Robotic Mobile Fulfillment Systems: a mathematical modelling framework for e-commerce applications

Adrien Rimélé[ab], Michel Gamache[ac], Michel Gendreau[ab], Philippe Grangier[d] and Louis-Martin Rousseau[ab]
[a]Department of Mathematics and Industrial Engineering, Polytechnique Montreal; [b]CIRRELT, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation; [c]GERAD, Group for Research in Decision Analysis; [d]IVADO Labs

ARTICLE HISTORY
Compiled October 20, 2020

ABSTRACT
Robotic Mobile Fulfillment Systems (RMFSs) are a new type of automated warehouse deployed in e-commerce. In this parts-to-picker system, a fleet of small robots is tasked with retrieving and storing shelves of items in the warehouse. Due to the nature of the e-commerce market, and the high flexibility of RMFSs, there are many opportunities to improve the productivity of the warehouse by optimising operational decisions. Online retailers promise extremely fast deliveries, which requires that new orders be included in the set of requests to fulfil as soon as they are revealed. For this reason, and because of the very dynamic nature of the robots' cycles, decision-making needs to be done in real time, in an uncertain environment. Because such a problem often lacks a formal description, we propose a mathematical framework that models the operational decisions taking place in an RMFS as a stochastic dynamic program. Our objective is to formalise optimisation opportunities, to allow researchers to develop more advanced methods in a well-defined environment. Embedded in a discrete event simulator, this model is illustrated by simulations to compare against standard storage decision rules.

KEYWORDS
warehouse; e-commerce; Robotic Mobile Fulfillment System; stochastic dynamic programming; mathematical modelling; simulations

1. Introduction

Warehouses play a central role in any supply chain. Regarding the impact of warehousing activities on the economy, Tompkins and Smith (1998) suggest that the real value of warehousing relies on having the right product in the right place at the right time. According to Gu, Goetschalckx, and McGinnis (2007), a warehouse's primary purposes are to act as a buffer to adapt to the variability of production flow; to consolidate products which come from different sources and must be grouped for shipping to customers; and to add marginal value to the product such as pricing, labelling or customisation. Gu, Goetschalckx, and McGinnis (2007) divide the operations that take place in a warehouse into four types: receiving, storing, picking, and shipping. Picking operations are of critical importance, as emphasised by De Koster, Le-Duc,

CONTACT Adrien Rimélé. Email: adrien.rimele@polymtl.ca

and Roodbergen (2007) and Shah and Khanzode (2017), who estimate that these operations constitute roughly 55% of all warehouse operating expenses, 60% of which is attributable to travelling within the warehouse.

In the business-to-consumer (B2C) segment, the growth of e-commerce in recent years is revolutionising the full sector. In 2016 alone, e-commerce sales increased by 23%, representing 8.7% of the total retail market (Boysen, Stephan, and Weidinger 2019). More than the growth of the e-commerce market, its specificities are the biggest challenge to warehousing operations. E-commerce involves enormous volumes of very small orders (1.6 lines per order on average); requires an enormous assortment of items; and faces uncertain demand with fast-changing trends, promotional events, etc. Facing ample competition, online retailers cannot rely on customer loyalty; instead, they operate under continuous pressure and need to fulfil orders as quickly as possible (Ramanathan 2010), compelling them to offer differentiated services such as same-day delivery from Amazon.

To deliver to customers ever faster, and because 'adaptation is urgent', new types of automated warehouses have appeared (Davarzani and Norrman 2015). One of them is the Robotic Mobile Fulfillment System, which is a parts-to-picker system where a fleet of small robots move between the storage area, where they retrieve and store full shelves of items, and picking stations, where human operators pick the required items from the shelves. Such a system presents great real-time decision-making opportunities, mainly because of its storage flexibility and its fleet of robots operating simultaneously. Indeed, when returning a shelf of items to the storage area, a robot can dynamically choose a new storage location to improve the general performance of the system: this makes the full layout adaptable. Also, several other decisions can be made online to improve the overall performance, such as the scheduling of the incoming orders, the selection of a shelf, or which picking station to use.

The contribution and foremost objective of this paper is to present a mathematical framework to model the dynamic decision-making occurring during the storage and retrieval operations in an RMFS. The nature of e-commerce orders, as well as uncertain processing times and demands, justify online decision-making, without batching. While such situations are typically handled by high-level decision rules evaluated through simulations, we believe that formalising the decision process with a stochastic dynamic model facilitates the development of more advanced solution approaches.

This work is presented as follows. Section 2 describes the Robotic Mobile Fulfillment System, as well as its optimisation opportunities, while Section 3 presents a literature review of existing work on this storage system. Section 4 presents the mathematical model, which formalises the problem under Powell's unified framework of stochastic optimisation (2019). Section 5 demonstrates a *basic* usage of the framework by adapting typical decision rules in a simulation study. Finally, Section 6 offers some conclusions and explores future research opportunities.

2. Robotic Mobile Fulfillment Systems

The Robotic Mobile Fulfillment System, also called *rack-moving mobile robot warehouse* or *Kiva warehouse*, was first introduced by Kiva Systems in 2006 (renamed Amazon Robotics after acquired by Amazon in 2012) (Wurman, D'Andrea, and Mountz 2008; Azadeh, De Koster, and Roy 2017; Boysen, De Koster, and Weidinger 2019). Similar systems using small robots to lift shelves of items have since been developed by other companies, including Alibaba, Knapp, Swisslog, Locus Robotics, and others

(Banker 2016; Kirks et al. 2012; Boysen, De Koster, and Weidinger 2019).

As mentioned in the introduction, an RMFS is a parts-to-picker system in which a fleet of small robots (sometimes called AGVs for automated guided vehicles) is in charge of retrieving and storing shelves (also called pods or bins) of items in the storage area. These robots move around following a system of waypoints organised as a regular grid. Human operators stand at picking stations, ready to pick items from a shelf to send them to the downstream system. The life cycle of a robot corresponds to a *dual command cycle*, which defines an immediate succession of a storage task and a retrieval task. When a robot leaves a picking station, any open storage location can be selected. Once the shelf has been stored, the robot goes directly to the next retrieval location to load another shelf and bring it to a picking station where it may wait in line behind other robots while a human operator is picking items. Interestingly, when a robot is loaded, it moves along the aisles, but when it is unloaded, it can pass under the stored shelves to avoid conflicts.

A typical and more common Automated Storage and Retrieval Systems (ASRS) corresponds to a single unit-load aisle-captive system in which there is one crane per aisle that moves both horizontally and vertically to retrieve bins (Roodbergen and Vis 2009). RMFSs have several advantages over such ASRSs (Wurman, D'Andrea, and Mountz 2008). In particular, the operators located at picking stations can receive shelves from all of (or most of) the storage area, so fewer operators are needed. There is better accountability and accuracy because orders are generally fully processed by one operator, which also reduces delays as it eliminates downstream dependencies. Since any location is accessible by every robot, and there is a large fleet of robots, there is no single point of failure. The system is easily implementable in any environment, and it is easy to extend if needed. Finally, and this is of great interest in terms of optimisation: RMFSs offer enormous storage flexibility. The allocation of a shelf to a storage location can be modified after every picking operation, which allows for the layout to be adapted in real time, whether in response to changing demand or simply to leverage immediate opportunities. For a more detailed and technical review on RMFSs, we refer the reader to the following papers: Wurman, D'Andrea, and Mountz (2008); D'Andrea and Wurman (2008); Enright and Wurman (2011); Azadeh, De Koster, and Roy (2017).

The logic employed by traditional warehouse management systems fails to accommodate the extremely fast delivery times necessary in the e-commerce sector. E-commerce orders must be fulfilled as quickly as possible, necessitating what Banker (2018) calls *order streaming logic*, which 'drops these orders [...] to the floor as soon as they are received'. This is of importance in optimisation because it limits the possibility of processing orders in batches, but instead favours dynamic decision-making in which new orders can be revealed at any time.

3. Literature review

The literature on RMFSs includes research on ASRSs, automated warehouses that were introduced in the 1970s (see Roodbergen and Vis (2009) for a survey of literature on the topic). While there are key differences between RMFSs and common ASRSs, some features are similar. Dual command cycles and the question of block (batching) sequencing versus dynamic sequencing are already central topics in ASRS research. In the case of dynamic sequencing, most if not all works rely on decision rules for storage allocation and sequencing of orders (van den Berg and Gademann

2000; Gagliardi, Renaud, and Ruiz 2014a). Common storage allocation rules include: *random storage*; *turnover-based storage* (the higher the turnover of a bin, the closer to the operator); *closest open location*; and *shortest leg* (select the open storage location closest to the next retrieval location). Through simulations, van den Berg and Gademann (2000) conclude that for dual command cycles, the shortest leg rule performs the best. Gagliardi, Renaud, and Ruiz (2014b) present a discrete event simulator and compare a random, a full turnover-based and a class-based storage policy (a bin is assigned to a class based on its turnover rate; its location within the class is random). Their results show that when realistic conditions are met, full turnover-based storage performs much worse than the other two policies. Their takeaway is that for real-world applications, theoretical results (such as analytical models) may be far from reality. Careful simulation studies should be performed for every special case since it is unlikely that a single simple rule could perform well in all conditions.

These earlier studies have greatly influenced more recent works on RMFSs. An RMFS could even be seen as a special (and more complex) type of ASRS, designated as a mini-load with dual command cycles system with a fleet of non-aisle-captive robots. Several works such as Lamballais, Roy, and De Koster (2017) and Lamballais, Roy, and De Koster (2020), focus on developing queueing networks to analytically estimate KPIs, such as maximum order throughput or average cycle time. Those models facilitate quick estimates of the impact of strategic decisions such as layout design, the number of shelves used to store an item, the ratio of the number of replenishment stations to picking stations, etc. Zou et al. (2017) use a semi-open queueing network to study assignment rules of robots to picking stations and demonstrate that their proposed rule outperforms a random assignment.

Boysen, Briskorn, and Emde (2017) study the problem of sequencing the processing of a set of orders at a picking station, to minimise the number of shelves that need to be brought to fulfil these orders. With diverse methods, such as a mixed integer programming (MIP) model, a decomposition method, and heuristics, they manage to reduce the fleet of robots needed by half. Bozer and Aldarondo (2018) compare a traditional mini-load ASRS with an RMFS, to identify configurations that yield a similar throughput. For a fixed set of parameters and defined decision rules, they conclude that a mini-load system with four aisles and a conveyor belt has similar productivity to an RMFS with 50 robots. They also find that carefully determining the number of picking stations is essential to maintain a high picker utilisation while avoiding congestion. Guan and Li (2018) derive association rules from historical demand data to decide which items to store together on a shelf, to maximise item similarity (the probability that items from the same shelf are ordered together) within the context of scattered storage. They propose a non-linear MIP and a genetic algorithm to solve it, and they obtain a higher item similarity of about 35%, but their results are not converted to actual productivity gains through simulations. Weidinger, Boysen, and Briskorn (2018) define a rack assignment problem for a batch of orders (static scheduling). Their objective is to assign each returning shelf to a storage location, with the surrogate objective of minimising the total loaded distance travelled by the robots. Because they consider the arrival time of each rack at the picking station as a known variable, they do not need to account for robots individually. They devise a variant of an adaptive large neighbourhood search to solve their model and evaluate their approach in terms of total travelling time, for which they obtain better results compared to their baseline. Like van den Berg and Gademann (2000), they note the good performance of the shortest leg decision rule, which operates completely online and presents an average cycle time that is only 3.5% higher than their method. In Mer-

schformann, Xie, and Li (2018), the authors introduce RAWSim-O, a very detailed discrete event simulator that realistically reproduces operations taking place in an RMFS. Merschformann et al. (2019) use this simulation framework to evaluate many combinations of decision rules. They test decision rules for decision problems such as picking and replenishment station assignment, pod selection, or storage assignment. Their study yields significant performance differences between combinations, which emphasises the importance of careful selection for each application. They also note the importance of a proper picking station assignment rule. The cross-dependencies existing between some rules suggests potential benefit in investigating integrated approaches.

Considering the variety of operational subproblems, the uncertainty about demand and operations completion times, as well as the dynamic nature of the operations that should respond in real time to new orders, we propose a mathematical framework that formalises the operational decision-making in such a dynamic stochastic setting. We follow the stochastic optimisation framework of Powell (2019) with the objective of guiding future research on integrated approaches.

4. Formulation

Before presenting the complete dynamic model in Section 4.2, we first describe the key elements considered in the framework and the types of operational decisions we consider.

4.1. *Modelling framework*

As explained in the presentation of RMFSs in Section 2, the operations of a robot consist of a succession of dual command cycles. These cycles require several decisions that we describe here. Since the system is very dynamic and complex, any decision policy needs to be tested through simulation. For this reason, our modelling has strong similarities with a discrete event simulator. First, a decision needs to be made, not at regular time steps, but when a robot becomes available for a new task. We refer to this as an *event*.

Because we do not consider batching of orders, new orders are revealed online and need to be included in the pool of orders to fulfil right away. Note that we consider one-line orders only (one type of item, possibly several units of it) because most e-commerce orders present very few lines. If one wanted to consider multi-lines orders, there are two cases: those with and without consolidation at the picking station. If consolidation is done at the picking station, orders can still be split up by lines and additional constraints can enforce those orders to be processed by the same operator. If not, consolidation is done in the downstream system and does not need to be considered at the retrieval level.

Furthermore, we do not consider the path planning problem, which consists of defining the exact movement of robots within the warehouse. However, because the travelling time taken by a robot may vary due to congestion, we consider stochastic travelling times in the general case. As such, we cannot know in advance exactly when a task will be completed. We assume all tasks to be non-preemptive, which means that when a robot is assigned to a task it cannot be interrupted before completion, even if a better opportunity occurs.

At a picking station, a human operator can only pick items from one shelf at a time. When a robot arrives at a picking station, it may have to wait in a queue, following a first-in, first-out procedure. Note that in the model, we do not explicitly represent the replenishment of shelves, but we do consider a notion of stock levels. Instead, we suggest that a replenishment task could be accounted for in our modelling by considering orders of negative quantities. Since stations are typically fully dedicated to either replenishment or retrieval tasks, simple constraints about which station can process a specific order can be added, but those are not explicitly mentioned in the model.

While dual command cycles (consecutive storage and retrieval tasks) are the standard sequences of tasks for a robot, we also define less frequent *opportunistic tasks*. The purpose of an opportunistic task is to avoid a trip to the storage area if the shelf carried by a robot could be used again to fulfil another order. We distinguish two types of opportunistic tasks. First, when a robot is leaving a picking station, it can go straight to the end of the waiting line of the same or another picking station. In this case, the robot saves the travelling time to the storage area and back. The second type of opportunistic task has the robot in place to deal immediately with another order. In this case, the robot saves both the travelling time and waiting time at a picking station.

Figure 1 presents the decision tree that guides decision-making when an event occurs. If the robot is located at a *picking station*, two types of decisions can be made: a storage task or an opportunistic task. On the other hand, if the robot is in the storage area, the robot must perform a retrieval task, which consists of choosing which order to fulfil, from which shelf, and to which picking station. To elaborate those decisions and their consequences, we first define useful sets and parameters. We will then describe the state variables along with exogenous information (random variables), the decisions that need to be made, the transition function, and the cost function.

[**Figure 1 near here**]

4.2. *Sets*

$\mathcal{I}$ = set of items that can be found in the warehouse.
$\mathcal{L}$ = set of storage locations.
$\mathcal{S}$ = set of storage shelves.
$\mathcal{S}_i$ = subset of shelves $\subset \mathcal{S}$ carrying item $i \in \mathcal{I}$.
$\mathcal{R}$ = set of automated robots.
$\mathcal{P}$ = set of picking stations.

All item types stored in the warehouse are denoted by $\mathcal{I}$. Storage locations $\mathcal{L}$ define locations where a shelf can be stored in the warehouse. Those shelves are defined by set $\mathcal{S}$, and $\mathcal{S}_i$ identifies the subset of shelves containing a specific item $i \in \mathcal{I}$. Robots $\mathcal{R}$ represent the small automated robots that can store and retrieve shelves. Human operators are located at picking stations represented by set $\mathcal{P}$.

4.3. *Parameters*

d_o = deadline of order $o \in \mathcal{O}^k$ (state variable, see Section 4.4.2).
$\underline{cost_o}$ = penalty cost for tardiness of order $o \in \mathcal{O}^k$ (expressed per late time unit).
$\overline{time}_{i,j}$ = expected time for a robot to travel from point i to point j; $(i, j) \in (\mathcal{L} \cup \mathcal{P})^2$.
$\overline{\overline{time}_o}$ = expected time required to pick all the items from order $o \in \mathcal{O}^k$ by an operator.
i_o = item type $\in \mathcal{I}$ of order $o \in \mathcal{O}^k$.
nb_o = number of units of item type $i_o \in \mathcal{I}$ of order $o \in \mathcal{O}^k$.

Because of different levels of priority, such as with *Prime vs Standard* service for Amazon, each order can be associated with distinct penalty costs for tardiness $cost_o$ with respect to deadline d_o. $\overline{time}_{i,j}$ represents the estimated, or expected time needed by a robot to travel from location i to location j; these locations can either be storage locations or picking stations. Depending on the application, if travelling times are deterministic, this value is simply an estimated time. Otherwise, with stochastic travelling times, the actual time taken by the robot is uncertain, but the expected value $\overline{time}_{i,j}$ may still be useful for decision-making. A human operator needs, on average, $\overline{\overline{time}_o}$ to pick the items of order o from a shelf; it may depend on the number of items to pick, the weight of the item, etc. i_o denotes the item type of order o. While we assume that orders are single-line orders, we still define nb_o as the number of units of item type i_o to be picked. This allows for an order to contain more than one unit of the same item but also, should an order be a replenishment task, we could have $nb_o < 0$.

4.4. *State variables*

The state of the warehouse at a given time step describes all of the relevant information needed to make operational decisions: the location of shelves and robots, availability of resources, orders to fulfil, etc. Following the framework of Powell (2019), the state of the system at a given time step k is denoted by S_k and is decomposed into three entities: the *physical state* variables R_k, the *other information* variables I_k, and the *belief state* variables B_k. We detail the role and the components of these entities below.

$S_k = (R_k, I_k, B_k)$

4.4.1. Physical state

The physical state defines all physical characteristics of the warehouse, such as the location and availability of the shelves, the robots, and others. We decompose the physical state as $R_k = (t^k, \mathcal{L}^k_{\mathcal{S}}, \mathcal{T}^k_{\mathcal{S}}, stock^k_{\mathcal{S},\mathcal{I}}, \mathcal{T}^k_{\mathcal{L}}, z^k, \mathcal{L}^k_{\mathcal{R}}, \mathcal{S}^k_{\mathcal{R}}, \mathcal{T}^k_{\mathcal{R}}, \mathcal{O}^k_{\mathcal{R}}, \mathcal{R}^k_{\mathcal{P}})$, of which the components are described below.

Running time

t^k = current time corresponding to time step k.

In our model, a time step k corresponds to an event when a decision must be made, as in a discrete event simulation. The time interval between two such events can be highly variable. We define the state variable t^k as the time of event k.

Shelves

l_s^k = location $\in \mathcal{L}$ of shelf $s \in \mathcal{S}$; $\mathcal{L}_\mathcal{S}^k = \{l_s^k \mid s \in \mathcal{S}\}$.
τ_s^k = time at which shelf $s \in \mathcal{S}$ is available; $\mathcal{T}_\mathcal{S}^k = \{\tau_s^k \mid s \in \mathcal{S}\}$.
$stock_{s,i}^k$ = stock level of item type $i \in \mathcal{I}$ on shelf $s \in \mathcal{S}$;
$stock_{\mathcal{S},\mathcal{I}}^k = \{stock_{s,i}^k \mid s \in \mathcal{S}, i \in \mathcal{I}\}$.

A shelf $s \in \mathcal{S}$ is currently assigned to location $l_s^k \in \mathcal{L} \cup \mathcal{P}$ and will be available by time τ_s^k. In this context, 'available' means the shelf (or location, robot, etc.) can be used right away. At time t^k, even if $l_s^k \in \mathcal{L}$, shelf s may not be available yet, but will be at time $\tau_s^k > t^k$, because it may not have reached its storage location. By convention, we set $\tau_s^k = +\infty$ when the arrival time has not been revealed yet or if the shelf is currently not assigned to a location. Note that in the case of stochastic travelling times, τ_s^k is only known when the shelf has been replaced, so its value will remain set to $+\infty$ until then. However, in a deterministic setting, the arrival time can be anticipated as soon as the storage decision is made. The stochastic case is presented here, but the details around the *belief state* variables will be similar to a deterministic setting. The variable $stock_{s,i}^k$ denotes the level of stock of item type i on shelf s. Depending on the number of items of type i required by an order, a shelf which contains this item may not be usable. Indeed, we consider that all items from an order must be retrieved from a single shelf (to relax this assumption, one may simply decouple a multiple-items order into several one-item distinct orders).

Locations

τ_l^k = time at which location $l \in \mathcal{L}$ is available for storage; $\mathcal{T}_\mathcal{L}^k = \{\tau_l^k \mid l \in \mathcal{L}\}$.

A location l is available for storage by time τ_l^k. This means that if $\tau_l^k \leq t^k$, the location is currently free. If $\tau_l^k = +\infty$, the location is not available for any planned horizon or a robot is on its way but has not yet retrieved the shelf.

Robots

z^k = robot $\in \mathcal{R}$ available for a task.
l_r^k = location $\in \mathcal{L} \cup \mathcal{P}$ of robot $r \in \mathcal{R}$; $\mathcal{L}_\mathcal{R}^k = \{l_r^k \mid r \in \mathcal{R}\}$.
s_r^k = shelf $\in \mathcal{S}$ carried by robot $r \in \mathcal{R}$; $\mathcal{S}_\mathcal{R}^k = \{s_r^k \mid r \in \mathcal{R}\}$.
τ_r^k = time at which robot $r \in \mathcal{R}$ is available; $\mathcal{T}_\mathcal{R}^k = \{\tau_r^k \mid r \in \mathcal{R}\}$.
o_r^k = order $\in \mathcal{O}$ that is currently processed by robot $r \in \mathcal{R}$; $\mathcal{O}_\mathcal{R}^k = \{o_r^k \mid r \in \mathcal{R}\}$.

Robot $z^k \in \mathcal{R}$ denotes the robot currently available for a task (defining an event). A robot $r \in \mathcal{R}$ is currently located at location l_r^k, is carrying shelf s_r^k and will be available next at time τ_r^k. Finally, o_r^k denotes which order robot r is currently processing.

Picking stations

$\mathcal{R}_p^k$ = set of robots $\subset \mathcal{R}$ assigned to picking station $p \in \mathcal{P}$; $\mathcal{R}_{\mathcal{P}}^k = \{\mathcal{R}_p^k \mid p \in \mathcal{P}\}$

Because a human operator needs $time_o$ to pick the items of an order o, the robots wait in a queue. Robots that are assigned to picking station p, and will thus wait in the same queue, are identified by $\mathcal{R}_p^k$.

4.4.2. Other information variables

An essential aspect of dynamic operations in e-commerce comes from new orders continuously entering the system. At a given time step, a set of orders to fulfil is known, but new orders can arrive at any moment. We represent this set of revealed orders as a state variable $\mathcal{O}^k$ of type *other information* as it evolves exogenously even if still influenced by the previous selection of orders to fulfil. Since it is the only variable of this type, we have: $I_k = (\mathcal{O}^k)$.

Orders

$\mathcal{O}^k$ = orders yet to be fulfilled by time step k.

4.4.3. Belief state variables

Depending on the application, some extra information about distributions of random variables may be known and useful to make decisions. As such, they can be stored in the *belief state* B_k. As mentioned above, since we consider stochastic travelling times, we do not know in advance when a robot will reach its destination. However, we can use expected values while making decisions. We define such estimations, as well as information about demand distributions with $B_k = (\overline{\mathcal{T}}^k_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{L}}, \overline{\mathcal{T}}^k_{\mathcal{R},\mathcal{P}}, \bar{\lambda}^k_{\mathcal{I}})$, described below.

$\bar{\tau}_j^k$ = believed (e.g., expected) time of availability of component $j \in \mathcal{R} \cup \mathcal{S} \cup \mathcal{L}$; $\overline{\mathcal{T}}^k_{\mathcal{R}\cup\mathcal{S}\cup\mathcal{L}} = \{\bar{\tau}_j^k \mid j \in \mathcal{R} \cup \mathcal{S} \cup \mathcal{L}\}$.

$\bar{\tau}_{r,p}^k$ = believed time of arrival of robot r at picking station p; $\overline{\mathcal{T}}^k_{\mathcal{R},\mathcal{P}} = \{\bar{\tau}_{r,p}^k \mid r \in \mathcal{R},\, p \in \mathcal{P}\}$.

$\bar{\lambda}_i^k$ = average demand rate of item i, can be a Poisson average or any other distribution information; $\bar{\lambda}^k_{\mathcal{I}} = \{\bar{\lambda}_i^k \mid i \in \mathcal{I}\}$.

$\bar{\tau}_j^k$ represents the believed, or estimated, time of availability of component j, which can be a robot, a shelf, or a location. This estimation can be useful when the true time of availability τ_j^k has not been revealed yet. It can be computed based on past data or on an expected travelling times model. For example, if component j represents a shelf, and if $t^k < \bar{\tau}_j^k < \tau_j^k = +\infty$, it means that shelf j is currently being brought back to a storage location, will be available at an estimated time $\bar{\tau}_j^k$, but its exact arrival time is yet unknown. Similarly, $\bar{\tau}_{r,p}^k$ designates the estimated time of arrival of robot r at picking station p. This information can be useful to estimate the order of the robots in the waiting queue at the picking station, to determine the processing order. Finally,

orders are arriving online, but it is reasonable to consider that some statistics about future orders are available based on current trends, forecasts, planned promotions, etc. As such, we consider here that the arrival of new orders can be reasonably well-modelled with Poisson distributions and that we have an estimate of their current averages $\bar{\lambda}_i^k$ that may evolve over time.

4.5. *Exogenous information*

Between stochastic travelling times and new orders entering the system, new information is continuously revealed, comprising most of the challenge of dynamic decision-making. We represent the exogenous information as a list of random variables $W_{k+1} = ((\hat{r}^{k+1}, \hat{\tau}^{k+1}), \widehat{\mathcal{T}}_{\mathcal{L}}^{k+1}, \widehat{\mathcal{T}}_{\mathcal{R},\mathcal{P}}^{k+1}, \widehat{\Theta}^{k+1})$ that are revealed during the operations:

$(\hat{r}^{k+1}, \hat{\tau}^{k+1})$ = robot becoming available at the next time step and its time of availability. Those information are linked as $\hat{r}^{k+1} = \underset{r \in \mathcal{R}}{argmin}(\hat{\tau}_r^{k+1})$, but only the time of availability of robot $\hat{r}^{k+1}$ is revealed by time step $k+1$.

$\hat{\tau}_l^{k+1}$ = arrival time of a robot at retrieval location l. Between two decision time steps, a robot may have arrived at a retrieval location to retrieve a shelf, the storage location then becomes available; $\widehat{\mathcal{T}}_{\mathcal{L}}^k = \{\hat{\tau}_l^k \mid l \in \mathcal{L}\}$.

$\hat{\tau}_{r,p}^{k+1}$ = arrival time of a robot r to picking station p. Between two decision time steps, a robot may have arrived at a picking station, in this case, its arrival time is revealed. Note that such a robot can very well be the next available robot $\hat{r}^{k+1}$; $\widehat{\mathcal{T}}_{\mathcal{R},\mathcal{P}}^k = \{\hat{\tau}_{r,p}^k \mid r \in \mathcal{R},\, p \in \mathcal{P}\}$.

$\widehat{\Theta}^{k+1}$ = new orders that entered the system between time step k and $k+1$. Realisation of a random variable $\mathbf{\Theta}_{\mathbf{t^k} \to \hat{\tau}^{\mathbf{k+1}}}$ of newly revealed orders between time t^k and $\hat{\tau}^{k+1}$.

In our framework, an event k is defined by the need for a decision, corresponding to a robot requesting a new task. Because of stochastic travelling and processing times, this event is uncertain and is represented by the pair of random variables $(\hat{r}^{k+1}, \hat{\tau}^{k+1})$, which designates the robot asking for a decision and the time at which it happens. The availability of other robots is governed by distinct random variables $\hat{\tau}_{\mathbf{r}}^{\mathbf{k+1}}$, $r \in \mathcal{R}$, but only the realisation of the earliest available robot is revealed. Between two events, other relevant *episodes* (events that do not require a decision) may happen. One such episode is a robot reaching a retrieval location l at time $\hat{\tau}_l^{k+1}$. This information is important because it frees the location for another storage task. Another similar episode is the arrival of a robot at a picking station $\hat{\tau}_{r,p}^{k+1}$ because it determines the order in the waiting queue. By convention, if such time has not been revealed during the two decision steps, its value $\hat{\tau}^{k+1}$ is set to $+\infty$. Finally, between the times of two events, new orders may be revealed in the system. This set of new orders, which arrive between times t^k and t^{k+1}, is represented by a random variable $\mathbf{\Theta}_{\mathbf{t^k} \to \mathbf{t^{k+1}}}$ for which a realisation is denoted by $\widehat{\Theta}^{k+1}$.

4.6. *Decisions*

For a given state of the system S_k, a decision, or action, x_k must be made. Two options are possible, depending on the location of the available robot z^k.

If robot z^k is currently located at a picking station (state: $l^k_{z^k} \in \mathcal{P}$, see Eq. 1), it is carrying a shelf, so its first option is to store the shelf back into the warehouse (storage task, $x_k \in X^1_k$). A decision about which location l to store the shelf at must be made (Figure 2, (1)). Of course, this location needs to be available at the decision time: $\tau^k_l \leq t^k$. If travelling times were deterministic, one could anticipate the availability of a storage location and verify that l will be available by the time the robot reaches it: $\tau^t_l \leq t^k + \overline{time}_{l(z^k)^k,l}$.

The other option is called an *opportunistic task*. Instead of storing its shelf, a robot can reuse it right away to fulfil another order o at picking station p ($x_k \in X^2_k$). We distinguish two cases of such an opportunistic task. First, the robot can bring its shelf to the end of the waiting queue of either the same (Figure 2, (3)) or a different (Figure 2, (2)) picking station (case $p \in \mathcal{P}$). Second, the robot can remain at its current position (Figure 2, (4)) to deal immediately with another order o (case $p = l^{k*}_{z^k}$ which designates the current picking station of the robot with an * as an exponent).

$$
\begin{aligned}
&\textbf{if } l^k_{z^k} \in \mathcal{P}: \\
&\quad x_k \in X^1_k \cup X^2_k \\
&\quad X^1_k = \{l \in \mathcal{L} \mid \tau^k_l \leq t^k\} \textbf{ (storage task)} \\
&\quad X^2_k = \{(o,p) \mid o \in \mathcal{O}^k,\, p \in \mathcal{P} \cup \{l^{k*}_{z^k}\},\, s_{z^k} \in \mathcal{S}_{i_o},\, stock^k_{s_{z^k},i_o} \geq nb_o\} \\
&\qquad \textbf{(opportunistic task)}
\end{aligned}
\tag{1}
$$

[Figure 2 near here]

If robot z^k is available but located at a storage location (state: $l^k_{z^k} \in \mathcal{L}$, see Eq. 2), it is waiting for a retrieval task. The decision is threefold: which order $o \in \mathcal{O}^k$ to fulfil, from which shelf $s \in \mathcal{S}$, and by which picking station $p \in \mathcal{P}$. Of course, candidate shelves must contain the item of the order ($s \in \mathcal{S}_{i_o}$), have a sufficient stock level ($stock^k_{s,i_o} \geq nb_o$), and must be available at decision time ($\tau^k_s \leq t^k$).

$$
\begin{aligned}
&\textbf{if } l^k_{z^k} \in \mathcal{L} \textbf{ (retrieval task)}: \\
&\quad x_k \in X^3_k \\
&\quad X^3_k = \{(o,s,p) \mid o \in \mathcal{O}^k,\, s \in \mathcal{S},\, p \in \mathcal{P},\, \tau^k_s \leq t^k,\, s \in \mathcal{S}_{i_o},\, stock^k_{s,i_o} \geq nb_o\}
\end{aligned}
\tag{2}
$$

4.7. *Transition function*

The transition function S^M describes how the state variables are modified from S_k to S_{k+1} when a decision x_k is made and exogenous information W_{k+1} is revealed.

$$S_{k+1} = S^M(S_k,\, x_k,\, W_{k+1})$$

Note that to slightly lighten the notation, state variables that *do not* change between two consecutive time steps are not detailed. In this case, one must understand that these variables keep the same value at the next step.

First, for all states and decisions, the transition function starts with Algorithm 1. This part is straightforward: the robot available for a task is set to the newly revealed available robot (line 1, Alg. 1), the running time to the time at which this robot becomes available (line 2, Alg. 1), and the newly revealed orders between times t^k and t^{k+1} are added to the pool of orders to fulfil (line 3, Alg. 1). Then, if a location has been freed up during the time interval, its true and believed times of availability are updated (lines 4-6, Alg. 1). Similarly, if a robot has reached a picking station, its believed reaching time is updated (line 7-10, Alg. 1).

Algorithm 1 Transition function - general update

1: $z^{k+1} = \hat{r}^{k+1}$
2: $t^{k+1} = \hat{\tau}^{k+1}$
3: $\mathcal{O}^{k+1} = \mathcal{O}^k \cup \widehat{\Theta}^{k+1}$
4: **for** $l \in \mathcal{L}$ **do**
5: **if** $\hat{\tau}_l^{k+1} < +\infty$ **then**
6: $\tau_l^{k+1} = \bar{\tau}_l^{k+1} = \hat{\tau}_l^{k+1}$
7: **for** $r \in \mathcal{R}$ **do**
8: **for** $p \in \mathcal{P}$ **do**
9: **if** $\hat{\tau}_{r,p}^{k+1} < +\infty$ **then**
10: $\bar{\tau}_{r,p}^{k} = \hat{\tau}_{r,p}^{k+1}$

The rest of the transition function depends, again, on the location of the available robot.

Transition - robot located at picking station ($l_{z^k}^k \in \mathcal{P}$, Algorithm 2)

When a robot is located at a picking station, it can either perform a storage task or an opportunistic task.

If a storage task is chosen (line 1, Alg. 2), then the decision concerns a storage location $x_k = l$. First, the locations of both the robot z_k and its shelf $s_{z_k}^k$ are set to location l (line 2, Alg. 2). Location l is now reserved with no information about when it will become available again, so its true and believed times of availability are both set to ∞ (line 3, Alg. 2). Because of uncertain travelling times, the time of availability of robot $\tau_{z^k}^{k+1}$ is unknown and set to ∞ (line 3, Alg. 2). However, its believed time of availability (and the one of its shelf) can be estimated as the current time plus the expected travelling time from the picking station to the storage location (line 4, Alg. 2). The robot is leaving picking station $l_{z^k}^k$ so its estimated time of arrival at this picking station is set to ∞ (line 5, Alg. 2), and the robot is removed from the robots assigned to the station (line 6, Alg. 2).

If the robot performs an opportunistic task instead, a new order o and a picking station p are selected (line 7, Alg. 2). The locations of the robot and its shelf are set to the picking station p (line 8, Alg. 2). The new order will pick items from the shelf, so the stock level of the shelf in items i_o is updated (line 9, Alg. 2). Order o is removed from the list of remaining orders to fulfil (line 10, Alg. 2) and the robot is assigned to this order (line 11, Alg. 2). By the next time step, if the same robot z^k has not yet arrived at station p and the decision was to send the robot at the end of a waiting queue ($p \neq l_{z^k}^{k*}$), its estimated time of arrival to station p is set to the running time plus the average travelling time between the two stations (line 14, Alg. 2). Then, if the new picking station is different from the current one, robot z^k is removed from

the robots assigned to station $l^k_{z^k}$ (line 16, Alg. 2) and added to the ones of station p (line 18, Alg. 2); its estimated time of arrival to station $l^k_{z^k}$ is also set to $+\infty$ (line 17, Alg. 2).

Finally, in both cases, if more robots are still assigned to station $l^k_{z^k}$ (line 19, Alg. 2), one can identify the next robot to be processed as the robot r that arrived the earliest (line 19, Alg. 2). If this robot r is not the one available at next time step (which would already be revealed), its believed time of availability is set to the running time plus the average time needed by the human operator to pick the items of its assigned order $\overline{time}_{o^k(r)}$ (line 22, Alg. 2).

Algorithm 2 Robot located at a picking station ($l^k_{z^k} \in \mathcal{P}$)

1: **if** $x_k = l \in X^1_k \subset \mathcal{L}$ **then** ▷ Storage task
2: $\quad l^{k+1}_{z^k} = l^{k+1}_{s^k(z^k)} = l$
3: $\quad \tau^{k+1}_l = \bar{\tau}^{k+1}_l = \tau^{k+1}_{z^k} = +\infty$
4: $\quad \bar{\tau}^{k+1}_{z^k} = \bar{\tau}^{k+1}_{s^k(z^k)} = t^k + \overline{time}_{l(z^k)^k, l}$
5: $\quad \bar{\tau}^{k+1}_{z^k, l^k(z^k)} = +\infty$
6: $\quad \mathcal{R}^{k+1}_{l^k(z^k)} = \mathcal{R}^k_{l^k(z^k)} \setminus z^k$
7: **else if** $x_k = (o, p) \in X^2_k \subset (\mathcal{O}^k, \mathcal{P} \cup \{l^{k*}_{z^k}\})$ **then** ▷ Opportunistic task
8: $\quad l^{k+1}_{z^k} = l^{k+1}_{s^k(z^k)} = p$
9: $\quad stock^{k+1}_{s^k(z^k), i_o} = stock^k_{s^k(z^k), i_o} - nb_o$
10: $\quad \mathcal{O}^{k+1} = \mathcal{O}^k \setminus o$
11: $\quad o^{k+1}_{z^k} = o$
12: $\quad$ **if** $\hat{\tau}^{k+1}_{z^k, p} = +\infty$ **then**
13: $\quad\quad$ **if** $p \in \mathcal{P}$ **then**
14: $\quad\quad\quad \bar{\tau}^{k+1}_{z^k, p} = t^k + \overline{time}_{l^k(z^k), p}$
15: $\quad$ **if** $p \in \mathcal{P} \setminus l^k_{z^k}$ **then**
16: $\quad\quad \mathcal{R}^{k+1}_{l^k(z^k)} = \mathcal{R}^k_{l^k(z^k)} \setminus z^k$
17: $\quad\quad \bar{\tau}^{k+1}_{z^k, l^k(z^k)} = +\infty$
18: $\quad\quad \mathcal{R}^{k+1}_p = \mathcal{R}^k_p \cup z^k$
19: **if** $\mathcal{R}^{k+1}_{l^k(z^k)} \neq \varnothing$ **then** ▷ Common to both tasks
20: $\quad$ **let** $r = \underset{r' \in \mathcal{R}}{argmin}(\bar{\tau}^{k+1}_{r', l^k(z^k)})$
21: $\quad$ **if** $r \neq \hat{r}^{k+1}$ **then**
22: $\quad\quad \bar{\tau}^{k+1}_r = t^k + \overline{time}_{o^k(r)}$

Transition - robot located in storage area ($l^k_{z^k} \in \mathcal{L}$, Algorithm 3)

When a robot is available at a storage location, it needs to perform a retrieval task which consists in making a decision $x_k = (o, s, p)$ of an order o to fulfil, from shelf s, at picking station p.

First, both the true and believed times of availability of the shelf just dropped by the robot are now set to the running time (line 1, Alg. 3). The robot is now carrying the selected shelf s (line 2, Alg. 3). The true and believed times of availability of this shelf, as well as the true time of availability of the robot, are all set to $+\infty$ (line 3,

Alg. 3) because the estimated time of when the items will be picked is hard to obtain at this stage (even if a rough estimate could be considered). The locations of the robot and shelf s become station p (line 4, Alg. 3) and the robot is added to the set of robots assigned to the picking station (line 5, Alg. 3). The stock level of the shelf in item type i_o is updated (line 6, Alg. 3). The order o is removed from the set of orders to fulfil (line 7, Alg. 3) and the order is assigned to robot z^k (line 8, Alg. 3). If the robot has not reached the retrieval location by time step $k+1$, the believed time of availability of storage location l_s^k is estimated to be the running time plus the travelling time between the drop-off and retrieval locations (lines 9-10, Alg. 3). Finally, if the robot has not arrived at the picking station by time step $k+1$, the believed time of arrival of robot z^k at station p is calculated as the running time plus the travelling time from drop-off to retrieval locations and retrieval location to picking station (line 11-12, Alg. 3).

Algorithm 3 Retrieval task ($l_{z^k}^k \in \mathcal{L}, x_k = (o, s, p) \in X_k^3 \subset (\mathcal{O}^k, \mathcal{S}, \mathcal{P})$)

1: $\tau_{s^k(z^k)}^{k+1} = \bar{\tau}_{s^k(z^k)}^{k+1} = t^k$
2: $s_{z^k}^{k+1} = s$
3: $\tau_s^{k+1} = \bar{\tau}_s^{k+1} = \bar{\tau}_{z^k}^{k+1} = +\infty$
4: $l_{z^k}^{k+1} = l_s^{k+1} = p$
5: $\mathcal{R}_p^{k+1} = \mathcal{R}_p^k \cup z^k$
6: $stock_{s,i_o}^{k+1} = stock_{s,i_o}^k - nb_o$
7: $\mathcal{O}^{k+1} = \mathcal{O}^k \setminus o$
8: $o_{z^k}^{k+1} = o$
9: **if** $\hat{\tau}_{l_s^k}^{k+1} = +\infty$ **then**
10: $\quad \bar{\tau}_{l_s^k}^{k+1} = t^k + \overline{time}_{l^k(z^k), l_s^k}$
11: **if** $\hat{\tau}_{z^k,p}^{k+1} = +\infty$ **then**
12: $\quad \bar{\tau}_{z^k,p}^{k+1} = t^k + \overline{time}_{l^k(z^k), l_s^k} + \overline{time}_{l_s^k, p}$

4.8. *Cost function*

The cost function defines the cost contribution of a decision to the global objective that needs to be optimised. While the previous parts of the model are quite universal, the objective function depends on the Key Performance Indicator (KPI) the decision-maker favours most. Here, we present two objective contributions related to travelling times and deadlines. The first one considers the full cycle time in a fully stochastic setting (as the rest of the model). The second one only considers the travelling time of the robots in a setting where travelling times are deterministic. The reason for this double illustration is that to truly assess the complete cycle time of a robot in the fully stochastic case, we need to wait for the robot to have finished its cycle, which makes the contribution of each decision less perceptible. In both cases, Figure 3 illustrates the time components that compose a complete cycle.

4.8.1. Stochastic case

In this case, the time taken by a robot to perform a complete cycle is considered in the objective, which includes the travelling time as well as any waiting time spent in a queue at a picking station. Because the real time taken by a robot to complete a task

is only revealed once the task has been completed, its contribution to the objective function is likewise delayed. For this reason, we add a physical state variable for each robot, representing the time at which it started its task:

$$\varphi_r^k = \text{time at which robot } r \in \mathcal{R} \text{ started its new task; } \varphi_{\mathcal{R}}^k = \{\varphi_r^k \mid r \in \mathcal{R}\}$$

And we update this state variable every time a robot is assigned to a new task. We add at the end of Algorithm 1, line 11:

Algorithm 4 Extra line in Algorithm 1

11: $\varphi_{z^k}^{k+1} = t^k$

Now, in every state, the time taken to perform the *previous* task can be computed by taking the difference between the current time t^k and the time the robot started this task $\varphi_{z^k}^k$ (Eq. 3). Also, if the robot is located at a picking station, an order was just completed, and we can add a potential penalty cost if the deadline of this order was violated: $cost_{o^k(z^k)} \times max[0, t^k - d_{o^k(z^k)}]$.

For a given state, decision and exogenous information, we denote the objective contribution by: $C(S_k, x_k, W_{k+1})$

$$\begin{aligned}
&\textbf{if } l_{z^k}^k \in \mathcal{P} \textbf{ (Robot located at a picking station)} : \\
&\quad C(S_k, x_k, W_{k+1}) = t^k - \varphi_{z^k}^k + cost_{o^k(z^k)} \times max[0, t^k - d_{o^k(z^k)}] \\
&\textbf{if } l_{z^k}^k \in \mathcal{L} \textbf{ (Robot located at a storage location)} : \\
&\quad C(S_k, x_k, W_{k+1}) = t^k - \varphi_{z^k}^k
\end{aligned} \tag{3}$$

4.8.2. Deterministic case - travelling time

We present the objective of the deterministic travelling time to have a more intuitive representation of the objective contribution. In Eq. 4, we distinguish the different types of decisions. In the case of a storage task or an opportunistic task, the costs are similar: the travelling time from the current picking station to either the storage location or another picking station (if the robot stays in place to deal immediately with another order, this cost is 0), and a potential penalty cost for violated deadlines. If the robot performs a retrieval task, the cost corresponds to the interleaving time between the drop-off and the retrieval locations plus the time between the retrieval location and the picking station.

$$\begin{aligned}
&\textbf{if } l_{z^k}^k \in \mathcal{P} \text{ and } x_k = l \in X_k^1 \subset \mathcal{L} \textbf{ (Storage task)} : \\
&\quad C(S_k, x_k, W_{k+1}) = \overline{time}_{l(z^k)^k, l} + cost_{o^k(z^k)} \times max[0, t^k - d_{o^k(z^k)}] \\
&\textbf{if } l_{z^k}^k \in \mathcal{P} \text{ and } x_k = (o, p) \in X_k^2 \subset (\mathcal{O}^k, \mathcal{P} \cup \{l_{z^k}^{k*}\}) \textbf{ (Opportunistic task)} : \\
&\quad C(S_k, x_k, W_{k+1}) = \overline{time}_{l(z^k)^k, p} + cost_{o^k(z^k)} \times max[0, t^k - d_{o^k(z^k)}] \\
&\textbf{if } l_{z^k}^k \in \mathcal{L} \text{ and } x_k = (o, s, p) \in X_k^3 \subset (\mathcal{O}^k, \mathcal{S}, \mathcal{P}) \textbf{ (Retrieval task)} : \\
&\quad C(S_k, x_k, W_{k+1}) = \overline{time}_{l^k(z^k), l_s^k} + \overline{time}_{l_s^k, p}
\end{aligned} \tag{4}$$

[**Figure 3 near here**]

4.8.3. Global objective

Finally, the global objective is, for every state, to make the decision that minimises the global cost of operations. This is defined recursively by Eq. 5, equivalent to Bellman equations.

$$\min_{\pi} \mathbb{E}_{S_0} \mathbb{E}_{W_1 \dots W_{+\infty} | S_0} \left\{ \sum_{k=0}^{+\infty} \alpha^k \, C_k(S_k,\, X_k^{\pi}(S_k),\, W_{k+1}) \mid S_0 \right\}, \tag{5}$$

where α is a discount rate $\in]0\,;1[$ and:

$$S_{k+1} = S^M(S_k,\, X_k^{\pi}(S_k),\, W_{k+1}) \tag{6}$$

More precisely, the objective is to define a policy π that minimises the expected infinite sum of discounted cost contributions $C_k(S_k,\, X_k^{\pi}(S_k),\, W_{k+1})$, considering the recursive relationship between two consecutive states defined by Eq. 6.

We may also prefer to not consider any discount rate in the objective function, for instance to properly minimise the expected average cycle time. In this case, we can define the global objective as in Eq. 7, which averages the infinite sum of cost contributions.

$$\min_{\pi} \lim_{h \to +\infty} \mathbb{E}_{S_0} \mathbb{E}_{W_1 \dots W_h | S_0} \left\{ \frac{1}{h} \sum_{k=0}^{h} C_k(S_k,\, X_k^{\pi}(S_k),\, W_{k+1}) \mid S_0 \right\} \tag{7}$$

5. Simulation study

Because the subproblem of storage allocation is recurrent both in traditional ASRSs and also in RMFSs, we propose illustrating a simple usage of our framework by comparing the performance of the four most common storage decision rules. The objective of this section is neither to propose a new method, nor a fully integrated simulation study, but to compare storage policies that should be considered as baselines when developing new methods. We study the performance of those baselines in terms of the two KPIs presented in Section 4.8: the travelling time and the full cycle time. The baselines are tested in isolation under simplification assumptions that are presented below.

5.1. *Assumptions*

To study storage allocation baselines in isolation, we make several simplifications. First, we consider a single picking station. A type of item is stored on exactly one shelf, which only contains this item, and replenishment is ignored. Both travelling and picking times are deterministic. Revealed orders are processed following their deadlines, with the condition that the required item is on a shelf that is not currently being carried by another robot. Because this online scheduling is not optimised, delay penalties in the objective functions presented in the model are removed. Only opportunistic tasks of type 1 (where the robot goes straight to the end of the waiting queue at a

picking station) are considered, and they are enforced as often as possible (if the next order needs the same shelf that the robot is currently carrying). Also, because the next order assigned to a robot is an external decision, we can use this information at the time when a storage decision is made. To do so, we add an *other information* variable to the ones defined in Section 4.4.2, representing this order. Let us denote this state variable by o^k_{next}. Note that the only event that requires a decision is when a robot is available at the picking station: $l^k_{z^k} \in \mathcal{P}$, we can then omit the case of a retrieval task when $l^k_{z^k} \in \mathcal{L}$ in Eq. 2.

5.2. *Storage allocation baselines*

In this section, we express the storage decision rules within our modelling framework. Because we enforce opportunistic tasks of type 1 whenever possible, an actual storage task only occurs when $s^k_{z^k} \neq \mathcal{S}_{i(o^k_{\text{next}})}$ (here $\mathcal{S}_i$ only designates one shelf instead of a set), which means that the robot is not carrying the shelf containing the required item. Otherwise, the opportunistic task is performed by default, which consists of processing the next order o^k_{next} at picking station $p = 1$ (because there is only one picking station): $x_k = (o^k_{\text{next}}, p = 1)$. We only represent this second case in the random storage policy.

Random storage

In random storage, the storage decision x_k is randomly drawn from the uniform distribution over the set of available locations at time t^k.

$$\begin{aligned}
&\textbf{if } s^k_{z^k} \neq \mathcal{S}_{i(o^k_{\text{next}})} : \\
&\qquad x_k \sim U(X^1_k) \\
&\qquad \textbf{where } X^1_k = \{l;\, l \in \mathcal{L},\, \tau^k_l \leq t^k\} \\
&\textbf{else:} \\
&\qquad x_k = (o^k_{\text{next}}, p = 1) \in (\mathcal{O}^k, \mathcal{P}) \subset X^2_k
\end{aligned} \tag{8}$$

Closest open location

The closest open location policy consists of selecting the available storage that is closest to the picking station as defined by Eq. 9.

$$\begin{aligned}
&\textbf{if } s^k_{z^k} \neq \mathcal{S}_{i(o^k_{\text{next}})} : \\
&\qquad x_k = \underset{l}{argmin} \left(time_{p=1,l} \mid l \in \mathcal{L},\, \tau^k_l \leq t^k \right)
\end{aligned} \tag{9}$$

Class-based storage

In class-based storage, a shelf is randomly stored within a class based on its turnover rate. Classes are distributed by increasing distances from the picking station and shelves with the highest turnovers are assigned to the closest classes. Because here a shelf contains exactly one item type, which can only be found on this shelf, the turnover rate of a shelf s corresponds to that of its item type ($\bar{\lambda}^k_i;\, s = \mathcal{S}_i$). Eq. 10 randomly draws the storage location from a uniform distribution over the set of available

locations within the class of shelf $s^k_{z^k}$. The item contained on the shelf is $i(o^k(z^k))$, which gives the turnover rate of the shelf $\text{Class}(s^k_{z^k})$.

$$\begin{aligned}&\textbf{if } s^k_{z^k} \neq \mathcal{S}_{i(o^k_{\text{next}})}:\\ &\quad x_k \sim U(\text{Class}(s^k_{z^k}))\\ &\quad \textbf{where } \text{Class}(s^k_{z^k}) \text{ defines the open locations of the class associated with}\\ &\quad \text{shelf } s^k_{z^k} \text{ based on its turnover rate } \bar{\lambda}^k_{i(o^k(z^k))}\end{aligned} \tag{10}$$

Shortest leg

For the shortest leg decision rule, Eq. 11 selects the open location that minimises the travelling time to the storage location plus the time from the storage location to the next retrieval location. The next shelf to retrieve for the robot at hand is $\mathcal{S}_{i(o^k_{\text{next}})}$, and its location is $l' = l(\mathcal{S}_{i(o^k_{\text{next}})})$. So, the selected location x_k is the one that minimises $time_{p=1,\,l} + time_{l,\,l'}$.

$$\begin{aligned}&\textbf{if } s^k_{z^k} \neq \mathcal{S}_{i(o^k_{\text{next}})}:\\ &\quad x_k = \underset{l}{argmin}\left(time_{p=1,\,l} + time_{l,\,l'} \mid l \in \mathcal{L},\, \tau^k_l \leq t^k,\, l' = l(\mathcal{S}_{i(o^k_{\text{next}})})\right)\end{aligned} \tag{11}$$

5.3. *Simulation study*

In this section, we run simulations to evaluate the four decision rules presented above with the two cost functions of Section 4.8: full-cycle time and travelling time with the average cost global objective (Eq. 7). We define a reasonably small warehouse storage area since we consider only one picking station, and we keep the setting parameters constant except the skewness parameter of the demand distribution that we describe below.

The storage area contains 108 storage locations for 103 shelves, and 8 robots are deployed. The speed of the robots is set to 0.6 m/s, accounting for acceleration, deceleration, loading, and unloading times. The picking time taken by the human operators is set to 5 seconds. The number of classes for the class-based storage policy is set to 6. Figure 4 gives a plan view of such a warehouse.

As in regular warehousing, some items are much more popular than others, which translates into skewed demand distributions. We generate orders following the principle of an ABC curve proposed by Hausman, Schwarz, and Graves (1976): $G(x) = x^s$, for $0 < s \leq 1$, represents the ranked cumulative % demand versus % (x) of item types. The smaller the s, the more skewed the demand distribution. To every item type i a Poisson distribution of average $\bar{\lambda}_i = \left(\left(\frac{i}{m}\right)^s - \left(\frac{i-1}{m}\right)^s\right) \times \frac{n}{N}$ is assigned, where m is the number of item types, n the expected total number of orders over the time horizon and N the number of discretized periods. Orders are then generated online at each discretized period, for each item. Then, for every generated order k, a completion time δ_k (time before deadline) is randomly drawn from a uniform distribution of interval $[1; \alpha \times T]$, where α defines the tightness of the completion times. In this simulation study, we run the simulation for 24 hours with time steps of 60 seconds to generate new orders. Several values of skewness parameter s are tested and the

tightness parameter α is set to 0.4. We also set $m = 103$, $n = 30,000$ and $N = 1440$ ($24\,h\,/\,60\,sec$).

[**Figure 4 near here**]

Table 1 presents the performance of the four decision rules in terms of travelling time for 8 values of the skewness parameter s. Note that, because of the stochasticity in the simulation process, each value presented in the table corresponds to the average over 100 runs. For each decision rule, one column presents the average travelling time and the other the relative gain compared to the random storage policy. First, we notice that the travelling time, for all storage policies, increases with the increase of parameter s. This is explained by the greater number of opportunistic tasks that can be performed with skewed distributions. Indeed, when an item is ordered much more often relative to others, the chances that two consecutive orders will require the same item are greater. Overall, the best-performing policy is the shortest leg (SL), with performance gains ranging from 10.6% to 13.6%. The only exception is for extremely skewed distributions ($s = 0.4$), for which class-based storage performs best with an improvement of 15.3%. However, the performance of class-based storage declines sharply when the value of s increases; it even becomes worse than random when $s = 1$. Interestingly, SL's performance remains reasonably constant, with some extra performance gains for smaller s. This can be explained by the fact that often-required shelves are frequently moved and may end up occupying, by chance, locations closer to the picking station, which results in shorter cycle times. We note the poor global performance of the closest open location (COL) method that remains independent of the value of parameter s, with performance gains around 5%.

[**Table 1 near here**]

Table 2 presents the same type of information but in terms of full cycle time, considering the waiting time spent by robots at the picking station as well. We note some relevant differences. First, as expected, all policies result in longer full-cycle times compared to travelling times only (which is a lower bound); robots take on average 10 seconds more to complete a full cycle. This simply means that robots are losing time waiting in the waiting queue. Then, while the cycle time for all policies still benefits from lower s values, the differences are less significant. This can come from opportunistic tasks that do not result in significant performance gains due to congestion at the picking station. The best-performing policies dependent upon s remain the same as before. Interestingly, SL, the best overall policy, does not seem to benefit from smaller s values, compared to random. It may be that conveniently-located, high-turnover shelves do not result in faster cycles, once again because of congestion.

[**Table 2 near here**]

While this study remains simple, with an illustrative objective only, we can draw some relevant insights from the results. First, considering the performance of class-based and SL policies on the travelling time only, we understand the existence of a trade-off between strategically locating shelves based on the turnover rate of the items, to favour future accessibility, and minimising immediate cycle times. Then, with regards to the waiting time spent in the picking station's queue, we understand that time saved in travelling can be wasted.

6. Conclusions and future work

In this work, we presented a dynamic stochastic optimisation framework that models real-time operational decisions within a Robotic Mobile Fulfillment System. This new

system of automated warehouses was developed specifically for the challenges of the e-commerce market. Among those challenges, online retailers need to fulfil orders extremely fast in a highly competitive environment. Thus, the necessity of considering orders as soon as they are revealed to the system. Coupled with the great flexibility of storage allocation of RMFSs, as well as the diverse operational decisions that need to be made, the global process results in a stochastic dynamic problem that is typically tackled by high-level decision rules. By formalising such a problem with a model, even though there is no obvious solving mechanism to be proposed, it will assist researchers in the development of more advanced methods to improve the performance of either specific subproblems in isolation or the integrated problem. Essential elements of the model are: stochastic demand (orders are revealed online); stochastic travelling and picking times for the robots and operators; irregular decision-making time steps that follow times at which events occur, similar to a discrete event simulation; definition of opportunistic tasks that model the possibility of combining two compatible orders to save travelling time; a waiting queue for robots at picking stations. To illustrate the model, we propose a simple simulation study using storage allocation decision rules from the literature. We transcribe these rules following the model's proposed notation and run experiments to evaluate their performance and gain insights about improvement opportunities.

In the future, based on the assessment of the storage decision rules illustration, we wish to build on the modelling framework to develop new storage policies other than traditional decision rules. We think that methods coming from approximate dynamic programming or reinforcement learning could be applicable here, to learn from repeated experiences and demand trends in order to improve performance. While considering storage allocation only, we understand that there exists a trade-off between minimising travelling times and avoiding saturation at picking stations. We want to build on this observation in our future methods.

Acknowledgements

We are grateful to the Mitacs Accelerate Program for providing funding for this project. We also wish to gratefully acknowledge the support and valuable insights of our industrial partners, JDA Software and Element AI.

References

Azadeh, Kaveh, René De Koster, and Debjit Roy. 2017. "Robotized Warehouse Systems: Developments and Research Opportunities." *SSRN Electronic Journal* 1–55.

Banker, Steve. 2016. "Robots In The Warehouse: It's Not Just Amazon." *Forbes Business* https://www.forbes.com/sites/stevebanker/2016/01/11/robots-in-the-warehouse-its-not-just-amazon.

Banker, Steve. 2018. "New Solution Changes the Rules of Warehouse Automation." *Forbes Business* https://www.forbes.com/sites/stevebanker/2018/06/12/new-solution-changes-the-rules-of-warehouse-automation.

Boysen, Nils, Dirk Briskorn, and Simon Emde. 2017. "Parts-to-picker based order processing in a rack-moving mobile robots environment." *European Journal of Operational Research* 262 (2): 550–562.

Boysen, Nils, René De Koster, and Felix Weidinger. 2019. "Warehousing in the e-commerce era: A survey." *European Journal of Operational Research* 277 (2): 396–411.

Boysen, Nils, Konrad Stephan, and Felix Weidinger. 2019. "Manual order consolidation with put walls: the batched order bin sequencing problem." *EURO Journal on Transportation and Logistics* 8 (2): 169–193.

Bozer, Yavuz A., and Francisco J. Aldarondo. 2018. "A simulation-based comparison of two goods-to-person order picking systems in an online retail setting." *International Journal of Production Research* 56 (11): 3838–3858.

D'Andrea, Raffaello, and Peter Wurman. 2008. "Future challenges of coordinating hundreds of autonomous vehicles in distribution facilities." *2008 IEEE International Conference on Technologies for Practical Robot Applications, TePRA* 80–83.

Davarzani, Hoda, and Andreas Norrman. 2015. "Toward a relevant agenda for warehousing research: literature review and practitioners' input." *Logistics Research* 8 (1).

De Koster, René, Tho Le-Duc, and Kees Jan Roodbergen. 2007. "Design and control of warehouse order picking: A literature review." *European Journal of Operational Research* 182 (2): 481–501.

Enright, John J., and Peter R. Wurman. 2011. "Optimization and coordinated autonomy in mobile fulfillment systems." *AAAI Workshop - Technical Report* WS-11-09: 33–38.

Gagliardi, Jean-Philippe, Jacques Renaud, and Angel Ruiz. 2014a. "A simulation modeling framework for multiple-aisle automated storage and retrieval systems." *Journal of Intelligent Manufacturing* 25 (1): 193–207.

Gagliardi, Jean-Philippe, Jacques Renaud, and Angel Ruiz. 2014b. "On sequencing policies for unit-load automated storage and retrieval systems." *International Journal of Production Research* 52 (4): 1090–1099.

Gu, Jinxiang, Marc Goetschalckx, and Leon F. McGinnis. 2007. "Research on warehouse operation: A comprehensive review." *European Journal of Operational Research* 177 (1): 1–21.

Guan, Mengcheng, and Zhenping Li. 2018. "Genetic Algorithm for Scattered Storage Assignment in Kiva Mobile Fulfillment System." *American Journal of Operations Research* 08 (06): 474–485.

Hausman, Warren, Leroy Schwarz, and Stephen Graves. 1976. "Optimal Storage Assignment in Automatic Warehousing Systems." *Management Science* 22 (6): 629–638.

Kirks, Thomas, Jonas Stenzel, Andreas Kamagaew, and Michael ten Hompel. 2012. "Cellular Transport Vehicles for Flexible and Changeable Facility Logistics Systems." *Logistics Journal* 2192(9084) (1).

Lamballais, Tim, Debjit Roy, and René De Koster. 2017. "Estimating performance in a Robotic Mobile Fulfillment System." *European Journal of Operational Research* 256 (3): 976–990.

Lamballais, Tim, Debjit Roy, and René De Koster. 2020. "Inventory allocation in robotic mobile fulfillment systems." *IISE Transactions* 52 (1): 1–17.

Merschformann, Marius, Tim Lamballais, René De Koster, and L. Suhl. 2019. "Decision rules for robotic mobile ful fi llment systems." *Operations Research Perspectives* 6 (March): 100128.

Merschformann, Marius, Lin Xie, and Hanyi Li. 2018. "RAWSim-O: A simulation framework for robotic mobile fulfillment systems." *Logistics Research* 11 (1): 1–11.

Powell, Warren B. 2019. "A unified framework for stochastic optimization." *European Journal of Operational Research* 275 (3): 795–821.

Ramanathan, Ramakrishnan. 2010. "The moderating roles of risk and efficiency on the relationship between logistics performance and customer loyalty in e-commerce." *Transportation Research Part E: Logistics and Transportation Review* 46 (6): 950–962.

Roodbergen, Kees Jan, and Iris F. A. Vis. 2009. "A survey of literature on automated storage and retrieval systems." *European Journal of Operational Research* 194 (2): 343–362.

Shah, Bhavin, and Vivek Khanzode. 2017. "A comprehensive review of warehouse operational issues." *International Journal of Logistics Systems and Management* 26 (3): 346–378.

Tompkins, J. A., and J. D. Smith. 1998. *The Warehouse Management Handbook, Second Edition.* Tompkins Press.

van den Berg, Jeroen P., and A.J.R.M. Gademann. 2000. "Simulation study of an automated storage/retrieval system." *International Journal of Production Research* 38 (6): 1339–1356.

Weidinger, Felix, Nils Boysen, and Dirk Briskorn. 2018. "Storage assignment with rack-moving mobile robots in KIVA warehouses." *Transportation Science* 52 (6): 1479–1495.

Wurman, Peter R., Raffaello D'Andrea, and Mick Mountz. 2008. "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses." *AI Magazine* 29 (1): 9–9.

Zou, Bipan, Yeming (Yale) Gong, Xianhao Xu, and Zhe Yuan. 2017. "Assignment rules in robotic mobile fulfilment systems for online retailers." *International Journal of Production Research* 55 (20): 6175–6192.

Table 1. Travelling times

Storage	Random		COL		Class-based		SL	
policy	t(s)	g(%)	t(s)	g(%)	t(s)	g(%)	t(s)	g(%)
s=0.4	46.94	0.0	44.58	5.03	40.12	**15.3**	41.49	13.6
s=0.5	49.99	0.0	47.57	4.84	44.44	11.67	44.46	**12.44**
s=0.6	52.93	0.0	50.33	4.91	48.12	9.56	47.30	**11.7**
s=0.7	53.47	0.0	50.86	4.88	48.70	9.38	47.41	**12.44**
s=0.8	54.41	0.0	51.52	5.31	52.56	3.59	48.33	**11.57**
s=0.9	54.52	0.0	51.82	4.95	54.35	0.33	48.55	**10.98**
s=1.0	54.51	0.0	51.82	4.93	55.72	-2.34	48.59	**10.62**

Table 2. Full cycle times

Storage	Random		COL		Class-based		SL	
policy	t(s)	g(%)	t(s)	g(%)	t(s)	g(%)	t(s)	g(%)
s=0.4	55.65	0.0	53.38	4.08	50.28	**10.1**	50.74	9.77
s=0.5	58.21	0.0	55.70	4.31	53.53	8.40	52.87	**9.98**
s=0.6	60.74	0.0	57.90	4.68	56.43	7.44	55.14	**9.92**
s=0.7	61.16	0.0	58.31	4.66	56.74	7.58	55.14	**10.6**
s=0.8	62.00	0.0	58.92	4.97	60.33	2.83	55.95	**10.0**
s=0.9	62.09	0.0	59.20	4.65	61.93	0.27	56.13	**9.62**
s=1.0	62.08	0.0	59.19	4.66	63.15	-1.81	56.18	**9.34**

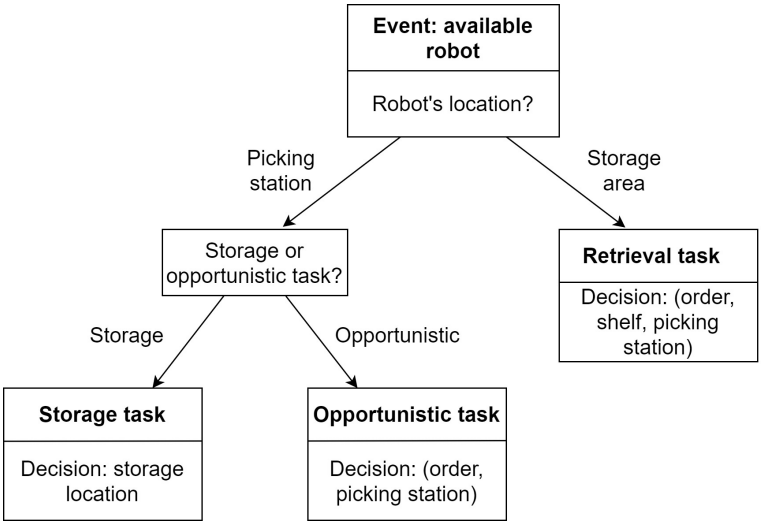


Figure 1. Decision tree

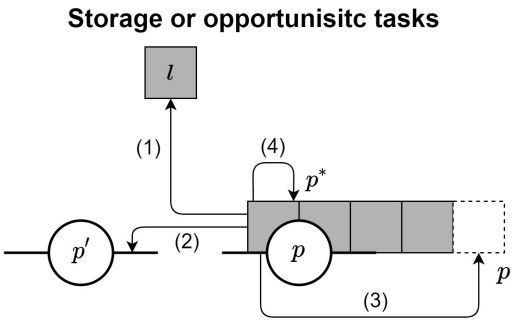


Figure 2. Possible decisions when a robot is located at a picking station

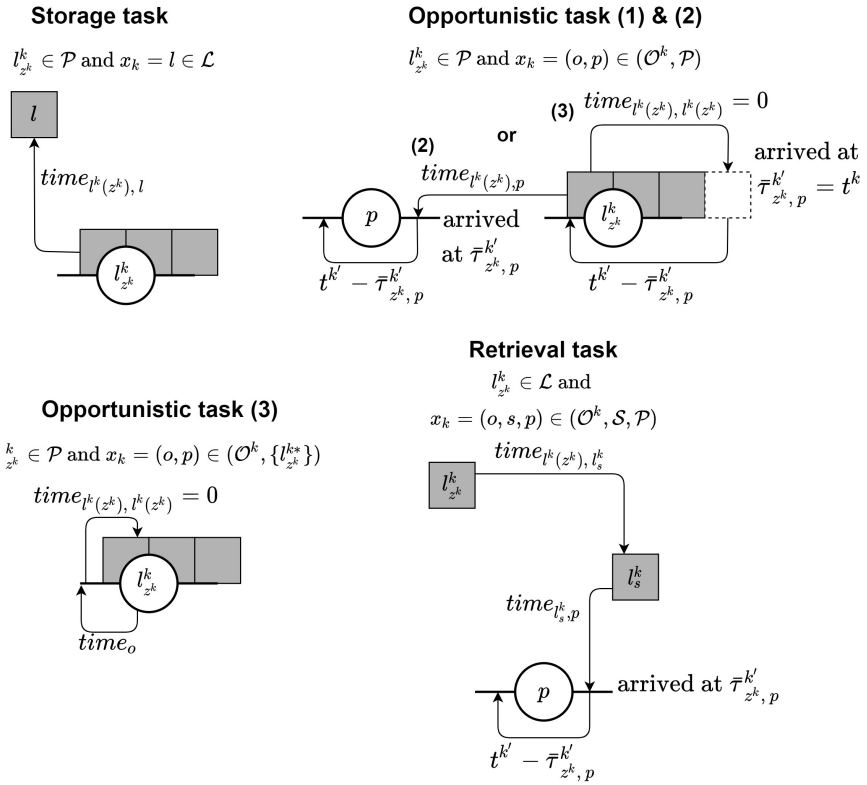


Figure 3. Illustration of the cost components in the complete cycle time



Figure 4. Simulation study - storage area

List of Figures

Figure 1 Decision tree ..... page 25

Figure 2 Possible decisions when a robot is located at a picking station ..... page 25

Figure 3 Illustration of the cost components in the complete cycle time ..... page 26

Figure 4 Simulation study - storage area ..... page 26