

WORKING PAPER SERIES

Robotized sorting systems: Large-scale scheduling under real-time conditions with limited lookahead

Nils Boysen/Stefan Schwerdfeger/Marlin Ulmer

Working Paper No. 5/2022



OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

FACULTY OF ECONOMICS
AND MANAGEMENT

Impressum (§ 5 TMG)

Herausgeber:

Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Der Dekan

Verantwortlich für diese Ausgabe:

Nils Boysen, Stefan Schwerdfeger, Marlin Ulmer
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Postfach 4120
39016 Magdeburg
Germany

<http://www.fww.ovgu.de/femm>

Bezug über den Herausgeber
ISSN 1615-4274

Robotized sorting systems: Large-scale scheduling under real-time conditions with limited lookahead

Nils Boysen, Stefan Schwerdfeger

Friedrich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Carl-Zeiß-Straße 3,
07743 Jena, Germany, nils.boysen@uni-jena.de, stefan.schwerdfeger@uni-jena.de, www.om.uni-jena.de

Marlin Ulmer

Otto von Guericke Universität Magdeburg, Management Science, Faculty of Economics and Management, Postfach 4120,
39016 Magdeburg, Germany, marlin.ulmer@ovgu.de, www.ms.ovgu.de

June 2, 2022

A major drawback of most automated warehousing solutions is that fixedly installed hardware makes them inflexible and hardly scalable. In the recent years, numerous robotized warehousing solutions have been innovated, which are more adaptable to varying capacity situations. In this paper, we consider robotized sorting systems where autonomous mobile robots load individual pieces of stock keeping units (SKUs) at a loading station, drive to the collection points temporarily associated with the orders demanding the pieces, and autonomously release them, e.g., by tilting a tray mounted on top of each robot. In these systems, a huge number of products approach the loading station with an interarrival time of very few seconds, so that we face a very challenging scheduling environment in which the following operational decisions must be taken in real time: First, since pieces of the same SKU are interchangeable among orders with a demand for this specific SKU, we have to assign pieces to suitable orders. Furthermore, each order has to be temporarily assigned to a collection point. Finally, we have to match robots and transport jobs, where pieces have to be delivered between loading station and selected collection points. These interdependent decisions become even more involved, since we (typically) do not possess complete knowledge on the arrival sequence but have merely a restricted lookahead of the next approaching products. In this paper, we show that even in such a fierce environment sophisticated optimization, based on a novel two-step multiple-scenario approach applied under real-time conditions, can be a serviceable tool to significantly improve the sortation throughput.

Key words: Warehousing; Robotized sorting systems; Dynamic scheduling; Multiple-scenario approach

1 Introduction

E-Commerce is booming. Customers order more online than ever, and retailers are in fierce competition to streamline their fulfillment processes. Fast, reliable, and at the same time low-cost processes have become a key success factor for many online and omni-channel retailers (Schiffer et al. 2022). An obvious and widely applied lever to reach these targets is automation. The first crane-operated high-bay racks have been installed in the 1950s (Roodbergen and Vis 2009). Since then, a lot of automated solutions for all elementary functions of facility logistics have been introduced and realized in warehouses and distribution centers all over the globe (for a survey, see

Azadeh et al. 2019). Most automated solutions, however, are based on fixedly installed hardware, such as cranes, lifts, industrial robots, and conveyors, which make these systems inflexible and hardly scalable on short notice. Automated solutions thus have to be generously dimensioned, such that they can successfully handle peak-loads, e.g., during end-of-season sales, Singles' Day, or Black Friday; whereas during the off-peak season costly capacity remains idle (Boysen et al. 2017). To avoid this, the next generation of automated warehousing solutions is rather based on more flexible autonomous mobile robots. Among the most prominent is certainly the shelf-lifting mobile robots system, for instance, applied by online retailer Amazon (Cezik et al. 2022). In this system, mobile robots drive directly under racks, lift them, and transport them toward stationary pickers (e.g., see Wang et al. 2022). The shelf-lifting mobile robots system is dedicated to the product retrieval function, whereas other robotized systems, e.g., Autostore, rather support the storage function of warehousing (see Azadeh et al. 2019). In this paper, we focus on another elementary function of facility logistics: sorting. In the warehousing context, sorting is required to separate customer orders that are picked under a batching and/or zoning policy. Both are frequently applied to improve picking performance (De Koster et al. 2007). Traditional sorting systems, such as tilt-tray, sliding shoe, or cross-belt sorters (for a survey, see Boysen et al. 2019a), apply fixedly installed conveyors to transport products along a given number of sortation lanes into which passing products are redirected. These systems are inflexible, occupy a lot of space on the shop floor, and are hardly scalable once erected.



Figure 1 Robotized sorting systems t-Sort of Tompkins Robotics (left; source: Tompkins Robotics) and Ranger Mobile Sort of Grey Orange (right; source: Grey Orange).

In this paper, we treat the next-generation solution for order consolidation: robotized sorting systems. This system is based on autonomous mobile robots, which take over picked pieces of stock keeping units (SKUs) at a loading station, transport them toward a collection point where the order of the respective piece is collected, and release them into a bin, container, or scuttle in the floor. The latter is either achieved by tilting a tray mounted on top of each robot (see Figure 1,

left) or by initiating a conveying mechanism (see Figure 1, right). By adding further robots and collection points (or removing them), a robotized sorting system is quickly adaptable to different capacity situations. Furthermore, the layout is very flexible, so that it can easily be added to existing facilities with restricted or winding shop floor. In the following section, we elaborate on the decisions to be taken when setting up and operating a robotized sorting system.

1.1 Decision problems and literature review

Sorting pieces, e.g., individual products, parcels, or bins, according to some criteria is an elementary function of facility logistics, which plays an important role in most warehouses and distribution centers. Sorting is, for instance, inevitable, if batching and/or zoning is applied in the picking area in order to improve picking performance. Picking orders from a large assortment is a laborious task, which either causes plenty of unproductive picker travel in picker-to-parts systems or excessively utilizes scarce material handling capacity in automated parts-to-picker systems (see [De Koster et al. 2007](#)). To avoid this, batching, i.e., the joint retrieval of multiple pieces of the same SKU aggregated over a batch of orders ([Boysen et al. 2019b](#)), and/or zoning, i.e., a parallel order picking in multiple areas ([Boysen et al. 2018](#)), are widely applied. This, however, requires that the stream of picked pieces arriving in the consolidation area has to be sorted according to customer orders. This and further applications of sorting in the warehousing domain are discussed by [De Koster et al. \(2007\)](#), [Boysen et al. \(2019b\)](#), and decision problems related to traditional (conveyor-based) sortation technology are treated by [Boysen et al. \(2019a\)](#). Robotized sorting systems, however, are among the latest innovations in the warehousing domain and have thus only rarely been treated by the operations research community yet. We elaborate on the most important decision problems related to these systems in the following.

System setup and layout planning: The main decision tasks to be addressed when setting up a robotized sorting system (or re-structuring an existing one) are decisions on the number and arrangement of collection points as well as loading stations, and the sizing of the robot fleet. The investment costs for these entities have to be weighed off against the required sortation performance, which heavily depends on their specific arrangement on the shop floor. Collection points can, for instance, be arranged in a grid structure or along the perimeter of an empty inner area reserved for robot travel. Driving lanes for robots can be uni- or bi-directional. The former leads to a more compact layout, but tends to prolong robot drives until a lane leading into the intended direction is reached. Once a specific layout is available, the resulting sortation performance is typically anticipated by queuing models ([Zou et al. 2021](#)), simulation, or analytical models ([Shi et al. 2022](#)). In this paper, we presuppose a given sorter layout and rather focus on the operational decision problems that are described in the following.

Scheduling: Once a specific system layout is realized and picked pieces approach a loading station on a conveyor, the operational scheduling requires three basic decisions:

(i) Piece-to-order (P2O) assignment: Real-world order data is typically heavily skewed (De Koster et al. 2007) and especially fast-moving SKUs are regularly demanded by many orders. Hence, the pieces of these SKUs that approach the loading station have to be assigned to orders. When taking this decision, it seems advantageous to reduce the order spread, i.e., the distance (counted in pieces) between the first and the last piece assigned to each order. The shorter the order spread, the quicker orders are completed, so that the collection points intermediately assigned to them are released earlier and can be reassigned to successive orders. In this way, sortation performance can be improved. For traditional conveyor-based sorting systems, the P2O assignment has been treated by Boysen et al. (2021). These systems, however, are either based on loop- or linear-shaped conveyors, so that a completely different problem structure arises.

(ii) Order-to-collection point (O2CP) assignment: Customer orders have to be assigned to collection points, where the pieces dedicated to the respective order are collected. Such an assignment, however, is only temporary and the collection point is relieved for re-assignment once the respective order is complete and forwarded to the packing and shipping area. Advantageous collection points close to the loading station should especially be assigned to orders, which require many pieces. Furthermore, orders not excessively spread over the arrival sequence should be preferred, because then advantageous collection points are not blocked over an excessively long time span. Thus, good O2CP assignments can relieve the scarce robot capacity and improve sortation performance. The assignment of orders to collection points has already been investigated for traditional (conveyor-based) sorting systems (see Meller 1997, Johnson 1997, Khir et al. 2021). For a survey, see Boysen et al. (2019a). Generally, however, the O2CP assignment has no major impact on sortation performance in traditional conveyor-based loop sorting systems. In these systems, the conveyor passes along any collection point anyway and released conveyor capacity, obtained by a piece being tilted into a collection point, cannot be reused during the current circulation; the empty tray moves onward and cannot be refilled before reaching the loading station of the next cycle. Freed capacity of a mobile robot, having delivered its current piece toward a close-by collection point, instead, can be reused earlier. The robot can prematurely return to the loading station, take over an additional piece, and thus improve sortation performance. Consequently, the O2CP point assignment is much more mission-critical for robotized sorting systems.

(iii) Piece-to-robot (P2R) assignment and robot scheduling: Once the designated orders and collection points of the pieces arriving at the loading station are determined, the transport jobs are readily defined. They have to be assigned to the fleet of mobile robots, whose paths along the shop floor have to be determined. These paths have to be coordinated among all robots, such that

robots blocking each other in deadlock situations are ruled out and requests are completed as soon as possible. Surveys on the research dedicated to these control issues are, for instance, provided by Vis (2006) and Carlo et al. (2014).

It can be concluded that we are the first to consider these three decisions jointly for a robotized sorting system. This leads us to the contributions of this paper.

1.2 Contributions and paper structure

In a robotized sorting system, making the operational scheduling decisions, namely, the P2O, O2CP, and the P2R assignments, is especially challenging: First, the typical interarrival time of products at the loading station is just a few seconds (see Zou et al. 2021). Scheduling algorithms have just these few seconds before the next product arrives at the loading station, and hence decisions must be taken in a most-demanding real-time environment. Second, a typical robotized sorting system provides merely a limited lookahead of the next pieces that approach the loading station. Beyond the lookahead, which is determined by the position of the identification device, typically a barcode scanner that registers the products on the conveyor, the approaching products remain unknown. It is only known that the remaining products of the current wave of orders will arrive but not their exact arrival sequence. Note that a wave denotes a subset of orders that are jointly picked and forwarded from the picking area to the sortation area. Here, the orders are sorted, and only then the next wave is released (Boysen et al. 2021). Third, waves in large e-commerce warehouses consist of hundreds of orders and even more pieces, so that also the amount of input data poses a big computational challenge. Finally, we face a holistic scheduling task, where three interdependent decisions are to be taken simultaneously. Note that two of these decisions in isolation are shown to be strongly \mathcal{NP} -hard. Thus, we face a most challenging and inhospitable scheduling environment, for which we derive the following contributions:

The first contribution of this paper is a suitable, runtime-efficient multiple-scenario approach that samples potential arrival sequences beyond the lookahead. This approach, based on a novel two-step scenario generation approach, is shown to fulfill our above challenges. To achieve this, we start with a thorough analysis of the deterministic problem setting, which presupposes perfect information on the complete arrival sequence. We formulate the holistic decision problem that joins our three operational decisions and provide a thorough complexity analysis including all subproblems. This allows us to derive a suitable decomposition heuristic that solves the deterministic problem even for hundreds of orders in fractions of a second with good solution performance. This decomposition heuristic is the workhorse for our multiple-scenario approach, which samples the arrival sequences beyond the lookahead, if only partial knowledge on the arrival sequence is available. Our multiple-scenario approach is shown to almost reach the solution quality of the theoretical benchmark based on perfect information and to clearly outperform rule-based approaches.

Our second contribution is the application of our algorithms to real-world order data, from which we derive important managerial insights. For instance, we show that a clever scheduling algorithm ensures that most orders are processed at near-by collection points in the hot zone close to the loading station. This implicates that the layout of the system and the arrangement of collection points on the shop floor is less of an issue, as long as we can ensure a compact hot zone containing a sufficient number of collection points. Furthermore, we thoroughly explore the impact of the wave sizes and the amount of lookahead on throughput performance.

The remainder of the paper is structured as follows. We start our analysis in Section 2, where we presuppose perfect knowledge on the complete arrival sequence. We formulate the holistic problem as a mixed-integer program, analyze computational complexity and derive a heuristic decomposition approach. The latter is the workhorse in our multiple-scenario approach, that addresses the case of limited lookahead on the arrival sequence in Section 3. Then, in Sections 4 and 5, we investigate the computational performance of our solution methods and explore managerial issues, respectively. Finally, Section 6 concludes the paper.

2 Complete information

This section presupposes perfect information. We assume that the complete arrival sequence of pieces approaching the loading station is known with certainty. Note that large e-commerce warehouses employ conveyor systems with a total length of several kilometers (Boysen et al. 2019a). Hence, given a proper placement of the identification device along these conveyors, this prerequisite may actually be fulfilled for some warehouses. We, first, formulate a mixed integer programming model for the holistic problem setting that unifies all three decisions (see Section 2.1). Then, we decompose the problem and consider each decision task individually (see Section 2.2). In each case, this includes a thorough analysis of computational complexity and the introduction of an efficient heuristic approach.

2.1 Mixed integer programming model for the holistic problem

Our holistic operational robotized sorting (HORS) problem has to take the following three decisions: (i) Each piece σ_t that arrives in $T = \{1, \dots, |T|\}$ subsequent slots at the loading station refers to a SKU $s \in S$ and has to be assigned to an order $j \in O$ that demands the specific SKU. The first and the last piece of slot set T assigned to a specific order determines this order's slot interval (also denoted *order spread*) during which it occupies its associated collection point. (ii) Each order $j \in O$ has to be assigned to a collection point $i \in P$, where two orders j and j' can only be assigned the same collection point if they have non-overlapping order spreads. (iii) Finally, each piece σ_t has to be transported toward the collection point $i \in P$ its designated order $j \in O$ got assigned to. The

resulting transport job has to be executed by a robot r of given robot fleet R with sufficient time between subsequent jobs.

In this decision context, HORS seeks to minimize the makespan, which is given by the point in time at which the last piece of the arrival sequence is removed by its dedicated robot from the loading station. Pieces that have arrived at the foremost position of the loading station and cannot instantaneously be picked up by a robot, block all subsequent pieces. Hence, any pick up delay deteriorates the overall throughput performance and should be avoided. Thus, minimizing the makespan also minimizes the sum of pick up delays, shortens the processing time for the current wave of orders, and allows to start subsequent waves earlier.

O	set of orders (indices j and j')
S	set of SKUs (index s)
S_j	set of SKUs required by order j (index s)
P	set of collection points (index i)
T	set of arrival slots, in which pieces subsequently arrive at the loading station, with $T = \{1, \dots, T \}$ and $ T = \sum_{j \in O} \sum_{s \in S_j} n_{j,s}$ (indices t and t')
R	set of autonomous robots (index r)
$n_{j,s}$	number of pieces of SKU $s \in S_j$ that are demanded by order j
σ_t	SKU of the t -th piece of the arrival sequence
Δ_i	robot driving time from the loading station to collection point i and back
$\bar{\Delta}$	big integer, i.e., $\bar{\Delta} = \max_{i \in P} \{\Delta_i\}$
$x_{j,t}$	binary variable: 1, if the t -th piece of the arrival sequence is assigned to order j ; 0, otherwise
$a_{j,t}$	binary variable: 1, if order j is active during the arrival of the t -th piece of the arrival sequence; 0, otherwise
$y_{j,i}$	binary variable: 1, if order j is assigned to collection point i ; 0, otherwise
$z_{r,t}$	binary variable: 1, if robot r processes the t -th piece of the arrival sequence; 0, otherwise
p_t	continuous variable: processing time of the t -th piece of the arrival sequence
Z_t	continuous variable: handover time of the t -th piece of the arrival sequence

Table 1 Notation for HORS

HORS can be expressed by a mixed integer program (MIP), which we dub HORS-MIP. The notation of this model is summarized in Table 1 and it consists of objective function (1) subject to constraints (2) to (14).

$$\text{HORS-MIP: Minimize } F(x, a, y, z, p, Z) = Z_{|T|} \quad (1)$$

subject to:

$$\sum_{\substack{j \in O: \\ \sigma_t \in S_j}} x_{j,t} = 1 \quad \forall t \in T \quad (2)$$

$$\sum_{\substack{t \in T: \\ \sigma_t = s}} x_{j,t} = n_{j,s} \quad \forall j \in O; s \in S_j \quad (3)$$

$$z_{r,r} = 1 \quad \forall r \in R \quad (4)$$

$$\sum_{r \in R} z_{r,t} = 1 \quad \forall t \in T \setminus R \quad (5)$$

$$\sum_{i \in P} y_{j,i} = 1 \quad \forall j \in O \quad (6)$$

$$a_{j,t} \geq x_{j,t} \quad \forall j \in O; t \in T; \sigma_t \in S_j \quad (7)$$

$$T \cdot (1 - a_{j,t} + a_{j,t+1}) \geq \sum_{t'=t+2}^T a_{j,t'} \quad \forall j \in O; t \in T \setminus \{|T| - 1, |T|\} \quad (8)$$

$$a_{j,t} + a_{j',t} + y_{j,i} + y_{j',i} \leq 3 \quad \forall j, j' \in O; j \neq j'; i \in P; t \in T \quad (9)$$

$$Z_r = r \quad \forall r \in R \quad (10)$$

$$Z_t \geq Z_{t-1} + 1 \quad \forall t \in T \setminus R \quad (11)$$

$$Z_t + \bar{\Delta} \cdot (2 - z_{r,t} - z_{r,t'}) \geq Z_{t'} + p_{t'} \quad \forall r \in R; t = 2, \dots, T; t' = \max\{1, t - \bar{\Delta} + 1\}, \dots, t - 1 \quad (12)$$

$$\Delta_i \cdot (x_{j,t} + y_{j,i} - 1) \leq p_t \quad \forall j \in O; i \in P; t \in T; \sigma_t \in S_j \quad (13)$$

$$x_{j,t}, a_{j,t}, y_{j,i}, z_{r,t} \in \{0, 1\} \quad \forall j \in O; i \in P; r \in R; t \in T \quad (14)$$

Objective function (1) minimizes the point in time when the last piece of the arrival sequence is picked up by its designated robot. Equalities (2) ensure that each piece is assigned to an order. The fulfillment of each order's demand is ensured by (3). Moreover, each piece is assigned to a robot by (4) and (5) and each order to a collection point (6). Constraints (7) enforce that an order can only receive a piece, if it is currently active at some collection point. Furthermore, constraints (8) ensure that orders remain at an assigned collection point during an uninterrupted interval of subsequent slots. Constraints (9) take care that each collection point can be assigned to at most one order during a slot. The handover time of the first $r = 1, \dots, |R|$ pieces is set to directly subsequent and undelayed slots each serviced by one robot of the fleet. After these first $|R|$ pieces, the next piece can only be picked up one time unit after its predecessor (11), if no robot waiting time occurs. On top of that, inequalities (12) ensure that the handover time of a piece may be further delayed, if the designated robot requires more driving time from the loading station to the collection point of its previous piece and back. The driving time of the current piece is set by (13). Finally, (14) defines the domains of the binary variables. Note that (4) and (10) are valid, because the first-come-first-served (FCFS) policy provides an optimal robot assignment (see Section 2.2.3). Further note that $Z_t > Z_{t'}$ for $t > t'$ holds, so that $\bar{\Delta}$ provides a sufficiently large big integer within (12). Finally, since $p_t \leq \bar{\Delta}$ holds, it is sufficient to consider at most $\bar{\Delta}$ time steps, i.e., $t' = \max\{1, t - \bar{\Delta} + 1\}$.

Regarding the computational complexity of HORS, the following sections will show that already the P2O and the O2CP assignment (in isolation) constitute strongly \mathcal{NP} -hard decision tasks. Since a reduction of our holistic problem setting to each of these subproblems is readily at hand, we claim strong \mathcal{NP} -hardness without a formal proof.

2.2 A heuristic decomposition approach

Unfortunately, our computational study will show that we cannot expect an off-the-shelf solver to successfully solve our holistic MIP in a challenging real-world setup, where waves are large and not

much decision time is at hand. Consequently, this section elaborates a decomposition approach that makes all three decisions subsequently in the following order: (i) piece-to-order assignment (P2O), (ii) order-to-collection point assignment (O2CP), and (iii) piece-to-robot assignment (P2R). Each of these subproblems (and their solution) is dedicated a separate section in the following.

2.2.1 Piece-to-order assignment (P2O)

Problem definition: The basic input data for P2O is a set of orders $O = \{o_1, o_2, \dots, o_n\}$ each defining the pieces of SKUs demanded by the respective order. Note that orders are multisets, because an order can demand multiple pieces of a SKU. Furthermore, we have the arrival sequence σ of pieces approaching the loading station. Note that order set O and arrival sequence σ are not independent of each other. σ constitutes a permutation of all individual pieces demanded by all orders in O . We seek an assignment defining the order each approaching piece of sequence σ is assigned to, such that each order receives all its demanded pieces and the sum of order spreads is minimized. Recall that the order spread of a specific order reaches from the foremost sequence position of a piece assigned to this order within σ up to the last position. We assume that the order spread is a good surrogate objective for this decision task, because of the following reasoning: If an order spread reduces, the time interval during which the assigned collection points is blocked reduces too. Especially advantageous collection points close to the loading station can then quickly be reassigned to subsequent orders, which decreases the robots' travel effort.

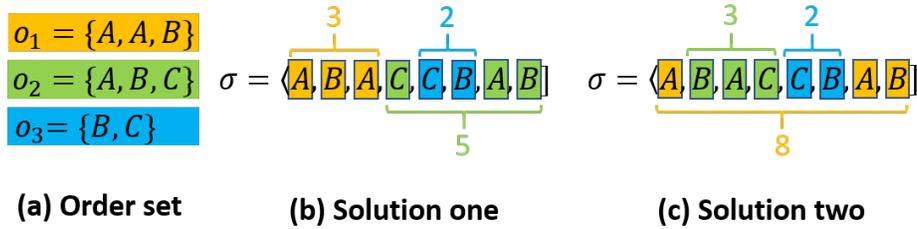


Figure 2 Example and two solutions for P2O

Example: The order set O with demands for three different SKUs A , B , and C and the loading sequence σ of our example are depicted in Figure 2. The assignment of pieces to orders of solution one leads to a sum of order spreads of 10, whereas solution two leads to an objective value of 13.

MIP: Applying the notation of Tables 1 and 2, P2O can be formulated as a MIP (dubbed P2O-MIP) as follows:

$$\text{P2O-MIP: Minimize } F_{P2O}(x, A) = |O| + \sum_{j \in O} (\bar{A}_j - \underline{A}_j) \quad (15)$$

$\underline{t}_j, \bar{t}_j$	Upper and lower bound on \underline{A}_j and \bar{A}_j , respectively, with $\underline{t}_j = \max\{t \in T: S_j \subseteq \{\sigma_t, \dots, \sigma_{ T }\}\}$ and $\bar{t}_j = \min\{t \in T: S_j \subseteq \{1, \dots, \sigma_t\}\}$
$\underline{A}_j/\bar{A}_j$	Continuous variable: first/last slot dedicated to order j

Table 2 Additional notation for subproblem P2O

subject to:

$$\sum_{\substack{j \in O: \\ \sigma_t \in S_j}} x_{j,t} = 1 \quad \forall t \in T \quad (16)$$

$$\sum_{\substack{t \in T: \\ \sigma_t = s}} x_{j,t} = n_{j,s} \quad \forall j \in O; s \in S_j \quad (17)$$

$$\underline{A}_j \leq t \cdot x_{j,t} + \underline{t}_j \cdot (1 - x_{j,t}) \quad \forall j \in O; t = 1, \dots, \underline{t}_j; \sigma_t \in S_j \quad (18)$$

$$\bar{A}_j \geq t \cdot x_{j,t} \quad \forall j \in O; t = \bar{t}_j, \dots, T; \sigma_t \in S_j \quad (19)$$

$$x_{j,t} \in \{0, 1\} \quad \forall j \in O; t \in T; \sigma_t \in S_j \quad (20)$$

Objective function (15) minimizes the sum of order spreads, each of which is defined by the first (defined by (18)) and last (defined by (19)) sequence position within σ that is assigned to the respective order. Constraints (16), (17), and (20) directly correspond to (2), (3), and (14) of our previous HORS-MIP. Note that the first (last) piece must (can) be assigned to order j at the latest (earliest) at \underline{t}_j (\bar{t}_j). Unfortunately, solving P2O turns out as a complex matter.

THEOREM 1. *P2O is strongly \mathcal{NP} -hard.*

Proof of Theorem 1: See online appendix A.

Heuristic: Given this complexity status, we suggest a priority-rule based heuristic that provides solutions extremely fast. This heuristic proceeds as follows. First, all orders are sorted in a non-increasing manner according to priority value $(\sum_{s \in S_j} n_{j,s})^2 / \hat{A}_j$. Here, \hat{A}_j represents the minimum order spread order j can receive by an assignment of the demanded pieces within arrival sequence σ , if all orders are planned independently. Thus, larger orders that require many robot drives but are nonetheless likely to not block a collection point for an excessive time period receive higher priority. According to the resulting order sequence, one order after the other is considered in a greedy manner and receives all demanded pieces among those still available, such that the resulting order spread is minimized. Ties are broken according to the minimum slot number of an order spread's final slot. The selected pieces are blocked for all subsequent orders. Then, we proceed with the next order, until all orders are completed and all pieces have been assigned. This approach has a runtime complexity of $\mathcal{O}(|O| \cdot |T|)$. Note that in preliminary tests, we have also evaluated further priority rules, i.e., largest-order-first, smallest-order-first, random-order-next, or balanced-order-first (see online appendix C). However, all these competitors were clearly outperformed by our priority rule, so that we exclusively apply this approach in the remainder of the paper.

2.2.2 Order-to-collection point assignment (O2CP)

Problem definition: Once the P2O assignment is given for the complete arrival sequence, the order spreads of all orders of set $O = \{o_1, o_2, \dots, o_n\}$ are known. Furthermore, we are given set P of collection points where orders are collected. A collection point can be assigned multiple orders, but not during overlapping time intervals given by the order spreads. In this setting, we seek a feasible assignment of orders to collection points, such that each order is assigned to exactly one collection point and no order spreads of any pair of orders assigned the same collection point overlap. Among all feasible assignments, we seek one which minimizes the total travel effort for the robots. Depending on the number of pieces each order receives and the position of the collection point toward the loading station, each assignment of an order $j \in O$ to a collection point $i \in P$ is assigned a weight w_{ij} defining the total robot travel when delivering the respective pieces toward the collection point and moving back empty to the loading station. Given these weights, we seek a feasible assignment that minimizes the sum of weights over all realized O2CP assignments.

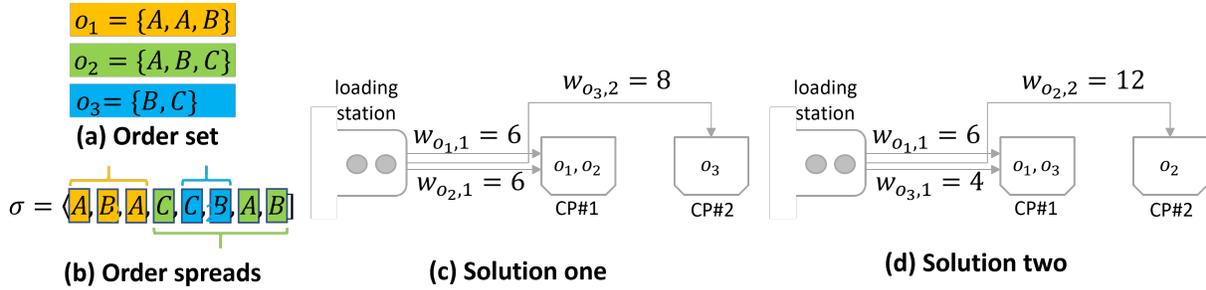


Figure 3 Example instance for the order-to-collection point assignment problem

Example: We assume that solution one of our previous example (see Figure 2) has been realized. The resulting input data is repeated in Figure 3(a) and (b). Within solution one, orders o_1 (yellow) and o_2 (green) are successively assigned to collection point 1 (denoted CP#1), which is possible because both order spreads do not overlap. Both orders receive three pieces, CP#1 is one distance unit from the loading station, and the assigned robot has to drive back and forth, so that weights $w_{o_1,1} = w_{o_2,1} = 3 \cdot 1 \cdot 2 = 6$ are realized. Order two receives two pieces to be delivered back and forth CP#2 via two distance units, which results in $w_{o_3,2} = 2 \cdot 2 \cdot 2 = 8$ and a total objective value of solution one of 20. Alternative solution two, instead, results in an objective value of 22.

MIP: Applying the notation of Tables 1 and 2, problem O2CP can be formulated as a MIP. The resulting O2CP-MIP consists of objective function (21) subject to constraints (22) to (24).

$$\text{O2CP-MIP: Minimize } F_{\text{O2CP}}(y) = \sum_{j \in O, i \in P} \Delta_i \cdot |S_j| \cdot y_{j,i} \quad (21)$$

subject to:

$$\sum_{i \in P} y_{j,i} = 1 \quad \forall j \in O \quad (22)$$

$$y_{j,i} + y_{j',i} \leq 1 \quad \forall j < j' \in O; i \in P; t = \max\{\underline{A}_j, \underline{A}_{j'}\}, \dots, \min\{\bar{A}_j, \bar{A}_{j'}\} \quad (23)$$

$$y_{j,i} \in \{0, 1\} \quad \forall j \in O; i \in P \quad (24)$$

Objective function (21) minimizes total travel effort for the robots, such that each order is assigned to exactly one collection point (22) and no order spreads of any pair of orders assigned the same collection point overlap (23). Note that (22), (23), and (24) (directly) correspond to (6),(9), and (14) of HORS-MIP. Further note that \underline{A}_j and \bar{A}_j are no longer variables. Instead, they are parameters within O2CP that have been determined by solving previous subproblem P2O. Unfortunately, solving this subproblem, too, turns out as a complex matter.

THEOREM 2. *O2CP is strongly NP-hard.*

Proof of Theorem 2: See online appendix B.

Heuristic: Thus, O2CP turns out as a challenging optimization problem whose optimal solution can be too costly, especially under real-time conditions. Therefore, we again employ a straightforward heuristic based on a priority rule. This rule is based on the same reasoning as before: Orders that demand many pieces but enable short blocking times of collection points should be prioritized. The main difference to our P2O, however, is that the order spreads are already known for each order. Hence, order j 's priority values amounts to $(\sum_{s \in S_j} n_{j,s})^2 / (\bar{A}_j - \underline{A}_j + 1)$. In decreasing order according to this priority value, one order after the other is considered and assigned to the closest collection point that can handle the order without conflict. Once assigned, the selected collection point is blocked during the respective order spread. Then, we proceed with the next order, until finally all orders are assigned to collection points. In preliminary tests, we compared this approach with a surrogate problem that minimizes the number of utilized collection points instead. This surrogate problem is efficiently solvable in polynomial time but is also outperformed by the straightforward heuristic described above. The surrogate problem and the computational benchmark are detailed in online appendix C.

2.2.3 Piece-to-robot assignment (P2R)

Given both previous assignment decisions, we have to assign a set J of transport jobs to robots R , such that the pickup of the final piece of the arrival sequence is minimized. A transport job $j \in J$ consists of a pickup of a piece at the loading station, its delivery to the collection point temporarily assigned to the current piece's order and back to the loading station. Given the previous assignment decisions, each transport job has a constant processing time p_j . Since each delayed pickup postpones

all subsequent pieces, it is not hard to see that we should start each transport job as soon as possible. Since we have identical robots, the P2R decision task is solved to optimality by a simple first-come-first-served (FCFS) policy, where each piece is assigned the first available robot. This result is summarized by the following theorem.

THEOREM 3. *P2R is solvable to optimality by FCFS.*

This completes our heuristic decomposition approach, which is also the workhorse if we have no perfect information on the complete arrival sequence.

3 Limited lookahead

In most practical settings, the complete arrival sequence of pieces in a wave is not fully known but only the next L pieces that approach the loading station. We call these L pieces the (limited) lookahead. The typical reason for such a setup is an identification device that is placed only a short distance before the loading station. Especially, in e-commerce, however, waves can be so large that the conveyor capacity, even if the identification device is placed right at the start of the conveyor, is not large enough to house the complete wave. In this case, the robots already remove pieces from the loading station, while further pieces of the current wave are still placed onto the conveyor. Again, the result is a limited lookahead. Thus, instead of determining a single a-priori plan on how to assign pieces to orders and orders to collection points, decisions are made dynamically whenever a piece or order needs to be assigned. In the following, we will describe the sequential decision process and present how we approach the uncertainty by means of a multiple-scenario approach (MSA, [Bent and Van Hentenryck 2004](#)).

3.1 Sequential decision process

We define the stochastic dynamic problem as a sequential decision process, i.e., via states, decisions, stochastic information, and transition (see [Powell 2022](#)). For the purpose of presentation, we omit heavy notation and focus on the notation necessary for the description of the MSA.

Decision points and states. A decision point $t \in \{1, \dots, T\}$ occurs in case a robot and a SKU are available at the handover point. Thus, t does not indicate the time passed but the product currently available at the handover point. A state s_t contains information about the orders O_t , summarizing orders still to be opened and their required pieces as well as orders currently at a collection point, the corresponding collection point, and their remaining pieces. Therefore, O_t also covers information about the availability of collection points. A state further contains information about the availability of robots represented by a $|R|$ -dimensional vector of the time steps until each robot returns back to the loading station $\tau_t = (\tau_{1t}, \dots, \tau_{R|t})$. For example, a value of $\tau_{it} = 0$ indicates that robot i is currently idling at the handover point, a value $\tau_{it} = 3$ indicates that this robot will

be available again in three time steps, etc. Because the robots are identical, this vector can be ordered by earliest availability times, $\tau_{1t} \leq \dots \leq \tau_{|R|t}$. Since a state requires the availability of at least one robot, $\tau_{1t} = 0$. Finally, a state contains the sequence $(\sigma_t, \dots, \sigma_{t+L})$ of L known pieces on the conveyor with σ_t being the product at the loading station. Note that at the end of a wave when only a few pieces are missing, the set of known pieces may be smaller than lookahead horizon L .

Decisions. Decisions x_t are made about the assignment of piece σ_t at the loading station to an order at a collection point or the opening of a new order at an empty collection point, respectively. A decision is feasible if piece σ_t is a (missing) piece of the corresponding order and, in case a new order is opened, the corresponding collection point is free. The decision changes the availability time of the corresponding robot and leads therefore to a newly sorted vector τ_t^x . The cost $C(s_t, x_t)$ of a decision x_t are the handover time between the current and next piece, i.e., the maximum of the time steps the next robot is available at the loading station τ_{1t}^x and the next product is available at the handover point, i.e., 1 time step: $C(s_t, x_t) = \max\{\tau_{1t}^x, 1\}$.

Stochastic information and transition. The stochastic information reveals a new piece $\hat{\sigma}_{t+L+1}$ from the set of remaining pieces at the end of the lookahead horizon. The transition $\mathcal{T}(s_t, x_t, \hat{\sigma}_{t+L+1})$ leads to a new state s_{t+1} at the time the next robot and piece become available at the loading station, i.e., $\max\{\tau_{1t}^x, 1\}$ time steps later. The lookahead is updated with respect to the assigned piece and the piece revealed by the stochastic information, $(\sigma_{t+1}, \dots, \sigma_{t+L}, \hat{\sigma}_{t+L+1})$. The availability times of the robots are $\tau_{it+1} = \max\{0, \tau_{it}^x - C(s_t, x_t)\}$, i.e., all times are reduced by the time span between states s_t and s_{t+1} . Further, the set of orders O_{t+1} are updated as follows: If an order was completed by decision x_t , the corresponding order is removed from the set of orders and the corresponding collection point becomes available again. Else, the piece is removed from the set of remaining products for that order.

3.2 Multiple-scenario approach

Given partial knowledge about the approaching products, we propose a multiple-scenario approach (MSA). The MSA is a general concept especially tailored to sequential decision problems with complex decision spaces (e.g., Azi et al. 2012, Ghiani et al. 2012, Schilde et al. 2014, Zolfagharinia and Haughton 2016, Voccia et al. 2019, Song et al. 2020, Ausseil et al. 2022). We refer to Soeffker et al. (2022) for a recent survey. The general idea of an MSA is to generate a set of W scenarios (samples), to derive a solution for each (deterministic) scenario, and to find the best-fitting solution by means of a consensus function. In the following, we present how we adapt the MSA concept to our problem.

We focus our MSA on the P2O and O2CP assignment as the FCFS policy is optimal for the P2R assignment (see Section 2.2.3). However, in contrast to other work on MSA, we face a problem with

two very different parts of decision making, P2O and O2CP. As we show later, the conventional MSA procedure struggles when determining both at once. Thus, we propose an adapted, *two-step* version of the MSA. Instead of solving all scenarios and deriving the full decision at once, we split the decisions into two steps. We first determine the P2O assignment and, given that, decide on the O2CP assignment afterward. For both steps, the two-step MSA generates W samples of arrival sequences, $\tilde{\sigma}^1, \dots, \tilde{\sigma}^W$. These pieces complement the known pieces $(\sigma_t, \dots, \sigma_{t+L})$ of the lookahead, with $(\sigma_t, \dots, \sigma_{t+L}) \cup \tilde{\sigma}^i$ being the overall required pieces to finish all orders O_t and therefore the entire wave.

P2O: For each scenario $\tilde{\sigma}^i$, we obtain a deterministic problem as defined in Section 2, however, with a partial fixed solution with regard to the active orders in O_t and their occupied collection points. We can then solve the resulting deterministic problem as described in Section 2.2.1. Note that we do not have to run the whole procedure but can terminate as soon as piece σ_t was assigned to any order. We then can check to which order $o_t^i \sigma_t$ was assigned to in scenario i . To determine decision x_t , we rely on the consensus solution, i.e., we assign the piece to the order most frequently chosen over the scenarios.

O2CP: To this end, if in state s_t , piece σ_t is assigned to any open order in O_t , the corresponding collection point is already fixed. Otherwise, in case a new order is opened, we again generate scenarios (but with fixed P2O assignment of σ_t) to decide on the O2CP assignment by consensus solution in a fashion similar to the previous step. We use the same scenarios of arrival sequences, but fix the assignment of the current piece σ_t .

The algorithmic details of MSA are summarized in Algorithm 1, adapted from the general procedure in [Soeffker et al. \(2022\)](#). First, W deterministic scenarios \hat{s} as combinations of current state s_t and sampled arrival sequence $\tilde{\sigma}^i$ are generated with function $Scenario_P2O(s_t, \tilde{\sigma}^i)$. Each scenario \hat{s} is then solved with $DEC(\hat{s})$. The solution \hat{x}^i is a solution of the full deterministic problem. From this solution, the order is extracted for each scenario via $ExtractOrder(\hat{x}^i)$ and stored in \bar{O} . Then, the most common order O^* is determined via function $HighestFrequency(\cdot)$. Next, the scenarios are updated by fixing O^* with function $Scenario_O2CP(s_t, \tilde{\sigma}^i, O^*)$. After again solving the individual scenarios, the collection points are extracted and the best collection point P^* is determined via $ExtractPoint(\hat{x}^i)$ and $HighestFrequency(\cdot)$, respectively. The result of the MSA is the tuple (O^*, P^*) that is then implemented.

4 Computational performance

This section explores the computational performance of our solution methods. Specifically, we want to show that our algorithms deliver good solutions in the short decision times available. First, we describe the data instances our computational studies are based on (Section 4.1). Then, the results

Algorithm 1: Multiple-Scenario Approach

Input : State s_t , Sampled Piece Sequences $\{\tilde{\sigma}^1, \dots, \tilde{\sigma}^W\}$
Output : Open Order O^* at Collection Point P^*

- 1 $\bar{O} \leftarrow \emptyset$ // Set of Opened Orders
- 2 $\bar{P} \leftarrow \emptyset$ // Set of Opened Collection Points
- 4 // Generate Scenario Solutions
- 5 **for** $i=1, \dots, W$ **do**
- 6 $\hat{s} \leftarrow \text{Scenario_P2O}(s_t, \tilde{\sigma}^i)$ // Generate Scenario i
- 7 $\hat{x}^i \leftarrow \text{DEC}(\hat{s})$ // Solve Decision Model for Scenario i with DEC
- 8 $\bar{O} \leftarrow \bar{O} \cup \text{ExtractOrder}(\hat{x}^i)$ // Extract Opened Order Decision
- 9 **end**
- 10 $O^* \leftarrow \text{HighestFrequency}(\bar{O})$ // Most Frequent Order
- 12 // Generate Scenario Solutions
- 13 **for** $i=1, \dots, W$ **do**
- 14 $\hat{s} \leftarrow \text{Scenario_O2CP}(s_t, \tilde{\sigma}^i, O^*)$ // Generate Scenario i with Fixed Order
- 15 $O^* \hat{x}^i \leftarrow \text{DEC}(\hat{s})$ // Solve Decision Model for Scenario i with DEC
- 16 $\bar{P} \leftarrow \bar{P} \cup \text{ExtractPoint}(\hat{x}^i)$ // Extract Opened Collection Point
- 17 **end**
- 19 // Selection
- 20 $P^* \leftarrow \text{HighestFrequency}(\bar{P})$ // Most Frequent Collection Point
- 21 **return** (O^*, P^*)

of our computational performance tests are elaborated in Sections 4.2 and 4.3 that treat the cases of perfect knowledge and limited lookahead, respectively.

4.1 Instance generation

Our computational tests are performed on two types of order data. We apply generated data, mainly for performance testing, in order to have a controlled environment, where all influencing parameters can systematically be explored. Furthermore, we apply a real-world data set from a large e-commerce retailer in South Asia, mainly to investigate managerial issues on representative order data.

Our data generator receives the following data as its own input: $|S|$ (the number of SKUs), $|O|$ (the number of orders), $|P|$ (the number of collection points), and $|R|$ (the number of robots). Note that $|S|$ does not refer to the complete assortment stored in a warehouse but only to those SKUs demanded by the current wave of orders.

The main input data are the orders, which we (i) generate with an artificial generator and (ii) take over from a real-world data set.

(i) *Generated orders:* First, we determine whether an order $j \in O$ contains $|S_j| = 2$ or $|S_j| = 3$ SKUs with equal probability. Note that this reflects the rather small orders of e-commerce (Boysen et al. 2019b). Then, we follow two different generation regimes, *ABC* and *EQ*, to determine set S_j of

SKUs demanded by order $j \in O$. The former subdivides all SKUs of set O into 20% A-, 30% B-, and 50% C-products (De Koster et al. 2007). Then, with a chance of 80% an A-product, with 15% a B-product, and with 5% a C-product is chosen. Once the class is selected, the SKU to be assigned is drawn from the set of SKUs belonging to that class with equal probability. Under the EQ regime, we do not subdivide our SKUs into different classes but choose among all SKUs with equal probability. Finally, we generate $n_{j,s}$, the number of pieces of SKU $s \in S_j$ that are demanded by order $j \in O$, which is a uniformly distributed random number from interval $[1, 3]$.

(ii) *Real-world order data* is obtained from the dataset of Bansal et al. (2021), which is publicly available: <https://data.mendeley.com/datasets/ggbkd8ck3x/1>. The dataset contains 100,000 records of customer orders from a large e-commerce retailer in South Asia. However, to derive a meaningful order set for a robotized sorting system, we first remove all orders demanding only a single SKU. Naturally, single-SKU orders, about 95% of the dataset, require no sortation. To remove further orders with a structure rather atypical for e-commerce (e.g., there are a few orders that demand several hundreds of pieces of a single SKU and others that demand hundreds of different SKUs), we delete all orders demanding more than ten SKUs or more than ten pieces per SKU. The revised dataset contains 2,091 orders. From this set, we randomly choose $|O|$ orders (without replacement) to obtain the set of picking orders.

Given the SKUs S_j demanded by all orders $j \in O$ and the respective numbers of requested pieces $n_{j,s}$, we generate a random arrival sequence σ of these pieces. In this sequence, pieces arrive subsequently at an arrival time of $\lambda = 3$ seconds. Finally, we have to position the $|P|$ collection points and derive the resulting driving times for the robots. Since this paper is not primarily concerned with layout issues, for which we refer to Zou et al. (2021) instead, we presuppose the most basic layout where collection points are arranged along a linear corridor. The details on how this layout translates into robot driving times are provided in online appendix D. Note that our solution procedures are not bound to such a basic layout but can handle any layout and facultative distance matrices.

In our studies, we vary the number of orders $|O|$ and set $|P| = |O|$ to ensure feasibility. For our *generated* instances, we select $|O| \in \{5, 10, 20, 50, 100\}$ orders, $|S| = 20$ SKUs, and $|R| = 5$ robots. Note that these parameters are small enough, so that the majority of instances can still be handled by our off-the-shelf solver Gurobi solving HORS-MIP. For our *real-world* data, we generate instances with $|O| \in \{50(5), 100(10), 200(15), 500(30), 1000(40)\}$ orders, where the numbers in brackets define the number $|R|$ of robots applied for the respective amount of orders. Note that the number $|S|$ of SKUs is directly obtained from the real-world order data and thus not part of the input.

The data generator and all our procedures have been implemented in C# (using Microsoft Visual Studio 2022). The tests have been carried out on an x64-PC with an Intel Core i9-11900 2.50 GHz

Type	$ O $	Gurobi solving HORS-MIP					DEC(MIP,MIP)				DEC(HEU,HEU)			
		gap	gap _b	best	opt	sec	gap _b	best	opt	sec	gap _b	best	opt	sec
ABC	5	25.50	0.00	10	0	300.42	3.75	2	0	0.05	4.42	0	0	0.00
EQ	5	28.96	0.22	9	0	300.43	2.54	4	0	0.01	2.61	4	0	0.00
ABC	10	56.84	9.77	0	0	300.39	0.84	6	0	46.74	1.78	6	0	0.00
EQ	10	63.17	5.73	0	0	300.41	1.14	7	0	0.05	1.66	4	0	0.00
ABC	20	116.86	49.15	0	0	301.87	1.01	6	0	300.22	1.54	5	0	0.00
EQ	20	128.13	39.11	0	0	301.88	0.00	10	0	109.16	6.40	0	0	0.00
ABC	50	–	–	–	–	–	4.69	0	0	301.04	0.00	10	0	0.00
EQ	50	–	–	–	–	–	0.00	10	0	301.20	3.36	1	0	0.00
ABC	100	–	–	–	–	–	49.64	0	0	340.77	0.00	10	0	0.01
EQ	100	–	–	–	–	–	29.24	0	0	315.46	0.00	10	0	0.00
avg		69.91	17.33	1.90	0.00	300.90	9.28	4.50	0.00	171.47	2.18	5.00	0.00	0.00

Table 3 Performance results of Gurobi solving HORS-MIP and our decomposition approach DEC in two versions on generated instances with $|S|=20$ SKUs and $|R|=5$ robots

CPU and 64.0 GB of RAM. Our default solver is Gurobi (version 9.5.0), which receives a general time limit of 300 seconds.

4.2 Computational performance: Complete knowledge on the arrival sequence

First, we investigate the performance of our solution approaches that are based on perfect information and require knowledge on the complete arrival sequence. Specifically, we compare off-the-shelf solver Gurobi when solving our HORS-MIP of Section 2.1 with our decomposition approach (see Section 2.2), which we denote DEC. For the latter, we differentiate whether the steps P2O and O2CP are solved by MIP formulations P2O-MIP and O2CP-MIP, respectively. We dub this approach DEC(MIP,MIP). Alternatively, we also employ our decomposition approach in the version DEC(HEU,HEU), where both subproblems are solved with the introduced heuristics. The benchmarking results are summarized in Table 3, where we report on the average optimality gap in percent determined by Gurobi (column 'gap'), the average gap in percent to the best solution found among all competitors (column 'gap_b'), the number of best solutions found (column 'best'), the number of solutions proven to be optimal (column 'opt'), and the average CPU-seconds (column 'sec'). Note that our two gaps, i.e., 'gap' and 'gap_b', can only be reported if at least a single feasible solution has been found by the respective solution approach. If it failed in all ten repetitions per data setting, we mark this case by '–'. For each number of orders $|O|$, ten instances as described in Section 4.1 have been obtained, so that in total 100 generated instances constitute this testbed. The results summarized in Table 3 suggest the following conclusions:

- Default solver Gurobi solving HORS-MIP is bound to small instances. Only in case of $|O|=5$ orders, it delivers the best solutions in all instances except of one. Not even for these smallest instances, however, Gurobi was capable of verifying any optimal solutions. The performance significantly deteriorates for increasing instance sizes and, in case of $|O| \geq 50$ orders, it was not

capable of determining any feasible solution within the given time limit. Note that equipping Gurobi with more solution time up to an hour, did not significantly improve these results.

- Decomposing the problem and solving subproblems P2O and O2CP subsequently (but still with Gurobi), as it is done by DEC(MIP,MIP), grants only interim relief. At the latest for $|O| = 100$ orders, this approach too, consistently hits the timeout and returns solutions with significant gaps to the best solutions.
- Decomposition approach DEC(HEU,HEU), instead, delivers the best compromise among all competitors. It barely requires any solution time (i.e., rounding with two decimal places delivers a solution time of 0.00 in almost all cases) and the gaps are small, even in those cases, where we can expect the two competitors to delivery near-optimal solutions.

Note that further setups of our decomposition approach are discussed and evaluated in online appendix C. Our computational results make DEC(HEU,HEU) to our method of choice. In a robotized sorting systems, the pieces arrive in quick succession just seconds apart, so that fast solutions with acceptable gaps are required. This finding is even more true, if not the complete arrival sequence is known, but a dynamic environment with only limited lookahead leaves just the interarrival times for the decisions. The performance of our methods for such an environment is investigated next.

4.3 Computational performance: Limited lookahead

This section explores the computational performance, if we do not have complete knowledge on the arrival sequence but only for a limited lookahead. Specifically, we investigate the performance of our multiple-scenario approach (MSA), which is suitable if in each step we only know the next $L < |T|$ pieces that approach the loading station. MSA is benchmarked with a straightforward rule-based approach (dubbed RULE), like it is often applied in real-world warehouses (Boysen et al. 2021): First, we sort all orders according to the number of pieces they demand (ties are broken randomly) into a priority list. For each slot $t = 1, \dots, T$, the current product σ_t is assigned to the active order with highest priority. If an active order has received the last demanded product, this order is completed and the assigned collection point is released. If no active order demands current product σ_t , the non-active order with highest priority is activated, assigned to the closest empty collection point, and the current piece is delivered there. The decision on the assignment of robots to transport jobs is made according to FCFS.

To evaluate the performance of these two competitors, we compare them with a theoretical benchmark on what is achievable if the complete arrival sequence was know, for which we apply DEC (i.e., in setup DEC(HEU,HEU) see Section 4.2). Note that for MSA we set the number of scenarios to $W = 60$ and the lookahead to $L = 120$ within our study. The impact of both parameters

data	O	R	DEC			RULE				MSA			
			gap _b	best	sec	gap _b	best	sec	mspp	gap _b	best	sec	mspp
ABC	5	5	0.85	7	0.00	2.75	4	0.00	0.00	0.85	7	0.00	0.00
EQ	5	5	0.70	8	0.00	1.83	4	0.00	0.00	0.70	8	0.00	0.00
ABC	10	5	0.87	7	0.00	6.14	1	0.00	0.00	0.87	7	0.00	0.00
EQ	10	5	0.70	6	0.00	3.79	3	0.00	0.00	0.70	6	0.00	0.00
ABC	20	5	0.27	8	0.00	11.60	0	0.00	0.00	0.59	6	0.00	0.02
EQ	20	5	1.10	4	0.00	6.72	0	0.00	0.00	1.96	1	0.01	0.05
ABC	50	5	0.00	10	0.00	24.14	0	0.00	0.00	17.16	0	0.65	2.08
EQ	50	5	0.00	10	0.00	13.24	0	0.00	0.00	12.87	0	0.39	1.23
ABC	100	5	0.00	10	0.00	32.38	0	0.00	0.00	30.27	0	5.79	9.24
EQ	100	5	0.00	10	0.00	24.29	0	0.00	0.00	16.59	0	3.41	5.48
avg			0.45	8.00	0.00	12.69	1.20	0.00	0.00	8.26	3.50	1.03	1.81
RW	50	5	0.40	6	0.00	12.01	0	0.00	0.00	0.64	3	1.97	8.64
RW	100	10	0.03	9	0.00	16.83	0	0.00	0.00	2.33	1	9.21	21.61
RW	200	15	0.00	10	0.00	23.15	0	0.00	0.00	7.27	0	34.46	40.46
RW	500	30	0.00	10	0.01	39.92	0	0.00	0.00	18.36	0	225.69	104.95
RW	1000	40	0.00	10	0.04	51.88	0	0.01	0.00	27.16	0	1309.19	299.39
avg			0.09	9.00	0.01	28.76	0.00	0.00	0.00	11.15	0.80	316.10	95.01

Table 4 Computational performance in case of limited lookahead: Rule-based approach (RULE) and multiple-scenario approach (MSA) against the theoretical benchmark provided by DEC on generated data (with $|S|=20$ SKUs) and real-world data RW

will be investigated later on. These solution procedures are tested on our generated and real-world instances, so that the testbed consists of 15 instance classes in total. For each class, we run 10 realizations and calculate the average values. The results are summarized in Table 6, where we report on the previous performance measures defined in Section 4.2. The only additional performance measure is called 'mspp' and denotes the milliseconds per piece. Since both MSA and RULE have to make a decision for each new piece that arrives at the loading station, they only have the interarrival time between products for their decisions. Therefore, 'mspp' reports on the average solution time in milliseconds of MSA and RULE when executed for each new arrival. The results of Table 6 suggest the following findings:

The good news is that both our solution methods for a limited lookahead RULE and MSA suffice the challenging real-time requirements of a robotized sorting system. For RULE, which just makes a myopic choice for the current piece at the loading station, this was to be expected. MSA, however, samples arrival sequences beyond lookahead $L = 120$ in $W = 60$ scenarios and solves DEC in each case. Nonetheless, even in the most demanding case with $|O| = 1,000$ orders, which leads to arrival sequences of several thousand pieces, the average mspp is just 299.39 milliseconds; even the maximum never exceeds the interarrival time reported by Zou et al. (2021) of 3 seconds. Regarding the solution quality, we can observe varying gaps of RULE and MSA to theoretical benchmark DEC. These gaps will be investigated in more detail below, for the moment we can assume that the gaps are heavily impacted by the lookahead-to-total-sequence-length ratio. The

smaller this ratio, the larger the information disadvantage to the DEC benchmark and thus the larger the gaps. We can, furthermore, conclude that utilizing the additional lookahead information with MSA improves over the myopic online decisions of RULE. While this conclusion being true on average, it does not hold for every single data setting. The improvement of MSA compared to RULE decreases slightly with larger instances and with the data generated under the EQ regime. The reason is twofold. First, if all $|S| = 100$ SKUs are equally likely, there are not many SKUs that are demanded by multiple orders. Thus, the P2O assignment is less of an issue and MSA loses parts of its optimization potential in relation to RULE. Second, for larger instances and under the EQ regime, the future becomes more noisy and less predictable compared to smaller instances and the ABC scheme. Thus, anticipation becomes more challenging (compare [Ulmer et al. 2019](#)). We provide further analysis on this phenomenon in online appendix E, where we systematically control the predictability of order arrivals.

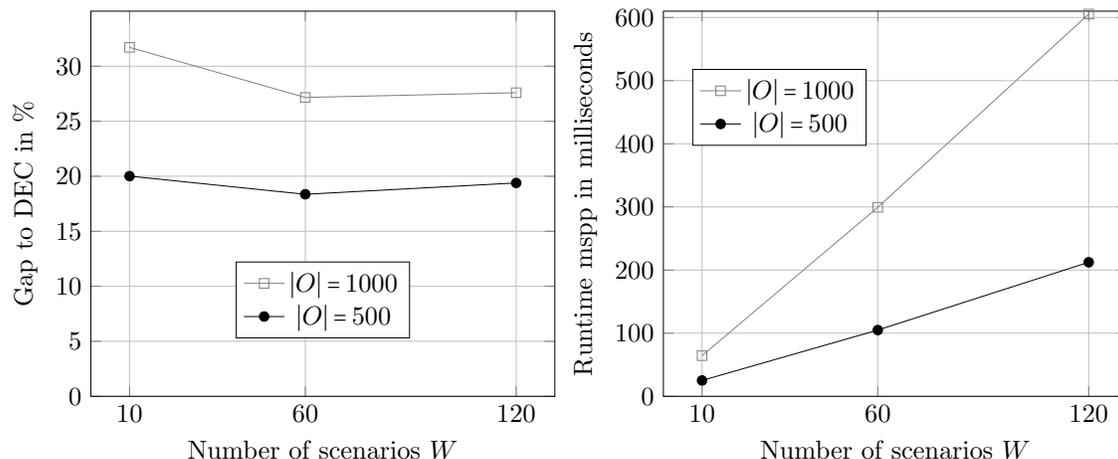


Figure 4 Impact of the number of scenarios W on solution quality (left) and time (right) of multiple-scenario approach MSA on the real-world dataset

The performance of MSA is impacted by the number of scenarios W that are generated and evaluated by DEC in each decision step. The impact of parameter W on the quality and solution time of MSA is depicted in Figure 4. These results show us mainly two things: First, the positive impact of additional scenarios quickly diminishes. Furthermore, as to be expected the number of scenarios W has a linear impact on the runtime. Each additional scenario adds a rather constant solution time per scenario that mainly depends on the length of the arrival sequence. Given these results, we apply $W = 60$ in all our further experiments with MSA.

In summary, we can conclude that our multiple-scenario approach MSA is fast enough to be applied even in the most demanding real-time environment with limited lookahead where a large number of orders are to be processed.

5 Analysis

This section applies our algorithms in order to gain managerial insights. In Section 5.1, we start with the deterministic case and explore the value of optimization (compared to simple rules of thumb). Then, in Section 5.2, we switch to the identification of further levers to improve the throughput performance, namely, the number of robots, the layout, and the wave size. Finally, in Section 5.3, we explore the value of anticipation in system setups with limited lookahead.

To quantify throughput performance in a compact and easily comprehensible way, we neither report solely on the objective values nor the gap to some best solution. Instead, we define throughput loss $\theta = \frac{Z_{|T|} - |T| \cdot \lambda}{|T| \cdot \lambda}$. This performance measure relates the total waiting time of all pieces of the current wave, i.e., $Z_{|T|} - |T| \cdot \lambda$, where λ is the interarrival time of products at the loading station, to minimum (undelayed) total processing time $|T| \cdot \lambda$. Thus, a throughput loss of $\theta = 0$ represents the best case, where no piece at the loading station is delayed and all products are instantaneously picked up by some robot upon arrival. A throughput loss of $\theta = 1$, instead, says that the total waiting time is just as long as the minimum processing time without delay, so that the overall processing time doubles. The experiments to derive the throughput loss for different system setups and solution methods are based on our real-world order data (see Section 4.1). Recall that the largest instances of this testbed contain $|O| = 1,000$ orders. This means that we have arrival sequences up to about $|T| = 4,400$ pieces in total, so that – at an interarrival time of $\lambda = 3$ seconds (Zou et al. 2021) – it takes up to about 3.5h to process all products if no delays occur.

5.1 Impact of optimization

First, we take a look at the value of optimization in an environment where we have complete knowledge on the arrival sequence. To do so, we benchmark the results of our heuristic decomposition approach DEC (see Section 2.2) with simple priority rule-based decision approach RULE (see Section 4.3) and an even more basic approach, which we dub the random approach (RND). RND proceeds exactly as RULE (see Section 4.3), it only selects randomly among the orders. Note that the application of very basic rules, analogously to RND, can sometimes be observed even for highly automated systems in warehouses, where there is no awareness for the additional value of optimization (Boysen et al. 2021). The resulting throughput loss for these three competitors, namely, DEC, RULE, and RND, are reported in Table 5. To explore the importance of the individual decisions that altogether constitute our holistic robotized sorting problem, namely, piece-to-order assignment P2O (see Section 2.2.1) and order-to-collection point assignment O2CP (see Section 2.2.2), we also report on the outcomes if either decision is taken by the heuristic applied within DEC or the basic rule of RND. The results of Table 5 suggest the following findings:

data	$ O $	$ R $	RND	RULE	HEU-RND	RND-HEU	DEC
RW	50	5	1.62	1.59	1.60	1.35	1.32
RW	100	10	0.91	1.59	1.60	0.66	0.59
RW	200	15	0.99	0.85	0.87	0.63	0.50
RW	500	30	1.12	0.80	0.89	0.64	0.29
RW	1000	40	1.96	1.35	1.36	1.24	0.55
avg			1.32	1.09	1.10	0.90	0.65

Table 5 Impact of optimization in case of complete knowledge on the arrival sequence: Throughput loss θ for most basic method RND, myopic decision rule RULE, two variants of combining the approaches (HEU) of DEC and RND, namely HEU-RND and RND-HEU, and decomposition approach DEC

- *Significant performance gains by optimization:* First, we can observe that sophisticated optimization, as provided by our DEC approach, promises a much smaller throughput loss than simple decision rule approaches RND and RULE. For our largest instances with $|O| = 1,000$ orders the improvements of DEC translate into an average reduction of the total processing time of 5 and 3 hours over RND and RULE, respectively.
- *O2CP decision more critical than P2O:* The average throughput loss of HEU-RND (i.e., the P2O decision is taken by the heuristic of Section 2.2.1 and O2CP by the basic rule of RND) is larger than that of RND-HEU (i.e., the P2O decision is taken by the basic rule of RND and O2CP by the heuristic of Section 2.2.2). Thus, we can conclude that a more sophisticated approach for the O2CP decision delivers better results, so that the assignment of orders to collection points seems the more critical decision for improving throughput performance. However, more sophisticated approaches on both stages promise an even better performance. These results lead us to our first important take-home message.

Finding 1: *Even if a robotized sorting system provides a fully-automated sortation process, not optimizing the basic decisions involved is a bad idea. Sophisticated optimization promises a significant improvement of throughput performance.*

5.2 How to improve throughput performance

This section explores further levers to improve the throughput performance of robotized sorting systems. Specifically, we investigate the impact of the following three entities: the size of the robot fleet, the system layout (i.e., the arrangement of collection points on the shop floor), and the wave size. Note that all these experiments are based on our largest real-world instances with $|O| = 1,000$ orders and $|R| = 40$ robots, if not explicitly stated otherwise.

Size of robot fleet: The most obvious lever to improve throughput performance is certainly an invest into additional robots. Increasing the robot fleet size $|R|$ should reduce the waiting times of the products at the loading station. The throughput loss of decomposition heuristic DEC and most basic method RND are compared for different fleet sizes $|R|$ within Figure 5.

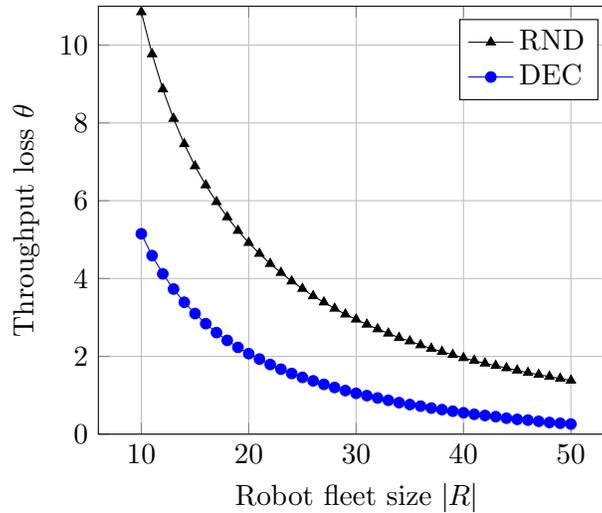


Figure 5 Impact of robot fleet size: Throughput loss θ of decomposition approach DEC and most basic method RND for varying fleet sizes $|R|$ on the largest benchmark instances with $|O| = 1,000$ orders

The results of Figure 5 show diminishing returns of additional robots for both solution methods. For the large number of orders to be processed in our instances, obviously, a robot fleet of $|R| = 10$ is too small, so that long waiting times accumulate and any additional robot greatly relieves this bottleneck. However, if we already have a substantial fleet size, e.g., of $|R| = 40$, adding further robots only provides small additional gains. Note that our analysis neglects robot interference on the shop floor. A larger robot fleet tends to also increase the risk of robots blocking each other (see [Zou et al. 2021](#)), so that the effect of diminishing returns should be even more pronounced in real-world systems. This leads us to another take-home message:

Finding 2: *A robot bottleneck should be avoided, because this adds considerable waiting time and deteriorates throughput performance. Once a sufficient fleet size is available, however, adding even more robots delivers only small returns. Hence, further improvement requires other levers.*

Layout: One of the big assets of a robotized sorting system is its layout flexibility and the substantial degrees of freedom on how to arrange the collection points on the shop floor. We leave the detailed exploration of the huge amount of potential layouts to future research. Instead, we only contribute the following layout-related take-home message:

Finding 3: *The performance of a robotized sorting system is rather insensitive regarding the specific arrangement of collection points, as long as there is a sufficient amount of quickly accessible collection points in the hot zone close to the loading station. Here, the majority of orders can be processed quickly without inducing much robot travel.*

To substantiate this finding, Figure 6 counts the number of orders processed at the individual collection points. We order the collection points in increasing distance from the loading station,

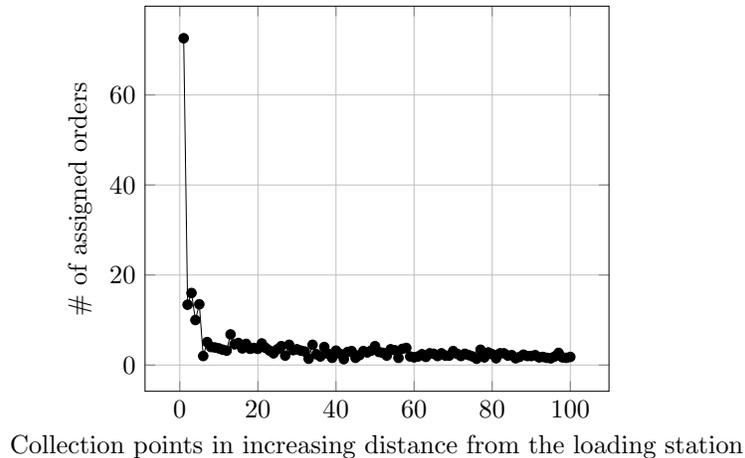


Figure 6 Number of orders assigned to each collection point, ordered in increasing distance from the loading station)

with collection point 1 (100) being closest to (farthest from) the loading station. These results show that there is a small number of close-by collection points where the majority of orders is collected. As long as these collection points from the hot zone can be kept very close to the loading station, the majority of orders can be handled without much effort. Farther collection points receive just very few orders, so that they must only rarely be visited and their positions on the shop floor become less influential. Recall that our study neglects robot interference. If most orders are processed in the hot zone, robot interference may be an issue here. Hence, the arrangement of collection points in the hot zone may still be important in order to enable a clear robot flow regulation.

Wave size: The picking area, where the products demanded by customer orders are picked, and the sortation area, where picked products are sorted according to orders, are coordinated via waves (see also [Boysen et al. 2019a,b](#)). A wave denotes a subset of orders that are jointly picked and released toward the sortation area, where the wave is sorted, before the next wave is initiated. The wave size specifies the number of orders that are unified to subsequent waves. Naturally, a wave size larger than the number of available collection points bears the risk that all collection points are occupied and yet another order cannot be serviced. Then, the current product dedicated to a new order cannot be removed from the loading station, so that more and more products accumulate and a so-called gridlock with substantial recovery work threatens ([Gallien and Weber 2010](#)). Hence, to process larger waves, more collection points are required to avoid gridlocks. This, however, increases the travel effort for the robots. Beyond that, smaller waves also reduce the order spreads, which cannot spread across their specific wave. Thus, it is to be expected that larger waves increase the throughput loss. We challenge this expectation with the help of Figure 7. On the left of this figure, we plot the number of open collection points during any slot t of the arrival sequence for different

wave sizes $\omega \in \{100, 250, 500, 1000\}$. Note that these results are obtained for a single of our largest instances. To obtain the arrival sequence, we select the next ω orders not yet processed, randomly mix the demanded products and append these products to the arrival sequence and so on until all orders are processed. On the right hand side of Figure 7, we report on the total throughput loss over the complete arrival sequence for different wave sizes ω . Note that here we average the throughput loss over all instances with $|O| = 1,000$ orders. These results confirm our expectations:

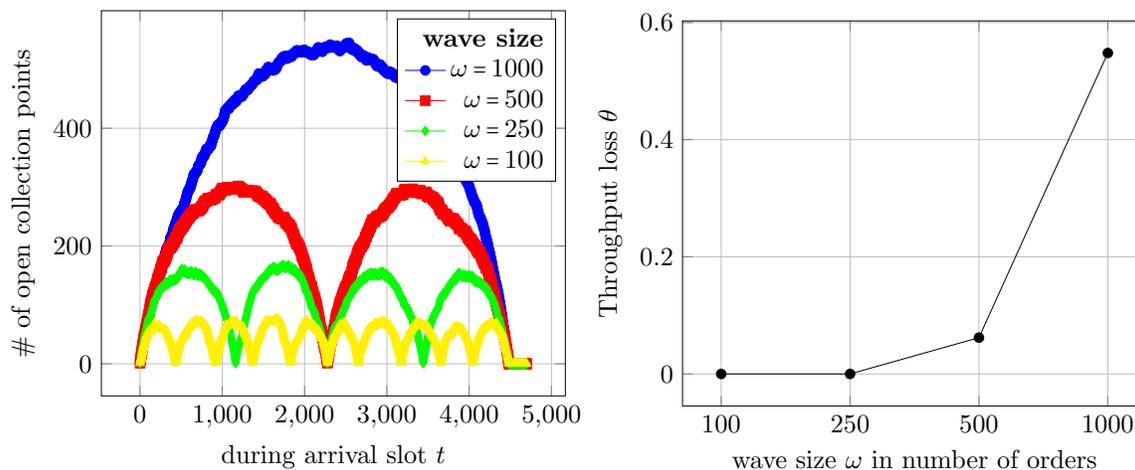


Figure 7 Impact of wave size ω : Number of open collection points during the arrival slots of the sortation process for different wave sizes (left) and performance loss θ over all subsequent waves depending on the wave size (right)

- *Larger waves require more collection points:* The development of the number of open collection points follows a wavy pattern over time (see Figure 7, left). We have a peak in the middle of each wave, where the maximum number of collection points is open. Before (after) this peak, we observe an excess (deficit) of collection points that are opened by new orders over those closed after order completion. Furthermore, we can confirm one of our expectations: Larger waves require more collection points, where, however, the maximum number of open collection points is smaller than the wave size.
- *Larger waves increase the throughput loss:* On the right hand side of Figure 7, we can observe the performance impact of the wave size. Again, our expectation is met. Smaller waves reduce the number of required collection points, so that the robots' travel distances reduce, and waiting is avoided. Larger waves, instead, lead to more travel and extended order spreads, so that advantageous collection points are blocked for a longer time. Thus, we observe significantly larger throughput losses for larger waves. These results lead us to the next take-home message.

Finding 4: *The wave size is a delicate planning task that must be determined with great care. Smaller waves are beneficial for the sortation stage. Here, fewer collection points are required and the reduced travel effort for the robots leads to significant performance gains. Smaller waves, however, also reduce the flexibility of the picker area. Here, the pick density per picking tour decreases, because the order pickers are less flexible and have to travel toward specific (potentially far distant) shelves in order to complete a small wave. Therefore, the picking area rather profits from larger waves, and we face a delicate performance tradeoff among picking and sortation.*

5.3 The impact of anticipation

Next, we turn to systems where we merely have partial knowledge on the upcoming arrival sequence. This means that the identification device, e.g., the barcode scanner that registers the approaching products on the conveyor, is positioned such that only a limited lookahead of the next L pieces is available. Alternatively, the picking and consolidation area can be linked by a rather short low-capacity conveyor, so that picking and sorting proceed concurrently. First, we analyze the functionality of our MSA.

Value of two-step scenario generation: In contrast to conventional MSA work, we split our MSA in two steps. First, we determine the P2O assignment by solving a first set of scenarios. Second, we determine the collection point with the order being already fixed with a second scenario set. To show the advantage of this two-step procedure, we compare our MSA to a single-step variant where the most frequent tuple (O, P) over all scenarios is selected. For a fair comparison, we double the number of scenarios in this single-step variant to 120 scenarios overall. The results are shown in Table 6. We observe that except for the smallest instance, our two-step MSA provides significantly smaller gaps compared to the single-step MSA benchmark. This means that splitting the decision making into two parts provides better decision making, likely, because the number of identical tuples can be small in the single-stage MSA. Furthermore, by fixing the first part in the determination of the second, the interplay between both decisions is given better consideration. Another interesting observation is that the runtime for the two-step MSA is noticeably shorter than for the single-step MSA, even though both solve the same total number of scenarios. In the two-step MSA, the second “round” of scenarios is less time-consuming though since an important part, the P2O assignment, is already fixed.

Finding 5: *When faced with a stochastic dynamic decision problem where decisions contain several different parts, separating the determination of the individual parts may not only provide better solutions compared to conventional MSA procedures, it may do so in significantly shorter time. Thus, it may be valuable to consider multiple steps in future scenario-based planning approaches for dynamic problems with different parts of decisions, e.g., vehicle routing (assignment, sequencing) or demand management (assortment, fulfillment).*

data	O	R	MSA				single-step MSA			
			gap _b	best	sec	mspp	gap _b	best	sec	mspp
RW	50	5	0.34	4	1.97	8.64	0.21	8	1.96	8.48
RW	100	10	0.03	8	9.21	21.61	1.10	2	10.27	24.10
RW	200	15	0.26	7	34.46	40.46	0.99	4	43.96	51.46
RW	500	30	0.00	10	225.69	104.95	2.33	0	408.89	190.04
RW	1000	40	0.45	5	1309.19	299.39	0.66	6	2532.58	578.93
avg			0.22	6.80	316.10	95.01	1.06	4.00	599.53	170.60

Table 6 Impact of two-step scenario generation: (Two-step) MSA ($W = 60$) vs. single-step MSA ($W = 120$)

Lookahead horizon: Table 7 reports on the throughput loss of our MSA approach (see Section 3) for different values of lookahead L . To give these results some proportion, we also report the results of rule-based approach RULE, which only applies the information on the current product at the loading station and thus only utilizes a lookahead of $L = 1$, and of heuristic decomposition approach DEC. Note that the latter is just a theoretical benchmark in this case, because it presupposes perfect knowledge on the complete arrival sequence.

Type	O	R	RULE	MSA				DEC
				$L = 10$	$L = 60$	$L = 120$	$L = 240$	
RW	50	5	1.59	1.46	1.35	1.33	1.32	1.32
RW	100	10	0.85	0.75	0.67	0.62	0.61	0.59
RW	200	15	0.85	0.75	0.67	0.61	0.56	0.50
RW	500	30	0.80	0.64	0.57	0.52	0.46	0.29
RW	1000	40	1.35	1.13	1.05	0.97	0.88	0.55
avg			1.09	0.95	0.86	0.81	0.77	0.65

Table 7 Impact of varying lookaheads L on throughput loss θ of multiple-scenario approach MSA in comparison with rule-based approach RULE and decomposition heuristic DEC

The results of Table 7 show that utilizing more information on the arrival sequence reduces the throughput loss. If MSA is bound to a short lookahead of only the next $L = 10$ products its performance is only slightly (yet consistently) better than RULE. The longer lookahead L , the more the results of MSA approach those of theoretical benchmark DEC. Naturally, the gap to DEC depends on the fraction of the total arrival sequence that is covered by the lookahead. Hence, we observe larger gaps in larger instances with more orders $|O|$ and longer arrival sequences. Finally, we also witness decreasing marginal returns of an increasing lookahead. These results can be condensed into the following managerial take-home message:

Finding 6: *In typical real-world warehouses, increasing the lookahead of the arrival sequence not only requires a simple repositioning of the identification device along the conveyor but comes along with further investment for restructuring the warehouse setup in general and the conveyor system in particular. Regarding the performance gains of an increasing lookahead, which have to outweigh*

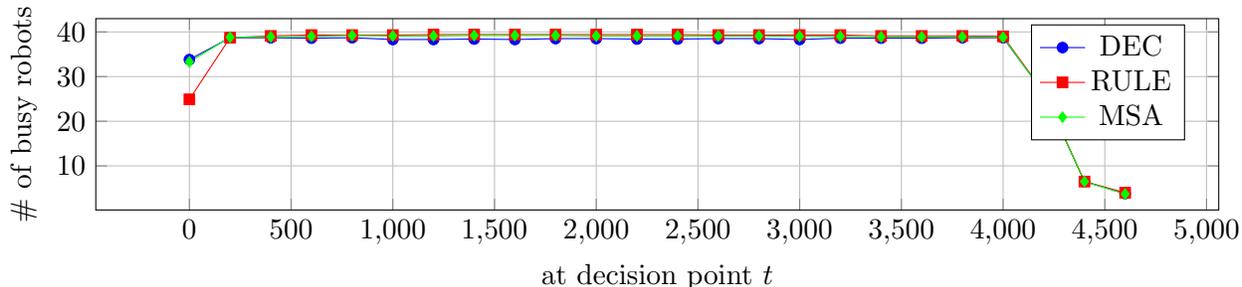


Figure 8 Average number of busy robots during the decision points t of the decision process

these investment costs, we have good news: Already a comparatively short lookahead on the next dozens of products promises a considerable reduction of the throughput loss. Even longer lookaheads deliver merely diminishing additional returns, so that in most cases a moderate restructuring should be sufficient to gain large parts of the cake.

Decision making: The final part of our analysis investigates how decision making changes in case of anticipation. To this end, we first plot the average number of busy robots over time. The results are shown in Figure 8. The x-axis depicts the time of the process by enumerating the decision points t (i.e., one in each arrival slots t). The y-axis shows the number of busy robots ($|R| = 40$) averaged over all results between the current and the next data point. The averaging avoids a confusing flicking of the graphs. We observe that the policies differ most significantly in the beginning of the horizon when the MSA (and omniscient policy DEC) already keep most of the robots busy compared to policy RULE. Later, all robots are more or less busy until the end of the process. The early difference indicates that both anticipatory policies invest to set the system up in a flexible and anticipatory way, e.g., by opening more orders and placing orders more distant from the loading station.

To corroborate this observation, we plot the average handover times (i.e., costs $C(s_t, x_t)$) over the subsequent arrival slots of the process in Figure 9. The lowest possible cost are one indicating seamless handovers of two consecutive products. We observe that both DEC and MSA have relatively high handover cost in the beginning compared to policy RULE. However, for RULE, the efficient routing in the beginning leads to inflexible and costly situations later. This is indicated by the steady increase in handover cost until decision point 1500. Based on its perfect foresight, DEC keeps the handover cost rather constant over time, and always below two. This means that in every state, at least twice the interarrival time later, another robot is available. As expected, MSA delivers a compromise between RULE and DEC as it lacks full information.

Finding 7: *Anticipation and flexibility are essential for an effective sortation process. This means that especially at the start of a wave, the decision making may carefully sacrifice some*

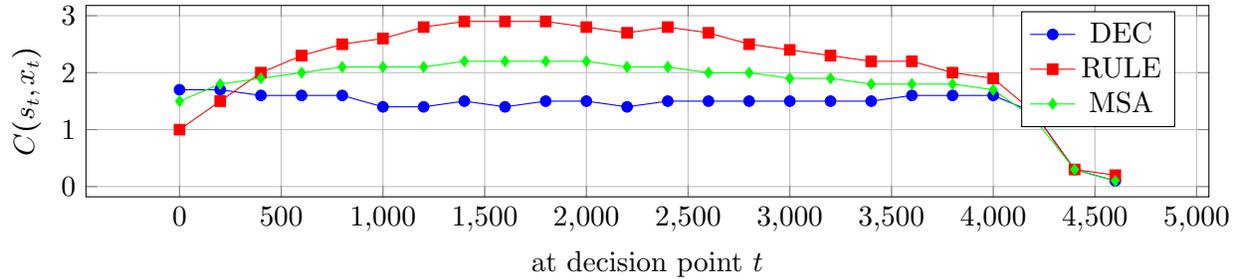


Figure 9 Average handover cost $C(s_t, x_t)$ during the decision points t of the decision process

efficiency by opening more orders at more distant consolidation points to receive a well-thought-out and flexible system setup, regardless of the level of information availability. This setup will pay off later during the process with less congestion and overall better performance.

6 Conclusions and outlook

This paper explores the operational planning of robotized sorting systems, which constitute a most challenging scheduling environment: A large number of products arrive subsequently at the loading station with an interarrival time of just a few seconds, so that the multiple interdependent decisions must be made in real time. Furthermore, most warehouse setups cannot provide perfect knowledge on the complete arrival sequence of products but limit the lookahead to a subset of approaching pieces. Based on an analysis of computational complexity and a thorough computational evaluation of the deterministic case, we provide a novel two-step multiple-scenario approach that meets these challenges and improves upon conventional single-step MSA procedures and practical decision rules. The application of our algorithms to real-world order data allows us to derive seven elementary take-home messages that deliver decision support for managers, who have to setup and operate a robotized sorting system.

Future research should extend our problem setting. First, we neglect robot interference. There exist layouts, based on one-way robot routing (see [Zou et al. 2021](#)), that actually eliminate most potential interference. However, especially in the hot zone close to the loading station, robots blocking each other may become an issue. How to successfully integrate robot interference into our scheduling framework is a valid field for future research. Furthermore, we only consider a single loading station. Even if there are larger systems with multiple parallel stations, partitioning the collections points among loading stations, such that the problem decomposes into multiple single-stations problems, is a valid solution approach. However, further improvement may be achievable without such a fixed partitioning of collection points. The price for the additional flexibility, however, is a much more involved parallel-station scheduling problem. Future research should explore whether the potential performance gains can actually justify the increasing problem complexity.

Finally, applying our novel two-step MSA procedure to other domains where multiple interdependent decisions must be made simultaneously could lead to a better decision making in stochastic dynamic environments.

References

- Ausseil, Rosemonde, Jennifer A Pazour, Marlin W Ulmer. 2022. Supplier menus for dynamic matching in peer-to-peer transportation platforms. *Transportation Science*, (to appear).
- Azadeh, Kaveh, René de Koster, Debjit Roy. 2019. Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, 53 (4), 917-945.
- Azi, Nabila, Michel Gendreau, Jean-Yves Potvin. 2012. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199 (1), 103-112.
- Bansal, Vishal, Debjit Roy, Jennifer Pazour. 2021. Performance analysis of batching decisions in waveless order release environments for e-commerce stock-to-picker order fulfillment. *International Transactions in Operational Research*, 28 (4), 1787-1820.
- Bent, Russell W, Pascal Van Hentenryck. 2004. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52 (6), 977-987.
- Boysen, Nils, Dirk Briskorn, 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, Dirk Briskorn, Stefan Fedtke, Marcel Schmickerath. 2019a. Automated sortation conveyors: A survey from an operational research perspective. *European Journal of Operational Research*, 276 (3), 796-815.
- Boysen, Nils, René De Koster, Felix Weidinger. 2019b. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277 (2), 396-411.
- Boysen, Nils, Stefan Fedtke, Felix Weidinger. 2018. Optimizing automated sorting in warehouses: The minimum order spread sequencing problem. *European Journal of Operational Research*, 270 (1), 386-400.
- Boysen, Nils, Konrad Stephan, Felix Weidinger. 2021. Efficient order consolidation in warehouses: The product-to-order-assignment problem in warehouses with sortation systems. *IIE Transactions*, (to appear).
- Carlo, Héctor J, Iris FA Vis, Kees Jan Roodbergen. 2014. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research*, 236 (1), 1-13.
- Cezik, Tolga, Stephen C Graves, Amy C Liu. 2022. Velocity-based stowage policy for semi-automated fulfillment system. *Production and Operations Management*, (to appear).
- De Koster, René, Tho Le-Duc, Kees Jan Roodbergen. 2007. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182 (2), 481-501.
- Gallien, Jérémie, Théophane Weber. 2010. To wave or not to wave? order release policies for warehouses with an automated sorter. *Manufacturing & Service Operations Management*, 12 (4), 642-662.
- Garey, Michael R, David S Johnson. 1979. *Computers and intractability*, vol. 174. Freeman San Francisco.
- Ghiani, Gianpaolo, Emanuele Manni, Barrett W Thomas. 2012. A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science*, 46 (3), 374-387.
- Johnson, M Eric. 1997. The impact of sorting strategies on automated sortation system performance. *IIE Transactions*, 30 (1), 67-77.

- Khir, Reem, Alan Erera, Alejandro Toriello. 2021. Two-stage sort planning for express parcel delivery. *IIE Transactions*, 53 (12), 1353-1368.
- Kolen, Antoon WJ, Jan Karel Lenstra, Christos H Papadimitriou, Frits CR Spieksma. 2007. Interval scheduling: A survey. *Naval Research Logistics*, 54 (5), 530-543.
- Kroon, Leo G, Arunabha Sen, Haiyong Deng, Asim Roy. 1996. The optimal cost chromatic partition problem for trees and interval graphs. *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 279-292.
- Mead, Carver, Lynn Conway. 1980. *Introduction to VLSI systems*. Addison-Wesley Reading, MA.
- Meller, Russell D. 1997. Optimal order-to-lane assignments in an order accumulation/sortation system. *IIE Transactions*, 29 (4), 293-301.
- Powell, Warren B. 2022. *Reinforcement Learning and Stochastic Optimization: A unified framework for sequential decisions*. John Wiley & Sons.
- Roodbergen, Kees Jan, Iris FA Vis. 2009. A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, 194 (2), 343-362.
- Schiffer, Maximilian, Nils Boysen, Patrick S Klein, Gilbert Laporte, Marco Pavone. 2022. Optimal picking policies in e-commerce warehouses. *Management Science*, (to appear).
- Schilde, Michael, Karl F Doerner, Richard F Hartl. 2014. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, 238 (1), 18-30.
- Shi, Ye, Hu Yu, Yugang Yu, Xiaohang Yue. 2022. Analytics for IoT-enabled human-robot hybrid sortation: An online optimization approach. *Production and Operations Management*, (to appear).
- Soeffker, Ninja, Marlin W. Ulmer, Dirk C. Mattfeld. 2022. Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research*, 298 (3), 801-820.
- Song, Yongjia, Marlin W Ulmer, Barrett W Thomas, Stein W Wallace. 2020. Building trust in home services—stochastic team-orienteeing with consistency constraints. *Transportation Science*, 54 (3), 823-838.
- Ulmer, Marlin W, Justin C Goodson, Dirk C Mattfeld, Marco Hennig. 2019. Offline-online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science*, 53 (1), 185-202.
- Vis, Iris FA. 2006. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170 (3), 677-709.
- Voccia, Stacy A, Ann Melissa Campbell, Barrett W Thomas. 2019. The same-day delivery problem for online purchases. *Transportation Science*, 53 (1), 167-184.
- Wang, Zheng, Jiuh-Biing Sheu, Chung-Piaw Teo, Guiqin Xue. 2022. Robot scheduling for mobile-rack warehouses: Human-robot coordinated order picking systems. *Production and Operations Management*, 31 (1), 98-116.
- Zolfagharinia, Hossein, Michael Haughton. 2016. Effective truckload dispatch decision methods with incomplete advance load information. *European Journal of Operational Research*, 252 (1), 103-121.
- Zou, B, De Koster M.B.M., Y. Gong, X. Xu, G. Shen. 2021. Robotic sorting systems: Performance estimation and operating policies analysis. *Transportation Science*, 55 (6), 1430-1455.

E-Companion

Appendix A: Complexity of P2O

This appendix proves that P2O is strongly \mathcal{NP} -hard. The proof is by reduction from the 3-Partition problem, which is well-known to be strongly \mathcal{NP} -complete (Garey and Johnson 1979). This problem is defined as follows:

3-Partition: Given $3q$ positive integers a_i ($i = 1, 2, \dots, 3q$) and a positive integer B with $B/4 < a_i < B/2$ and $\sum_{i=1}^{3q} a_i = qB$, does there exist a partition of the set $\{1, 2, \dots, 3q\}$ into q sets $\{A_1, A_2, \dots, A_q\}$, each having exactly three elements, such that $\sum_{i \in A_j} a_i = B$, $\forall j = 1, 2, \dots, q$?

The (Pseudo-polynomial) transformation scheme, generating a P2O instance from a given instance of 3-Partition, is defined as follows: We introduce $n = 3q$ orders and three SKU denoted X , Y , and Z . Each order builds the counterpart to an integer a_i of 3-Partition and demands one piece of X , a_i pieces of Y , and one piece of Z . Arrival sequence σ consists of q subsequences defined as follows: first, we have three pieces of X , then B pieces of Y , and finally three pieces of Z . This subsequence, identically repeated q times, builds the arrival sequence σ of P2O. The question we ask is whether a solution for P2O with a total order spread not exceeding $3q \cdot (B + 4)$ can be obtained.

A feasible solution for an instance of 3-Partition can be transformed to a feasible solution of the corresponding P2O instance by assigning the three counterpart orders corresponding to the integers of a subset to a subsequence. These orders spread from the pieces of SKU X at the start up to those of SKU Z at the rear and, since the sum of the three integers of 3-Partition is B , the B pieces of SKU Y are sufficient to satisfy the demands for this SKU of the three assigned orders. Each subsequence, thus, has an order spread of $3 \cdot (B + 4)$ and summarized over all q subsequences this does not exceed bound $3q \cdot (B + 4)$.

On the other hand, each feasible solution for any P2O instance is also a feasible solution for 3-Partition. If each subsequence is not assigned three orders, whose demands sum up to exactly B pieces of SKU Y , either one of these orders is not completed within the subsequence or an order of a later subsequence is already started earlier. In both cases, bound $3q \cdot (B + 4)$ can no longer be reached. Instead, three orders with a total demand for B pieces of SKU Y are unified within each subsequence, which directly correspond to subsets of 3-Partition, and the one-to-one mapping between both problems is readily available.

Appendix B: Complexity of O2CP

This appendix proves that O2CP is strongly \mathcal{NP} -hard. This is done by reduction from a well-known graph coloring problem, the optimal cost chromatic partition (OCCP) problem, which was initially introduced by Mead and Conway (1980) and is defined as follows:

OCCP: Given a graph $G = (V, E)$ with n nodes and a sequence of coloring costs (k_1, \dots, k_n) , find a proper coloring $C(v) \in \{1, \dots, n\}$ of each node $v \in V$ such that the total coloring costs $\sum_{v=1}^n k_{C(v)}$ are minimum. A coloring is defined as proper, if adjacent nodes have different colors.

The nodes of OCCP directly correspond to the orders of O2CP, and two nodes connected by an edge equal two orders with overlapping order spreads. If we, furthermore, think of collection points as colors, then the relationship between both problems is readily available. However, our O2CP extends OCCP, because we do not have constant costs per color but assignment specific costs w_{ij} where the costs per color vary among nodes. Kroon et al. (1996) have shown that OCCP remains strongly \mathcal{NP} -hard, even if G is an interval graph (as long as there are at least four different coloring costs). Since the order spreads of O2CP correspond to the underlying intervals of an interval graph, \mathcal{NP} -hardness of our O2CP directly follows.

Appendix C: Alternative setups for our decomposition approach

This appendix benchmarks our heuristic decomposition approach (DEC) of Section 2.2 with three alternative setups. The computational study of this appendix shows that our DEC setup outperforms these competitors. Specifically, we introduce an alternative approach for each stage:

- *P2O*: Within DEC, we assign pieces to orders according to composite rule (CR) that relates the number of demand pieces per order to the minimum order spread (see Section 2.2.1). We have also tested the following alternative decision rules: largest-order-first (LOF), smallest-order-first, random-order-next, or balanced-order-first. LOF, which only applies the nominator of the composite rule CR, turned out as the best of these competitors. To not overload this appendix with results and tables, we thus only report the computational results if LOF is applied for the P2O assignment.
- *O2CP*: Unfortunately, O2CP turns out as being strongly \mathcal{NP} -hard and its solution can thus be too time-consuming, especially when repeatedly solved in an MSA approach. Alternatively, we could also apply a surrogate objective and only minimize the number of utilized collection points instead. If a minimum number of collection points is ensured, we can select those closest to the loading station, which also tends to reduce the robots' travel distances. The advantage of this surrogate is that the resulting optimization problem (i.e., assign orders with given order spreads to a minimum number of collection points) is efficiently solvable. To do so, we can apply the so-called left-edge procedure, which is one of the basic algorithms of classical interval scheduling (Kolen et al. 2007): Assign the jobs (orders) to the machines (collection points) in order of non-decreasing starting times (first slots of order spreads), using a machine used before whenever possible. The left-edge algorithm solves our surrogate problem to optimality in $\mathcal{O}(|O| \log |O|)$. Once a solution for the surrogate problem is at hand, we can fix the actual

collection points each machine of the solution refers to in a greedy manner according to the travel distances caused by the assigned orders, i.e., $\sum_{j \in O, i \in P} \Delta_i \cdot |S_j| \cdot y_{j,i}$. We label this alternative method for O2CP as LE (for left-edge).

If we combine these alternative solution methods for both stages in a full-factorial manner, we receive four different setups for DEC. The computational performance of these competitors on our real-world data instances is summarized in Table EC.1.

Type	O	R	DEC(CR,CR)			DEC(CR,LE)			DEC(LOF,CR)			DEC(LOF,LE)		
			gap _b	best	sec									
RW	50	5	0.17	8	0.00	0.64	3	0.00	0.21	5	0.00	1.01	1	0.00
RW	100	10	0.55	5	0.00	2.78	0	0.00	0.50	5	0.00	2.45	0	0.00
RW	200	15	0.14	9	0.00	5.10	0	0.00	1.56	1	0.00	5.71	0	0.00
RW	500	30	0.00	10	0.01	11.28	0	0.01	3.46	0	0.01	13.36	0	0.01
RW	1000	40	0.00	10	0.04	17.06	0	0.04	4.83	0	0.03	19.97	0	0.02
avg			0.17	8.40	0.01	7.37	0.60	0.01	2.11	2.20	0.01	8.50	0.20	0.01

Table EC.1 Computational results of four different setups of decomposition approach DEC on our real-world data instances

While all four DEC setups lead to very similar (negligible) computational times DEC(CR,CR) clearly outperforms its competitors with regard to solution quality. Consequently, DEC(CR,CR) is our method of choice that is explained in detail in Section 2.2 and applied in all our further experiments.

Appendix D: System layout and driving times of robots

This appendix details the layout of the robotized sorting system that is applied in our computational studies. Given a linear layout of collections points $i \in P$ as depicted in Figure EC.1 and the parameters of Table EC.2, the robot driving time from the loading station to collection point i and back, is given by

$$\Delta_i = \left\lceil \frac{\bar{\Delta}_i}{\lambda} \right\rceil \quad \text{with} \quad \bar{\Delta}_i = t_d + t_w + 4 \cdot t_{90} + 2t(i \cdot (w_a + w_d)) + 2t(2 \cdot w_a + w_d). \quad (\text{EC.1})$$

Here, $t(d)$ denotes the driving time of a robot to cover a distance of d units. A detailed explanation of $t(d)$ is provided below. As can be seen from formula (EC.1) and Figure EC.1, a robot tour with a delivery to collection point i takes t_w for loading, t_d for unloading, $4 \cdot t_{90}$ for turning right, $2t(2 \cdot w_a + w_d)$ for vertical movements, and $2t(i \cdot (w_a + w_d))$ for horizontal movements.

We denote the time it takes a robot to pass a segment of length d as $t(d)$. Recall that we consider a robot starting (ending) with an initial (final) speed of 0, a maximum speed of v_m , and accelerating (decelerating) with a rate of a ($-a$). Thus, the robot either accelerates until it reaches top speed

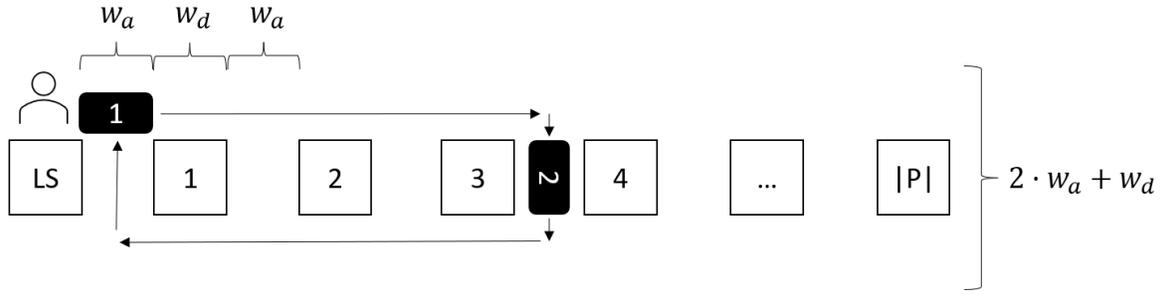


Figure EC.1 Layout of the robotized sorting system

$ P /2$	10	system length, given by the number of collection points per aisle
w_a, w_d	0.6m	width of an aisle/cross-aisle and a collection point
a	1m/s^2	acceleration/deceleration rate of a robot
t_d	2s	time for a robot to drop off a piece into a collection point
t_w	5s	time for loading a piece onto the robot
t_{90}	1s	time for a robot to take a 90° turn
v_m	2m/s	maximum velocity of a robot
λ	3s	interarrival time of products at loading station

Table EC.2 Parameters according to Zou et al. (2021)

v_m or it has covered $d/2$. Deceleration and acceleration are symmetric; in between there is a part where the robot moves with constant top speed, but only if d is long enough.

The time a robot needs to reach its top speed v_m is $\bar{t} = v_m/a$. The distance covered during the first t time units, $t \leq \bar{t}$, equals $d(t) = 0.5 \cdot a \cdot t^2$. Thus, a robot has covered a distance of $\bar{d} = 0.5 \cdot a \cdot \bar{t}^2$, when it reaches its top speed due to the second equation of motion. Symmetrically, a robot covers a distance of \bar{d} decelerating from v_m to 0.

If the length d does not exceed $2 \cdot \bar{d}$, then the robot accelerates (covering a distance of $d/2$) and immediately decelerates (covering a distance of $d/2$), potentially without reaching v_m . The time $t'(d/2)$ necessary for covering a distance of $d/2$ while accelerating with a can be determined using the second equation of motion by

$$t'(d/2) = \sqrt{d/a}.$$

Hence, if $d \leq 2 \cdot \bar{d}$, we have

$$t(d) = 2 \cdot t'(d/2) = 2 \cdot \sqrt{d/a}.$$

If the length d of a segment is at least $2 \cdot \bar{d}$, then the robot accelerates to v_m covering a distance of \bar{d} , moves with top speed covering a distance of $d - 2 \cdot \bar{d}$, and decelerates to 0 covering a distance of \bar{d} . Hence, if $d \geq 2 \cdot \bar{d}$, we have

$$t(d) = 2 \cdot \bar{t} + \frac{d - 2 \cdot \bar{d}}{v_m}.$$

Summarizing, we obtain

$$t(d) = \begin{cases} 2 \cdot \sqrt{d/a} & \text{if } d \leq 2 \cdot \bar{d} \\ 2 \cdot \bar{t} + \frac{d - 2 \cdot \bar{d}}{v_m} & \text{if } d \geq 2 \cdot \bar{d} \end{cases} = \begin{cases} 2 \cdot \sqrt{d/a} & \text{if } d \leq 2 \cdot v_m^2/a \\ 2 \cdot v_m/a + \frac{d - v_m^2/a}{v_m} & \text{if } d \geq 2 \cdot v_m^2/a \end{cases} \quad (\text{EC.2})$$

Appendix E: Impact of disorder on the throughput performance

This appendix investigates the impact of another potential lever to improve the sortation performance: A reduction of the level of disorder of the arrival sequence. In the very best case, order after order arrives at the loading station. This reduces the order spreads to a minimum, near-by collection points of the hot zone could be utilized even more often, and the total transport effort decreases considerably. Note that such a presorted arrival sequence renders a subsequent sortation superfluous. However, even if a perfectly arranged arrival sequence is rather theoretical, less disorder, where orders are only rarely intermixed with products dedicated to other orders, could still be a lever to improve the sortation throughput. To challenge this claim, we start with the perfect inbound sequences for our largest instances with $|O| = 1,000$ orders and $|R| = 40$ robots. In a perfect sequence, all pieces demanded by the first order are followed by all pieces for the second order and so on. Given this starting point, we increase the level of disorder in a stepwise manner by shifting another randomly selected product to a different (randomly selected) sequence position. For each resulting instance, the corresponding throughput loss θ is determined by the MSA, the (perfect information) decomposition approach DEC and method RULE. Figure EC.2 displays the performance loss in relation to the disorder level of the arrival sequence, where the latter is measured by the percental number of shifted products.

The results of Figure EC.2 suggest that reducing the level of disorder can improve the sortation throughput, especially given the simple policy of RULE. In an optimized system, where the MSA (or DEC) is applied for the operational decisions, the impact of the disorder level is much less distinct as the MSA can handle even very disordered sequences relatively well. However, with increasing disorder level, the gap between DEC and MSA slightly widens as well. This, again, highlights that MSAs perform best when the future can be predicted accurately. This suggests the following conclusion:

An “ideal” pre-sequencing can be very valuable in case sorting is steered with simple decision rules. However, when optimizing via MSA (or DEC), the loss resulting from more disorder is less substantial. Since less disorder comes at the price of more restricted picking and picking (at least under a picker-to-parts policy (see De Koster et al. 2007)) is likely more labor-intensive and thus more costly, additional restrictions on the picking side can typically not be outweighed by a higher sortation performance. Thus, reducing the level of disorder is a possible, yet not necessarily practical lever to improve throughput performance.

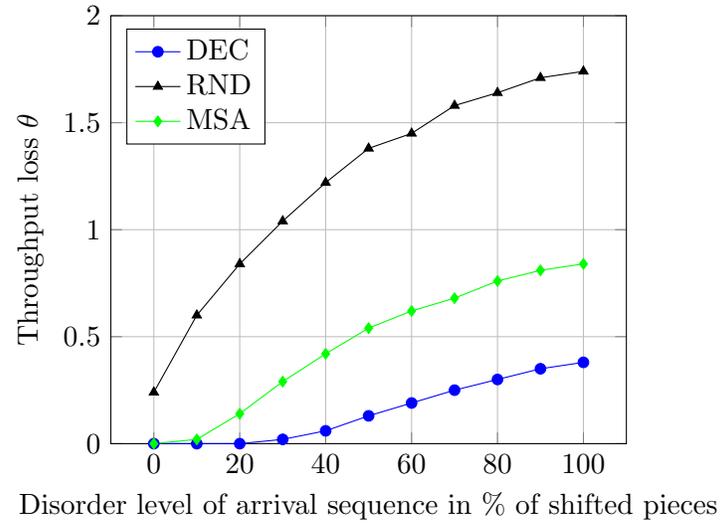


Figure EC.2 Disorder level of arrival sequence: Impact of the percental number of pieces shifted from the perfect arrival sequence to other positions an throughput loss θ

Otto von Guericke University Magdeburg
Faculty of Economics and Management
P.O. Box 4120 | 39016 Magdeburg | Germany

Tel.: +49 (0) 3 91/67-1 85 84
Fax: +49 (0) 3 91/67-1 21 20

www.fww.ovgu.de/femm

ISSN 1615-4274