Operations Research Perspectives 6 (2019) 100128

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

Decision rules for robotic mobile fulfillment systems

M. Merschformann*[,a], T. Lamballais[b], M.B.M. de Koster[b], L. Suhl[a]

[a] DS&OR Lab, Paderborn University, Warburger Str. 100, Paderborn, 33098, Germany
[b] Rotterdam School of Management, Erasmus University, PO Box 1738, Rotterdam, 3000 DR, The Netherlands

ARTICLE INFO

Keywords:
Logistics
Warehouse control
Simulation
Robotic mobile fulfillment system
Parts-to-Picker system

ABSTRACT

The Robotic Mobile Fulfillment Systems (RMFS) is a new type of robotized, parts-to-picker material handling system, designed especially for e-commerce warehouses. Robots bring movable shelves, called pods, to workstations where inventory is put on or removed from the pods. This paper simulates both the pick and replenishment process and studies the order assignment, pod selection and pod storage assignment problems by evaluating multiple decision rules per problem. The discrete event simulation uses realistic robot movements and keeps track of every unit of inventory on every pod. We analyze seven performance measures, e.g. throughput capacity and order due time, and find that the unit throughput is strongly correlated with the other performance measures. We vary the number of robots, the number of pick stations, the number of SKUs (stock keeping units), the order size and whether returns need processing or not. The decision rules for pick order assignment have a strong impact on the unit throughput rate. This is not the case for replenishment order assignment, pod selection and pod storage. Furthermore, for warehouses with a large number of SKUs, more robots are needed for a high unit throughput rate, even if the number of pods and the dimensions of the storage area remain the same. Lastly, processing return orders only affects the unit throughput rate for warehouse with a large number of SKUs and large pick orders.

1. Introduction

The rise of e-commerce has created the need for new warehousing systems. Traditional, manual picker-to-parts systems work best when orders are large, i.e. consist of many SKUs so that consolidation has to be organized well. However, e-commerce orders are typically small and e-commerce warehouses are often large as they need to contain large assortments of products, which results in long walking distances for the pickers. In contrast to manual picker-to-part systems, automated parts-to-picker systems eliminate the time pickers spend traveling. Thus, they can achieve higher pick rates.

The Robotic Mobile Fulfillment System (RMFS) is an automated parts-to-picker system. Robots transport movable shelves, called "pods", that contain the inventory, back and forth between the storage area and the workstations. As RMFSs eliminate picker walking time, high pick rates can be expected. The systems are mainly used by Amazon, which bought the company that invented the RMFS, Kiva Systems, and has since deployed more than 100,000 robots in its warehouses (see [18]). Recently, competitors such as Swisslog, Interlink, GreyOrange, Mobile Industrial Robots and Scallog have been rolling out their versions of an RMFS.

The RMFS is described in more detail in [5] and [19]. They mention that numerous operational decision problems are yet to be examined in depth, for example the assignment of customer orders to workstations or of pods to storage locations. Each of these decision problems comes with a trade-off. An order may be assigned to a workstation if it is nearing its due time, but assigning another order that has lines in common with other orders assigned to that workstation may result in more picks per pod and hence a reduction in the number of pod trips. Furthermore, assigning a pod to a storage location that is close to the workstation reduces travel time, but keeping the inventory sorted by assigning pods to favorable storage location if they are likely to be needed in the near future may reduce travel times more.

These trade-offs are linked to the number of robots in the system. As an example, with more robots, more trips can be done and hence the order due times can become a more important criterion than the number of picks per pod when selecting a pod to be transported to a workstation. The trade-offs are also linked to the resources and conditions in the warehouse. For example, the more SKUs a warehouse contains, the more difficult it becomes to assign orders to pick stations in such a way that multiple products can be picked from a single pod.

As these examples indicate, a need exists for finding methods to

* Corresponding author.
E-mail addresses: marius.merschformann@upb.de, marius.merschformann@uni-paderborn.de (M. Merschformann), lamballaistessensohn@rsm.nl (T. Lamballais), rkoster@rsm.nl (M.B.M. de Koster), leena.suhl@upb.de (L. Suhl).

https://doi.org/10.1016/j.orp.2019.100128
Received 27 March 2019; Received in revised form 7 October 2019; Accepted 21 October 2019
Available online 01 November 2019

address the decision problems in an RMFS, for research on the performance of RMFSs across performance measures, and for examining performance while varying aspects like the number of robots. This paper addresses this need. We study the pick order assignment, replenishment order assignment, pick pod selection, replenishment pod selection, and pod storage assignment decision problems and propose several decision rules for each. To see which trade-offs in performance may exist, we use different performance measures. Furthermore, we vary three aspects of the RMFS, namely whether or not return orders need to be processed, the size of the orders, and the number of SKUs in the warehouse. This study focuses on both the pick process and the replenishment process, because a more efficient replenishment process frees up robots for pick tasks. Lastly, the number of pick stations and the number of robots per pick station is varied. Varying these numbers shows how many pick stations and robots are needed to provide pickers with a near continuous supply of pods.

In the following we outline the main contributions of this work:

- A hierarchical definition of the core RMFS control decision problems
- A number of intra-logistic-typical and RMFS-tailored decision rules suitable for controlling RMFS
- A large two-stage experiment providing insights about the performance of the rules and importance of the decision problems
- The source code of all decision rules and the evaluation framework itself is available for open-access at: https://github.com/merschformann/RAWSim-O

Section 2 describes the RMFS in more detail, Section 3 points out related work, Section 4 the decision problems, and Section 5 the decision rules, while Section 6 explains the evaluation framework, Section 7 shows the results of the analysis, and Section 8 provides conclusions and directions for future research. For a description of the realistic simulation built for evaluating the decision rules refer to Appendix A.

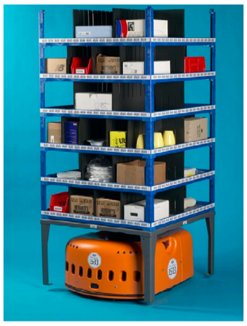
2. The robotic mobile fulfillment system

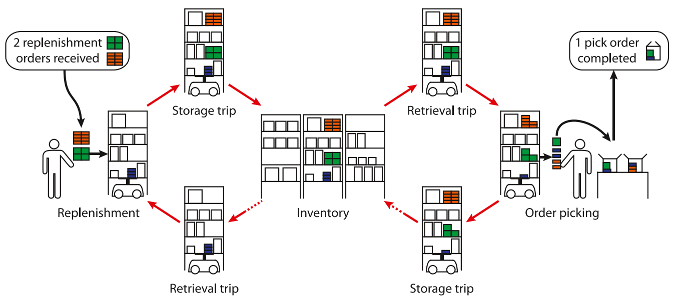
An RMFS consists of shelves on which products are stored (called pods), robots that can move underneath and also carry them (see Fig. 1a), and work stations.

Fig. 1 b shows the storage and retrieval processes, where the robots transport pods between the workstations and the storage area. Starting at the replenishment station, in the example, two replenishment orders with 4 and 8 units of two SKUs (green & orange) are stored on a pod that was retrieved from the inventory by a robot. Some units of the blue SKU, also relevant to the process example, have already been available on the pod at this point. After the pod was handled at the station it is stored in inventory again. Next, if the pod is selected for picking at a pick station, it is brought to that station. The operator at the station then picks the units matching the open order lines at the station from the pod and puts them into the bins for the respective pick orders. As soon as a pick order is completed it leaves the pick station and is handled by further warehouse systems. If zoning is in place at the warehouse, the pick order may only be a part of a larger customer order and must be consolidated further with the other partial pick orders in a following sortation process. If the customer order is already completely fulfilled at the pick station, it may be packed into a carton and prepared for shipping immediately with no further handling. The latter may only be possible in e-commerce operations where lines per order are small.

Each pair of storage and retrieval trip is one robot cycle in an RMFS. During one cycle the robot does not set-down or leave the rack until it is returned to a storage location. Note that, the pod may be brought to further replenishment or pick stations between the retrieval and the storage trip, if further replenishment or immediate picking can be done with it. While the operation of the robot is cyclic the flow of the inventory units through the system starts at a replenishment station (by storing a replenishment order) and exits at a pick station (by fulfilling a pick order). However, in contrast to other systems there is quite some overhead inventory movement, because all contained units, not only needed ones, are moved when a pod is brought to a station. The same happens during replenishment operations, if non-empty pods are moved to a replenishment station.

Robots navigate their paths through the warehouse using a waypoint system, which is laid out as a grid. A path is a sequence of connected waypoints and all robots have to be guided concurrently along their paths while avoiding collisions and deadlocks. Robots that are not carrying a pod can move underneath stationary pods and hence take other paths than robots that do carry pods. The system layout is depicted in Fig. 2 and consists of a storage area where the pods are stored, pick and replenishment stations grouped around the storage area, maneuvering areas between the storage area and the workstations, and per workstation a buffer area. A robot carries a pod from the storage area, via the maneuvering area, to the buffer area of the destination workstation. Pods are picked or replenished one at a time per station. Workers at the replenishment stations replenish the pods with new inventory. In contrast, workers at the pick stations pick product units to fulfill orders. A picker picks for multiple unfinished/incomplete pick orders at the same time. For both operations the robots need to stop with a pod at a waypoint representing the access point of the respective station. In the buffer area next to each workstation, robots carrying pods can wait for their turn. In the middle of the layout a number of waypoints is used as possible storage locations where pods can be put when they are not used. Every storage location is directly reachable from an aisle and access to a storage location cannot be blocked by stored pods. Travel in the aisles is single-directional to avoid gridlock


(a) Robot carrying a pod (see Enright and Wurman (2011))



(b) The internal storage / retrieval process in RMFSs (red: robot & pod movement

Fig. 1. The essential elements of an RMFS.

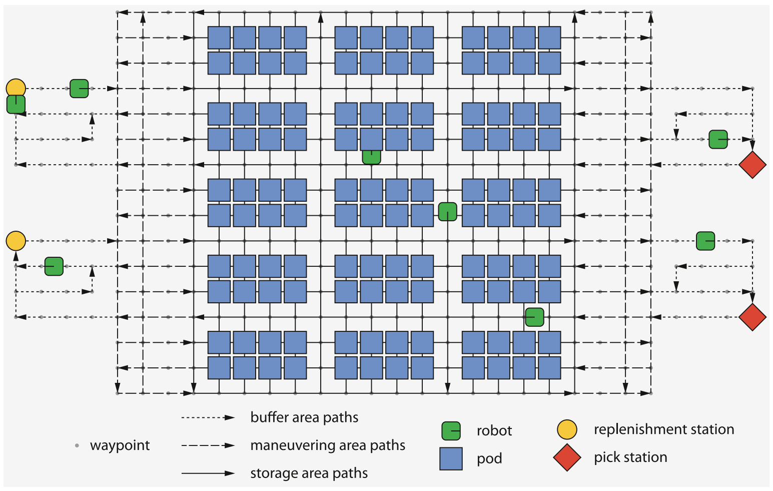


Fig. 2. A top view of an RMFS layout. The lines indicate the directed graph used for robot navigation.

and reduce congestion.

The system has the ability to adapt to changing demand conditions. E.g., if order arrival rates of some SKUs drop, pods containing those SKUs can be relocated further away from the pick stations. This relocation frees up storage locations near the pick stations for pods containing SKUs with high order arrival rates. Pods can be relocated when returning from a workstation, hence the inventory can be continually sorted in response to changing demand.

3. Related work

To this date no detailed discrete event simulation based research on control topics has been done for RMFS. Moreover, most research on RMFSs to date uses queueing networks to study design questions on the strategic level. This work aims to close the gap by delivering insights about RMFS using a very detailed simulation framework that integrates most dynamic effects an operator faces. Next, we first outline the queuing network based research and close this section with simulation based work.

Nigam et al. [14] create queueing networks similar to earlier queueing networks used for autonomous vehicle storage and retrieval systems (AVS/RS) and automated storage and retrieval systems (AS/RS) (see [8] and [17]). Their queueing networks capture both pick and replenishment operations but cannot model robot movement realistically. They estimate the order throughput time for single-line orders. Lamballais et al. [10] create a different queueing network for both single- and multi-line orders, with and without zoning in the storage area, that captures only the pick operations, but that does include realistic robot movement. Their model can accurately estimate the expected order cycle time, workstation utilization and robot utilization. Lamballais et al. [10] determine how the storage area dimensions and the workstation placement around the storage area affect the maximum order throughput, by evaluating a large number of possible designs. Lamballais et al. [11] develop a queueing network that addresses problems on a tactical level. They show the effect of the number of pods per SKU and of the replenishment level of a pod on order throughput, and they show what the optimal ratio of the number of pick stations to the number of replenishment stations is. They find that it is better to replenish pods before they are entirely empty, even with multiple pods per SKU. Zou et al. [20] use semi-open queueing networks to analyze the policy for assigning robots to pick stations. The authors find that the random policy is significantly outperformed by the proposed handling-speeds-based assignment rule when facing varying service rates of the pickers. Zou et al. [21] build a semi-open queueing network for evaluating the effects of battery management in RMFS. The strategies of battery swapping, automated plug-in charging and inductive charging at the pick station are compared. The authors come to the conclusion that battery swapping is generally more expensive than plug-in charging while inductive charging outperforms both in throughput and costs, if robot prices and retrieval times are low.

Enright and Wurman [5] and Wurman et al. [19] mention several decision problems on the operational level that they encountered in practice. One of the few studies that address decision problems on the operational level is by Boysen et al. [2]. They provide methods for optimally batching the pick orders and sequencing both the pick orders and the pods transported to the stations. They show that an optimized pick order processing requires only half the number of robots that a pick order process based on simple decision rules would need. Bozer and Aldarondo [3] devise a simulation study to compare the RMFS to a miniload order picking system. The authors find that for the assumed scenarios and parameters a miniload system with four aisles and one conveyor loop yields approximately the same performance as an RMFS with 50 AGVs. In the experiments one fixed control logic is used. Guan and Li [6] focus on scattered storage assignment in RMFS, i.e., assigning products to multiple pods. The main objective is to distribute inventory, such that the number of pods needed to fulfill a given set of orders is minimized. Since the set of orders is unknown at the time of replenishment, product similarity (based on association rules) is used to determine the pods contents. The authors formulate a MIP model and solve it in two-stages using a genetic algorithm. In the conducted experiments the method reduces the number of pods significantly over a random storage assignment. The authors do not validate the results with a simulation model or similar technique. Guan and Li [7] propose a zoning-based approach for positioning pods in an RMFS based on their content. The authors devise a MIP model and solve it in a three-stage algorithm design. The results suggest shorter order completion and travel time when compared to the reference data of the case study. The exact reference method is not described in detail. Roodbergen et al. [16] utilize a simulation based approach in order to optimize the warehouse layout of a manual order picking system for an industrial partner. The authors devise an integrated approach taking on certain design decisions as well as selecting control policies. The simulation is thereby used “as a solution tool and an evaluation system” (see [16]). Ribino et al. [15] devise a simulation model to emulate and study an AGV-based sortation system for in- and outbound activities. Using this methodology, the authors were able to recommend effective layouts, number of AGV and other aspects. Chen et al. [4] use a simulation based approach for evaluating the performance of policy sets for manual order picking systems. The authors make use of DEA as a tool for obtaining a comparable performance indicator among the policy sets. Beckschäfer et al. [1] use a discrete event simulation approach similar to this work for assessing storage policies for Automated Grid-based Storage systems. The authors find that even simple strategies improve the system efficiency, which encourages research on more complex strategies. Lamballais et al. [9] develop a Markow decision process (MDP) model for addressing the resource reallocation problem, i.e., the problem of deciding how many workers and robots to allocate to the pick process and replenishment process continually throughout time. The assumptions related to replenishment differ strongly across the papers mentioned above, and the number of approaches to replenishment in practical applications is diverse as well.

4. Decision problems

This section introduces the decision problems considered in this paper and places them within the context of other decision problems in an RMFS. Requests to the system occur via pick orders or replenishment orders. Upon receipt, pallets are broken up into smaller parts consisting of multiple units of one SKU. A replenishment order is a request to place one such part, i.e. a number of units of one specific product, on a pod.

We structure the decisions at the operational level in four steps: (1) Order Assignment (OA), the assignment of pick or replenishment orders to workstations, (2) Task Creation (TC), the creation of tasks for the robots, (3) Task Allocation (TA), the allocation of tasks to robots, and (4) Path Planning (PP), the creation of paths along which the robots will move. There are two kinds of Order Assignment decisions: the assignment of pick orders to pick stations, called the Pick Order Assignment (POA) problem, and the assignment of replenishment orders to replenishment stations, called the Replenishment Order Assignment (ROA) problem. In the second step, a task is defined as transporting a specific pod to a specific workstation and back to a specific storage location. Therefore, for each workstation, the Task Creation decision problem includes the two subproblems of (2.1) deciding which pod to select for transportation, the Pod Selection (PS) decision problem, and (2.2) deciding at which storage location to return the pod, the Pod Storage Assignment (PSA) decision problem. The Pod Selection (PS) decision problem differs for the pick and replenishment process, because for the pick process the due times of the pick orders is important in selecting a pod. Pod selection in the pick process is called Pick Pod Selection (PPS) and pod selection in the replenishment process is called Replenishment Pod Selection (RPS). The storage capacity of the pods is modeled in one-dimensional manner. Task Creation uses the pick order and replenishment order assignments to select suitable pods and subsequently converts the requests for the selected pods into tasks for pod transportation between the workstations and the storage area. Task Allocation creates a trip by building a sequence of tasks for the robots to execute. These sequenced tasks implicitly define trips and serve as input for the Path Planning algorithms, where a path is generated for a robot to follow.

Fig. 3 shows an overview of the decision problems at the strategic, tactical and operational level in an RMFS, with the problems addressed in this paper in bold. As can be seen in Fig. 3, this paper focuses on decision problems at the operational level. We use the term "decision rule" to refer to a fairly simple method to solve a decision problem. The aim of this paper is to evaluate several decision rules per decision problem. Some decision rules may closely resemble common best practices, whereas others may be more specific to RMFS. The Task Allocation decision problem is intertwined with the Path Planning decision problem, which has been addressed by Merschformann et al. [12]. Therefore we do not consider the Task Allocation and Path Planning decision problems. We do address Pick Order Assignment (POA), Replenishment Order Assignment (ROA), Pick Pod Selection (PPS), Replenishment Pod Selection (RPS), and Pod Storage Assignment (PSA). For Pick Order Assignment, we assume there is a constant backlog, and the pick stations are always filled to full capacity with pick orders. Whenever a pick order is fulfilled and leaves its pick station, a

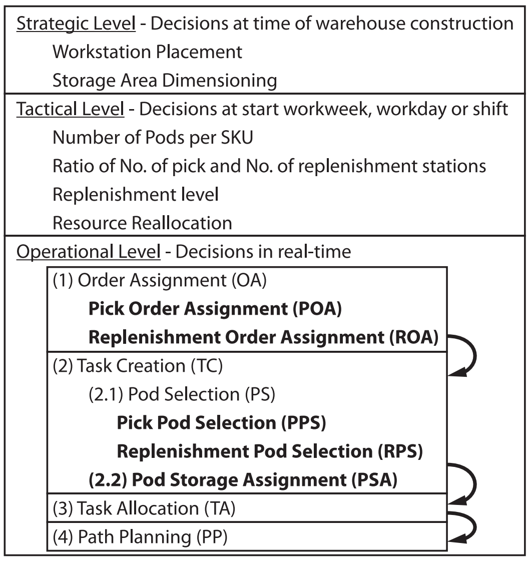

Strategic Level - Decisions at time of warehouse construction
Workstation Placement
Storage Area Dimensioning
Tactical Level - Decisions at start workweek, workday or shift
Number of Pods per SKU
Ratio of No. of pick and No. of replenishment stations
Replenishment level
Resource Reallocation
Operational Level - Decisions in real-time
(1) Order Assignment (OA)
Pick Order Assignment (POA)
Replenishment Order Assignment (ROA)
(2) Task Creation (TC)
(2.1) Pod Selection (PS)
Pick Pod Selection (PPS)
Replenishment Pod Selection (RPS)
(2.2) Pod Storage Assignment (PSA)
(3) Task Allocation (TA)
(4) Path Planning (PP)

Fig. 3. Hierarchical overview of the decision problems and their relations.

pick order has to be selected from the backlog and assigned to the pick station. For replenishment orders, we assume that the sequence of replenishment orders inbound to the system cannot be altered anymore. This assumption resembles the situation in conventional conveyor-based material handling components that do not allow sequence modification but only load routing. Moreover, we aim to avoid taking decision problems outside of the system's boundaries into account, e.g., different dispatching rules of preceding systems. The replenishment stations have a finite capacity, which is modeled as a one-dimensional storage capacity (like for the pods) in order to emulate buffer characteristics of the station. If a replenishment order arrives and multiple replenishment stations have capacity left, the ROA decision rule determines to which replenishment station the replenishment order is assigned. If no station is available, replenishment orders are put in a replenishment order backlog. When a replenishment order is fulfilled at one of the replenishment stations, a new replenishment order is chosen from the replenishment order backlog according to the FCFS rule. Table 1 summarizes the decision problems addressed in this paper.

At this point we also introduce the concept of "pile-on" (sometimes also called "hit-rate"). Pile-on as a concept refers to the average number of units that are picked from a pod every time a pod is presented to a picker at a pick station. Pile-on as a metric measures the number of units (across all SKUs) picked from a pod when presented to a picker at a pick station, averaged across every visit of a pod to a pick station during the entire time horizon. In other words, pile-on is measured in "units picked per pod visit to a pick station". The higher the pile-on is, the fewer pods need to be transported between the pick stations and the storage area, which may reduce the number of robots needed.

5. Decision rules

To solve the operational problems, we define several decision rules per decision problem that are evaluated in a realistic simulation. Several Path Planning algorithms for the RMFS are compared in [12], therefore this decision problem will not be addressed in this paper. Thus, we selected WHCA$^*_v$, one of the best performing algorithms from the paper, as the path planning engine for this work. Additionally, we fix the Task Allocation algorithm to a simple method that first assigns two-thirds of the robots to pick operations and the rest to replenishment operations. Then, it aims to equally distribute the robots across the respective stations. This means a robot will only do tasks related to the station it is assigned to. This section will therefore only describe decision rules for the Pick Order Assignment, Replenishment Order

Table 1
Decision problems.

Abb.	Name	Description	Trigger
POA	Pick Order Assignment	Choosing a pick order from the backlog	When another pick order is fulfilled, creating room for the next one to be assigned
ROA	Replenishment Order Assignment	Selecting the replenishment station for the next replenishment order	When a replenishment order arrives at the system and one or more replenishment stations have capacity left and after RPS assignment (latter depends on the active ROA rule)
PPS	Pick Pod Selection	Selecting a pod to transport to a pick station	When a robot working for a pick station needs a new task
RPS	Replenishment Pod Selection	Select a pod for the next replenishment order	When a replenishment order arrives at the system and a pod has sufficient capacity left and after ROA assignment (latter depends on the active RPS rule)
PSA	Pod Storage Assignment	Choosing a storage location for a pod	When a pod leaves a workstation

Assignment, Pick Pod Selection, Replenishment Pod Selection and Pod Storage Assignment decision problems.

While replenishment and pick operations are similar in the sense that high throughput should be achieved with few resources, the main asymmetry between both is that for the former the goal is to fill the inventory as quickly as possible and for the latter to empty it as quickly as possible. This means that for replenishment operations we aim to replenish pods fast to have them available for pick operations early while preparing pod content such that it allows for a high pile-on during pick operations. For pick operations we aim to achieve a high pile-on and keep trips short to fulfill as many orders as possible while also considering due times of the pick orders. Furthermore, we do not allow the sequence of replenishment orders to be modified. In contrast, for pick orders we allow to arbitrarily choose one order from the backlog. Lastly, pick orders have due times. All of this leads to different strategies we focus on per decision problem, instead of fully symmetric rules between pick and replenishment decision problems.

For a more precise description of some of the rules we introduce the notation shown in Table 2.

5.1. Pick order assignment rules

A pick station has to be chosen for every pick order submitted to the system and the pick order itself has to be chosen from the order backlog. In this work, we consider a pick order backlog of constant size, i.e., as soon as an order is removed from the backlog a new one is generated to replace it. This and the immediate replacement of orders completed at a station lead to only one option available to assign any pick order to: the slot of the just completed order. Hence, the choice of station is not a degree of freedom in this work. The rare occasions of multiple orders to be completed at the same time are handled by assigning the orders to the pick stations randomly. Hence, we only investigate rules for selecting the next pick order from the backlog to fill the only open slot at a station. We devise six rules to solve this problem:

Table 2
Overview of the symbols used in the rule descriptions.

Symbol	Explanation
$\mathcal{P}$	Set of all pods p
$\mathcal{W}^L$	Set of unused storage location waypoints w
$\mathcal{P}_s^I$	Set of pods heading to station s
$\mathcal{I}$	Set of all SKUs i
O^B	Set of pick orders o currently in backlog
O_s^S	Set of pick orders assigned to station s
$W(e)$	Current waypoint/location w of element e
$C(p, i)$	Number of units of SKU i contained in pod p
$L(o, i)$	Required units necessary to fulfill line i of order o
$D(o, i)$	Remaining units necessary to fulfill line i of order o
$T(w_1, w_2)$	Expected travel time from waypoints w_1 to w_2
$Class(e)$	The class of element e for class-based rules (e is storage location or pod)
t_o^D	Due time of order o
t_o^S	Time of assignment to the station of order o
t	Time of decision rule invocation (moment of decision)

"Random", "FCFS", "Due-Time", "Fast-Lane", "Common-Lines" and "Pod-Match":

Random The Random rule randomly selects a next pick order from the backlog and is used as a benchmark.

FCFS The FCFS rule assigns the pick order that was first received. The rationale behind this is to keep pick order throughput times short.

Due-Time The Due-Time rule selects the pick order with the earliest due time from the backlog and assigns it to a station. This is a greedy approach aiming to finish the pick orders before their deadline.

Fast-Lane The Fast-Lane rule randomly selects a pick order from the backlog like the Random rule, but keeps one slot at each pick station open for immediately completable pick orders. I.e., only pick orders (o), for whom all lines and all units of inventory are available on the next pod (p_n) will be assigned to this station's "fast-lane" order slot (see Eq. (1)). Thus, orders assigned to the "fast-lane" slot are processed shortly after assignment. The next pod of the station is either a not completely processed pod the picker is currently working on or the next pod in the station's queue, if no such pod is available. In cases where no pod reached the station's queue yet, we consider the pod with the shortest remaining path to estimate the next pod. When facing multiple options we use a random tie-breaker. Note that this rule can be combined with any other proposed POA rule. The reason we combine it with random selection is to better assess the impact of the idea itself.

$$\forall\ i \in \mathcal{I}: L(o, i) \leq C(p_n, i) \tag{1}$$

Common-Lines The Common-Lines rule compares the station's (s) currently assigned pick orders with all orders from the backlog and selects the one with most lines in common for assignment (see Eq. (2)). The rationale behind this is to increase pile-on by exploiting synergies among the pick orders. When facing multiple options we use a random tie-breaker.

$$\underset{o \in O^B}{\operatorname{argmax}} \sum_{o' \in O_s^S} \sum_{i \in \mathcal{I}} y_{oo'i} \quad \text{with } y_{oo'i} = \begin{cases} 1 & L(o, i) > 0 \wedge L(o', i) > 0 \\ 0 & otherwise \end{cases} \tag{2}$$

Pod-Match The Pod-Match rule selects the pick order from the backlog that matches best the pods heading to the station (s) at the moment of assignment best. I.e., the more units of the pick order are already available in the pods the better the match (see Eq. (3)). When facing multiple options we use a random tie-breaker.

$$\underset{o \in O^B}{\operatorname{argmax}} \sum_{p \in \mathcal{P}_s^I} \sum_{i \in \mathcal{I}} (\min(C(p, i), D(o, i))) \tag{3}$$

5.2. Replenishment order assignment rules

As a result of the assumptions that replenishment orders arrive in a fixed sequence, we investigate only two different approaches for assigning replenishment orders to the stations, i.e., immediate Random assignment and batching of customer orders that go on the same pod. Hence, we construct two rules for replenishment assignment: “Random” and “Pod-Batch”:

Random The Random rule randomly selects a next station with sufficient remaining capacity to allocate incoming replenishment orders to. If no such station is available, the order will wait until one becomes available again. This rule can operate independently of the chosen Replenishment Pod Selection rule.

Pod-Batch The Pod-Batch rule tries to use a pod already selected to go to a replenishment station for assigning the next replenishment order. In other words, the Pod-Batch rule first waits for the Replenishment Pod Selection (Section 5.4) rule to decide which orders are assigned to which pod, and then uses the same replenishment station for the orders of one pod. If the replenishment orders do not fit one station, they wait until a station with sufficient capacity becomes available. Note that, during this time all consecutive orders are also blocked, because the sequence cannot be altered.

5.3. Pick pod selection rules

Every time a robot working for a pick station s requests a next task, a pod suitable for picking at pick station s must be selected. We require for all rules that at least one unit can be picked from the pod. This means that no pod is brought to a station completely in vain and additionally it implies a pile-on of at least 1. The six PPS rules used in this paper are the “Random”, “Nearest”, “Pile-on”, “Demand”, “Lateness”, and “Age” rules:

Random The Random rule randomly selects a pod that offers at least one useful unit for picking.

Nearest The Nearest rule selects the pod which has the least estimated path time towards the station according to the path planning algorithm and that offers at least one useful unit for picking.

Pile-on The Pile-on rule selects the pod that offers most units necessary to fulfill the orders at the station (see Eq. (4)). Ties are broken by favoring pods with which more orders can be completed. If ties still persist, they are broken randomly.

$$\underset{p\in\mathcal{P}}{\operatorname{argmax}} \sum_{i\in I} \sum_{o\in O_s^S} (\min(C(p, i), D(o, i))) \tag{4}$$

Demand The Demand rule selects the pod whose content is most demanded considering the current pick order backlog situation, i.e. the pod with most units demanded in the backlog is chosen (see Eq. (5)). Ties are broken randomly.

$$\underset{p\in\mathcal{P}}{\operatorname{argmax}} \sum_{i\in I} \sum_{o\in O^B} \min(C(p, i), D(o, i)) \tag{5}$$

Lateness The Lateness rule aims to finish late pick orders by selecting a pod that offers units needed to fulfill open order lines with most lateness at the station, i.e., for one order the time the order is late is summed as fractions of the open picks (see Eq. (6)). If no order is late, the resulting ties are broken by using the same metric but replacing $\max(t - t_o^D, 0)$ with t_o^D, thus, selecting pods for orders whose due times are most imminent.

$$\underset{p\in\mathcal{P}}{\operatorname{argmax}} \sum_{i\in I} \sum_{o\in O_s^S} \left(\frac{\min(C(p, i), D(o, i))}{\sum_{i'\in I} D(o, i')} \max(t - t_o^D, 0) \right) \tag{6}$$

Age The Age rule aims to finish the oldest pick orders of a station by selecting a pod that offers units needed to fulfill the oldest open order lines, i.e. for one order the time the order spent assigned to the station is summed as fractions of the open picks (see Eq. (7))

$$\underset{p\in\mathcal{P}}{\operatorname{argmax}} \sum_{i\in I} \sum_{o\in O_s^S} \left(\frac{\min(C(p, i), D(o, i))}{\sum_{i'\in I} D(o, i')} (t - t_o^S) \right) \tag{7}$$

5.4. Replenishment pod selection rules

For every replenishment order, a suitable pod with sufficient remaining storage capacity needs to be chosen. The decision is taken right before the replenishment order is assigned to a replenishment station. Depending on the selected ROA and RPS rules both are either invoked simultaneously or, if there is a dependency between the two, one after the other. An example for the latter case is the combination of the Pod-Batch ROA rule with the Emptiest RPS rule, because the Pod-Batch rule relies on an already selected pod for the replenishment order. Since Replenishment Pod Selection determines the composition of the pods, it offers many possibilities to create pods with different features, e.g. high frequency pods that combine frequently ordered products, or family-based pods combining products that are often ordered together. If all replenishment orders assigned to the same pod are assigned to the same replenishment station, only one trip is necessary to place all replenishment orders on the pod, which reduces the number of robot movements.

The five RPS rules used in this paper are the “Random”, “Emptiest”, “Nearest”, “Least-Demand” and “Class” rules:

Random The Random rule selects a random pod with sufficient remaining capacity.

Emptiest The Emptiest rule assigns replenishment orders to the emptiest pod and reuses the same pod for subsequent replenishment orders until it is full or used at a station.

Nearest The Nearest rule assigns an incoming replenishment order to the nearest pod with sufficient remaining capacity. This rule needs to await the ROA assignment first to make a decision.

Least-Demand With the Least-Demand rule an incoming replenishment order is assigned to the pod currently offering the least demanded inventory, i.e. the pod with the least units offered when compared to the aggregated demand by assigned and backlogged pick orders is selected. Thus, this pod is not useful for pick-operations at the time of selection and by this it is not disadvantageous to block it for replenishment operations.

Class The Class rule assigns incoming replenishment orders to a pod of the same class as the replenishment order, i.e. fast moving SKUs to pods with other fast moving SKUs. The classes are built by a background mechanism for which the cumulative relative amount of pods per class are given. For this work we use “0.1, 0.3, 1.0”, i.e., three classes where the first class holds 10% of the pods for the highest frequency SKUs, the second class holds 20% and the last class holds the remaining ones, which are the ones with the lowest frequency SKUs. To assign a replenishment order of SKU s of a certain class, the emptiest pod is selected from the pods of that particular class (see Eq. (8)). Similar to the Emptiest rule, a selected pod is used for the subsequent incoming replenishment orders of the same class until no more replenishment orders fit the pod or until the respective pod completes its visit to a replenishment station.

$$\underset{\{p\in\mathcal{P} \mid Class(p)=Class(s)\}}{\operatorname{argmin}} \left(\sum_{i\in I} C(p, i) \right) \tag{8}$$

5.5. Pod storage assignment rules

For each pod an unoccupied storage location has to be selected, every time after visiting a pick or replenishment station. PSA is an important aspect of the RMFS, because being able to change the storage location of pods after every visit to a workstation is what makes continuous automatic sorting possible. For PSA, five decision rules are examined, namely the “Random”, “Fixed”, “Nearest”, “Station-Based” and “Class” rules.

Random The Random rule chooses a random free storage location.
Fixed The Fixed rule maintains the initially assigned storage location for all pods.
Nearest The Nearest rule stores pods at the nearest unoccupied storage location in terms of shortest estimated path time. This path time is determined using an A* algorithm that takes the time needed for turning the robot (with or without pod) into account.
Station-Based The Station-based rule is a variant on the Nearest rule, i.e. instead of bringing the pod to a storage location that is nearest to the robot’s position the storage location with shortest path time to a pick station is selected. The greatest difference with the Nearest rule is in the storage locations chosen for pods returning from a visit to a replenishment station.
Class The Class rule brings pods back to storage locations of the same class, where classes are constructed in a similar fashion as in the RPS decision problem, but based on the shortest path time to a pick station. Within a class, a storage location for a pod is selected analogously to the Nearest rule (see Eq. (9)).

$$\underset{\{w \in \mathcal{W}^L | Class(w) = Class(p)\}}{\arg\min} (T(W(p), w)) \tag{9}$$

Table 3 provides an overview of the decision rules per decision problem and shows how the decision rules are labeled across decision problems. Note that choosing a rule for one decision problem may jeopardize strategies chosen for others. For example, a random Pick Order Assignment may have a negative impact on a Class-based approach for assigning replenishment orders to storage locations, because it does not respect the units currently positioned near the pick station while assigning orders to it. Hence, a selection respecting mutual influences has to be done to provide an efficient compilation of rules that is able to adequately overcome the planning problems in such a system.

6. Evaluation framework

This section describes the evaluation framework used to carry out the research in this paper. Two central concepts to the evaluation framework are the Rule Configuration (RC) and the Warehouse Scenario (WS). The RC specifies for each decision problem, which decision rule is used. The WS specifies the warehouse layout, number of robots, number of workstations, number of SKUs, whether or not return orders are part of the operations of the warehouse, and pick order size. During one simulation run the RC and WS do not change, so they can be seen as an input to a simulation run.

Table 3
Overview of the decision rules per decision problem.

Decision problem	Decision rules
POA	Random, FCFS, Due-Time, Fast-Lane, Common-Lines, Pod-Match
ROA	Random, Pod-Batch
PPS	Random, Nearest, Pile-on, Demand, Lateness, Age
RPS	Random, Emptiest, Nearest, Least-Demand, Class
PSA	Random, Fixed, Nearest, Station-Based, Class

The evaluation framework consists of two phases, one varying the RCs, the other varying the WSs. Phase 1 evaluates all 1620 possible RCs on one WS. For phase 1, we compare eight performance measures: (1) Unit throughput rate, (2) pick order throughput rate, (3) order turnover time, (4) distance traveled per robot, (5) order offset, (6) fraction of orders that are late, (7) pile-on (8) the pick station idle time. Unit throughput rate is the number of picked units of all SKUs per hour. Pick order throughput rate is the number of pick orders fulfilled per hour. Order turnover time is the average time between submitting a pick order to the backlog and fulfilling it. Order offset is the average time between the due time and the completion time of the pick orders. Thus, a value smaller than zero shows how much in advance pick orders are completed. The rationale behind this is that follow-up processes at the distribution center are not deterministic, hence, pick orders completed earlier may improve the overall service level. The pick station idle time is measured as an average across all pick stations in the system.

Phase 1 selects the RCs with the highest unit throughput rate. However, among these selected best RCs, the variety in the decision rules may be low. For a particular decision problem, all of the selected RCs may use the same decision rule. To ensure more diversity in the RCs in phase 2, we define 6 so-called “benchmark RCs”, see Table 4. The benchmark RCs were chosen such, that all decision rules across all decision problems appear in at least one of the benchmark RCs. Each benchmark RC has been given a name that reflects a characteristic that the decision rules have most in common.

Phase 2 evaluates the selected RCs from phase 1 and the benchmark RCs, while varying the warehouse scenarios. Since we are specifically interested in efficiency of RCs we neglect layout decisions for this work. Thus, we choose one specific layout, using the style described in Section 2. The concrete layout instance comprises 1149 pods and 1352 storage locations (85% filled) and is shown in Fig. 4. When varying the number of pick stations during phase 2 we add workstations in the order given in Fig. 4.

6.1. Parameters

In the following we describe the used parameters in more detail. The parameters shared for both phases are outlined in Table 5. We set a continuous simulation horizon of 48 h in order to decrease the impact of side effects like recurring replenishment overflows, which cause replenishment pauses described previously. Within a duration of 48 h we observe sufficient repetitions of such patterns to achieve a reasonable mitigation of these side effects.

Furthermore, for each RC and WS combination in phase 1 and in phase 2 we conduct 10 runs to lessen the effect of randomness. To keep the system under continuous pressure, like described above, we keep a constant pick and replenishment order backlog of 200 orders each. At simulation start inventory is generated until 70% overall storage utilization to avoid cold starting the system. This is done using the same process used for generating replenishment orders during simulation and using assignment rules suiting the respective RPS rule in place. The storage capacity of a pod is set to 500 slots while the storage

Table 4
Benchmark RCs.

Benchmark RC	POA	ROA	PPS	RPS	PSA
Demand	Due-Time	Pod-Batch	Demand	Least-Demand	Fixed
Speed	Fast-Lane	Pod-Batch	Lateness	Emptiest	Nearest
Nearest	FCFS	Random	Nearest	Nearest	Nearest
Class	Common-Lines	Pod-Batch	Age	Class	Class
Greedy	Pod-Match	Pod-Batch	Pile-on	Emptiest	Station-Based
Random	Random	Random	Random	Random	Random

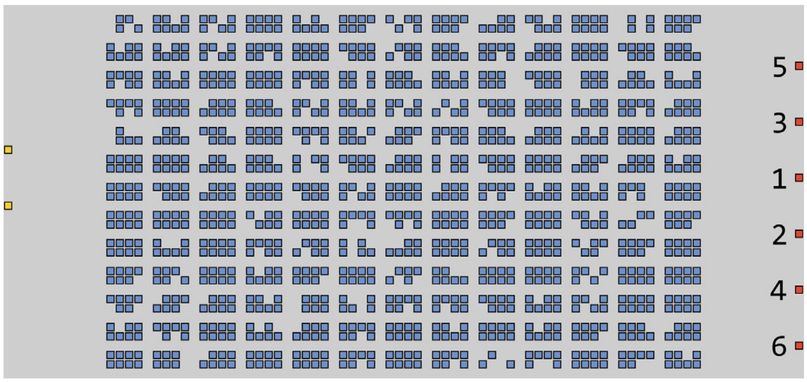


Fig. 4. Top view of the layout, including pick station indices, with the storage area in the middle, replenishment stations to the left, and pick stations to the right.

Table 5
Parameters shared across all simulations.

Parameter	Value
Simulation	
Simulated duration of warehouse operations	48 h
Number of simulation repetitions	10 repetitions
Size of pick order backlog	200 pick orders
Size of repl. order backlog	200 repl. orders
Layout	1149 pods, 1352 storage locations in 2 × 4 blocks, 12 aisles and 12 cross-aisles
Orders	
Number of units per repl. order	uniform distribution between 4 and 12
Amount of priority orders in pick orders	20%
Priority pick order due time	backlog submission time + 30 min.
Normal pick order due time	backlog submission time + 120 min.
Threshold when pick order generation starts	60% of inventory capacity
Threshold when pick order generation stops	10% of inventory capacity
Threshold when repl. order generation starts	65% of inventory capacity
Threshold when repl. order generation stops	85% of inventory capacity
Inventory	
Initial inventory in the storage area	70% of inventory capacity
Space on a pod	500 slots
SKU frequency / popularity	Exponential distribution, $\lambda = \frac{1}{2}$
SKU size (in slots)	uniform distribution between 2 and 8
Robot movement	
Robot acceleration/deceleration	$0.5\frac{m}{s^2}$
Robot maximum velocity	$1.5\frac{m}{s}$
Time needed for a full turn of a robot	2.5 s
Time needed for lifting and storing a pod	3 s
Time needed for picking a unit	8s
Time needed for handling a unit at a pick station	15 s
Time needed for putting a repl. order on a pod	20 s
Stations	
Repl. station capacity	two times pod capacity
Pick station capacity	8 pick orders

consumption of one SKU unit is drawn from a uniform distribution between 2 and 8 slots, thus, a full pod contains 100 units in average. The popularity of the SKUs is determined by drawing a value from an exponential distribution with parameter $\lambda = \frac{1}{2}$ for each SKU to emulate a typical ABC curve in e-commerce. This popularity is the relative frequency parameter between all SKUs, thus, the frequency (if divided by the sum of all frequencies) is the probability of choosing a particular SKU when generating an order line for both replenishment and pick orders. One replenishment order restocks between 4 and 12 units of one SKU following a uniform distribution. To emulate due times we distinguish between priority and normal orders that have to be completed in 30 minutes respectively 120 min. This reflects the need for preferring important orders.

The movement behavior of the robots is emulated by using a maximum velocity of $1.5\frac{m}{s}$ with acceleration and deceleration rates of $0.5\frac{m}{s^2}$. We set the rotational speed to $\frac{4}{5}\pi\frac{rad}{s}$, i.e., 2.5 s for a full turn. Turning takes the same amount of time regardless of whether a robot is carrying a pod. The time for lifting and setting down a pod is set to 3 s. This should reflect the capabilities of mobile robots used in similar industry applications reasonably close. These values are based on observations of similar systems in operation and discussions with a supplier of such systems. For the actual pick operation of one unit at a pick station we assume a constant time of 8 s. The complete time for handling one unit including additional operations, like putting the product unit in the correct pick order tote, is set to 15 s. This distinction is considered to allow for an early release of the robot, such that no unnecessary robot waiting times are caused. This is not distinguished for replenishment operations, since we assume that a robot can only leave after fully completing the put operation to the pod. The time of a put operation of one replenishment order is set to 20s.

The parameters in Table 5 are shared across all conducted experiments, while the parameters in Table 6 depend on phase and scenario. For the first phase we assess all possible RCs for one fixed warehouse scenario. Note that the RPS rule Nearest and the ROA rule Pod-Batch rely on each others assignments for taking their decisions (since they are using them as inputs), which leads to no decision at all. Hence, the combination of these rules is forbidden. For the fixed warehouse scenario we set the number of robots to 4 per pick station, i.e. 8 robots in

Table 6
Varied parameters for the phases (distributions as: (mean, std.dev., min, max)).

Parameter	Phase 1 values	Phase 2 values
Rule configurations (RCs)	1620 RCs	6 Benchmark RCs + 4 best RCs from phase 1
# pick stations	2	1, 2, 3, 4, 5, 6
Robots per pick station	4	2, 3, 4, 5, 6
# SKUs	1000	1000, 10,000
Return orders	0%	0%, 30%
Pick order size	*Mixed* - line & unit dist.: (1,1,1,4) & (1,0.3,1,3)	*Small* - line & unit dist.: (–,–,1, 1) & (–,–,1, 1) *Mixed* - line & unit dist.: (1,1,1,4) & (1,0.3,1,3) *Large* - line & unit dist.: (1,1,2,4) & (1,0.3,2,3)
# RC	1620	10
# WS	1	360
# RC × WS	1620	3600
# simulation runs	16,200	36000

the system at whole. Furthermore, we set the number of pick stations to 2, the number of SKUs to 1000 and exclude the processing of return orders. The order setting is set to Mixed. This means the number of lines per pick order and the number of units per order line are generated following truncated normal distributions with parameters shown in Table 6. This is done to resemble e-commerce pick order characteristics of generally small orders with occasional larger ones in between. The full set of RCs analyzed in phase 1 is given by the full set of allowed combinations of all given decision rules. This results in 1620 RCs, and since phase 1 has 1 WS and 10 runs are conducted per RC and WS combination, this results in 16,200 simulation runs for phase 1.

For phase 2 we limit the RCs to the 6 benchmark RCs and the 4 best ones from phase 1, i.e., the 4 RCs with highest throughput rate. Moreover, we vary the number of pick stations from 1 through 6 and the number of robots per pick station from 2 through 6. This leads to a range from 2 robots in the system to 36 robots across all WSs. In addition to WSs with 1000 SKU, we also assess WSs with 10,000 SKUs stored in the system. For the order size we define two additional settings of small and large orders. For the Small pick order size, only single line / single unit pick orders are generated. For the Large pick order size, the distributions from the Mixed order setting are used but the min parameter for both is set to 2. Lastly, in WSs where we emulate the processing of return orders, 30% of the generated replenishment orders are single unit. The total number of RC and WS combinations for the phase 2 is therefore 3600 (10 RCs, 360 WSs), which leads to 36,000 simulation runs.

7. Computational results

This section shows the results from phase 1 and phase 2 of the evaluation framework. Throughout this section, the unit throughput rate is presented as a percentage of the upper bound on the unit throughput rate. The unit throughput rate is presented in this way to facilitate interpretation and comparison of results across experiments. Moreover, the RMFS is supposed to have high pick rates as it eliminates the need of walking for the workers, while the robots are supposed to supply the pickers with a constant stream of pods to pick from. Presenting the unit throughput rate as a percentage shows clearly to what extent these aims are achieved. The upper bound is discussed in more detail in the appendix. The length of the confidence intervals is always less than 1% of the mean, based on 10 runs per RC and WS combination, and therefore does not add much information. Based on 10 runs per RC and WS combination, we observe only small standard deviation in unit throughput rate. For the first phase, it is less than 1.32% for all combinations and less than 0.61% for 95% of them. In the second phase, it is less than 0.57% for all combinations and less than 0.17% for 95% of them. Therefore, we consider the results sufficiently stable for the experiments conducted in this work. The repetition count and simulation horizon are large enough to mitigate random effects.

7.1. Phase 1

The first phase aims to investigate throughput performance and the impact per decision problem of decision rules on throughput. Furthermore, we assess the behavior of the different output measures depending on decision rule selection. For this, Table 7 shows how across these simulations the eight previously introduced performance measures correlate with each other. At first, we can observe that as the unit throughput rate score improves, the other performance measures improve as well. As the unit throughput rate score increases, pick order throughput rate and pile-on increase as well, whereas the order turnover time, the distance that robots travel, the order offset, the fraction of orders that miss their due time, and the station idle time decreases. Although it is not clear what the exact causal relationships are, the correlations suggest that pile-on and the distance traveled by the robots are the main drivers behind these improvements. With higher pile-on, more units are picked per pod, so order lines are fulfilled more quickly and fewer trips are needed to fulfill the pick orders. This also causes longer processing times for each pod at the pick station, which in turn increases the time for the next robot to queue and become ready at the station. In other words: a more continuous input of inventory at the pick station is achieved. Additionally, fewer trips for the pick process free up robots to do more replenishment tasks. With less distance traveled by the robots we expect pods to be presented at the pick stations more continuously. Similar to the pile-on this effect enables more continuous picking, which in turn increases the overall unit throughput rate. Both measures, pile-on and the traveled distance, are intermediate measures affected by the choice of strategy for the different decision problems, i.e., a better score in both decrease the idle time at the stations, which in turn increase the throughput. An increased throughput, in the constant pick order backlog setting of this work, also decreases the turnover time of pick orders and the due time offset. Only the number of orders being late is not strongly correlated with the two main throughput drivers. The two main throughput drivers can also be observed when looking at a scatter plot of all simulation runs of the first phase (see Fig. 5). Here we can see the best results in unit throughput rate score are achieved with a high pile-on and less distance traveled per robot. The group of simulation runs with least distance traveled per bot and a pile-on around 4 are RCs involving the Nearest PPS rule, while the simulation runs with highest pile-on (greater 5) at the top of the plot are RCs involving the Demand PPS rule. In both groups we find runs with the highest unit throughput rate score, hence, a higher throughput is not only achieved by a high pile-on. In particular within the top ten RCs in terms of unit throughput rate score the pile-on ranges between 3.84 and 6.36, while the distance traveled per bot ranges between 68.04 km to 80.36 km. Hence, pile-on and the traveled distance enable higher throughput, but may also compensate for each other. This is particularly interesting, because both come at operational costs. For traveled distance this is energy consumption and robot wear, while for pile-on it may be costs arising from potentially more complex replenishment processes. Furthermore, within both groups better results are obtained with RCs also involving the Pod-Match POA rule, which causes an additional boost in pile-on.

In Fig. 5 we also observe a ’cutoff’ of simulation runs in the upper right and bottom left areas. This can be explained by the longer handling time at the station resulting from a higher pile-on. I.e., the longer a robot needs to wait at a station for the picking to finish the less it can travel in the meantime. Thus, rules increasing pile-on may help reducing the necessary travel distance, and by this also robot wear and energy consumption.

The pick order throughput rate is neglected completely in the remainder of this work, because it almost completely aligns with the unit throughput rate score. The reason for this is the constant backlog of 200 pick orders over 48 h: With a pick order throughput rate of 241.963 completed orders per hour in average, omitting certain pick orders is almost impossible. Hence, we cannot observe a potential temporary throughput gain by preferring smaller or larger orders. In order to investigate the trade-off between picking many units and completing more pick orders an experiment with a fixed set of backlogged pick orders over a fixed period of time should be devised. For this, the possibly tedious processing of leftover pick orders, which are presumably harder to pick quickly, needs to be investigated. We leave this work for future research.

Table 8 shows for each decision problem the unit throughput rate score for each of the decision rules, averaged across all simulations in phase 1. We calculate the multiplier by dividing the highest unit throughput rate by the lowest. As the multiplier in unit throughput rates is rather large for the POA decision problem, system integrators and RMFS suppliers may benefit from carefully selecting a POA decision rule and from investigating better decision rules for this decision problem. The multiplier for the Replenishment Order Assignment is near 1, indicating that using a different decision rule does not offer much

Table 7
Correlations between the different performance measures for first phase.

	Unit throughput	Order throughput	Order turnover time	Distance traveled	Order offset	Late orders	Pile-on	Station idle time	μ	σ
Unit throughput	-	-	-	-	-	-	-	-	0.556	0.189
Order throughput	1.000	-	-	-	-	-	-	-	241.963	82.234
Order turnover time	-0.950	-0.950	-	-	-	-	-	-	3549.625	1220.445
Distance traveled	-0.952	-0.952	0.880	-	-	-	-	-	122598.768	19433.860
Order offset	-0.950	-0.950	1.000	0.880	-	-	-	-	-2565.458	1224.985
Late orders	-0.590	-0.591	0.685	0.549	0.684	-	-	-	0.187	0.115
Pile-on	0.899	0.899	-0.802	-0.796	-0.802	-0.448	-	-	2.438	1.450
Station idle time	-1.000	-1.000	0.950	0.952	0.950	0.591	-0.899	-	0.450	0.186

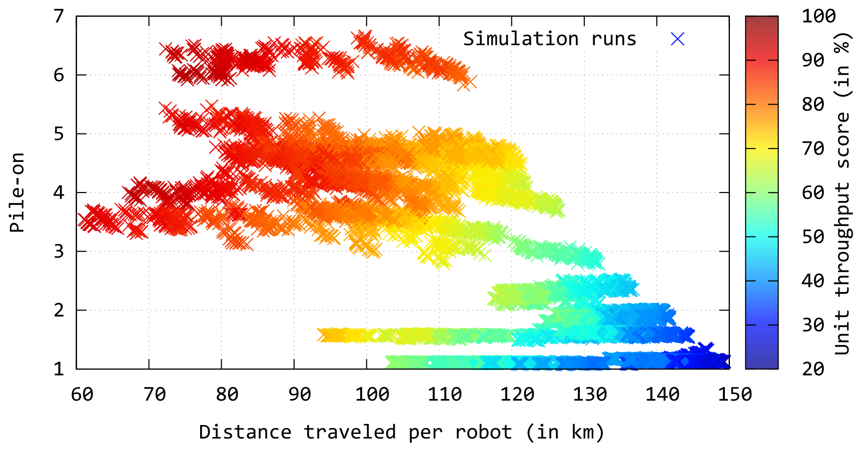


Fig. 5. Scatter plot for pile-on vs. traveled distance per robot colored by the achieved throughput rate score for all simulation runs of the first phase.

performance improvements. However, we note that we keep the sequence of incoming replenishment orders fixed at all times in this work, which limits improvement potential. Nevertheless, we expect limited degrees-of-freedom in replenishment operations to be more realistic, because the sequence will typically be a result of preceding operations or systems. Moreover, the limited number of replenishment stations diminishes the impact of ROA decision rules even more. Furthermore, the impact of the Pod Storage Assignment selection rule seems to be fairly low. This may be a reason of the quite small layout. We expect the impact of PSA decision rules to increase with the size of the instance layout, because the effect on the traveled distance would grow by a large amount.

In the following we analyze the achieved throughput performance per decision rule. For this, Fig. 6 shows the box-plots of unit throughput rate scores for each decision rule colored per decision problem. The boundaries of the boxes are determined by the upper and lower quartile while the line in the middle indicates the median value. The whiskers extend from the boxes to the minimum and maximum values. The first observation is that throughput performance of the RMFS is most sensitive to the choice of POA decision rule among the defined decision rules. This aligns with the previously observed correlations, because the choice of POA immediately affects the pile-on, which is identified as a major performance driver. The best performing POA strategies are Fast-Lane and Pod-Match, which both look at the incoming pods at a pick station when assigning new pick orders from the backlog. This suggests that a strategy aligning pick orders with the content of incoming pods seems most promising for throughput efficiency. This backs up the findings of Boysen et al. [2]. Although the Common-Lines rule exploits a similar greedy strategy, it achieves substantially less throughput. Hence, only matching pick orders to each other but not to the content of

Table 8
Average unit throughput rates as percentages of the upper bound for all rules, together with the **best** / **worst** performance multiplier per decision problem.

							Mult. $\left(\frac{\text{best}}{\text{worst}}\right)$
POA	Common-Lines	Due-Time	Fast-Lane	FCFS	Pod-Match	Random	
	50.93%	41.93%	76.13%	41.81%	**81.18%**	**41.71%**	1.946
ROA	Random	Pod-Batch					
	53.71%	**57.99%**					1.080
PPS	Age	Demand	Lateness	Nearest	Pile-on	Random	
	61.50%	52.70%	**48.63%**	**62.16%**	59.82%	48.88%	1.278
RPS	Class	Nearest	Emptiest	Least-Demand	Random		
	56.16%	58.42%	**59.63%**	57.71%	**47.56%**		1.254
PSA	Class	Fixed	Nearest	Random	Station-Based		
	55.91%	54.08%	**58.79%**	**53.60%**	55.70%		1.097

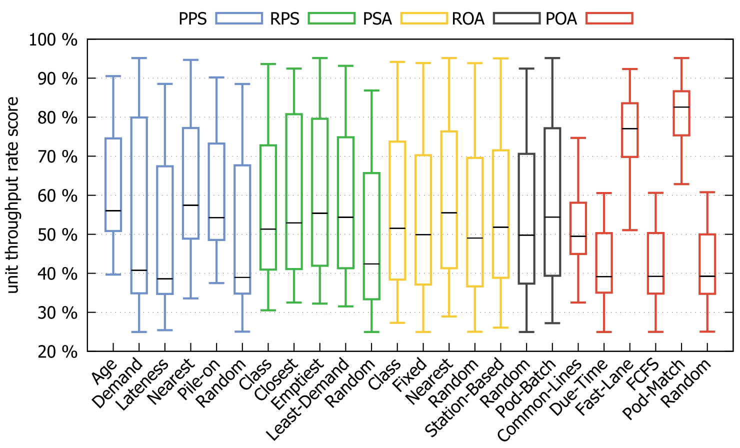


Fig. 6. Unit throughput rate performance of all runs involving the given rule.

the pods squanders throughput capabilities of the system. All other POA decision rules achieve similar throughput performance, since they do not consider order characteristics that would affect pile-on or traveled distance.

When looking at the PPS rule box-plots the average best throughput performance with least variance is achieved by the Age, Nearest and Pile-on rules. All of them focus either on maximizing the pile-on or minimizing the traveled distance. Although the Age rule does only indirectly maximize pile-on, it achieves a higher average pile-on of 2.92 among all RCs containing it than the actual Pile-on rule, which achieves an average pile-on of 2.79. The Demand rule has the highest spread across PPS rules with a very low median, but also provides some top performing RCs (see Table 9). This suggests that the throughput performance of the rule has a higher dependency on the selection of other rules.

Although the variation among the ROA decision rules is small, we observe a slightly better throughput performance by the Pod-Batch rule. This is a reason of the smaller number of trips necessary when batching replenishment orders.

Many of the top performing RCs contain the Emptiest or Nearest RPS decision rule. The main reason for the good throughput performance again seems to rely on fewer and shorter trips. The Emptiest rule decreases the number of trips, because more replenishment orders are stored in pods at once until it is full. E.g., only 31.03% pods need to be brought to replenishment stations in average when compared to the Random rule. The Nearest rule benefits from a similar effect since the same (nearest) pod is used for further replenishment orders even while it is already approaching. Furthermore, Nearest decreases the distance per replenishment trip, because nearer pods are used. The Random rule performs worst for RPS. The main reason for this is that too many trips are caused by randomly selecting pods while only storing few replenishment orders per trip.

Among the PSA decision rules we observe the best throughput performance for the Nearest strategy. This is again mainly caused by the shorter trips for the robots. When comparing the Nearest and the Station-based rule we see the benefit from shorter trips for replenishment operations increasing throughput of pick operations. However, this depends on the queue length at stations and the distribution of robots between replenishment and picking. I.e., if longer queue times are expected at replenishment stations than in our devised scenarios, moving pods nearer to the pick stations when returning them to the inventory may improve overall throughput performance. The Fixed and Random decision rules differ little in their performance. The main reason for this is that the storage location per pod in the Fixed rule is randomly selected. Thus, leading to a very similar behavior.

Due to the large sample sizes, the results of ANOVA and Tukey's range tests rejected the hypotheses that the means were equals at the 0.05 significance level within groups and pair-wise, with five exceptions. The null hypothesis of equal means was not rejected at the 0.05 significance level for POA rules FCFS and Due-Time, for Random and Due-Time, and for Random and FCFS. Furthermore, for PPS rules Random and Lateness the hypothesis of equal means could not be rejected, and for PSA rules Station-Based and Class.

Table 9
RCs with best throughput score selected from first phase (performance is unit throughput rate score).

RC rank	POA	ROA	PPS	RPS	PSA	performance
1	Pod-Match	Pod-Batch	Demand	Emptiest	Nearest	94.81%
2	Pod-Match	Pod-Batch	Demand	Emptiest	Station-Based	94.63%
3	Pod-Match	Pod-Batch	Nearest	Emptiest	Nearest	94.43%
4	Pod-Match	Pod-Batch	Demand	Emptiest	Class	94.00%

7.2. Phase 2

From the 1620 RCs in phase 1, the four with the highest unit throughput rate (see Table 9) together with the benchmark RCs form the set of ten RCs used in phase 2. The main purpose of phase 2 is to examine how well the RCs perform under different circumstances. In the following we analyze the results obtained for the 12 warehouse scenarios and 30 resource settings described before (see Section 6.1).

Table 10 shows the results, with the entries being the unit throughput rate as a percentage of the upper bound. In each cell the result of the best performing RC for the respective scenario and station / robot configuration is shown. The unit throughput rate scales well when adding more pick stations, the scaling is (almost) completely independent of the scenario characteristics. However, the necessary number of robots to achieve a given unit throughput rate greatly depends on the scenario characteristics, e.g., for more SKUs more robots are necessary to achieve a high unit throughput rate. The number of SKUs, does have a major impact on performance overall, where the main reason is that pile-on is considerably lower for the 10,000 SKU scenarios. A reason for this is the lower likeliness to have a pod with a good combination of SKUs matching the orders of the pick stations available. Thereby, if larger orders have to be processed with the system, this helps mitigating the negative effect of handling lots of SKUs. The main reason for this are the larger number of order lines active at a station when picking larger orders. I.e., more open order lines increase the likeliness of having a well matching pod available for the inventory required at a pick station. Processing return orders has an

Table 10
Best unit throughput rate score for all scenarios, robots per pick station and numbers of pick stations. Scenario abbreviations: [SKU count: 1000 (1K), 10,000 (10K)]-[Order size: Small (S), Medium (M), Large (L)]-[Return orders: yes (R), no (N)].

Stations	1					2					3					4					5					6				
Robots	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6
1K-S-N	44	82	91	97	97	59	89	94	97	98	64	90	95	97	98	60	87	93	97	98	57	87	93	97	98	59	87	94	97	98
1K-S-R	46	82	92	97	97	59	89	94	97	98	63	90	95	97	98	61	88	93	97	98	56	87	93	97	98	55	86	93	97	98
1K-M-N	45	83	92	97	98	60	90	95	98	98	64	90	95	98	98	60	88	93	98	98	57	88	94	98	98	59	88	94	98	98
1K-M-R	45	82	92	97	98	59	89	95	98	98	63	91	96	98	98	62	89	93	98	98	56	88	94	98	98	56	87	94	98	98
1K-L-N	54	83	93	99	99	66	90	97	99	99	68	91	96	99	99	67	88	94	99	99	63	86	94	98	99	63	87	94	99	99
1K-L-R	50	80	92	99	99	64	88	96	99	99	65	90	97	99	99	67	88	94	99	99	62	86	93	98	99	62	84	94	98	99
10K-S-N	21	39	55	68	78	27	44	59	72	81	28	46	61	73	80	29	47	59	69	77	30	47	57	68	77	31	45	58	68	76
10K-S-R	20	40	56	70	80	27	45	61	74	82	29	47	62	74	82	30	48	61	70	78	31	48	58	67	76	31	46	57	66	74
10K-M-N	23	41	58	71	81	28	46	61	75	84	29	48	64	76	83	30	49	61	71	80	31	48	60	71	80	32	47	60	71	79
10K-M-R	21	41	59	73	83	28	48	63	76	85	30	49	65	77	84	31	50	63	72	81	32	49	60	70	79	32	47	59	69	77
10K-L-N	36	63	84	94	98	44	71	89	96	99	46	73	87	94	98	47	69	83	92	97	46	67	84	91	97	45	68	83	91	97
10K-L-R	30	52	76	89	96	37	62	81	92	97	40	64	83	91	96	41	64	76	87	94	42	61	74	84	93	41	59	73	84	92

increased negative effect, if the order size of customer orders is large. However, in general, whether return orders are processed has a lesser effect on throughput performance than the other warehouse scenario variations. The reason behind this may be that even though approximately 19.76% more time is spent on replenishment operations by the robots when compared to the scenarios without return order processing, replenishment operations are overall quick enough to mitigate the effect. Replenishment operations only consume 20.29% out of the overall time consumed by the robots in average across all phase 2 simulation runs. Furthermore, we can conclude that with 1000 SKUs, the unit throughput rates are close to their theoretical maximum even with relatively few robots per stations.

Table 11 shows the unit throughput rate score for the RCs for all combinations of number of robots (n_r) and number of stations (n_s), averaged across WSs and presented as whole percentages. From Table 11 we can see that the Ranked RCs from phase 1 perform similarly and better than the benchmark RCs. Among the benchmark RCs, the Greedy benchmark outperforms the others consistently across all settings and is the only one whose unit throughput rate scores approached those of the ranked RCs.

7.3. Managerial insights

In the following, we briefly outline high-level findings for practitioners. First, in our experiments RMFS demonstrates excellent scalability characteristics. The throughput scales almost linearly with the number of stations for all studied scenarios. The number of robots shows similar behavior while few robots are used, but reaches a saturation point when approaching the maximal throughput of the stations. In connection with the prior, it should be noted that the number of robots necessary is highly dependent on the warehouse scenario a system is facing (e.g., number of SKUs, customer order characteristics, etc.). As a rule-of-thumb, anything decreasing the handling time per pod presentation (e.g., pile-on) will increase the number of robots required to get high throughput performance of the stations.

Next, we want to emphasize the importance of smart decision logic and algorithms for attaining high throughput with few resources (i.e., pickers, robots). In our experiments we observe a substantial performance dispersion exclusively caused by the selection of decision rules. Thus, there should be a strong focus on decision logic implementation when designing an RMFS. In particular, decision rules specifically tailored to RMFS should be considered (e.g. Pod-Match). In other words, the better the control logic the less equipment is needed, which in turn reduces the total-cost of ownership of the system (mobile robots, maintenance, support, etc.).

Finally, the effect of compensating inefficient decision logic with more robots will be exhausted when congestion and other blocking effects increase. Specifically, in scenarios with many SKUs and small orders (e.g.: e-commerce) this saturation point will be reached earlier.

8. Conclusion

In this work we studied the throughput performance of decision rules for multiple decision problems occurring in the control of RMFS. By analyzing a total of eight output measures for a total of 1620 RCs, we

Table 11
Unit throughput rate scores for the RCs in phase 2 (green ≡ best, red ≡ worst).

Stations	1					2					3					4					5					6				
Robots	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6
RC #1	31	61	77	87	92	43	68	81	89	93	45	70	82	89	93	46	69	79	87	91	44	68	78	86	91	44	67	78	86	90
RC #2	29	59	76	86	91	41	67	80	89	93	44	69	81	88	92	44	68	78	86	91	42	66	77	85	90	42	65	76	85	89
RC #3	33	61	76	86	91	44	69	81	88	93	47	70	82	88	92	47	69	79	86	91	45	68	78	85	90	46	67	78	85	90
RC #4	29	58	75	85	90	40	65	79	87	91	42	67	79	87	91	43	66	77	85	90	41	65	75	83	89	41	64	75	83	88
Demand	14	25	37	48	58	18	29	41	52	62	20	31	42	53	62	20	32	42	52	60	21	32	41	50	58	21	31	40	49	56
Speed	24	40	57	70	80	29	48	63	75	83	32	51	65	76	83	34	52	64	74	81	35	52	63	72	80	35	51	63	72	79
Nearest	19	34	49	62	73	26	41	56	68	78	28	43	57	69	78	29	44	57	67	75	30	44	55	64	72	30	43	54	63	70
Class	26	43	56	67	75	32	47	59	69	77	33	48	61	70	76	34	50	60	68	75	35	49	59	67	73	35	48	58	66	72
Greedy	34	57	72	82	88	43	64	77	85	89	44	66	77	85	89	45	65	75	83	88	44	63	74	81	87	43	62	73	81	86
Random	12	23	23	34	45	12	23	29	37	47	15	23	32	41	51	15	24	34	44	53	15	25	35	45	54	16	26	36	45	53

found strong correlations between these. Most interestingly a high pile-on and a short distance traveled by the robots together almost immediately account for the success of a decision rule applied to RMFS. Hence, we propose using these two output measures as the key tactics when designing decision strategies for RMFS that aim to achieve high throughput. In the investigated high pressure situation further performance measures like the turnover time of pick orders were also highly correlated with the unit throughput rate, which is why we focused on the throughput itself as the main metric for a successful RMFS.

Furthermore, we found that varying the decision rule used for solving the Pick Order Assignment affected the unit throughput rate the most. The average unit throughput rate was twice as high for the best decision rule as it was for the worst. This finding indicates that system engineers and warehouse operators should pay most attention to the Pick Order Assignment decision problem. Moreover, the unit throughput rate score ranges from 25.24% for the worst RC assessed in phase 1 to 94.81% for the best scoring RC. Hence, the right combination of decision rules plays a crucial role when controlling an RMFS. We propose that future research may assess how to scale beyond the throughput performance of the merely simple decision rules investigated in this work. However, we observe some cross-dependencies between different strategies for the core decision problems featured in this paper, e.g., the Demand PPS rule is part of the best performing and the worst performing RC. Thus, an integrated and realistic evaluation or validation of new decision methods for RMFS is highly important, since dependencies exist and side-effects should not be neglected. Additionally, we found that the number of different SKUs in the system has a strong impact on the unit throughput rate. This finding is probably due to a decrease in pile-on for a higher number of SKUs. This effect is considerably less for larger orders, presumably because for larger pick orders pile-on tends to be higher. Having to process return orders seems to affect the unit throughput rate more, if the pick orders are large. Moreover, we found that the performance of the “greedy” benchmark consistently came close to the best ranked configurations of decision rules.

This paper has studied solutions to several operational problems, which lead towards promising directions for future research. Each decision rule in this study has looked at an operational problem in isolation, but heuristics that try to integrate multiple operational problems and optimize these problems jointly may achieve substantial increases in order throughput or reductions in resources used. Investigating rules and heuristics that increase pile-on, i.e. the number of picks per handled pod, would also be of great use to practitioners.

While many decision rules and parameters were varied to deliver insightful results we expect even more insight when varying the layout itself. For example, we expect a larger impact of the PSA rule selection when facing huge layout instances. This was not done in this work in order to keep a certain focus and to keep computational resource utilization for the conducted experiments tractable.

Another aspect to focus future research on is the order process design around the RMFS. In this work, we assumed a constant backlog of orders to compare throughput performance in continuous operation. While it is noteworthy that simply varying the size of the order backlog may already affect the performance, another typical approach is to evaluate the time it takes to process a batch of orders, to handle order waves or to face hybrids of the prior. Since this work has shown the high impact of related processes like POA, we see high potential for performance improvements in this area. We would like to see future research in these directions.

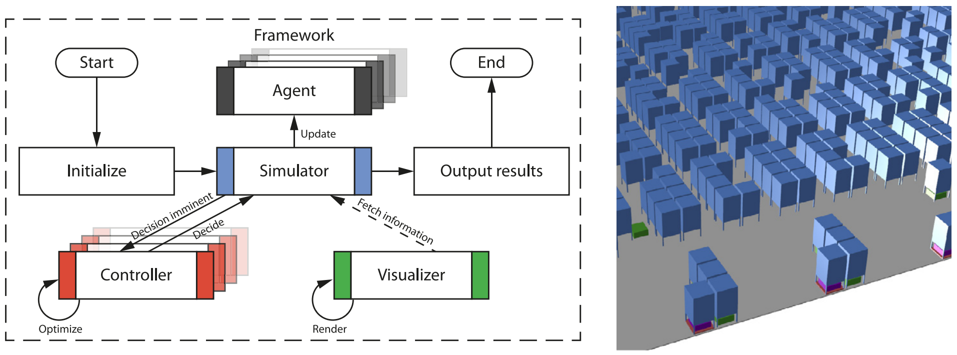
RMFSs are a new category of automated systems and concepts specific to RMFSs have not received much scholarly attention. An example would be cache zoning / priority zoning, that is the implementation of special zones near the workstations where pods are stored that will be needed in the near future. Another example would be a study of automatic sorting of the system without explicit zones. Since pods can be relocated to another storage location each time they are transported to and from a workstation, the inventory can be sorted automatically to some degree during operations. It is not clear at which speed automatic sorting takes place or how much performance benefits from it. Automatic sorting is a unique feature of RMFSs, but as with so many other aspects of RMFSs, it remains to be explored.

Acknowledgements

We would like to thank the Paderborn Center for Parallel Computing (PC^2) for the use of their HPC systems for conducting the experiments. Marius Merschformann is funded by the International Graduate School - Dynamic Intelligent Systems, University of Paderborn.

Appendix A. Simulation Framework

In this work we use the simulation framework “RAWSim-O”. A more detailed description of it can be found in [13] while the source code is available at https://github.com/merschformann/RAWSim-O. “RAWSim-O” is an agent-based and event-driven simulation focusing at a detailed view of an RMFS. The basic simulation process is managed by the core simulator instance (see Fig. A.7a), which is responsible for obtaining the next event and updating the agents. Agents can either represent real entities like robots and stations or virtual entities like process managers, e.g. for emulating order processes. Every decision that has to be made is passed to the corresponding controller. The controller can either immediately decide or can buffer multiple requests in order to optimize and release the decision later on. However, in this work we only consider ad hoc decision rules



(a) Overview of the simulation process. (b) Visualization screenshot

Fig. A1. RAWSim-O simulation framework.

with the former approach. To allow visual feedback, the ongoing simulation can optionally be rendered in 2D and 3D. The implementation was done in C#.

The level of detail of the simulation is especially high for the simulated movement behavior of the robots. We consider the robot's momentum by emulating acceleration and deceleration behavior, collision avoidance and turning speed (see Table 5). The emulation employs a continuous time-horizon. The times for activities other than robot movement, e.g. lifting or storing a pod, or picking one unit at a pick station, are constant (see Table 5). Robots that do not carry a pod can traverse underneath stored pods by using the waypoints at which the pods are stored. Furthermore, in the buffers of the workstations, robots can take short-cuts if the buffer is (partially) empty.

Information about the system's state is tracked in a high level of detail, because some decision rules differ with regard to the information they require. For example, all pods and all units on all pods are tracked exactly. All of the decision rules proposed in this work differ in their computational complexity and therefore also in the computational time they require to reach a decision. They are, however, simple enough to be considered as ad hoc decisions even for large system sizes.

Some information is not completely known beforehand, but becomes available over time. This is the case for incoming pick orders and replenishment orders submitted to the system over time by external processes. While each replenishment order consists of a number of physical units of one SKU, each pick order consists of a set of order lines, each for one SKU, with corresponding units necessary to fulfill the line. We assume for both pick and replenishment orders, that there is a constant order backlog. A constant order backlog means that when an order from the backlog is assigned to a workstation, it is immediately replaced by a newly generated order. By keeping the order backlogs constant, we aim to analyze the system's behavior under constant pressure. However, it also leads to the phenomenon that the system's storage space utilization (utilized space divided by total space available) in the storage area is affected by the performance of the decision rules controlling it, because no further virtual manager steers the process. E.g., if a combination of rules is leading to quick replenishment, the storage space utilization will increase. In contrast, it will decrease, if the rules are replenishing slowly. Situations in which the storage space utilization is nearing 100%, and only few storage places for new replenishment orders are available, lead to an inefficient replenishment process. To avoid such situations, we pause replenishment order generation, if storage space utilization exceeds 85% and it is continued after it drops below 65% again. Analogously, we pause the pick order generation, if storage space utilization drops below 10% and resume after it exceeds 60% again. The latter is done to avoid draining the inventory completely. Since in both cases either the replenishment stations or the pick stations will become inactive due to no further orders to process, the robots will be reassigned to the remaining active stations. This redistribution of robots across the active stations is done at any time a station becomes active or inactive, i.e. at the beginning and end of order generation pauses.

If a new replenishment order is received, first the rules for ROA and RPS are responsible for choosing a replenishment station and a pod (see Fig. A.8). The time the decision is taken depends on the active rules. The execution of the assignment can earliest be done as soon as there is sufficient capacity on a pod and a station available. Technically, it results in an insertion request (shown as red cylinders), i.e., a request that requires a robot to bring the pod to the workstation. Multiple of these requests are then combined to an insertion task and assigned to a robot by a TA rule. Similarly, after the POA rule selects a pick order from the backlog and the assignment is committed to a pick station, an extraction request (shown as blue cylinders) is generated, i.e. a request that requires bringing a suitable pod to the chosen station. Up to this point, the physical units of SKUs for fulfilling the pick order are not yet chosen. Instead, the decision is postponed and taken right before combining different requests to extraction tasks by PPS and assigning them to robots by TA. This allows the implemented rules to exploit more information when choosing a pod for picking. Hence, in this work we consider PPS as a decision closely interlinked with TA. Furthermore, the system generates store requests (shown as orange cylinders) each time a pod has to be transported to a storage location. The PSA rule only decides the storage location for a pod that is not needed anymore and has to be returned to the storage area. If all requests are already being handled by other robots, the robot will be assigned an idle task, thus, the robot dwells at a dwelling point until needed. Dwelling points can be used to reduce congestion effects if there are only a few active stations compared to the number of robots, e.g. robots waiting at a storage location block others that try to pass by. For this, the robot will park at a free storage location to avoid causing conflicts with other robots. The dwell point policy uses locations in the middle of the storage area to avoid blocking prominent storage locations next to the stations. Another type of task would be charging, which is necessary when robots run low on battery, however, in this work we assume the battery capacity to be infinite, so this type of task is ignored. All of the tasks result in trips (shown as green cylinders). The only exception is when a pod can be used for another task at the same station. The trips are planned by a path planning (PP) algorithm and the resulting paths are executed by the robots. Fig. A.8 shows an abstract overview of these dependencies.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.orp.2019.100128.

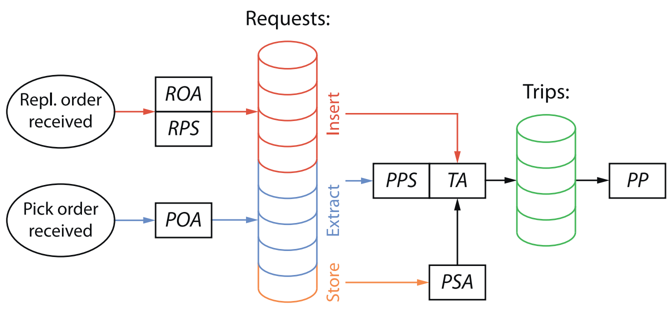


Fig. A2. Order of decisions to be done induced by receiving pick and replenishment orders.

References

[1] Beckschäfer M, Malberg S, Tierney K, Weskamp C. Simulating storage policies for an automated grid-based warehouse system. In: Bektas T, Coniglio S, Martinez-Sykora A, Voß S, editors. Computational logistics Lecture Notes in Computer Science Cham: Springer; 2017. p. 468–82. https://doi.org/10.1007/978-3-319-68496-3_31.

[2] Boysen N, Briskorn D, Emde S. Parts-to-picker based order processing in a rack-moving mobile robots environment. Eur J Oper Res 2017;262(2):550–62.

[3] Bozer YA, Aldarondo FJ. A simulation-based comparison of two goods-to-person order picking systems in an online retail setting. Int J Prod Res 2018;29(1):1–21. https://doi.org/10.1080/00207543.2018.1424364.

[4] Chen CM, Gong Y, de Koster MBM, van Nunen JAEE. A flexible evaluative framework for order picking systems. Prod Oper Manage 2010;19(1):70–82.

[5] Enright J, Wurman PR. Optimization and coordinated autonomy in mobile fulfillment systems. In: Sariel-Talay S, Smith SF, Onder N, editors. Automated action planning for autonomous mobile robots. 2011.

[6] Guan M, Li Z. Genetic algorithm for scattered storage assignment in kiva mobile fulfillment system. Am J Oper Res 2018;08(06):474–85. https://doi.org/10.4236/ajor.2018.86027.

[7] Guan M, Li Z. Pod layout problem in kiva mobile fulfillment system using synchronized zoning. J Appl Math Phys 2018;06(12):2553–62. https://doi.org/10.4236/jamp.2018.612213.

[8] Heragu SS, Cai X, Krishnamurthy A, Malmborg CJ. Analytical models for analysis of automated warehouse material handling systems. Int J Prod Res 2011;49(22):6833–61.

[9] Lamballais, T., Merschformann, M., Roy, D., Suhl, L., & De Koster, M. B. M. (2017a). Optimal policies for resource reallocation in a robotic mobile fulfillment system. Working paper.

[10] Lamballais T, Roy D, de Koster MBM. Estimating performance in a robotic mobile fulfillment system. Eur J Oper Res 2017;256:976–90.

[11] Lamballais, T., Roy, D., & de Koster, M. B. M. (2017c). Inventory allocation in robotic mobile fulfillment systems. Working Paper, available at SSRN.

[12] Merschformann, M., Xie, L., & Erdmann, D. (2017). Path planning for robotic mobile fulfillment systems. Working paper, available at arXivarXiv:1706.09347.

[13] Merschformann M, Xie L, Li H. RAWSim-O: a simulation framework for robotic mobile fulfillment systems. Logist Res 2018;11(8). https://doi.org/10.23773/2018_8.

[14] Nigam S, Roy D, de Koster MBM, Adan IJBF. Analysis of class-based storage strategies for the mobile shelf-based order pick system. Progress in material handling research: 2014. 2014.

[15] Ribino P, Cossentino M, Lodato C, Lopes S. Agent-based simulation study for improving logistic warehouse performance. J Simul 2018;12(1):23–41. https://doi.org/10.1057/s41273-017-0055-z.

[16] Roodbergen KJ, Vis IFA, Taylor GD. Simultaneous determination of warehouse layout and control policies. Int J Prod Res 2014;53(11):3306–26. https://doi.org/10.1080/00207543.2014.978029.

[17] Roy D, Krishnamurthy A, Heragu SS, Malmborg CJ. Performance analysis and design trade-offs in warehouses with autonomous vehicle technology. IIE Trans 2012;44:1045–60.

[18] Wingfield N. As amazon pushes forward with robots, workers find new roles. The New York Times 2017. https://nyti.ms/2xUhVgM

[19] Wurman PR, D'Andrea R, Mountz M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI Magazine 2008;29(1):9–19.

[20] Zou B, Gong Y, Xu X, Yuan Z. Assignment rules in robotic mobile fulfilment systems for online retailers. Int J Prod Res 2017;55(20):6175–92.

[21] Zou B, Xu X, Gong YY, de Koster R. Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system. Eur J Oper Res 2018;267(2):733–53.