industries whose identity mostly revolves around logistics are especially able to exploit the massive scaling effects of autonomous robotic fleets (e.g., [9,10]). Yet, automated material handling is still an active research field. Common research questions are the evaluation and benchmarking of different factory layouts, numbers of vehicles deployed, dispatching strategies or related factors [11], often before the background of a specific case study.

Coelho et al. [12] employ a simulation to evaluate the performance of three configurations regarding the number of pickers in the supermarket (i.e., a central storage area from which material is distributed) and behavioural adaptations for the Automated Guided Vehicle (AGV) during operation. They calculate a benefit in terms of different metrics for those configurations. Ma et al. [13] report an evaluation with a discrete-event simulation (DES) for trading off numbers of vehicles, layouts, idle/charging policies with performance indicators. Other recent approaches incorporate machine learning algorithms, e.g., for the estimation of travel times to achieve better-informed scheduling decisions [14], or load balancing within the factory through the adoption of entropy-based calculations to improve overall system performance [15]. Liu et al. [16] propose to use a genetic algorithm to factor various heuristics into a meta-heuristic, here, to take into account stochastic variations in transport time and to improve compliance with required windows of delivery time. Instead of a practical evaluation of the methods, e.g., in a simulation, they compare the methods based on fitness functions directly calculated from the produced task assignments. Similar research questions are actively being discussed in the field of on-demand mobility, taxi dispatching, or ride-hailing systems. Here, the layout of the navigation graph usually cannot be adjusted, but the same mechanisms apply, e.g., for order matching, vehicle dispatching or idle vehicle repositioning [17,18].

We find that many previous works put a focus on layouts that facilitate highly constrained traffic networks, e.g., limiting vehicle traffic along unidirectional edges, no overtaking, etc. These restrictions and assumptions can be explained by the observation that vehicle navigation used to be bound to rail systems (e.g., Overhead Hoist Transport, OHT) or guide paths for many of the considered use cases. As a result, many experimental evaluations are performed using event-based simulation models, and dispatching approaches are formulated with event-based triggering mechanisms in mind (c.f., Table 1). This does not translate very well to a fleet of highly mobile systems that can change direction at any time.

Table 1. Comparison of previous evaluation setups regarding the modelled Automated Material Handling System (AMHS) (Automated Guided Vehicle, AGV; Overhead Hoist Transport, OHT; Autonomous Mobile Robot, AMR), its navigation constraints, and the employed simulation model (Discrete-Event Simulation, DES; Agent-Based Model, ABM).

Source	AMHS Type	Navigation Graph	Simulation Model
[19]	AGV	unidirectional/bidirectional	DES
[20]	AGV	unidirectional/bidirectional	DES
[21]	AGV	unidirectional	DES
[22,23]	OHT	unidirectional	ABM
[24]	OHT	unidirectional	DES
[15]	AGV	unidirectional	ABM
[6]	AGV	unidirectional/bidirectional	DES
[16]	AMR	unconstrained	unknown

1.2. Proposed Approach

With this in mind, as our main contributions we present adapted formulations of well known dispatching approaches, specifically to make them better suited for a synchronous evaluation of dispatching decisions. For the validation of this approach, we present an agent-based simulation testbed and a modular DT, whose role it is to provide the interface between the target environment and the dispatching algorithm. We present an extensive experimental evaluation with a broad range of target application environments, from which we can draw more general insights about the performance of the methods, than would be possible with only a single target use case scenario (c.f., Figure 1).

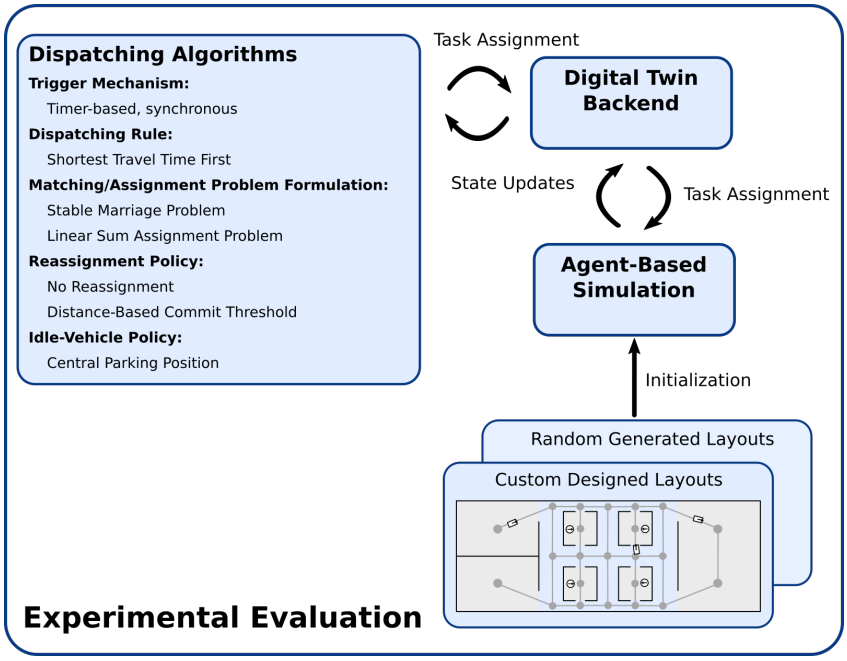


Figure 1. Dispatching method formulation and design of experimental evaluation.

Based on the literature research, we select and adapt the MOD STTF method [20] for our experimental evaluation, both with reassignment feature (MOD STTF) and for comparison without (STTF). As stated before, methods found in the literature are typically designed around event-based triggers and proceed with an asynchronous evaluation of dispatching decisions. This does not translate well to continuous agent-based models or physical implementations with a fleet of AMR, where the specific events, designed around navigation constraints, are not available. Therefore, we introduce a timer-based mechanism to evaluate the dispatching decisions cyclicly in a synchronised fashion. We formulate the dispatching problem in terms of the Stable Marriage Problem (SMP), which yields

equivalent results to the original MOD STTF formulation, and the Linear Sum Assignment Problem, which performs a global optimisation according to the selected dispatching rule. The methods can then be implemented using the specific algorithms available to solve these problems.

With our experimental evaluation we demonstrate the viability of the approach, which is the adaptation and application of the described methods to the domain of AMR-based AMHS, the presented modular simulation testbed, and the DT interface. We assess the performance of the approaches in a broad range of operation scenarios, i.e., the frequency and spatial distribution of transport requests, the fleet utilisation, and the size and structure of the environment. In order to achieve this, we use the following variations on the environment parameters.

- The makespan is dominated by the travel time vs. processing duration,
- The vehicle utilisation is either high or low,
- Different degrees of complexity regarding agent navigation and spatial distribution of stations are considered.

In our earlier work [25], we presented the testbed environment that includes an agent-based simulation and a digital twin. We will use this simulation environment for our experimental evaluation. The digital twin represents the backend for this simulation that encapsulates and provides access to its data and controls.

Section 2 gives a brief literature overview and lays out the selected approaches for Task Allocation and Vehicle Dispatching. Section 3 provides the detailed description of the proposed digital twin concept and implementation. Section 4 presents the simulation environment and how it is embedded in the overall system architecture. The experimental evaluation is given in Section 5. Concluding remarks and notes on future research directions can be found in Section 6.

2. Task Allocation and Dispatching

This section first provides a general introduction into the topic of task allocation and dispatching, followed by a selection of methods used within this work. Finally, the selected methods and how they are implemented is described in the corresponding subsections.

The dispatching of tasks across the available fleet of agents is one of the cornerstones of an autonomous intralogistics system. Our simulated environment produces one type of product on parallel workstations. If a new production job is assigned to a workstation, the required parts are identified. Move requests are issued to fetch and deliver those parts to the workstation. Once completed, another move request is issued to fetch and deliver the product from the workstation to the output buffer storage. These move requests are assigned to the available vehicles based on the specific dispatching policy employed. These algorithms facilitate heuristics in order to identify and assign the agent to a task that seems most suitable to accomplish it efficiently. Goal of the optimisation is an increase in the plant's overall throughput, or conversely, the minimisation of the mean duration between shipments at any of the environment's configured output storage. The specifics of the implemented dispatching policy contributes to the overall performance, e.g., regarding empty travel time, pick-up durations, or the overall makespans. To this end, we evaluate different approaches in different environment configurations for comparison. We would like to gain insights not only about the relative performance of the approaches compared to each other, but also how they perform in response to different environment configurations.

Vehicle dispatching in itself is a well-studied problem in the fields of logistics at warehouses or container terminals, and taxi operations. Different approaches exist, some of which support multiple loads to be transported, shared rides, reassignment, or guarantees on the delivery time. Egbelu and Tanchoco [26] is without a doubt one of the central sources on AGV dispatching in a job shop environment. They presented a number of rules according to which matching decisions would be made, e.g., Shortest Travel Time First (STTF), whereby the vehicle would be assigned to a job that would travel to the pick-up location in the shortest time. They distinguished *vehicle initiated* and *work centre initiated*

task assignment (dispatching) rules, implying different rules could be employed whenever a vehicle becomes available (again) versus a new move request is generated at a workstation. Each rule calculates a priority according to which a vehicle should pick a task from the set of available move requests, or a workstation to claim service from a vehicle from the set of available vehicles, respectively.

Later works started to refine these rules. Bozer and Yen [20] propose methods that support preempting task execution in order to redispatch tasks to other agents becoming available later (Modified Shortest Travel Time First, MOD STTF), and another method providing vehicles with task queues to let them participate in the assignment procedure even while they are still busy, i.e., to facilitate a predictive consideration of future vehicle availability (Bidding-Based Dynamic Dispatching, B^2D^2). While previously dispatching decisions would only be triggered upon insertion or completion of a move request, they proposed to review the assignments of vehicles until they are actually committed to their assignment. Where this commitment would be calculated based on the remaining travel distance. They also let vehicles travel uncommitted to a parking point, in case they are not directly assigned another task. With the employed dispatching rule, decisions are based on the greedy (Greedy, in the sense that the closest vehicle is always preferred, disregarding the overall sum of distances travelled by all the vehicles (c.f., Section 2.2).) minimisation of travel time for individual vehicles. In their implementation, the trigger for reevaluation of the assignment rules is the vehicle passing intersection points along their path. Thus, the distinction according to the initiation of the rule evaluation by vehicles or work centres becomes irrelevant.

The idea of bidding- or auction-based mechanisms for dispatching was picked up by several authors, as Koenig et al. [27] show in their survey. They give a good overview about several methods employing bidding mechanisms for task negotiation, but they also point out a potential shortcoming of the greedy assignment evaluation, i.e., one out of two agents might be assigned with priority to one out of two tasks, while the other agent might be entirely unsuitable to fulfill the remaining task.

The topic is also studied heavily for applications, such as container shipping terminals. Common extensions in that field involve multi-load carrying capabilities or the learning of agent preferences [28–30]. Ride-sharing and taxi operations are other important fields of application [31–34].

2.1. Modified Shortest Travel Time First

The Modified Shortest Travel Time First (MOD STTF) algorithm was originally proposed by Bozer and Yen [20]. Therein, the STTF rule is employed, which was presented and investigated earlier by Egbelu and Tanchoco [26]. By this rule, the priority for an assignment is given by the remaining distance an agent needs to travel in order to pick up a load, such that the lowest estimated travel time always wins the assignment.

The modification that was later added (i.e., the MOD part in the name) revolves around the concept of uncommitted assignment; thereby agents could be assigned a task tentatively. They would then start to travel towards the assigned pick up location, but could be reassigned to other tasks or released by other agents, that become available later at a closer distance. The window during which this reassignment can occur is denoted as *uncommitted travel*. It is open until the remaining empty travel distance falls below a configurable threshold value, at which point the agent is then committed to the assignment. If this value is set to an infinite positive value we obtain the standard STTF dispatching (without reassignment) as a special case. If there are no constraints regarding navigation capabilities or computational performance to consider, it seems sensible to set it to zero to always allow reassignment in favour of a better match. This is intended to react to:

- New loads or move requests that are added, which the agent could reach quicker,
- Or likewise other agents that become available, which are able to pick up and execute the task earlier than the agent that was originally (tentatively) assigned to it.

According to the original formulation, if a new move request is added, one of the unassigned vehicles is assigned according to the STTF rule. The current task assignment shall be revisited for each agent individually whenever certain events occur, i.e.,

- An agent completes its task or is released from an uncommitted task,
- An agent passes by certain locations while it is not committed to a task.

While they acknowledge the performance could potentially be improved by more frequent reevaluation of the rule, they argue the benefit would not justify the additional complexity.

As can be seen from the above described trigger mechanism for the re-evaluation of dispatching decisions, the method is designed around applications that fit an event-based model, e.g., due to navigation constraints imposed onto the agents. This assumption does not hold for our target application of an AMHS facilitating free-ranging autonomous mobile robots (AMR). Our target application uses a microscopic agent-based model. Therefore, the agents ability to change their path is not restricted to discrete locations, so the dispatching decisions do not need to be synchronised to these events either. Additionally, the exact point where a vehicle *"travels through a station or intersection"* is hard to track in this environment and can not be detected precisely. Therefore, we opted to implement the triggering mechanism based on timer events, and reconsider the assignment for all unassigned and uncommitted agents simultaneously instead of asynchronously for individual agents. Thus, to apply the method to the domain of AMR, the algorithm is adapted based on two different formulations, the Stable Marriage Problem (SMP) and the Linear Sum Assignment Problem (LSAP). The following subsections present the specific methods.

2.2. Stable Marriage Problem

As we are re-evaluating all uncommitted assignments simultaneously, we can use a stable matching algorithm to compute the assignments. In fact, we can formulate this setting in terms of a Stable Marriage Problem (SMP). Thereby a stable matching (bijection) is sought that matches the elements of two disjoint sets according to mutually stated preferences. Here, at each evaluation step, we wish to match all idle or uncommitted agents $a_i \in A_{idle} \cup A_{uncommitted}$ with any one unassigned or uncommitted move request $r_j \in R_{unassigned} \cup R_{uncommitted}$. The preference of an agent a_i towards a move request r_j is derived from the travel distance τ_{ij}, such that a move request ranks higher in preference, the lower the associated travel distance τ_{ij}. Symmetrically, the preference of a move request r_j towards an agent a_i is received in the same way. Thereby the matching is said to be stable in the sense that no pair of agent and move request could be formed that both rank each other higher in preference than their assigned matches.

The solution to this formulation can be obtained by the application of the *Gale–Shapley Algorithm* [35]. According to this algorithm stable matching between two equal sized sets of suitors and reviewers are formed iteratively. New matches are formed tentatively by iterating the suitors' preference lists in order. During this process previously formed matches are compared to newly proposed matches until all suitors eventually find a match. If during this procedure previously matched suitors are released by more preferable candidates to the reviewer, they are enqueued again to propose to the next preferable reviewer (c.f., Algorithm 1).

This matching is equivalent to the outcome of the asynchronous evaluation of the STTF assignment rule, as thereby agents would be released and reassigned until the matching would arrive at the same stable state in the same way. In order to extend this formulation to resemble the B^2D^2 method, one would just need to add vehicles committed to their move request to the input set $a_i \in A_{committed}$, whose preference would be derived from the travel distance to complete their current task, plus the empty travel distance to pick up the next.

The original formulation of the SMP and the presented solution algorithm assumes both sets are equal in size. In our case, that would correspond to an equal number of agents and move requests. This assumption will obviously not hold true in practice. McVitie and Wilson [36] investigate this situation and propose different strategies to handle it. They

recognise that the algorithm's solution speed degrades significantly when the number of suitors exceeds the number of reviewers. In that case the preference lists of the unmatched suitors need to be iterated exhaustively before they can conclude they will not receive a match. One of the approaches they propose is to swap the role of the two sets depending on their size, such that the smaller set is put into the role of the suitor and the larger acts as the reviewer. They note that, by doing this, a different stable solution may be returned (*female-optimal* vs. *male-optimal*). We assume the solutions will not actually differ in our case, due to the symmetry in the preference calculation we employ for both sets. Therefore, we integrated this mechanism to achieve the best efficiency for our implementation.

Algorithm 1: Gale–Shapley Algorithm

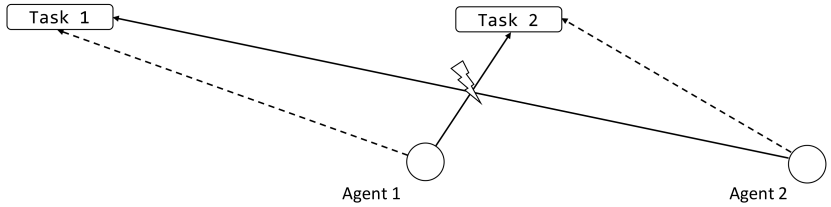
```
Input  : Suitors, Reviewers
Output : Assignment
// Iterate suitors until all are matched
UnassignedSuitors ← Suitors
for thisSuitor ← popfront(UnassignedSuitors) do
    // Check reviewers in order of decreasing preference
    for reviewer ← popfront(preferences(thisSuitor)) do
        prevSuitor ← Assignment(reviewer)
        if ∃prevSuitor then
            thisPreference ← preference(reviewer, thisSuitor)
            prevPreference ← preference(reviewer, prevSuitor)
            if prevPreference ≥ thisPreference then
                // Previous assignment is more preferable,
                // skip to next reviewer
                continue
            end
            // New assignment is more preferable,
            // re-enqueue previous suitor
            pushback(UnassignedSuitors, prevSuitor)
        end
        // Assign match tentatively,
        // proceed with next suitor
        Assignment(reviewer) ← thisSuitor
        break
    end
end
```

As Bozer and Yen [20] describe, unassigned vehicles are sent to a designated parking point, where they will wait for new assignments. The number and location of these points will have an impact on the performance of the fleet. It makes sense to distribute the vehicles at a central location, or (if those are predictable) in the vicinity of the pick-up positions of incoming move requests.

2.3. Linear Sum Assignment Problem

The formulation as a Linear Sum Assignment Problem (LSAP) [37,38] can be used as an alternative method to the SMP described above. The probability that vehicles have to cross their ways in order to pick up tasks nearby, intersecting with paths of other vehicles (c.f. Figure 2a), can be reduced with the LSAP formulation compared to SMP. We found this effect described in the literature (e.g., Kümmel [39]). However, how much this effect actually influences the overall performance in a continuous AMR-based application is not well studied. Figure 2b provides numbers to the example. The matrix shows the distance measures for the combinations, that would be used as the cost function. The highlighted

combination would be picked by the algorithm according to the derived preference lists produced in Figure 2c. According to the overall sum of travelled distance, the inverted combination would represent a (globally) more efficient solution to the problem.



(**a**) Example configuration: spatial distribution of two agents and tasks.

Pick-up Location	Agent 1	Agent 2
Task 1	2	**5**
Task 2	**1**	2

(**b**) Cost matrix.

Entity	Preference List
Agent 1	**Task 2**, Task 1
Agent 2	Task 2, **Task 1**
Task 1	Agent 1, **Agent 2**
Task 2	**Agent 1**, Agent 2

(**c**) Preference lists.

Figure 2. Example configuration that demonstrates navigation conflicts due to task assignment based on STTF rule (solid arrows). The assignment would favour matches based on individual preference, instead of globally optimal sum of travel times (dashed arrows). The cost matrix and preference lists associated to the example are given. The combination that results from the greedy assignment based on the STTF rule (highlighted) is globally suboptimal.

In fact, we found that other authors have used this approach before to address the AGV dispatching problem [21,23,24]. In particular, the approach presented by Kim et al. [23] shares similarity to the one described below, in that they also formulate the problem as an assignment problem and use the same reassignment policy as defined with MOD STTF, i.e., to consider both *idle* and *uncommitted* vehicles eligible for dispatching.

Here, we start with a cost matrix that is populated with the distances τ_{ij} between the agents a_i and the move requests r_j. The objective is to find the optimal assignment of agents to move requests, in order to minimise the overall sum of distance travelled. It is commonly formulated as a zero-one linear programming problem, e.g.,

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \tau_{ij} x_{ij} \qquad \text{with } x_{ij} \begin{cases} 1 & \text{if agent } i \text{ is assigned to move request } j \\ 0 & \text{otherwise} \end{cases}$$

such that

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, \ldots, n$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, 2, \ldots, n$$

One popular solution approach is by using the so called *Hungarian method*, which is what most authors state to use. For our application, we found the implementation provided by the *scipy* library [40] very useful, which claims to provide an implementation based on the *Jonker–Volgenant algorithm* [41].

Additional flexibility or adjustments to the application scenario are commonly added by extending the function that is used for the population of the cost matrix. Thereby the approach can incorporate more complex constraints or requirements, such as due dates, load balancing etc.

We implemented the reassignment mechanism and the idle behaviour in the same way as described in the previous Section 2.2. That is, vehicles can be released or reassigned until their assignment is committed, which happens when they reach a specified threshold distance. We also employed the same approach to idle vehicle positioning as described there. That is, unassigned vehicles are sent to a central parking location to await future incoming move requests.

3. Digital Twin Representation

In this work, the main purpose of the Digital Twin (DT) is to decouple the application-specific physical entities and environment from the fleet management. The DT reflects the current state of the environment and provides an interface to communicate state and actions between environment and dispatching algorithms. Therefore, the physical environment, either a simulated or a real-world application, has to be interfaced to the DT. Within the next paragraphs a short general introduction into the topic of DT is given. The Sections describe the relevant set of parameters our DT implementation is designed to track for the given application.

The DT provides the basis for storage and retrieval of any application-relevant information about a physical or virtual entity that is required for operation or analytics at a facility [3]. Other authors emphasise the significance of the DT as a tool to aid decision-making regarding supply chain management [42] or product life cycle management [43]. It serves as the interface and central point of access for any services the user wishes to build around the physical production environment or entity. Implementation-wise, the DT will necessarily be tightly integrated with the facility's Enterprise Resource Planning (ERP) solution. Sometimes the implementation will even be provided by the ERP vendor or at least interface with it to acquire the relevant information. At last, the DT will be interfacing with a Manufacturing Execution System (MES) that actually controls the operation of the intralogistics system [44].

We found studies regarding the specific case of DTs for manufacturing systems and intralogistics. Jiang et al. [45] propose the conceptualisation of the relevant processes around several basic elements, including *controller, executor, buffer, logistics path*, etc. Following the notation defined in the overview by Jones et al. [3], for our implementation we specify several types of *entities* and *processes* and the *environment*, within which the former exist and operate. The below specification provides the description of the DT implementation, which is an integral part of the setup for our application and experimental evaluation. We define all the relevant entities, parameters, and processes that need to be twinned for the successful integration with the dispatching methods and provide details about the technical implementation of the twinning procedure, i.e., the act of synchronisation of the physical environment with the virtual twin. Note that in the context of this paper, the physical environment is represented by an agent-based simulation model. Due to the modularity that is achieved by adopting this DT approach and model, the transfer to an application with a real-world application is straight-forward. The design of the DT that is described below was presented before in our earlier publication [25]. In the following we provide an updated version of that description.

3.1. Application Instance

We define the Digital Twin of the whole environment as a `graph` structure. Figure 3 shows a graphical visualisation of a simple example environment. Different types of agents are visualised as blocks or circles with an arrow to indicate their orientation. The agents can use the associated navigation graph for route planning. In the visualisation, the nodes and edges are visualised as circles and lines.

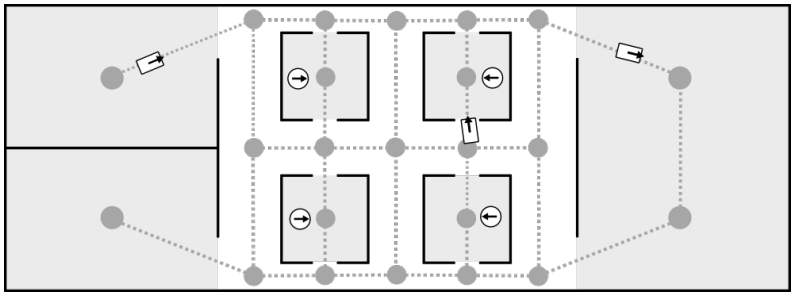


Figure 3. Visualisation of an example environment configuration. Displayed are the areas for storage of workpieces in light grey (left and right) and assembly (centre), that are partially enclosed by obstacles (walls). Additionally, a number of agents are shown (rectangles and circles with arrows indicating the orientation) and the edges and vertices of the navigation graph, that are used to support the agents' navigation (darker grey circles and dashed lines).

3.2. Structural Layout

The structural layout of the environment, i.e., *walls* and *areas*, can be described based on an actual floor plan of the facility or manually arranged using a web-based editor. We defined different area types to differentiate logically between different spaces, for example for picking up items or for production areas. They are only used as a visual aid, but they could be used in a more complex scenario, for example it could be included in planning to bring certain items not directly to an agent but in the general area this agent is stationed.

While walls are generally not traversable, the other areas can be traversed unless blocked by temporary obstructions, e.g., doors or mobile entities. Navigation planning might consider to include restrictions on the area type, e.g., in order to avoid transport travel through the workspace of production processes.

3.3. Navigation Graph

In extension of the structure of the physical environment, the digital twin representation specifies a navigation graph that mobile agents can use for efficient motion planning. When the assigned goal position for an agent's task is not in direct line of sight to the agent, i.e., when an obstacle needs to be circumnavigated, a graph search algorithm will be used (here: Dijkstra) to find the shortest path along the navigation graph to the goal position. It will then use the nodes along this path as waypoints for the navigation towards the goal position.

The navigation graph can be designed and provided manually, or created using a probabilistic roadmap algorithm [46] or other approaches for automatic generation of navigation graphs [47].

3.4. Processes and Tasks

The business processes in our environment are modelled around an assembly process that requires a number of input parts and produces an output part. The actual assembly process can take place at the locations of specific workstations, where worker agents can manufacture one item at a time. For this purpose, the required input parts have to be transported to these locations by dedicated transportation agents. Below, Figure 4 visualises the task tree of the production process, modelled as a Petri Net [48].

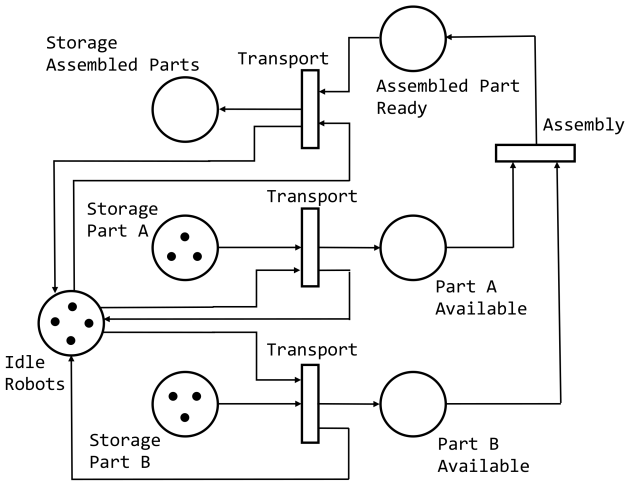
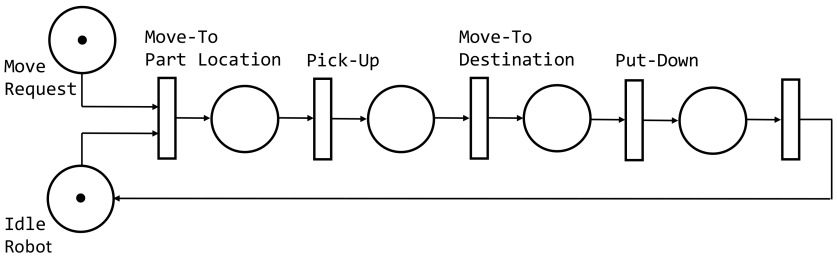
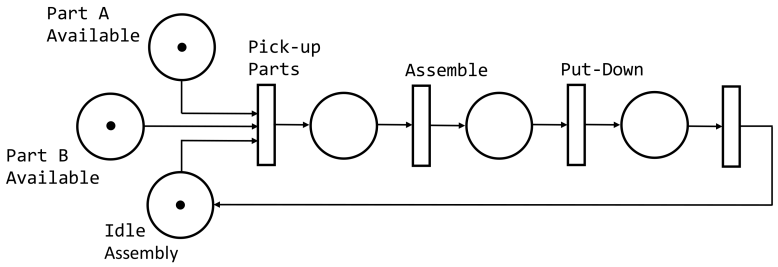


Figure 4. Petri net reflecting the task structure for the manufacturing processes in an example application environment.

For the actual implementation, we distinguish the task types *idle*, *move-to*, *pick-up*, *put-down*, and *assemble*. Each of these tasks typically have an expected non-zero duration, which could actually be infinite, e.g., in case of deadlocks. Tasks of the types *pick-up*, *put-down*, and *assemble* take the identifiers of the components that are supposed to be affected by the given action. In addition, tasks of type *move-to* also specify the target coordinates. To simplify the visualisation, the model presented above does not include the transitional states that would reflect the associated action execution of the transitions *Transport* and *Assembly*. Below, Figure 5 provides the actual model for these processes too.



(**a**) Petri net for the Transportation process.



(**b**) Petri net for the Assembly process.

Figure 5. Petri nets for the subprocesses in the environment.

Preconditions for the execution of these processes is the ability of the agent to actually execute them, e.g.,

- The availability of the specified parts in the vicinity of the agent is checked before task execution.

- Only assembly agents are able to execute the *assemble* action, while, on the other hand, only transportation agents can move. Assembly agents are modelled as being permanently stationary.
- The capability of an agent to carry one or many instances of a load item can be constrained arbitrarily. In our implementation, only one item of any kind can be picked up by a transportation agent. Assembly agents can pick up one item of each type for the execution of the assembly process.

The task instances are created and persisted in the environment's DT and can then be assigned to agents for execution.

A new task starts with the state *"queued"* which means that it has been assigned to an agent and will be executed in the near future, probably at the *"planned_start"* timestamp. When it is getting executed it changes its state to *"started"*. There should always be only one task for each agent in the *"started"* state. After the execution, the task changes to the success-state if it was *"successful"* or *"failed"* if something goes wrong. It can also be cancelled by the user. For logging purposes, we store those tasks and assign them the *"cancelled"* state.

3.5. Components and Modules

The environment allows the definition of any parts that play a role in the manufacturing process, either as components, tools, material, or supplements. Complex part types can include a tree of child components that was factored into the construction of the specific part at hand. In the specific implementation we distinguish the part types *wing* and *wing movable*, which combine into *wing assembled*. This very simple task tree stands representative of the much more complex tree of production steps and requirements in a real world scenario. It should be, however, sufficient to demonstrate the capabilities of the implemented testbed and for the evaluation of the selected approaches.

A component is either a workpiece that needs to get transported and processed or the final product that needs to be transported to the *despawn* position which can be a place to store the component before it gets transported elsewhere.

3.6. Agents

Each environment can contain multiple agents, described by their pose and type, their current and subsequent tasks and the currently transported payload. To reduce redundancy in a fleet of agents of the same model we can store most data in agent type that the single agent instance can overwrite. The agent type can be selected from a number of predefined agent types, that can be configured separately. The agent type is used to specify the generic properties and capabilities of the agent, such as whether or not it can transport or assemble certain parts, and also to select the graphical model that is used in the visualisation of the environment. The agent type can be selected from a number of predefined agent types, that can be configured separately.

The agent also carries the information which tasks are currently allocated to it (if any). During operation, one or more tasks can be added to the agent's task list, that it will execute sequentially, ordered by their planned start time, until the list is exhausted.

3.7. Persistence Layer

We implemented our DT using a document-oriented NoSQL database system for persistence, i.e., a MongoDB instance. Each of the above described entities are represented as documents in dedicated collections within this database. We chose to build our implementation on top of a MongoDB database, because capabilities to stream changes to clients directly and its built-in support for representing GeoJSON and geospatial queries are readily available. We acknowledge that other backend systems (e.g., Relational Database Management Systems, RDBMS) might work just as good.

For the act of twinning, the database can be interfaced from Python via an asynchronous programming interface that provides the required domain-specific functionalities,

e.g., regarding reading and updating properties of agents, their assigned tasks and the environment structure itself. Specific functions allow the planner to update the agents' task plans and others enable the agent implementation to query, execute, and cycle to the next task, once the current goal has been completed. The data model allows the representation of several separate environments in parallel, distinguished by an `environment_id` identifier.

4. Agent-Based Simulation

Simulations are the standard tool to evaluate the performance and to investigate the behaviour of AGV dispatching systems. They enable the assessment of different configurations, policies or management directives, without incurring the associated costs for a real world implementation. In our case, the simulation will be used to assess inherent properties of selected algorithms related to multiple environments, artificially constructed to investigate relevant situations and to provoke interesting behaviours. The testbed we developed for this evaluation is implemented in a modular fashion, so we can attach and replace planning and dispatching algorithms, visualisation frontends, and, perspectively, the simulation engine. Even the substitution of the simulation with an actual facility is possible.

In the following Section 4.1, we first describe how the simulation is integrated with the DT. Section 4.2 then lays out how the various elements and dynamics of the environment are modelled in detail. Although the design of the simulator still largely resembles the state we described in one of our earlier works [25], we here present the updated and more recent state for completeness.

4.1. Integration with the Digital Twin

We implemented the simulation as a modular testbed environment, that would allow us to tackle different research problems both jointly and individually. The three main components, namely the digital twin, the simulation, and the task allocation logic, are interfaced in a way such that we can easily swap out one for another implementation (c.f., Figure 6). The digital twin of the environment is represented by a database, a dispatching algorithm performs the task allocation based on the state of the environment, and the simulator evaluates the task execution. The simulation and the task allocation software are both retrieving and updating the DT, i.e., the state of the database continuously.

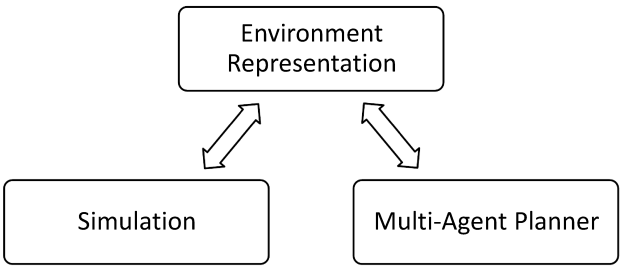


Figure 6. Modular components of the system architecture. Both the multi-agent simulation and the multi-agent planner directly interface only with the database used to track the state of the environment.

The modularity enables us to integrate different simulation backends or various task allocation approaches easily. It will also make the transition to an actual physical production environment straight-forward, because they share the DT as the common interface.

Our DT implementation already facilitates the necessary interfaces for the synchronisation with the state updates of the respective environment. A small convenience layer was added in between to simplify the call semantics of the raw database model API. For the quick initialisation of the experiment runs, a script was set up to parse the desired structure of the environment from a configuration file and initialise the simulation, as well as the DT accordingly. The general procedure would operate as follows.

1. Launch DT database.
2. Initialise the DT from the configuration file according to the experiment definition.
3. Initialise the simulation from the current state of the DT.
4. Run the simulation loop, which updates the DT accordingly.
5. Run the task allocation loop, which interacts with the simulation via the DT.
6. Wait until the experiment completes.

4.2. Simulation Characteristics

Applications investigating Task Allocation problems often facilitate DES-based models (e.g., [13,49–51]). Thereby, any changes to the environment (especially vehicle movements) occur by temporally and spatially discretised steps along the navigation graph. While this greatly simplifies the required computations for the model updates, this abstracts away several dynamics that we wished to not neglect perspectively. In contrast, we implemented the simulation using an agent-based model with a microscopic approach. We found it promising to have the system behaviour to model the actual dynamics in a real facility more closely. Following this approach also enables us to investigate related problems too, e.g., the decentralised navigation aspects, avoidance of collisions, and reactive approaches to respond to congestion (c.f., De Ryck et al. [8]).

After the DT is initialised, updates to its state are computed at a fixed rate by the simulator. The updates are then fed back to the DT. In principle, the simulation can run faster than real time. We chose to synchronise it with the wall time for the purpose of visualisation however.

The simulation is integrated out of two main components, the agent navigation backend and the task handlers. Both are triggered from a main loop at the desired interval period. During each iteration, first the updates by the navigation backend are calculated and updated to the DT. Then, the task handlers are run for each agent. These check the task completion status, e.g., whether the agent has arrived at the goal position, and would terminate the task accordingly and if available, proceed with the next. The actual task handlers are explained in more detail in the following subsections.

4.2.1. Tracking of Agents and Parts

We chose to feed the actual ground truth states of agents and components into the DT. In practice, the facility would need to employ special mechanisms to track agents and material, which would report their poses only with limited accuracy. However, Real-Time Location Systems (RTLS) are readily available, and, given the highly digitised setting we describe, we would assume that inventory will be tracked with the help of such a system. This, can be combined with local refinement mechanisms, e.g., agents would employ a Simultaneous Localisation and Mapping (SLAM) algorithm augmented by this global pose estimate, such that we can, therefore, consider this problem outside of the scope of this work.

4.2.2. Agent Navigation and Obstacle Avoidance

Agents need to navigate the environment in order to provide effective transportation of material. Many approaches exist for the control and coordination of AMR systems. The most distinctive characteristic is that of *centralised* versus *decentralised* coordination or control [8]. While the central control server that coordinates the entire fleet of the facility was the predominant form of implementation, recent works increasingly seem to embrace the decentralised approach due to expected improvements regarding scalability, flexibility, and robustness [8,52,53].

In our implementation, agent motion is simulated using a non-holonomic, differential drive kinematics model. In order to move towards their goals, they can adapt their velocity. Pose updates are calculated accordingly and communicated to the centralised environment representation.

When an agent receives a new move request it will first look up the pose of the corresponding navigation goal. Then it will find the required immediate movement direction using the environment's navigation graph and the static navigation obstacles defined in the DT. The current agent pose and the goal pose are matched onto the navigation graph and the shortest path is calculated using a graph search algorithm. For this purpose, we use an implementation of Dijkstra's algorithm. The agent will then use the roadmap nodes along the path as waypoints towards the goal. If the goal pose is found in direct line of sight, the agent will head directly towards the actual goal pose.

We integrated the collision-free agent navigation scheme Optimal Reciprocal Collision Avoidance (ORCA) provided by the RVO2 library for collision avoidance and conflict resolution [54]. The theoretical background is described in Snape et al. [55] and van den Berg et al. [56]. The basic principle is the continuous adaptation of the agent's linear and angular velocity in order to avoid collisions with mobile and stationary obstacles along their predicted relative trajectories. Thereby each agent slightly adjusts their own speed and heading towards a value outside the so called *Velocity Obstacles* [57] the other entities represent. Depending on the parameterisation of the time step, temporal horizons, and safety margins, we can rely on a collision-free multi-agent navigation within the simulation.

The selected approach provides a sufficient baseline operation in configurations with little congestion or navigation constraints. However, during the experiments it became obvious, that these circumstances can lead to deadlocks in crowded environments. Other approaches include priority-based schemes or swarm behaviours to mitigate this [53,58].

4.2.3. Part Dynamics

As the simulation models only the internal logistics of the facility, we needed to find a suitable abstraction with the external interfaces for providing material and consuming the output of the products. We defined this abstraction at the buffer storage, where parts from the outside world are unloaded, or delivered to, and in analogy, where the manufactured output of the plant is prepared for shipment or further processing elsewhere.

This abstraction is modelled as special locations throughout the environment, at which material is generated and, thus, introduced into the production flow. We refer to this type of location as a *component spawn point*. They can be configured such that they produce their assigned type of material at a desired rate. Similarly, we call the special location at which readily manufactured products are consumed *component despawn point*. At these locations, items of the matching type will be collected and dematerialised at the configured rate.

The configuration allows to simulate different scenarios, which might be worth considering in specific investigations. For our analyses, we chose to configure both to produce and collect items at a fixed rate. In order to avoid overflowing the input storage area with superfluous parts, new material is only created, when none is nearby.

5. Experiments

In order to gain insights about how the different dispatching approaches actually behave when employed under various conditions, we evaluated a number of artificial experiments with different environmental properties. Most studies we found in our literature review evaluate newly proposed algorithms against a baseline approach in a specific environment setting designed after very specific case studies. While this allows to draw insights valid for this specific environment configuration at hand, we would like to present a more detailed picture of the algorithms' performance. In order to accomplish this, we distinguish the environments according to the following characteristics.

- The makespan of the products is either dominated by the processing time versus transport time (i.e., denoted *P/T ratio* [19]).
- The time spent by the vehicles is dominated by travel time versus idle waiting time (i.e., denoted by vehicle utilisation).
- The distances between stations are homogeneous versus heterogeneous (i.e., concerning the existence of isolated remote stations).

- A high complexity of the navigation graph with walls and corridors versus unconstrained navigation on an empty plane.

In the following, we first describe the generation and configuration of scenarios for the experiments in Section 5.1. Then, we describe the performance metrics according to which the performance will be evaluated in Section 5.2. In Section 5.3, we present the results of our experiment and conclude with a discussion in Section 5.4.

5.1. Scenario Classification and Generation

The relevant characteristics for the performance evaluation are *P/T ratio*, *vehicle utilisation* and *layout homogeneity*. We implemented a script that is able to generate environments randomly according to these criteria.

5.1.1. P/T Ratio

The P/T ratio was proposed by Kim and Tanchocoj [19] to quantify the so called criticality of the transport system for the system performance. It calculates the ratio of average processing time per operation $\bar{t}_{processing}$ to average transport time per trip $\bar{t}_{transport}$. The load pick-up and drop-off times are added to those times, but the vehicles' empty travel or idle times are not. Multiple transports during the makespan of a single product are factored into the average value individually. For a makespan incorporating n processing steps and m transports, we thus arrive at the following formulation.

$$P/T\ ratio = \frac{m \sum_{i=0}^{n} t_{processing,i}}{n \sum_{j=0}^{m} t_{transport,j}} \tag{1}$$

In order to generate environments with a desired effective P/T ratio, we can adjust the expected processing time for the given configuration. For their simulation studies Kim and Tanchocoj [19], Kim et al. [59], the authors used P/T ratio values of 5 to 20. We chose to use a value of 5 to correspond to a high P/T ratio, indicating that processing time tends to be the bottle neck; and a value of 1 to correspond to a low P/T ratio, indicating that transportation time tends to dominate the makespan. We chose to use lower values, as we expect to receive more interesting results from the perspective of the transportation system this way.

The average transportation time can be calculated from the average distance between the buffer storage and the assembly stations, under the assumption of a constant average velocity. Then we can calculate the average processing time accordingly, in order to arrive at a value for the P/T ratio in the desired range.

5.1.2. Vehicle Utilisation

The vehicle utilisation can be adjusted by modifying the number of agents available to dispatch. This criterion does display some association with the aforementioned P/T ratio, as for the same number of vehicles, a lower P/T ratio would generally increase the vehicle utilisation. Because idle and waiting times are not tracked in that criterion, we can adjust the vehicle utilisation by the number of vehicles available for dispatch, without affecting the P/T ratio.

For the purpose of our experiments we define high vehicle utilisation to correspond to little to no vehicle idle times, while low vehicle utilisation would correspond to a larger amount of idle time. If there are fewer agents to fulfill the same amount of tasks, the more work would fall onto the individual, until, in our case, at some point the vehicle utilisation would saturate at a value of one. This is where each agent will never be idle, but always find an open task to pick up. For the purpose of this evaluation, we aimed at high vehicle utilisations greater than 0.95, and low vehicle utilisations of smaller than 0.9. We adjusted the numbers of agents until we found the actual vehicle utilisation observed in test runs to correspond to our declared expectations.

We started our search with agent numbers derived from the number of assembly stations in the environment and the P/T ratio as below Table 2 indicates.

Table 2. Deployed numbers of vehicles per assembly station to achieve different variations in vehicle utilisation in relation to P/T ratio for experiment configuration.

Vehicle Utilisation	P/T Ratio	
	Low: 1	**High: 5**
high	1	0.2
low	2	0.4

5.1.3. Layout Homogeneity

The homogeneity of the layout corresponds to the spread of the distribution of travel distances between nodes in the navigation graph. In a homogeneous environment all nodes are located at similar distance to each other. Our definition of a heterogeneous environment, in contrast, implies that some nodes are situated at a much further distance to the rest of the graph. These nodes are outliers so to say.

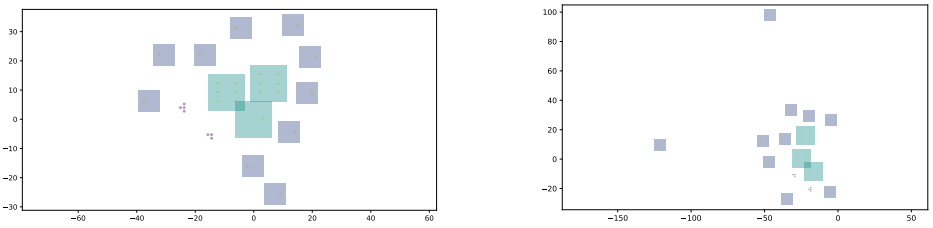
In order to achieve this, we used a simple custom algorithm to distribute the stations on a plane. The first station is placed at the origin of the environment. For the placement of each subsequent station we determine the centre of the environment using the median coordinates of the cluster of stations. Then a random angle $\alpha \in [0, 2\pi)$ is chosen at which the station is attempted to be placed at a distance $r_{placement}$ according to the following formula.

$$\vec{p}_{i+1} = \vec{p}_{median,i} + r_{placement} \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \tag{2}$$

Placement fails if this position is closer than a minimum distance to any previously placed station. Then a small increment is added to the placement radius and the iteration is repeated with another random angle. In case the placement succeeds, the radius is reset to its original value. This process is repeated until the desired number of stations were placed.

The list of placed coordinates can be shuffled to avoid patterns in the relative proximity of special stations (i.e., parking position and input/output storage). Without this step, these stations would always be placed near each other, as they are always assigned to the first positions in the output list. In order to evaluate this effect we added this configuration under the name *Pseudorandom* to the list of layout configurations.

For the heterogeneous configuration, we used the same procedure to add a few more stations at a relatively large radius ($r_{outlier} = 10 r_{placement}$). Afterwards the same number of stations is dropped from the previously obtained intermediate result. We chose to let only assembly stations appear in these outlier positions. Otherwise the outcome of the results would be dominated too much from the fact of which type of station would be placed remotely. For the random configurations we chose to designate a number of two (out of ten) assembly stations as outliers. Figure 7 shows examples of the random environments we obtain this way. Notably, these environments do not contain walls, such that the vehicles can navigate on straight lines between the stations.



(**a**) Homogeneous Layout. (**b**) Heterogeneous Layout.

Figure 7. Example random environments.

5.2. Performance Metrics

Beamon [60] describe a number of measures that can help to gauge the effectiveness and the efficiency of manufacturing systems as a whole and material handling systems in particular. For our evaluation we chose to track multiple measures to acquire detailed insights about the utilisation of both assembly stations and the vehicle fleet, as well as regarding the overall system performance. The selected methods are described in the following paragraphs.

5.2.1. Throughput

We figured the most straight-forward measure to evaluate the system performance would be the throughput $r_{shipped}$ in terms of the (average) number of shipped products $n_{shipped}$ per time Δt, as follows.

$$r_{shipped} = \frac{n_{shipped}}{\Delta t} \tag{3}$$

If desired, one can derive the average *flowtime* from this value, i.e., the time it takes to produce a single product. The required input data can be collected easily within our framework and it gives a good overall impression of the aggregated system performance, including all involved factors.

5.2.2. Vehicle Travel Efficiency

For the evaluation of the fleet efficiency we propose a measure derived from the *total vehicle travel distance*. A few relevant metrics are presented in Beamon [60] already, e.g., total amounts or ratios calculated from empty and loaded vehicle travel. We found convincing arguments in the criticism of these measures. Therefore, a larger vehicle utilisation or consequently larger amounts of (loaded) vehicle travel do not necessarily correspond to better system performance. Thus, these metrics alone (utilisation and travel distance) could hardly be considered consistent with the actual goals of a production facility.

Therefore, we propose a measure that presents a figure of the invested effort relative to the actual requested task. Similarly to how the *Value Added Efficiency* [60] measures the ratio of *productive time* divided by *spent time*, for the purpose of our evaluation, we define the *Vehicle Travel Efficiency* η_{travel} as the ratio of the minimum distance loads were requested to be moved $d_{requested}$ divided by the total amount of loaded d_{loaded} and empty d_{empty} distance travelled to satisfy the requirements.

$$\eta_{travel} = \frac{d_{requested}}{d_{loaded} + d_{empty}} \tag{4}$$

The requested transport distance we define as the euclidean distance between origin and destination of the move request. We calculate this statistic over the whole fleet and all requested transports.

5.2.3. Throughput Effort

Similarly, we propose a second measure that aims to give a better picture of the performance of the transportation system. Here, we put the effort in terms of total distance travelled into relation to the facility's output in terms of throughput $n_{shipped}$. We define the *throughput effort* TE as follows.

$$TE = \frac{d_{loaded} + d_{empty}}{n_{shipped}} \tag{5}$$

5.3. Experimental Evaluation

The experiments are conducted according to the following procedure. We run a *set* of each dispatching method for each *run configuration*. We have implemented the following approaches that employ the same cost function (i.e., the distance between vehicle and pick-up point of a move request), the same reassignment policy (distance-based commit threshold), and the same idle vehicle positioning scheme (central parking points). For comparison, we add a method to the evaluation that does not allow reassignment, which corresponds to the original STTF formulation. The only difference is that we let idle vehicles travel uncommitted to the same parking points as with the other methods.

- SMP-based dispatching with reassignment (MOD STTF) as described in Section 2.2,
- LSAP-based dispatching with reassignment as described in Section 2.3, and
- SMP-based dispatching without reassignment (STTF), by configuring an infinite commit distance threshold.

The environment definitions for the *run configurations* are determined by the specific *layout* and the *combinations* of P/T ratio and vehicle utilisation (high and low, as described in Section 5.1). While the layout is fixed for the custom environments, we generate a new environment layout for each set of method evaluations and repeat this procedure for $N = 100$ runs.

5.3.1. Tracking of Performance Metrics

As suggested by Bechtsis et al. [6], Kim and Tanchocoj [19], we let the system stabilise for an initialisation period after the start of the experiment run before we begin to track data for the above-mentioned metrics of evaluation (warming period). When this period ends, we initialise the starting values of the trackers and start to run the update loop. Each run spans a duration of 15 or 30 min. We found that the custom configurations with a high P/T ratio required more time to converge to terminal values for the performance metrics, due to the lower volume of transports that is taking place. Therefore, these configurations are set to run for the longer time period.

We track these statistics by querying the experiment database (i.e., the digital twin) from a separate process in an update cycle at a fixed rate. At every iteration of this update loop, we evaluate whether vehicles did move, assembly stations were occupied, tasks were completed, and parts shipped. Based on these numbers, the above described metrics are calculated.

The values are recorded as a time series that is expected to converge to some value for each configuration. As some of the metrics tend to show spikes and fluctuations around specific events, we calculate a mean value over the last 300 seconds to determine the outcome value for each run. Statistical significance is analysed using *Welch's t-Test* (two-sided), with a $p < 0.05$.

5.3.2. Random Environments

For the evaluation, we use randomly generated environment configurations in accordance with all combinations of the above described characteristics: layout homogeneity, P/T ratio, and vehicle utilisation (c.f. Section 5.1). The run configurations and the corresponding aggregated experiment results can be found in Table 3. The observations we draw from these results are presented in the following and discussed in Section 5.4.

Table 3. Experiment run configurations and results (mean values and standard deviation) for random layout with varying compactness, P/T ratio and vehicle utilisation. Entries typeset in bold face are significantly better than the other methods; entries typeset in italics are significantly worse; entries marked with an asterisk (*) are significantly better or worse than only one of the other methods (two-sided Welch's *t*-Test, $p < 0.05$).

ID	Layout	P/T Ratio	Vehicle Utilisation	Method	Throughput	Total Travelled Distance	Travel Efficiency	Throughput Effort
1a	Random, homogeneous	Low	Low	STTF	11.161 ± 1.236	*20761.680 ± 1047.617*	*0.459 ± 0.025*	*197.684 ± 17.493*
				SMP	11.158 ± 1.275	**19262.971 ± 1121.363**	0.486 ± 0.026	183.544 ± 16.823
				LSAP	11.337 ± 1.106	19797.229 ± 1038.545	0.488 ± 0.026	185.417 ± 17.377
1b	Random, homogeneous	Low	High	STTF	* *8.779 ± 0.794*	*13680.104 ± 305.938*	0.555 ± 0.037	* *165.896 ± 18.968*
				SMP	8.830 ± 0.993	13533.635 ± 330.924	0.554 ± 0.044	163.985 ± 22.361
				LSAP	* **9.040 ± 0.826**	13502.728 ± 299.939	**0.576 ± 0.040**	* **159.007 ± 17.855**
2a	Random, homogeneous	High	Low	STTF	* **4.347 ± 0.317**	*8588.156 ± 273.222*	*0.436 ± 0.023*	*209.104 ± 14.457*
				SMP	* *4.219 ± 0.294*	**8104.583 ± 315.185**	0.451 ± 0.025	203.178 ± 12.950
				LSAP	4.288 ± 0.311	8211.191 ± 281.293	0.452 ± 0.022	202.741 ± 14.412
2b	Random, homogeneous	High	High	STTF	3.873 ± 0.303	*6357.876 ± 117.071*	*0.521 ± 0.022*	* *174.139 ± 15.288*
				SMP	3.857 ± 0.360	**6175.416 ± 133.402**	0.528 ± 0.031	170.228 ± 16.721
				LSAP	3.907 ± 0.361	6213.036 ± 129.481	0.533 ± 0.028	* **169.051 ± 16.613**
3a	Random, heterogeneous	Low	Low	STTF	8.870 ± 0.686	*21012.921 ± 881.238*	*0.474 ± 0.020*	*250.750 ± 18.382*
				SMP	8.773 ± 0.746	20075.962 ± 1181.685	0.493 ± 0.022	242.283 ± 18.396
				LSAP	8.860 ± 0.614	20326.877 ± 950.286	0.497 ± 0.020	242.665 ± 17.594
3b	Random, heterogeneous	Low	High	STTF	*6.651 ± 0.500*	*14503.515 ± 191.597*	*0.543 ± 0.029*	*231.370 ± 20.493*
				SMP	7.177 ± 0.634	14210.794 ± 225.944	0.578 ± 0.036	210.648 ± 22.508
				LSAP	7.188 ± 0.535	14235.488 ± 204.584	**0.589 ± 0.031**	210.103 ± 18.839
4a	Random, heterogeneous	High	Low	STTF	* *3.156 ± 0.247*	*8376.131 ± 289.323*	*0.441 ± 0.016*	*281.675 ± 19.978*
				SMP	3.173 ± 0.229	8146.303 ± 295.392	0.457 ± 0.018	272.300 ± 18.916
				LSAP	* **3.233 ± 0.228**	8183.529 ± 293.946	0.457 ± 0.016	268.536 ± 20.019
4b	Random, heterogeneous	High	High	STTF	2.719 ± 0.221	*6566.425 ± 139.664*	*0.494 ± 0.019*	*256.344 ± 19.806*
				SMP	2.789 ± 0.275	6323.421 ± 201.663	0.520 ± 0.022	241.048 ± 20.601
				LSAP	2.779 ± 0.252	6352.495 ± 190.592	0.517 ± 0.020	242.861 ± 20.421

Table 3. *Cont.*

ID	Layout	P/T Ratio	Vehicle Utilisation	Method	Throughput	Total Travelled Distance	Travel Efficiency	Throughput Effort
5a	Pseudorandom	Low	Low	STTF	12.879 ± 1.033	*21286.354 ± 783.250*	*0.480 ± 0.014*	*174.947 ± 11.259*
				SMP	12.861 ± 1.011	20245.555 ± 742.402	0.499 ± 0.016	166.704 ± 11.844
				LSAP	13.038 ± 1.011	20214.539 ± 728.698	0.501 ± 0.013	164.245 ± 12.528
5b	Pseudorandom	Low	High	STTF	*9.740 ± 0.671*	*13316.138 ± 249.436*	*0.594 ± 0.017*	*145.000 ± 13.789*
				SMP	10.017 ± 0.694	13133.049 ± 256.683	0.611 ± 0.024	139.039 ± 13.244
				LSAP	**10.237 ± 0.656**	13064.809 ± 243.917	**0.619 ± 0.021**	**135.207 ± 11.617**
6a	Pseudorandom	High	Low	STTF	4.739 ± 0.310	*8504.851 ± 233.476*	*0.444 ± 0.019*	*189.687 ± 10.347*
				SMP	4.711 ± 0.301	8000.677 ± 296.529	0.466 ± 0.019	179.492 ± 12.094
				LSAP	4.693 ± 0.325	7982.351 ± 249.713	0.471 ± 0.021	179.909 ± 12.027
6b	Pseudorandom	High	High	STTF	*4.337 ± 0.306*	*6224.724 ± 106.318*	*0.549 ± 0.012*	*152.148 ± 13.376*
				SMP	4.429 ± 0.340	6087.419 ± 129.220	0.574 ± 0.015	145.826 ± 13.575
				LSAP	4.467 ± 0.331	6117.970 ± 100.459	0.572 ± 0.013	145.256 ± 13.115

General Observations

Looking at the differences between different run configurations we observe a strong relationship between the utilisation of the vehicle fleet and the output performance in terms of throughput and efficiency. A higher vehicle utilisation results in a decrease in throughput and an increase in efficiency (higher travel efficiency, lower throughput effort).

Regarding the throughput rate, within run configurations we notice some significant differences between the methods. The only configuration, where one method clearly outperforms the other two is 5b, where the LSAP method scores significantly better than both STTF and SMP. For the other configurations we see no significantly best performance. In particular, related to a high vehicle utilisation and a low P/T ratio (3b, 5b), the STTF method performs significantly worse than the other methods. A similar pattern can be observed with configuration 1b, but here the difference is significant only between STTF and LSAP. For most configurations with high vehicle utilisation and a high P/T ratio however (2b, 4b), no clear difference could be obtained between the methods. As an exception, we obtain a small but significant difference with configuration 6b, where STTF performs worse than the other two. Most configurations with low vehicle utilisation show no clear outliers in terms of throughput performance (1a, 3a, 5a, 6a). We see significant differences only between the respective best and worst performances with configurations 2a and 4a. Here, STTF performs significantly better than only SMP (2a), and significantly worse than only LSAP (4a).

For all configurations, we observe that the STTF method generates significantly more traffic in general. Consequently, STTF also scores worse with the efficiency-related performance measures, with a lower travel efficiency and a higher throughput effort. For most configurations, this difference is significant. The only exceptions to this are configurations 1b and 2b, where the difference is only significant between STTF and LSAP.

The performance of the LSAP method regarding the efficiency-related measures are generally better than or equal to that of SMP. The SMP method scores best with the total travelled distance metric, which seems to be correlated with a lower throughput rate and, thus, does not lead to better relative efficiency scores.

We notice no clear difference to the above described observations with the pseudo-random layout configuration (that is, the parking positions, as well as input and output storage are systematically located near each other). Only for the configuration 5b, with low P/T ratio and high vehicle utilisation, we observe a significantly superior performance by the LSAP method across all metrics but the travelled distance, and a significantly inferior performance by the STTF method for all performance metrics. Here too, in general, we observe a clear benefit in terms of efficiency with LSAP and SMP when compared to STTF. For the other configurations (5a, 6a, 6b), no clear difference was observed between the former two methods.

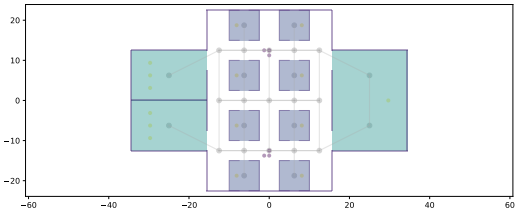
5.3.3. Custom Environments

We compare this to a small number of handcrafted layouts (c.f., Figure 8). In contrast to the random environments, navigation within these configurations is restricted by walls, which consequently facilitates navigation conflicts and congestion. Additionally, these environments are intentionally structured in specific ways to simulate certain general characteristics.

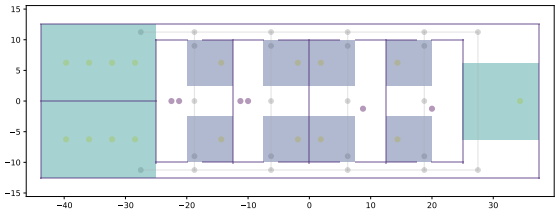
Custom 1 In this environment the navigation graph is very broad. Distances between the stations are relatively short and navigation is possible along many paths in parallel. The input storages are located on the left side, while the output storage is located on the right. In order to navigate from the output storage (common drop-off location) to the input storage (common pick-up location), the vehicles need to pass the assembly stations (common pick-up locations). The idle positions are located centrally between the assembly stations. From these last two conditions, we expect transports to prioritise the transports from the assembly stations to the output storage for high vehicle utilisations.

Custom 2 The general structure is similar to the *Custom 1* environment. Only the assembly stations are located along two long corridors, such that traffic will concentrate along there. The idle positions are again located between the assembly stations. Here, we again assume that vehicles will prioritise transports originating from the assembly stations. Here, the emphasis on navigation conflicts is increased slightly, which might give an advantage to the LSAP approach according to the hypothesis postulated in Section 2.3.

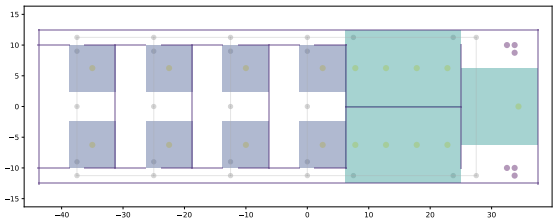
Custom 3 The environment is very similar to *Custom 2*, but here the input storage are located between the assembly stations and the output storage. This way, we assume that transports, especially those facilitating reassignment, will prioritise the delivery of input parts and thus lead to a larger assembly utilisation and throughput. Thus, we would expect a larger advantage of the reassignment-based dispatching.



(**a**) Custom 1.



(**b**) Custom 2.



(**c**) Custom 3.

Figure 8. Custom environments. Light blue coloured areas represent input/output storage. Greyish areas represent assembly stations. Purple areas/lines represent walls. Thin grey dots and lines indicate the navigation graph (nodes and edges).

The experiment variations regarding the target values for P/T ratio and vehicle utilisation correspond to those defined before (c.f., Section 5.1). The evaluated run configurations and the corresponding results can be found in Table 4.

General Observations

As we also noticed before in Section 5.3.2, we received a strong relationship between the utilisation of the vehicle fleet and the output performance in terms of throughput and efficiency. Here too, other things being equal, a higher vehicle utilisation results in a decrease in throughput and an increase in efficiency (higher travel efficiency, lower throughput effort).

For the Custom 1 layout we notice that LSAP achieves the best performance for low P/T ratios independent of the vehicle utilisation (7a, 7b). STTF method scores significantly better than the others with configuration 8b, featuring high P/T ratio and high vehicle utilisation. The SMP method scores significantly worse than the others with configurations 7a, 8a, and 8b. Regarding the total travelled distance, the SMP method scores best for all four configurations. As before, STTF scores worst on this metric. As to the efficiency-related scores we note that, as before, STTF achieves significantly lowest values for most configurations (7a, 7b, 8a), the only exception being configuration 8b (high P/T ratio, high

vehicle utilisation), where it scores best. The LSAP method scores significantly better at both scores with configurations 7a, 7b (low P/T ratio). With configuration 8a, LSAP outperforms SMP only at the throughput effort, but not at the travel efficiency.

The Custom 2 layout indicates the best throughput performance with the STTF method for configurations 9a, 10a, and 10b, but worst performance for configuration 9b. The SMP method achieves significantly lowest scores with configurations 9a, 10a (low vehicle utilisation). LSAP scores best with configuration 9b (low P/T ratio, high vehicle utilisation). Regarding the total travelled distance, STTF scores worst for all configurations. Here, STTF performs best only for configuration 9a. The efficiency-related metrics do not show a clear results here. For configurations 9a and 10a we find LSAP significantly outperforms the other two methods, and STTF being outperformed by the other two. Configuration 10b (high P/T ratio, high vehicle utilisation), on the other hand, shows best performance on the side of STTF for both scores in contrast, here SMP being lowest. Additionally, for configuration 9b (low P/T ratio, high vehicle utilisation), we observe best performance by STTF in terms of travel efficiency (worst here: SMP), but lowest performance in terms of throughput effort (best here: LSAP).

Regarding Custom 3 layout, STTF performs significantly worst throughput-wise at configurations 11a and 12b, but best at configuration 12a. For the configurations 11b and 12a, SMP achieves the lowest throughput performance, whereas it scores best at configuration 11a. The total travelled distance demonstrates the same result as before, with STTF scoring lowest for all configurations. Here, best performance is achieved by LSAP for configuration 11b and SMP for configuration 12a. With the efficiency scores, we reproduce the pattern of worst performance achieved by STTF only for configurations 11a, 12a, and 12b. Configuration 11b does not show a clearly worst performance regarding the travel efficiency, here SMP scores worst with the throughput effort. The best efficiency scores are achieved by SMP for configurations 11a, 12b (both only significant for travel efficiency), and 12a (both scores). Best efficiency with configuration 11b is achieved by the LSAP method.

The reassignment-based methods SMP and LSAP tend to outperform STTF especially on the efficiency scores, but for some configurations we observe the opposite effect. Compared against each other, we find that LSAP scores better than SMP, but this again does not hold for all configurations. LSAP and STTF tend to outperform SMP in terms of throughput rate for most, but not for all configurations.

5.4. Discussion

In general we demonstrated the successful application of the presented method formulations for AGV dispatching to the domain of AMR-based AMHS. With the adjustment to the triggering mechanism we were able to dispatch the vehicles independent of event-based triggers. We demonstrated the suitability of the modular DT interface to allow for the interaction of the dispatching algorithms with the target environment, which was represented by an agent-based simulation for this evaluation. We are not aware of previous studies that presented a related similar application and investigation regarding this domain before.

The results of the experimental evaluation indicate that none of the presented methods clearly outperforms the others independent of the environmental and operational setting. We identified some relationships that have an influence on the observed performance characteristics, which we will discuss below.

Table 4. Experiment run configurations and results (mean values and standard deviation) for different custom layouts with varying P/T ratio and vehicle utilisation. Entries typeset in bold face are significantly better than the other methods; entries typeset in italics are significantly worse (two-sided Welch's *t*-Test, $p < 0.05$).

ID	Layout	P/T Ratio	Vehicle Utilisation	Method	Throughput	Total Travelled Distance	Travel Efficiency	Throughput Effort
7a	Custom 1	Low	Low	STTF	7.049 ± 0.082	*17033.533 ± 115.754*	*0.388 ± 0.003*	*254.615 ± 2.923*
				SMP	*6.688 ± 0.076*	**16090.131 ± 133.817**	0.391 ± 0.003	253.484 ± 2.716
				LSAP	**7.103 ± 0.072**	16925.475 ± 120.085	**0.393 ± 0.003**	**251.076 ± 2.382**
7b	Custom 1	Low	High	STTF	5.021 ± 0.069	*11390.737 ± 31.318*	*0.412 ± 0.005*	*239.142 ± 3.283*
				SMP	5.024 ± 0.075	**11280.797 ± 42.300**	0.416 ± 0.004	236.694 ± 3.560
				LSAP	**5.116 ± 0.062**	11291.923 ± 36.160	**0.424 ± 0.004**	**232.636 ± 2.786**
8a	Custom 1	High	Low	STTF	3.168 ± 0.027	*19835.327 ± 163.384*	*0.381 ± 0.004*	*255.785 ± 3.170*
				SMP	*3.072 ± 0.031*	**18937.697 ± 212.304**	0.390 ± 0.004	251.757 ± 3.033
				LSAP	3.167 ± 0.034	19433.974 ± 148.317	0.389 ± 0.003	**250.583 ± 2.835**
8b	Custom 1	High	High	STTF	**2.590 ± 0.032**	*14697.560 ± 46.144*	**0.420 ± 0.003**	**231.784 ± 2.832**
				SMP	*2.437 ± 0.050*	**14115.601 ± 222.631**	*0.414 ± 0.004*	*236.675 ± 3.370*
				LSAP	2.509 ± 0.046	14336.829 ± 151.047	0.418 ± 0.003	233.478 ± 3.243
9a	Custom 2	Low	Low	STTF	**5.702 ± 0.172**	*16413.436 ± 338.958*	*0.331 ± 0.006*	*303.418 ± 6.594*
				SMP	*5.469 ± 0.110*	**15244.047 ± 252.284**	0.334 ± 0.005	293.812 ± 5.588
				LSAP	5.659 ± 0.120	15526.577 ± 257.915	**0.341 ± 0.005**	**289.197 ± 4.904**
9b	Custom 2	Low	High	STTF	*4.163 ± 0.102*	*11117.251 ± 51.001*	**0.371 ± 0.005**	*281.668 ± 6.964*
				SMP	4.221 ± 0.099	10939.330 ± 76.650	*0.359 ± 0.005*	273.352 ± 6.075
				LSAP	**4.309 ± 0.101**	10958.066 ± 65.727	0.367 ± 0.005	**268.146 ± 6.070**
10a	Custom 2	High	Low	STTF	**2.704 ± 0.014**	*19911.928 ± 190.178*	*0.321 ± 0.003*	*300.786 ± 3.229*
				SMP	*2.638 ± 0.036*	19071.070 ± 320.373	0.328 ± 0.004	295.236 ± 3.209
				LSAP	2.670 ± 0.028	19055.368 ± 254.640	**0.332 ± 0.003**	**291.516 ± 3.233**
10b	Custom 2	High	High	STTF	**2.226 ± 0.023**	*14510.323 ± 75.381*	**0.371 ± 0.003**	**266.231 ± 2.315**
				SMP	2.136 ± 0.049	14067.319 ± 239.584	*0.359 ± 0.003*	*269.060 ± 2.759*
				LSAP	2.146 ± 0.045	14064.195 ± 226.746	0.362 ± 0.003	267.802 ± 2.538

Table 4. *Cont.*

ID	Layout	P/T Ratio	Vehicle Utilisation	Method	Throughput	Total Travelled Distance	Travel Efficiency	Throughput Effort
11a	Custom 3	Low	Low	STTF	*6.461 ± 0.183*	*16507.787 ± 289.716*	*0.395 ± 0.007*	*269.322 ± 7.096*
				SMP	**6.636 ± 0.113**	16079.754 ± 298.170	**0.419 ± 0.006**	255.359 ± 5.107
				LSAP	6.602 ± 0.112	16060.706 ± 269.011	0.414 ± 0.006	256.373 ± 5.344
11b	Custom 3	Low	High	STTF	4.824 ± 0.106	*10818.640 ± 53.430*	0.480 ± 0.008	236.500 ± 5.494
				SMP	*4.732 ± 0.103*	10793.823 ± 58.644	0.479 ± 0.007	*240.519 ± 5.250*
				LSAP	4.848 ± 0.117	**10772.672 ± 76.030**	**0.483 ± 0.007**	**234.335 ± 5.590**
12a	Custom 3	High	Low	STTF	**2.597 ± 0.016**	*19009.812 ± 317.116*	*0.354 ± 0.006*	*299.021 ± 5.803*
				SMP	*2.568 ± 0.024*	**17447.097 ± 437.971**	**0.396 ± 0.013**	**277.525 ± 6.769**
				LSAP	2.586 ± 0.018	18115.463 ± 505.119	0.377 ± 0.013	286.071 ± 7.972
12b	Custom 3	High	High	STTF	*2.318 ± 0.021*	*14141.857 ± 63.204*	*0.428 ± 0.003*	*249.208 ± 2.298*
				SMP	2.350 ± 0.025	13919.971 ± 95.202	**0.445 ± 0.005**	241.985 ± 2.516
				LSAP	2.348 ± 0.025	13942.287 ± 88.966	0.442 ± 0.004	242.592 ± 2.341

5.4.1. Efficiency-Related Performance Metrics

We find the proposed efficiency-related performance metrics *Travel Efficiency* and *Throughput Effort* add a valuable dimension for the comparison of the methods, which he considered absolute measures of performance and effort do not provide. Beamon [60] already state the need for incorporating multiple measures for performance analysis and decision-making. The previously available metrics did not capture the efficiency of an AMHS directly. With our evaluation we could show that the proposed measures provide a suitable means to assess this relationship.

5.4.2. Evaluation of Vehicle Reassignment

For the random layouts, most configurations do not indicate significant differences in terms of throughput performance between the reassignment-based methods (keys SMP and LSAP) and the method, which does not facilitate reassignment (STTF). We observe some exceptions to this, where one or both of the reassignment-based methods would outperform the STTF method without, which seems to be linked to high vehicle utilisations. This matches our assumption that with increased fleet utilisation any increase in efficiency becomes relevant to maximise the absolute system performance. Correspondingly, we did measure a clear tendency that the reassignment mechanism would benefit the overall efficiency of the transportation system. Thus, the associated scores were reliably better with reassignment (SMP and LSAP) than they were without (STTF).

As soon as the vehicle utilisation is high enough that not every trip starts and ends in the parking position, and the volume of transportation is large and spatially interleaved enough, we expect to see a benefit to reassignment. While we originally expected this benefit to also manifest in the throughput rate, we can explain this by the structure of the problem. The previous evaluations in the literature mainly focused on stations with infinite input/output queues, evaluating performance measures such as *mean load waiting time* (i.e., the average time it takes to pick something back up), often restricting navigation to a graph (or even one or two loops only) that can be traversed unidirectionally (e.g., Bozer and Yen [20]). Under these conditions, the adoption of a heuristic or cost function based on the shortest travel time to pick-up for dispatching seems sensible. However, both metrics (*mean load waiting time* and *shortest travel time first*) do not reflect the actual throughput performance especially considering multi-step assembly processes, and, as such, cannot be expected to result in increased throughput performance either. We were able to skew the implicit prioritisation characteristics implicitly by layout design (c.f. pseudorandom random layout), i.e., by co-locating the vehicle parking positions and the input/output storage. A better approach would probably involve multi-attribute cost functions, which is in line with previous research.

In addition, we designed a number of specific fixed custom layouts according to certain considerations, in an attempt to reflect actual conditions in realistic environments a bit better. Although the results we obtained with these custom layouts indicated the same tendencies, they were not as clear as we expected. Here, we observed statistical differences in throughput, but we could not make out a clear pattern as to why these would occur. The results of several configurations with the SMP method display both lowest throughput performance and lowest travelled distance. This seems to suggest that the result for the lower throughput performance is not *inefficient* use of the vehicle fleet, but rather *ineffective* fleet use. Additionally, with several configurations of the custom layouts the STTF method without reassignment would outperform the others, as well at the throughput but occasionally also at the efficiency scores. We suspect confounding factors play into these effects, e.g., the implicit task prioritisation effect described above, or issues with oscillations of (re-)assignments that are especially prevalent with the more complex custom navigation graphs. Interestingly this pattern does not manifest with the random environments, so we expect this effect to be highly specific to the specific layout configurations.

In some cases, we noticed oscillations in the assignments. This issue manifested especially, when we first used the straight-line distance for the dispatching decision, but the

vehicles would need to follow the navigation graph instead, and thus multiple vehicles at the parking position would constantly swap assignment. With the calculation of the distance along the navigation graph, this problem occurred much less frequently. However, in hindsight, the integration of an additional (small) cost term to account for the cost of switching the assignment might be sensible.

In general, occasionally vehicles would need to travel detours when their assignment was changed en-route. This can look especially dramatic, when vehicles almost reach half-way to their destination and are then called back into the opposite direction. Probably this effect would not be too obvious in environments with a very restricted navigation graph, e.g., with a loop layout. This certainly counteracts potentially higher efficiency gains, but, as the data indicates, is still superior versus keeping the original assignments. For almost all configurations, vehicle reassignment reduces the total travelled distance significantly. Instead, this signifies the importance of incorporating predictive capabilities in the dispatching decisions, i.e., by considering future vehicle availability and future task insertion predictively. For this case, we assume that later reassignment would very rarely be required in the first place.

5.4.3. Comparison between LSAP- and SMP-Based Dispatching with Reassignment

We could not identify a strong difference in performance between the two reassignment-based approaches LSAP and SMP in general. We noticed differences only for some of the random and custom environment layout configurations, that indicate a tendency that LSAP tends to outperform SMP for some configurations. However, the effect size is too small or the causing conditions too specific to stand out more clearly. To our knowledge these methods were previously not compared directly to each other.

A related investigation by Kim et al. [23] indicates the superiority of the LSAP-based approach, which might be due to the navigation constraints that stem from the OHT-based AMHS their work is focused on. That is, vehicles are bound to single-lane unidirectional traffic along the flow path and are unable to overtake others during pick-up and put-down operations. Constraints that would resemble or even amplify the negative impact of the navigation conflicts are described in Section 2.3. However, they do not compare their approach to MOD STTF directly, and the difference they report must at least partially be attributed also to the different approaches regarding vehicle pool and reassignment policy they chose to implement for their first reassignment-based method [22]. As these navigation constraints do not affect AMR-based AMHS in the same way, it seems safe to assume that the impact of this effect would be much smaller for our application, which is in line with our observations.

5.4.4. Layout-Specific Implicit Task Prioritisation

It seems that, by the single-attribute STTF dispatching rule, an implicit prioritisation of different types of jobs is imposed by the structural layout of the environment. By the proximity of certain pick-up locations to common drop-off locations, vehicles tend to prioritise the jobs located there, over those located at a further distance. Then, when vehicles need to pass the input storage frequently after delivering a part, they tend to implicitly prioritise the input part delivery over the output part removal. When they instead need to pass many assembly stations on their way to the input storage, they will tend to prioritise removal of output parts over the input part delivery, which may thus affect the assembly utilisation negatively.

6. Conclusions

We presented formulations and experimental evaluations of three different approaches to the vehicle dispatching problem for a fleet of autonomous mobile robots, based on the Stable Marriage Problem (SMP) and the Linear Sum Assignment Problem (LSAP). These formulations make the approaches particularly well suited for a synchronous computation

of dispatching decisions for the whole fleet instead of cascading asynchronous dispatching decisions for each individual vehicle.

We compare the relative performance of the methods in a broad range of environment configurations, covering differences in terms of floorplan or factory layout, transportation volume (P/T ratio), and capacity of the material handling system (vehicle utilisation). The evaluation focuses specifically on Automated Material Handling Systems (AMHS) based on Autonomous Mobile Robots (AMR), contrasting systems that are highly-constrained by unidirectional guide/flow paths which are much more commonly investigated. For this, we use our own testbed simulation, which was designed and implemented based on an agent-based model (ABM) around a digital twin representation as a backend. The digital twin bridges the fleet management with the execution layer of the individual agents. Overall, the methods were adapted and applied to the AMR domain successfully. From our experimental evaluation, we conclude that the approach is suitable for many real-world applications. For our experimental evaluation, we tracked the absolute performance measures *throughput rate* and *total travelled distance*. In addition, we proposed two metrics related to the efficiency of the transportation system, *Travel Efficiency*, which puts total travelled distance in relation to the requested transport distance, and *Throughput Effort* which represents the average travel distance per throughput unit.

Generally, the results are quite similar for a broad range of environment configurations. We found that consistently, for each method individually, a lower vehicle utilisation results in a larger throughput rate. That is, throughput can be significantly improved by not operating the vehicle fleet near full capacity or maximum utilisation. This comes at the cost of some decline in efficiency however, which we found to increase with higher vehicle utilisation. We showed that the methods facilitating the reassignment window can indeed improve the efficiency of the transport system, but not so much the actual throughput. Instead, while we found that for most settings the method without reassignment would generally perform worse in terms of total travelled distance, we noticed that often it would also score better at the throughput rate. Contrary to our initial assumption we observed that for most settings, there is no significant difference between the LSAP and SMP formulation with reassignment.

In particular, the random environments show a clear tendency towards the above-mentioned observations. We identified some peculiarities with the custom layouts, where we were able to reproduce issues probably related to the employed dispatching rule (minimum distance to the pick-up location). While this rule optimises for the average load waiting time, it does not necessarily lead to a better system performance. Instead, we found that the structural layout of the environment can dictate implicit job prioritisation, which can be amplified by reassignment. That is, when the input storages are located in the vicinity of common drop-off locations (e.g., the output storage), the delivery of input parts would be prioritised. In contrast, when the vehicles would need to travel along the assembly stations on their way to the input storage, they would implicitly prioritise the delivery of output products. The method without reassignment would be less affected, as the time window for this effect to occur would be much smaller.

All in all, we identify several potential angles of approach for further improvements. We found the dispatching heuristic needs to reflect the system performance accurately, which is in line with common criticism of single-attribute dispatching heuristics. Regarding a practical implementation with AMR, this heuristic should take into account costs that stem from kinematic constraints, e.g., effort required for vehicle heading change. We are not sure if all performance-related considerations can be factored into a single cost value though, or if one should envision, e.g., a multi-stage dispatching architecture for differently prioritised jobs. In addition to the consistency of the cost function, another important consideration is the dispatching latency. In particular, when the margin around the decision boundary between assignments is small, real-time constraints become relevant, to prevent negative impact by decisions based on an outdated state of the environment. From following the simulation execution, we noticed potential improvements to the system efficiency by

the introduction of predictive capabilities, i.e., regarding future vehicle availability and demand forecast.

Overall, the problem of agent dispatching is a very important topic, relevant to a wide array of applications. This goes far beyond material handling systems or factory intralogistics alone. This can cover taxi dispatching, on-demand mobility, delivery drones, customer service, the coordination of first responders in search and rescue applications or even sports team games. Thus, we are eager to integrate our approach via the modular digital twin backend with an actual application scenario.

Author Contributions: Conceptualisation, F.M.g.B.; methodology, F.M.g.B.; software, F.M.g.B. and A.B.; validation, F.M.g.B. and M.L.; formal analysis, F.M.g.B. and M.L.; investigation, F.M.g.B. and M.L.; resources, M.L.; data curation, F.M.g.B. and A.B.; writing—original draft preparation, F.M.g.B. and A.B.; writing—review and editing, F.M.g.B., A.B. and M.L.; visualisation, F.M.g.B.; supervision, M.L.; project administration, M.L.; funding acquisition, M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the *German Federal Ministry for Economic Affairs and Climate Change* (BMWK), formerly known as *German Federal Ministry for Economic Affairs and Energy* (grant number 20X1724C). The APC was covered by funding provided by the federal state of Bremen, Germany (grant number 201-001-10-3/2021-3-2).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The full source code and the data logged during the experiments can be provided upon request.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ABM	Agent-Based Model
AGV	Automated Guided Vehicle
AGVS	Automated Guided Vehicle System
AMHS	Automated Material Handling System
AMR	Autonomous Mobile Robot
B^2D^2	Bidding-Based Dynamic Dispatching
DES	Discrete-Event Simulation
DT	Digital Twin
ERP	Enterprise Resource Planning
LSAP	Linear Sum Assignment Problem
MES	Manufacturing Execution System
MOD STTF	Modified Shortest Travel Time First
OHT	Overhead Hoist System
ORCA	Optimal Reciprocal Collision Avoidance
RTLS	Real-Time Location System
SLAM	Simultaneous Localisation and Mapping
SMP	Stable Marriage Problem
STTF	Shortest Travel Time First

References

1. Ghobakhloo, M. Industry 4.0, digitization, and opportunities for sustainability. *J. Clean. Prod.* **2020**, *252*, 119869. [CrossRef]
2. Olsen, T.L.; Tomlin, B. Industry 4.0: Opportunities and challenges for operations management. *Manuf. Serv. Oper. Manag.* **2020**, *22*, 113–122. [CrossRef]
3. Jones, D.; Snider, C.; Nassehi, A.; Yon, J.; Hicks, B. Characterising the Digital Twin: A systematic literature review. *CIRP J. Manuf. Sci. Technol.* **2020**, *29*, 36–52. [CrossRef]

4. Zeb, A.; Kortelainen, J.; Rantala, T.; Saunila, M.; Ukko, J. On the alleviation of imminent technical and business challenges of long-lasting functional digital twins. *Comput. Ind.* **2022**, *141*, 103701. [CrossRef]
5. Le-Anh, T.; De Koster, M. A review of design and control of automated guided vehicle systems. *Eur. J. Oper. Res.* **2006**, *171*, 1–23. [CrossRef]
6. Bechtsis, D.; Tsolakis, N.; Vouzas, M.; Vlachos, D. Industry 4.0: Sustainable material handling processes in industrial environments. In *Computer Aided Chemical Engineering*; Elsevier: Amsterdam, The Netherlands, 2017; Volume 40, pp. 2281–2286.
7. Efthymiou, O.K.; Ponis, S.T. Current status of industry 4.0 in material handling automation and in-house logistics. *Int. J. Ind. Manuf. Eng.* **2019**, *13*, 1382–1386.
8. De Ryck, M.; Versteyhe, M.; Debrouwere, F. Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *J. Manuf. Syst.* **2020**, *54*, 152–173. [CrossRef]
9. Enright, J.J.; Wurman, P.R. Optimization and Coordinated Autonomy in Mobile Fulfillment Systems. In Proceedings of the Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011; pp. 33–38.
10. Nantee, N.; Sureeyatanapas, P. The impact of Logistics 4.0 on corporate sustainability: A performance assessment of automated warehouse operations. *Benchmarking Int. J.* **2021**, *28*, 2865–2895. [CrossRef]
11. Fragapane, G.; De Koster, R.; Sgarbossa, F.; Strandhagen, J.O. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *Eur. J. Oper. Res.* **2021**, *294*, 405–426. [CrossRef]
12. Coelho, F.; Macedo, R.; Relvas, S.; Barbosa-Póvoa, A. Simulation of in-house logistics operations for manufacturing. *Int. J. Comput. Integr. Manuf.* **2022**, *35*, 989–1009. [CrossRef]
13. Ma, N.; Zhou, C.; Stephen, A. Simulation model and performance evaluation of battery-powered AGV systems in automated container terminals. *Simul. Model. Pract. Theory* **2021**, *106*, 102146. [CrossRef]
14. Wang, X.; Wu, W.; Xing, Z.; Chen, X.; Zhang, T.; Niu, H. A neural network based multi-state scheduling algorithm for multi-AGV system in FMS. *J. Manuf. Syst.* **2022**, *64*, 344–355. [CrossRef]
15. Zamiri Marvizadeh, S.; Choobineh, F. Entropy-based dispatching for automatic guided vehicles. *Int. J. Prod. Res.* **2014**, *52*, 3303–3316. [CrossRef]
16. Liu, L.; Qu, T.; Thürer, M.; Ma, L.; Zhang, Z.; Yuan, M. A new knowledge-guided multi-objective optimisation for the multi-AGV dispatching problem in dynamic production environments. *Int. J. Prod. Res.* **2022**, 2122619. [CrossRef]
17. Zardini, G.; Lanzetti, N.; Pavone, M.; Frazzoli, E. Analysis and control of autonomous mobility-on-demand systems. *Annu. Rev. Control. Robot. Auton. Syst.* **2022**, *5*, 633–658. [CrossRef]
18. Liu, Y.; Jia, R.; Ye, J.; Qu, X. How machine learning informs ride-hailing services: A survey. *Commun. Transp. Res.* **2022**, *2*, 100075. [CrossRef]
19. Kim, C.W.; Tanchocoj, J. Operational control of a bidirectional automated guided vehicle system. *Int. J. Prod. Res.* **1993**, *31*, 2123–2138. [CrossRef]
20. Bozer, Y.A.; Yen, C.K. Intelligent dispatching rules for trip-based material handling systems. *J. Manuf. Syst.* **1996**, *15*, 226–239. [CrossRef]
21. Briskorn, D.; Drexl, A.; Hartmann, S. Inventory-based dispatching of automated guided vehicles on container terminals. In *Container Terminals and Cargo Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 195–214.
22. Kim, B.I.; Oh, S.; Shin, J.; Jung, M.; Chae, J.; Lee, S. Effectiveness of vehicle reassignment in a large-scale overhead hoist transport system. *Int. J. Prod. Res.* **2007**, *45*, 789–802. [CrossRef]
23. Kim, B.I.; Shin, J.; Jeong, S.; Koo, J. Effective overhead hoist transport dispatching based on the Hungarian algorithm for a large semiconductor FAB. *Int. J. Prod. Res.* **2009**, *47*, 2823–2834. [CrossRef]
24. Im, K.; Kim, K.; Park, T.; Lee, S. Effective vehicle dispatching method minimising the blocking and delivery times in automatic material handling systems of 300 mm semiconductor fabrication. *Int. J. Prod. Res.* **2009**, *47*, 3997–4011. [CrossRef]
25. Maas genannt Bermpohl, F.; Bresser, A. Towards Autonomous Intralogistics: A Testbed Environment for the Coordination of a Robotic Fleet. In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)—Workshop on Planning and Robotics (PlanRob), Online, 22–23 October 2020.
26. Egbelu, P.J.; Tanchoco, J.M. Characterization of automatic guided vehicle dispatching rules. *Int. J. Prod. Res.* **1984**, *22*, 359–374. [CrossRef]
27. Koenig, S.; Keskinocak, P.; Tovey, C. Progress on agent coordination with cooperative auctions. In Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; Volume 24, pp. 1713–1717.
28. Grunow, M.; Günther, H.O.; Lehmann, M. Strategies for dispatching AGVs at automated seaport container terminals. *Spectr.* **2006**, *28*, 587–610. [CrossRef]
29. Choe, R.; Kim, J.; Ryu, K.R. Online preference learning for adaptive dispatching of AGVs in an automated container terminal. *Appl. Soft Comput.* **2016**, *38*, 647–660. [CrossRef]
30. Ma, X.; Bian, Y.; Gao, F. An improved shuffled frog leaping algorithm for multiload AGV dispatching in automated container terminals. *Math. Probl. Eng.* **2020**, *2020*, 1260196. [CrossRef]
31. Agatz, N.; Erera, A.; Savelsbergh, M.; Wang, X. Optimization for dynamic ride-sharing: A review. *Eur. J. Oper. Res.* **2012**, *223*, 295–303. [CrossRef]

32. Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; Ye, J. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 905–913.
33. Zhang, L.; Hu, T.; Min, Y.; Wu, G.; Zhang, J.; Feng, P.; Gong, P.; Ye, J. A taxi order dispatch model based on combinatorial optimization. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 2151–2159.
34. Kümmel, M.; Busch, F.; Wang, D.Z. Taxi dispatching and stable marriage. *Procedia Comput. Sci.* **2016**, *83*, 163–170. [CrossRef]
35. Gale, D.; Shapley, L.S. College admissions and the stability of marriage. *Am. Math. Mon.* **1962**, *69*, 9–15. [CrossRef]
36. McVitie, D.G.; Wilson, L.B. Stable marriage assignment for unequal sets. *BIT Numer. Math.* **1970**, *10*, 295–309. [CrossRef]
37. Dantzig, G.B.; Ramser, J.H. The truck dispatching problem. *Manag. Sci.* **1959**, *6*, 80–91. [CrossRef]
38. Burkard, R.E.; Cela, E. Linear assignment problems and extensions. In *Handbook of Combinatorial Optimization*; Springer: New York, NY, USA, 1999; pp. 75–149.
39. Kümmel, M. Taxis, Passengers and Stable Marriage. Ph.D. Thesis, Technische Universität München, Munich, Germany, 2016.
40. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef] [PubMed]
41. Jonker, R.; Volgenant, A. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* **1987**, *38*, 325–340. [CrossRef]
42. Santos, C.H.D.; de Queiroz, J.A.; Leal, F.; Montevechi, J.A.B. Use of simulation in the industry 4.0 context: Creation of a Digital Twin to optimise decision making on non-automated process. *J. Simul.* **2022**, *16*, 284–297. [CrossRef]
43. Liu, M.; Fang, S.; Dong, H.; Xu, C. Review of digital twin about concepts, technologies, and industrial applications. *J. Manuf. Syst.* **2021**, *58*, 346–361. [CrossRef]
44. Bai, Y.; You, J.B.; Lee, I.K. Design and Optimization of Smart Factory Control System Based on Digital Twin System Model. *Math. Probl. Eng.* **2021**, *2021*. [CrossRef]
45. Jiang, H.; Qin, S.; Fu, J.; Zhang, J.; Ding, G. How to model and implement connections between physical and virtual models for digital twin application. *J. Manuf. Syst.* **2021**, *58*, 36–51. [CrossRef]
46. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
47. Yang, L.; Worboys, M. Generation of navigation graphs for indoor space. *Int. J. Geogr. Inf. Sci.* **2015**, *29*, 1737–1756. [CrossRef]
48. Reisig, W. *A Primer in Petri Net Design*; Springer: Berlin/Heidelberg, Germany, 2012.
49. Prajapat, N.; Tiwari, A. A review of assembly optimisation applications using discrete event simulation. *Int. J. Comput. Integr. Manuf.* **2017**, *30*, 215–228. [CrossRef]
50. Schmaler, R.; Schmidt, T.; Schoeps, M.; Luebke, J.; Hupfer, R.; Schlaus, N. Simulation based evaluation of different empty vehicle management strategies with considering future transport jobs. In Proceedings of the 2017 Winter Simulation Conference (WSC), Las Vegas, NV, USA, 3–6 December 2017; pp. 3576–3587.
51. Gyulai, D.; Bergmann, J.; Lengyel, A.; Kádár, B.; Czirkó, D. Simulation-based digital twin of a complex shop-floor logistics system. In Proceedings of the 2020 Winter Simulation Conference (WSC), Orlando, FL, USA, 14–18 December 2020; pp. 1849–1860.
52. Berman, S.; Edan, Y. Decentralized autonomous AGV system for material handling. *Int. J. Prod. Res.* **2002**, *40*, 3995–4006. [CrossRef]
53. Parker, L.E. Path planning and motion coordination in multiple mobile robot teams. *Encycl. Complex. Syst. Sci.* **2009**, 5783–5800.
54. van den Berg, J.; Guy, S.J.; Snape, J.; Lin, M.; Manocha, D. RVO2 Library: Reciprocal Collision Avoidance for Real-Time Multi-Agent Simulations. 2016. Available online: http://gamma.cs.unc.edu/RVO2/ (accessed on 18 February 2020).
55. Snape, J.; van den Berg, J.; Guy, S.J.; Manocha, D. Smooth and Collision-Free Navigation for Multiple Robots Under Differential-Drive Constraints. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4584–4589.
56. van den Berg, J.; Guy, S.J.; Lin, M.; Manocha, D. Reciprocal n-Body Collision Avoidance. In *Robotics Research: The 14th International Symposium ISRR*; Springer Tracts in Advanced Robotics; Springer: Berlin/Heidelberg, Germany, 2011; Volume 70, pp. 3–19.
57. Fiorini, P.; Shiller, Z. Motion Planning in Dynamic Environments Using Velocity Obstacles. *Int. J. Robot. Res.* **1998**, *17*, 760–772. [CrossRef]
58. Draganjac, I.; Miklić, D.; Kovačić, Z.; Vasiljević, G.; Bogdan, S. Decentralized control of multi-AGV systems in autonomous warehousing applications. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1433–1447. [CrossRef]
59. Kim, C.; Tanchoco, J.; Koo, P.H. AGV dispatching based on workload balancing. *Int. J. Prod. Res.* **1999**, *37*, 4053–4066. [CrossRef]
60. Beamon, B.M. Performance, reliability, and performability of material handling systems. *Int. J. Prod. Res.* **1998**, *36*, 377–393. [CrossRef]