

# Evaluating automated storage and retrieval system policies with simulation and optimization

Michele Barbato, Alberto Ceselli, Marco Premoli

**Abstract** In this paper we present a methodology to evaluate policies for automated storage and retrieval system (AS/RS) in warehouses. It is composed by four steps: (i) formal definition of the physical AS/RS and descriptive modeling on a simulation framework; (ii) model validation and finding of potential bottlenecks by the statistical analysis of data logs; (iii) definition of operational optimization policies to mitigate such bottlenecks; (iv) evaluation of the policies using the simulation tool through key performance indicators (KPI). In particular, we take into consideration a unit-load AS/RS, we present a new simulation model combining discrete events and agent based paradigms. We consider an industrial test case, focusing on scheduling policies that exploit mathematical optimization, and we evaluate the effects of our approach on real world data. Experiments prove the effectiveness of our methodology.

**Keywords:** AS/RS policies; simulation; matching; mathematical optimization

## 1 Introduction

Automated storage and retrieval systems (AS/RSs) are widely used in warehouses and distributions centers all around the world since their introduction in the 1960s. Their main advantages are high space utilization, reduced labor costs, short retrieval times and improved inventory control. In order to get the best performance from an AS/RS, its design and operational planning should be wisely chosen.

A vast literature has accumulated over the years, both on general warehousing systems [2, 8, 7] and on diverse optimization problems arising in the design of AS/RS policies [3, 10]. While all these works carry out numerical studies, very few of them explicitly tackle the task of providing a full descriptive model of such a complex system: in [4] several rules of thumb to design a simulation tool for warehouse

---

Michele Barbato, Alberto Ceselli, Marco Premoli  
Università Degli Studi di Milano, Department of Computer Science, 18, via Celoria, Milano (Italy)  
e-mail: {firstname}.{lastname}@unimi.it

are presented, such as key performance indicators, simulation initialization, object-oriented design patterns to implement the simulation tool; authors also present a case study using discrete-event paradigm; authors in [6] aim to present a discrete event simulation model that can be easily extended to most of the features found in AS/RS industrial settings; [9] is one of the few examples in which the agent-based paradigm is used; authors propose a multi-agent model for a mini-load system, claiming that autonomous vehicles control is easier in such paradigm.

Indeed, there is a large gap in the literature. On one side, optimization algorithms for AS/RS assume simplified models and hypothetical critical sub-systems, thereby providing solutions whose practical implementation potential still needs to be validated. On the other side, simulation approaches propose more realistic and reliable descriptive models, disregarding however the algorithmic resolution of key AS/RS optimization problems. In this paper we aim at reducing this gap.

We present a four-steps methodology to identify and evaluate optimization policies for AS/RS: (i) formal definition of a physical AS/RS and modeling on a simulation framework (Sect. 2.1), (ii) model validation and finding of potential bottlenecks by statistical analysis of data logs (Sect. 2.2), (iii) definition of operational optimization policies to mitigate such bottlenecks (Sect. 2.3), (iv) policies evaluation using the simulation tool through *key performance indicators* (KPI) (Sect. 2.4).

In particular, we present a new mixed multi-agent and discrete-event simulation model for a unit-load AS/RS that, besides being very common in practice, matches the technology used by our industrial partners [1], and we propose a scheduling policy to tackle the found bottleneck exploiting mathematical optimization, which in turn extends a shift-based scheduling policy found in literature.

Finally, in Section 3 we draw our conclusions.

## 2 Design and Evaluation of AS/RS Optimization Policies

In the following we detail the four steps of our methodology.

### 2.1 Step (i): AS/RS descriptive model

The AS/RS under study has one aisle with racks on both sides. Each rack has double depth homogeneous cells with fifty-three lanes and nine levels. Stock-keeping-units (SKU) are represented by uniform size bins, with the same size of a rack slot. SKUs are moved with a double-shuttle aisle-captive crane, with separate drives for horizontal and vertical movements. The AS/RS has two I/O points, one in the front-end and one central, both with FIFO conveyor buffers to access the crane. The central I/O is connected to working stations where operators acts on SKUs. To move the SKU from (resp. to) the central-bottom I/O to (resp. from) the working stations an automated shuttle of capacity two is used, which moves on an horizontal rail.

Working stations are used to compose customers orders. An order contains multiple items and is collected in a dedicated SKU (*order bin*). After the arrival of a SKU containing an item (*item bin*) in a working station, the item has to be processed before being assigned to an order bin; hence, the item bin has to wait on a working station before it to be stored again in the warehouse.

Daily orders have no release date and have to be completed within the working day without priorities. Items can be inserted in an order bin in any sequence. Multiple orders can be processed simultaneously (i.e. multiple order bins can be present on working stations) by multiple operators. Operators do not have to coordinate with each other. The complete list of orders to build is known at the beginning of the working day. The quantity of items needed to build all orders is available at the beginning of the working day.

**AS/RS descriptive model.** To build a model of the AS/RS process described previously we characterize it by means of entities, events and activities. We can identify six types of entities:

- operators: resident entities, characterized by a cycle of activities that act on the SKUs; an operator  $O$  requests an empty order bin and a sequence of item bins, one at a time, to be retrieved on working stations; each time the production process on an item bin is over,  $O$  requests its storage back to the rack; when the order is complete,  $O$  requests the storage of the order bin and the cycle starts over;
- the system controller, or *orchestrator*: it is the second resident entity, characterized by a cycle of activities that act on the SKUs, on the crane and on the shuttle; the orchestrator listens to operators requests and translates them into actions to be executed by the last two entities, which in turns moves SKUs in the AS/RS;
- three resource entities, in charge of the execution of the orders given by the orchestrator to move SKUs: crane, shuttle and the conveyors. These resources need to synchronize in order to pass SKUs to each other: for example if one of the conveyor is full, the crane can not deliver an SKU on the conveyor until another SKU exits on the other side of the conveyor, and it has to stop and wait.
- SKUs: can be identified as the transient entities, characterized by the type, either item bin or order bin, and by activities like (i) storage and retrieval from the front-end I/O, (ii) storage and retrieval from the working stations, (iii) handling inside the warehouse and inside the working stations; each activities begin with an event generated either by an operator or by the system controller.

In Fig. 1 we present the schema of the relationships among model entities: an operator interacts with the orchestrator sending requests and receives notifications by the orchestrator and by the requested SKU; the orchestrator receives operators requests and translate them into orders for the crane and the shuttle; these latter move SKUs with pick and drop synchronizations; the synchronization among resources (crane, shuttle and conveyor) is done through the moved SKU, which acts as a mediator; at last, an SKU notifies the operator when it arrives at the working station.

We can conveniently model each entities as an agent in our model and every relationship with a message passing interface.

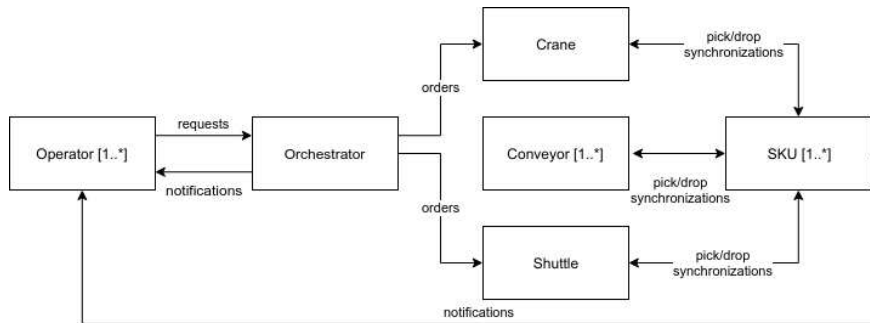


Fig. 1 Simulation Model Entities Relationship

At the same time, a discrete events approach can be used to model order composition of single operators, which generate requests of SKUs to the orchestrator: (i) the operator starts an order composition requesting to the orchestrator an empty order bin and the first item bin to process; (ii) it awaits until both SKUs arrive; (iii) when both SKUs arrive, it starts the working process on the item bin, which can require a non-deterministic amount of time; (iv) at the end of the working process, it requires the storage of the item bin to the orchestrator; (v) if the order is not finished, it requires the retrieval of the next item bin to the orchestrator, otherwise (vi) if a new order is available, the process goes back to step (i), otherwise it ends.

A critical analysis of our modeling framework allows to identify both strengths and weaknesses of our approach.

**Strengths.** The synchronization among resource entities is simplified; resource entities act autonomously and their coordination is modeled via a message passing interface that might be asynchronous.

Resource agents statechart can be generalized and it abstracts from the features of the system; it includes few general states: (i) idle state, where the resource waits new orders from orchestrator; when new orders arrive, a loop of order execution starts and (ii) the resource moves to the order execution location; when it arrives (iii) it sends the order (either pick or drop) to the SKU and awaits for its answer, which notifies the end of the order; if any order is left, the loop continues from (ii), otherwise the resource goes back to idle state (i).

A resource has no knowledge on the status and features of the external system; on the contrary, the orchestrator has a centralized view of the complete system, and it considers the current status of this latter to generate the sequence of orders for the resources. A resource stores parameters characterizing its current internal state, changed by the single order. For example the crane stores its current location (aisle, lane, level) and the SKUs currently on board.

**Weaknesses.** The modeling complexity is shifted towards: (i) the statechart of the SKU entity, which acts as a mediator among resource entities and has to consider all

combinations of orders from the resources and its current position, in order to enable customization of execution time of pick and drop actions for each combination; (ii) the orchestrator agent, which listens for operators request and has to implement an on-line scheduling algorithm to generate the sequence of SKU movements. This algorithm has to consider the complete features and statuses of the system, which is not visible to the single components.

However, we stress that the complexity of the orchestrator agent, that looks like a weakness at a descriptive modeling level, turns out to offer a large degree of flexibility as soon as optimization models and policies are implemented to define its logic. It therefore fits perfectly in our setting.

## 2.2 Step (ii): Model Validation And Bottleneck Analysis

Thanks to an industrial partner we had access to a dataset with the mechanical movements of SKUs in an AS/RS with the characteristics described in Section 2.1.

In particular we had access to all timestamps of arrival of an entity in a position in the warehouse, such as the arrival of the crane at a cell, and the end of pick and drop of an SKU from a resource (crane, shuttle and conveyor), covering 188 days.

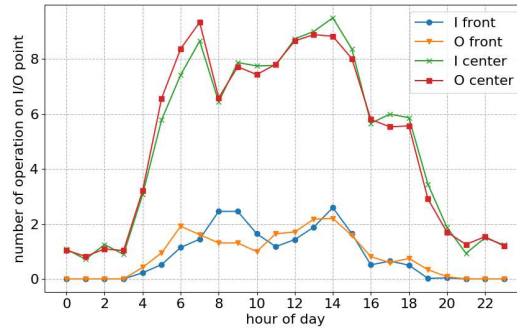
Unfortunately, we had no access neither to the time, and purpose, for which operators sent their request to the system, nor to the orders composition.

**Travel time model and validation** First, we make use of the available data logs to validate the implementation of the travel time models within our simulation tool.

Let  $(c, l)$  represent the position of a rack cell, where  $c$  is the horizontal lane and  $l$  is the vertical level. We model crane movements from location  $(c_1, l_1)$  to  $(c_2, l_2)$ , with the following deterministic function:  $\max(|c_1 - c_2| \cdot v_x^c, |l_1 - l_2| \cdot v_y^c) + f^c$  where we sum a constant value  $f^c$ , with which we model the picking/dropping time and the time to reach a constant speed, to the traveling time modelled with Chebyshev distance metric where  $v_x^c$  is the constant horizontal speed (seconds per lane) of the rack and  $v_y^c$  is the constant vertical speed (seconds per level). It would be easy to model a travel time that depends also on the load of the crane, which can be computed by the crane agent starting from the currently loaded SKUs; however the data log analysis showed that the the speed of the considered crane is not affected by the load.

We model shuttle movements from working station  $w_1$  to working station  $w_2$  with the following linear function:  $|w_1 - w_2| \cdot v_x^s + f^s$ , where we sum a constant value  $f^s$ , which models the picking/dropping time and the time to reach a constant speed, to the travelling time modeled as a uniform linear motion with a constant horizontal travel speed  $v_x^s$  (seconds per working station). Finally, we model conveyors travel time with the a constant value  $f^r + t^r$  where  $f^r$  is the constant time of picking/dropping on the conveyor and  $t^r$  is the constant time to travel all the conveyor length. Values of all parameters were estimated analyzing the real world dataset.

We implemented our simulation tool with Anylogic, a Java based simulation software supporting discrete-event, system dynamic and agent-based modelling, and we validated our simulation model using five complete days of our dataset, comparing



**Fig. 2** Average Hourly Usage of AS/RS I/O points

the time  $\tau_r$  measured by our industrial partner on the real system with the time  $\tau_s$  arising from simulations of single actions for both the crane and the shuttle, for a total of around 2500 actions for both.

Differences of crane movements  $\Delta^\tau = \tau_r - \tau_s$  could be approximated by a normal distribution with mean  $\mu \simeq 0.339$  sec. and standard deviation  $\sigma \simeq 3.688$ , where the mean time for a single action is around 45 sec. For shuttle movements, differences  $\Delta^\tau$  have a median value of  $-0.73$  sec., a first quartile of  $-1.135$  sec. and a third quartile of  $-0.36$  sec., having a mean time for the single action around 30 sec.. We also measured the overall time spent by crane and shuttle in mechanical operations, finding the average gap between real data and simulated one to be about 1.79%. That is, simulated data match well data of the real world system.

**Identification of bottlenecks** Intuitively, we consider KPIs and event categories, and we evaluate if the disaggregation of events in categories makes KPI patterns to arise. Following this approach, we found a decomposition of storage and retrieval operations arising from the two different I/O points to be suitable for our approach. In Fig. 2 we present the average hourly usage of each I/O points in the dataset. It is clear that central I/O points, towards working stations, have higher frequency of usage with respect to front-end I/O points, which in turn occur throughout the day.

Moreover, we know from our industrial partner that: (i) the front-end I/O operations can be executed without overlapping with central I/O operations, at the beginning and at the end of the working day; (ii) all front-end I/O operations are known at the beginning of the day and, within them, storage operations can be executed independently from retrieval operations.

From the union of the insights coming from the analysis of the data and the industrial knowledge a new scheduling policy emerges to improve SKUs movements, that is the disaggregated optimization of front-end I/O operations and central I/O operations. Such problem has been identified in [5] as the *shift-based sequencing* problem for the AS/RS with twin-shuttle crane and single depth rack. Given a set of SKUs to retrieve (or to store), authors: (i) model the set as a weighted undirected

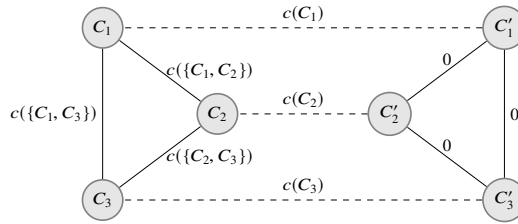


Fig. 3 Example of graph  $T$  with  $C^* = \{C_1, C_2, C_3\}$

graph, whose edges weights are given by a distance metric obeying the triangle inequality and represent the time to perform an action on pairs of SKUs; (ii) the sequence of actions to perform is given by the solution of a minimum-cost perfect matching problem over the built graph, which can be solved at optimality with a polynomial time algorithm.

### 2.3 Step (iii): Optimizing Bottleneck

In cases where the distance metric does not obey the triangle inequality, the modeling proposed in [5] does not guarantee optimality; in particular it always forms pairs of SKUs on which to perform an action, while with generic distance metric moving a single SKU might be less expensive. We propose the following variant: let  $C^*$  be the set of all SKUs that have to be retrieved; let  $c(C)$  be the time required to retrieve SKUs  $C$ , i.e. the sum of the time for the crane to go from the I/O point to the cell containing  $C$ , to pick  $C$  on the crane and to return to the I/O point; let  $c(\{C_1, C_2\})$  be the time required to retrieve the pair of SKUs  $C_1$  and  $C_2$ , i.e. the sum of time for the crane to go from the I/O point to the cell containing  $C_1$ , to go from  $C_1$  to  $C_2$ , to return to the I/O point and to pick both  $C_1$  and  $C_2$  on the crane.

Let us consider  $G$  and  $H$  two complete graphs on  $|C^*|$  vertices. Each vertex of  $G$  corresponds to one SKU in  $C^*$ . We assign weight  $c(\{C_1, C_2\})$  to each edge  $\{C_1, C_2\}$  of  $G$ . We assign weight 0 to each edge of  $H$ . We then link each vertex of  $G$  with its counterpart vertex in  $H$ . If  $C \in V(G)$  and  $C' \in V(H)$  are linked we assign weight  $c(C)$  to edge  $\{C, C'\}$ . Let  $T$  denote the weighted graph obtained so far. By using the edges  $\{C, C'\}$  with  $C \in V(G)$  and  $C' \in V(H)$ , we see that  $T$  always has a perfect matching and that a minimum weight perfect matching of  $T$  gives an optimal retrieval plan for  $C^*$ . An example of a weighted graph  $T$  is presented in Fig. 3.

**Retrieval and Storage.** In the case of SKU retrieval in a double depth rack, we need to consider that several SKUs might be not directly accessible by the double-shuttle crane, i.e. they lie in the back slot while the front slot is storing another SKU. The retrieval for such SKU is possible only when both shuttles of the crane are free. To consider this scenario, let  $B \subseteq C^*$  be the set of such not directly accessible SKUs, and  $A = C^* \setminus B$  be the set of SKUs directly accessible by a single shuttle crane. We

set to  $+\infty$  the cost  $c(\{C_1, C_2\})$  of all edges  $\{C_1, C_2\} \in T$  connecting a pair of vertices  $C_1, C_2 \in B$ , while edges connecting all other pairs of vertices remain unchanged. Operatively, to retrieve a pair of SKU  $C_1 \in B$  and  $C_2 \in A$ ,  $C_1$  will be retrieved first, when both shuttles of the crane are free.

As for the storage, we define this simple storage policy for the operations under optimization: incoming SKUs are stored only in the front slot of each cell in the rack.

**Optimization policy.** Our simple shift-based scheduling policy for a double-depth rack AS/RS with twin-shuttle crane is defined as follows:

- if an SKU has to be retrieved and it is not used in a working station throughout the day, it is retrieved at the beginning of the day, otherwise at the end of the day;
- if an SKU has to be stored and it is not used in a working station throughout the day, it is stored at the end of the day, otherwise at the beginning of the day;
- the position where to store incoming SKUs at the beginning of the day is chosen among front slot cells that are not used throughout all day. Among those cells, the nearest to the central I/O point are chosen;
- the position where to store incoming SKUs at the end of the day is chosen among all free front slot cells;
- the decomposed retrieval and storage operations, both for the morning and the evening, are optimized with the min-cost perfect matching algorithm.

The original order of center I/O operations does not have to be changed, and we can compare original and optimized operations.

## 2.4 Step (iv): Experimental Evaluation

We propose the following experimental setup to evaluate our scheduling policy using our simulation tool: given a complete day of operations from our dataset, we apply our shift-based scheduling policy. We compare the original and the modified sequence of operations, feeding them to our simulation tool in a trace-driven style. In this scenario, we do not use the orchestrator and the operators agents, but we directly order crane and shuttle to move SKUs in a given sequence. These resources will execute one order at a time, synchronizing independently with each other; once they finish an order, they will ask for the next one until no more orders are available.

**Key Performance Indicator.** As KPI, an interesting measure would be the total waiting time of an SKU by an operator, and the total amount of time to complete all orders. However, given that we can not recover operators information with the available data, we consider as KPI the total amount of time spent by the crane to move all SKUs, comparing the original actions with the modified ones.

**Experimental results.** In Table 1 we present the results of the execution of the shift based scheduling on ten different days of our dataset. We compare the time  $t$



d.	$I_o$	front-end I/O actions			central I/O actions			all actions			mechanical times
		$t_o^f$	$t_m^f$	$\frac{(t_o^f - t_m^f)}{t_o^f}$	$t_o^c$	$t_m^c$	$\frac{(t_o^c - t_m^c)}{t_o^c}$	$t_o$	$t_m$	$\frac{(t_o - t_m)}{t_o}$	$\frac{(\phi_r - \phi_m)}{\phi_r}$
1	1.51%	1671	1423	14.87%	16657	16590	0.40%	18328	18013	1.72%	7.88%
2	0.80%	2313	1942	16.04%	14787	14936	-1.01%	17100	16878	1.30%	3.54%
3	1.28%	2887	2200	23.78%	13549	14024	-3.50%	16436	16224	1.29%	1.58%
4	1.17%	1923	1555	19.16%	14463	14754	-2.01%	16386	16309	0.47%	7.57%
5	1.32%	2113	1715	18.80%	13601	14033	-3.18%	15713	15748	-0.22%	1.00%
6	0.40%	2867	2021	29.51%	14134	14516	-2.70%	17001	16537	2.73%	1.63%
7	0.99%	3119	2309	25.96%	11674	11909	-2.01%	14793	14218	3.89%	2.80%
8	1.44%	4326	3709	14.26%	12009	12307	-2.48%	16335	16016	1.95%	0.24%
9	3.81%	7363	5696	22.64%	12689	12921	-1.82%	20052	18616	7.16%	9.57%
10	0.33%	2019	1788	11.47%	18014	17927	0.49%	20033	19714	1.59%	0.30%

**Table 1** Comparison of total time spent by the crane (in seconds)

(in seconds) spent by the crane in the original (subscript  $o$ ) and modified (subscript  $m$ ) sequence of actions, for the front-end I/O actions (superscript  $f$ ), the central I/O actions (superscript  $c$ ) and all set of actions (no superscript). In particular we present the percentage gain or loss of time given by the modified sequence  $((t_o - t_m)/t_o)$ : a positive value is a gain, a negative value a loss.

For every instance,  $I_o$  is the percentage of central I/O actions that in the original sequence are nested with front-end I/O actions. In average,  $I_o$  is around 1% on all instances, except instance 9 where its value is 3.81%. This latter also corresponds to a gain of 7% given by our scheduling policy, which in fact always grants  $I_m = 0\%$ .

In all instances the modified sequence provides a substantial improvement in execution time for the front-end I/O actions, in the range 10 – 30%. However, the central I/O actions have higher execution times with a worsening in the range 1 – 3%, due to the new positions of SKUs entering from the front-end I/O point, defined by our simple storage policy. Considering the complete sequence of actions, the modified sequence grants only a slight improvement in the range 0.5 – 3%, with a single substantial gain around 7% but also one worsening of 0.22%.

Although our policy leads to a substantial time gain on the optimized operations, the advantage on the total operations time is limited. The time loss on central I/O operations indicates that a better storage policy is needed for the shift-based scheduling, to make a cleverer use of the rack cells.

Finally, in last column of Table 1, we report relative gaps between the overall time spent by crane and shuttle in mechanical operations measured on the real data ( $\phi_r$ ), and that obtained by the modified sequence in our simulation ( $\phi_m$ ). The overall picture is similar to that concerning SKUs operation times.

### 3 Conclusions

We proposed a generic approach to the identification of critical AS/RS sub-components, and the design of corresponding optimization policies. It makes use of both descriptive and optimization models.

Our framework turned out to be more flexible than those proposed in the literature; one of its key features is to shift complexity in the logic component of an orchestrator, that in turn can be effectively encoded by optimization models.

The building of a descriptive model turns out to be an asset of our methodology, allowing to (indirectly, but reliably) evaluate the effect of optimization on the real world system. In our case, the design and application of a custom optimization policy allows to reduce operational times of certain system functions at the expense of others. We indeed believe this to be a common issue when optimization policies are applied in practice. The potential of combining optimization and simulation actually unfolds at this stage: the decision maker can evaluate the impact of different optimization strategies, choosing the one yielding a suitable balancing for his/her system.

**Acknowledgment** Partially funded by Regione Lombardia, grant agreement n. E97F17000000009, Project AD-COM.

## References

- [1] AD-COM. Proj. Website. <https://www.ad-com.net>, 2019. last access 04/2019.
- [2] N. Boysen, R. de Koster, and F. Weidinger. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 2018.
- [3] N. Boysen and K. Stephan. A survey on single crane scheduling in automated storage/retrieval systems. *European Journal of Operational Research*, 254(3):691–704, 2016.
- [4] V. Colla and G. Nastasi. Modelling and simulation of an automated warehouse for the comparison of storage strategies. In *Modelling, Simulation and Optimization*. IntechOpen, 2010.
- [5] D. R. Dooly and H. F. Lee. A shift-based sequencing method for twin-shuttle automated storage and retrieval systems. *IIE Transactions*, 40(6), 2008.
- [6] J.-P. Gagliardi, J. Renaud, and A. Ruiz. A simulation modeling framework for multiple-aisle automated storage and retrieval systems. *Journal of Intelligent Manufacturing*, 25(1):193–207, 2014.
- [7] J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse operation: A comprehensive review. *European journal of operational research*, 177(1):1–21, 2007.
- [8] J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, 203(3):539–549, 2010.
- [9] M. Güller and T. Hegmanns. Simulation-based performance analysis of a miniload multishuttle order picking system. *Procedia CIRP*, 17, 2014.
- [10] T. Wauters, F. Villa, J. Christiaens, R. Alvarez-Valdes, and G. V. Berghe. A decomposition approach to dual shuttle automated storage and retrieval systems. *Computers & Industrial Engineering*, 101:325–337, 2016.