Tadumadze, Giorgi; Wenzel, Julia; Emde, Simon; Weidinger, Felix; Elbert, Ralf

Assigning orders and pods to picking stations in a multi-level robotic mobile fulfillment system

Flexible Services and Manufacturing Journal

Provided in Cooperation with:
Springer Nature

Suggested Citation: Tadumadze, Giorgi; Wenzel, Julia; Emde, Simon; Weidinger, Felix; Elbert, Ralf (2023) : Assigning orders and pods to picking stations in a multi-level robotic mobile fulfillment system, Flexible Services and Manufacturing Journal, ISSN 1936-6590, Springer US, New York, NY, Vol. 35, Iss. 4, pp. 1038-1075, https://doi.org/10.1007/s10696-023-09491-0

This Version is available at:
https://hdl.handle.net/10419/309572

Flexible Services and Manufacturing Journal (2023) 35:1038–1075
https://doi.org/10.1007/s10696-023-09491-0

Assigning orders and pods to picking stations in a multi-level robotic mobile fulfillment system

Giorgi Tadumadze[1] · Julia Wenzel[2] · Simon Emde[3] · Felix Weidinger[1] · Ralf Elbert[2]

Accepted: 26 February 2023 / Published online: 6 May 2023

Abstract

This paper addresses the operational planning problem of assigning orders and pods (i.e., mobile shelves) to picking stations in a multi-level robotic mobile fulfillment system (RMFS), which deals with two issues: deciding on which picking station handles which order, and from which pods to pick the ordered items, considering the limited storage capacity of the pods. Due to the relatively poor space utilization of single-level RMFS warehouses, such systems are often spread over multiple floors in practice. Therefore, we explicitly consider multi-level warehouse layouts with isolated levels (or zones) where a pod can only be brought to a station if both of them are on the same level. We optimize the problem with regard to a multi-criteria objective function that consists of three workload-oriented objectives: we aim to balance the total workload among all pickers, minimize the total order-consolidation effort for the packers, and the pod movement effort for the mobile robots. After formalizing the planning problem as a multi-objective optimization problem, we provide two mixed-integer

✉ Giorgi Tadumadze
giorgi.tadumadze@tu-darmstadt.de

Julia Wenzel
wenzel@log.tu-darmstadt.de

Simon Emde
simon.emde@uni-jena.de

Felix Weidinger
felix.weidinger@tu-darmstadt.de

Ralf Elbert
elbert@log.tu-darmstadt.de

1 Fachgebiet Management Science/Operations Research, Technische Universität Darmstadt, Hochschulstraße 1, 64289 Darmstadt, Germany

2 Fachgebiet Unternehmensführung und Logistik, Technische Universität Darmstadt, Hochschulstraße 1, 64289 Darmstadt, Germany

3 Lehrstuhl für Wirtschaftsinformatik, insb. Business Intelligence, Friedrich-Schiller-Universität Jena, Carl-Zeiß-Straße 3, 07743 Jena, Germany

linear programming models. Additionally, we propose a matheuristic that reduces the model size to the desired granularity so that realistically sized problem instances can be solved within less than four minutes of computation time. Moreover, we derive some managerial insights, such as the impact of the number of warehouse levels and picking waves on the objective values. We find evidence that running the RMFS warehouse in a multi-level facility can substantially compromise the consolidation effort at packing stations since it leads to a higher number of split orders. Furthermore, splitting the planning horizon into multiple short waves can lead to a higher number of pod-to-station assignments and, thus, to a raised pod-movement workload for mobile robots.

Keywords Robotic mobile fulfillment systems · Order assignment · Multi-level warehouse · Multi-objective optimization · Lexicographic approach

1 Introduction

Warehouse operators traditionally face a trade-off. On the one hand, they aim for short delivery times and high throughput. On the other hand, labor and investment cost should not be excessive (Boysen et al. 2017). Order picking is one of the most labor and capital intensive warehouse processes and is estimated to account for about 55 % of total warehouse operating expenses (De Koster et al. 2007). While fully or partially automated picking systems promise to lower labor costs, they come at the price of substantial investment and comparatively low flexibility (Boysen et al. 2017). The usage of automated warehouse systems has nevertheless increased; however, it is still relatively low (Azadeh et al. 2019; Jaghbeer et al. 2020). Interest in partially automated robotic mobile fulfillment systems (RMFS) has substantially grown over recent years, especially in the context of e-commerce (Lamballais et al. 2017; Jaghbeer et al. 2020; Fragapane et al. 2021), which is characterized by small-size stock-keeping units (SKUs) and a high volume of orders with few lines each (Azadeh et al. 2019).

In comparison to conventional order picking systems, in which order pickers move to shelves in the warehouse to retrieve items, order picking in an RMFS increases productivity, because human pickers waste less time on unproductive walking since robots move the shelves containing the items to be picked while the human pickers are stationary at a picking station. RMFS consist of the following basic components: the pod locations on the warehouse floor (Fig. 1a), the mobile racks (pod) containing the items and the mobile robots themselves (Fig. 1b), and the picking stations (Fig. 1c). The rack locations indicate the inventory area where the pods are stored, typically in a grid layout. Each robot navigates through the warehouse using QR codes and guide strips. The mobile robots move both in the aisles and underneath the pods, allowing them to ferry entire pods between the storage area and the picking stations.

This work is part of a research project in cooperation with different German logistic providers and experts from the industry with the aim of identifying potential applications for RMFS instead of conventional manual, or fully automated order picking systems. Our discussions with industry partners showed that compared with a typical automated storage and retrieval system (AS/RS), a single-level RMFS suffers from

poorer space utilization. This is because individual pods are not as tall as the high-bays in typical AS/RS as a human picker must still be able to reach all items. To mitigate this, RMFS can be operated in a multi-level facility, which enables to exploit the height of the warehouse facility in a better way. On the downside, running RMFS on multiple levels can cause non-avoidable split orders, because some orders may contain a set of SKUs, which is not stored on a single level. Split orders lead to an additional consolidation effort, because partially picked orders must be consolidated later. For more details on the effect of the split orders in RMFS, we refer to Xie et al. (2021), while order consolidation is studied by Boysen et al. (2019b) among others.

A typical workflow of the order picking process in a multi-level RMFS can be characterized as follows: The warehouse consists of multiple levels, which are connected by conveyor systems. Each warehouse level is operated by a specific robot fleet and consists of a pod storage area with multiple pods as well as stationary picking stations (see Fig. 1a). Each pod consists of several shelves, where different SKUs are stored (see Fig. 1b). Once a specific SKU is required at a picking station, a mobile robot navigates to a pod hosting the ordered SKU, and brings it to the dedicated picking station. Particularly, it moves underneath the pod, lifts it, and carries it to the destination. Each picking station is operated by a stationary human picker, who retrieves the ordered SKUs from the pods and collects the picked items for each order into a special tote (see Fig. 1c). Once all required SKUs are picked from a pod, the robot transports the pod back to the storage area. After completing an order by picking all SKUs, the picker places the completed tote onto a conveyor belt to be transported towards the packing station, which is typically located on the first floor (see Fig. 1d). At the packing stations, orders are taken from the totes, packed into cardboard boxes, and transferred to the shipping area, where they are loaded into outgoing trucks. Moreover, packing stations are also used to consolidate split orders, which were picked at multiple picking stations.

Typical objectives of automated warehousing systems are mainly throughput, lead time, and operational efficiency (Jaghbeer et al. 2020). The continuous improvement of these aspects is currently a significant field of interest for research. Several problems resemble the planning steps in classic warehouses. The presence of mobile robots, however, changes decision problems fundamentally, such that established procedures are usually not applicable (Boysen et al. 2017). The essential operational problems to be tackled are:

SKU storage assignment on pods Which SKUs should be stored on which pod?
Order assignment Which picking station should handle which order?
Pod assignment From which pods should the SKUs be picked?
Order processing In what sequence should orders be processed at a station, and when should the pods visit the stations?
Pod storage assignment Where should each pod be placed on the warehouse floor?
Traffic management Which robot should perform which transport job at which time? Which path should it take?

This paper deals with the second and the third item on this list: The order and pod assignment problem, which we dub as OPAP. Given a set of orders to process, we

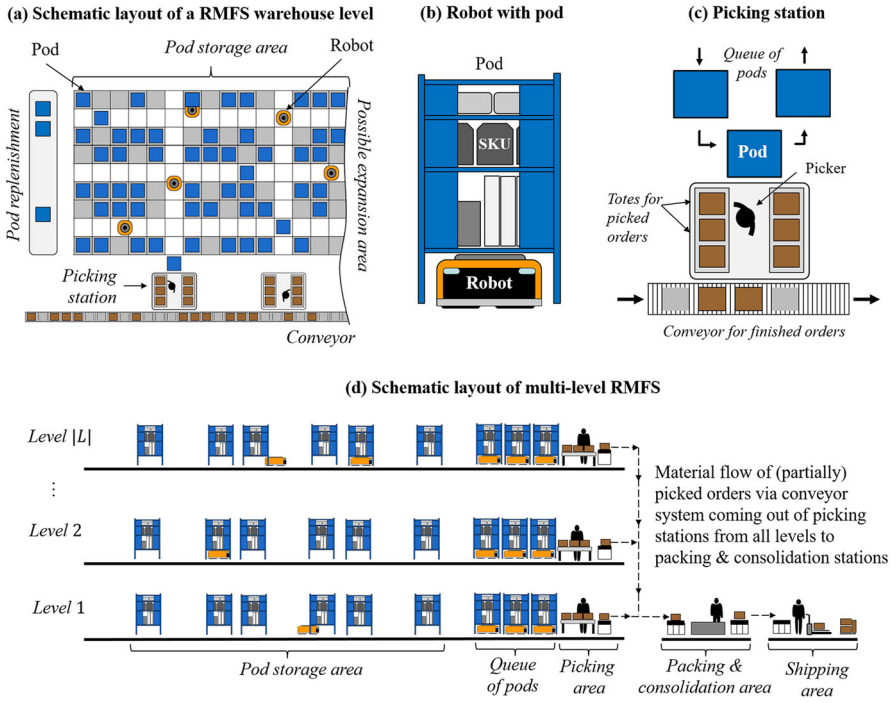


Fig. 1 Schematic depiction of an RMFS warehouse

want to answer which of a given set of picking stations should handle which order and from which pods the ordered articles must be picked.

In current practice, orders are typically assigned one after another using rules-of-thumb, greedily comparing similarities of currently processed and newly arrived orders (Wurman et al. 2008). Therefore, it stands to reason that a more holistic view considering all open orders at once will yield better results. The OPAP aims to fulfill all orders while balancing workload among all pickers and with as few resources (i.e., packers and robots) as possible. In contrast to the majority of the existing works, we explicitly consider limited storage space of the pods as well as multi-level warehouse layouts. We optimize towards three goals, which take workload-related aspects of the warehouse workers and mobile robots into consideration.

Balancing the order-processing workload for pickers The OPAP cannot influence the total effort of picking SKUs from pods since all orders must be fulfilled completely. Therefore, instead of minimizing the picking workload, our first goal is to balance the total workload over all pickers. In other words, we aim to distribute the workload evenly among all picking stations and keep all pickers equally busy, which implicitly also improves the cycle time.

Minimizing order-consolidation workload for packers In a multi-level warehouse, some orders cannot be picked at a single picking station, but must be split over multiple stations. Splitting an order is related to double-handling because partially picked orders must be consolidated later at a packing station. Our second goal is to minimize split orders to reduce order consolidation effort at packing stations.

Note that minimizing split orders minimizes the workload of workers at the packing stations and increases the overall efficiency of pickers, since each partial order entails some extra non-value-adding steps like preparing and dispatching a tote.
Minimizing workload for robots In RMFS, the order picking process generates workload not only for human workers but also for robots, which move pods with ordered SKUs between the storage area and picking stations. Therefore, our third goal is to minimize the workload of robots by minimizing the total number of pod movements. In other words, we want to assign orders and pods to stations in such a way that the picker can pick as many items from each pod as possible. The rationale is that the more SKUs can be picked per pod, the fewer pod moves are necessary to fulfill the orders. This way, minimizing the number of pod movements not only relieves the robots, but also increases the pickers' efficiency by reducing the inefficient setup times between pod switches at the stations, enabling pickers to concentrate on more productive tasks.

The contribution of this paper is as follows: We present the order and pod assignment problem in a multi-level RMFS warehouse. While order and pod assignment in single-level RMFS has been addressed in the literature in the past, very little work has been done on multi-level RMFS warehouses, despite their frequent use in practice. Moreover, our approach includes several realistic problem characteristics, such as limited pod capacities, workload balancing, and split orders. We formalize the planning problem as a mathematical optimization problem and provide a mixed-integer programming (MIP) model. Moreover, to solve large problem instances within an acceptable amount of time, we propose a matheuristic. Particularly, we first represent OPAP as a multicommodity flow network and propose an alternative path-based MIP formulation to solve the resulting multicommodity flow problem. Then, based on this formulation, we propose a heuristic path-reduction procedure, which offers additional flexibility to users by allowing them to heuristically reduce the problem size to the desired level and solve the reduced problem within a shorter computation time. Finally, we explore the computational performance of the proposed solution procedures and derive managerial insights, such as the impact of the number of warehouse levels and the length of the planning horizon on the proposed objective values.

The remainder of this paper is organized as follows. In Sect. 2, we discuss the pertinent literature. Section 3 formally defines the order and pod assignment problem, discusses the model assumptions, and provides the first MIP model. In Sect. 4, we reformulate the problem as a multicommodity flow problem, propose an alternative MIP model formulation, and, based on this, develop a matheuristic procedure. Section 5 presents the results of computational experiments, exploring the algorithmic performance of the proposed solution approaches, and gives some managerial insights. Finally, Sect. 6 concludes the paper.

2 Literature review

RMFS has attracted considerable attention in recent years. Beginning with the work of Wurman et al. (2008), several papers have been published that summarize and reflect

the current state of the art of RMFS through literature reviews. In the beginning, RMFS were considered as part-to-picker systems (Azadeh et al. 2019). Since the number of such systems has increased, the classification has to be reconsidered. Jaghbeer et al. (2020) classify RMFS as robot picker systems, which can be defined as a subcategory of automated order picking systems. Besides, an increasing number of papers deal with individual problems resulting from the use of RMFS, underlining the importance of efficient solution methods in the RMFS planning process.

A substantial body of literature on different operational planning problems in RMFS has been published lately. For works on SKU storage assignment on pods, we refer to e.g., Guan and Li (2018), Kim et al. (2020), Lamballais Tessensohn et al. (2020). The pod storage assignment problem during order processing is studied by Weidinger et al. (2018). Li et al. (2020) study the SKU and pod storage assignment problems, considering the energy consumption of the robots. Specifically, the authors solve the SKU storage assignment problem via association analysis and a clustering approach, propose a turnover-rate-based decentralized storage policy for pod storage assignment, and develop a novel order picking evaluation scheme with consideration of the energy consumption of the robots. The order processing problem at a single picking station was introduced by Boysen et al. (2017), which is extended for multiple picking stations by Wang et al. (2022). Regarding the traffic management of robots, Zou et al. (2017) deal with the problem of assigning robot to picking stations, while path planning algorithms for mobile robots are proposed by Merschformann et al. (2017). Note that these are only some works on precedent and subsequent planning problems of OPAP. We refer to Azadeh et al. (2019) and Boysen et al. (2019a) for an extensive literature review on robotized and automatized warehouse systems.

In the following, we focus on the works similar to ours, which also deal with the order assignment and pod assignment problems. To distinguish our approach from the existing models and solution methods from the literature, we highlight the novel features and characteristics of our model, which are not considered in existing works. In particular, we consider the following features:

Consideration of a multi-level warehouse Pods and the items on them are spread over multiple storeys. Not all picking stations are on the same level as all pods.
SKU capacity restrictions on pods Since pods have limited storage capacity, the number of stored items on each pod is limited, i.e., just because an SKU is on a specific pod, does not mean that an infinite number of items of the SKU can be picked from that pod.
Balancing picking workload among all pickers The workload should be distributed among the picking stations as equally as possible.
Consideration of split orders If an order consists of multiple lines, items can be picked at different stations; split orders are then consolidated at packing stations.

The results of our observation are summarized in Table 1. Concretely, it commences with the work of Merschformann et al. (2019), who deal with multiple planning problems at the operational level. Specifically, they cover the following decision problems: order assignment (including pick order assignment, replenishment order assignment), task creation (including pick pod selection, replenishment pod selection, and pod storage assignment), task allocation, and path planning. The authors propose several

Table 1 Delimitation of our work from existing related works

Existing literature	Multi-level warehouse	SKU capacity restrictions on pods	Balancing picking workload	Split orders	Method
Merschformann et al. (2019)	○	•			SIM, HEU
Valle and Beasley (2021)		•	○		MIP, HEU
Wang et al. (2022)			○		MIP, HEU, SIM
Xie et al. (2021)			•	•	MIP, SIM
Our work	•	•	•	•	MIP, SIM

Legend: ○ Partial consideration, • Full consideration
HEU heuristics, MIP mixed-integer programming, SIM simulation

decision-rule-based heuristics for each planning problem and evaluate their performance via a simulation model. The proposed planning problems also cover the order assignment and pod assignment problems, and there are partial commonalities with the OPAP. The simulation environment "RAWSim-O", used by Merschformann et al. (2019) and introduced by Merschformann et al. (2018), has an opportunity to model a multi-level warehouse. However, the algorithms proposed by Merschformann et al. (2019) do not seem to consider multi-level warehouse layouts. Similarly to OPAP, the authors consider the limited number of stored SKUs on pods, however, they do not aim at balancing the picking workload among all pickers and do not allow orders to be split among multiple stations.

At the first glance, the works of Valle and Beasley (2021), Wang et al. (2022), and Xie et al. (2021) seem to have the most things in common with ours. However, after looking into the details, there are some important differences, which we describe below.

Valle and Beasley (2021) solve three problems, order allocation, rack assignment, and rack sequencing in an integrated manner. The authors formulate a MIP model and propose two matheuristics. In particular, they decompose the problem into two sub-problems, whereas the order and rack allocation problems are solved in the first stage and the rack sequencing problem in the second stage. In contrast to our model, their model does not consider multiple levels and split orders. Moreover, instead of aiming at an equal distribution of the total picking workload among all pickers through the objective function, their model ensures that the number of assigned orders to each picker $p \in P$ equals the predefined picker capacity C_p.

Wang et al. (2022) extend the work of Valle and Beasley (2021) by allowing pods to visit more than one picking station before returning to the storage area. The authors formulate an integer programming model and propose a two-phase heuristic, where the order assignment problem is solved in the first stage and the order and rack sequencing problem in the second stage. Moreover, Wang et al. (2022) explore the impact of different aspects such as the number and capacity of picking stations, SKU diversity, and queue length on the system's performance. Due to considering an additional problem of rack sequencing, the resulting integrated problem is more complex than

OPAP, which is why the proposed approaches can only solve small instances (with up to 100 orders, 100 pods, 10 SKUs, and 4 picking stations). Moreover, in contrast to our model, the model of Wang et al. (2022) does not consider a multi-level layout. The authors also assume that if a pod contains a specific SKU, there is always enough quantity of items to fulfill all orders. Regarding the picker workload, the proposed model ensures that the number of assigned orders to each station does not exceed the predefined constant capacity C. Finally, by ensuring that each order is assigned to exactly one picking station, the authors do not allow split orders.

The problem formulated by Xie et al. (2021) is maybe the most similar to OPAP in the published literature. The authors solve the order and pod assignment problem and allow orders to be picked among several picking stations. Particularly, Xie et al. (2021) propose MIP models for two types of split orders: split-among-stations, and split-over-time, and explore their impact on the solution quality in a simulation study. However, in contrast to our model, the model of Xie et al. (2021) do not consider a multi-level warehouse layout and assume that the number of items of the same SKU is not limited. Moreover, in contrast to Xie et al. (2021), apart from a standard MIP model, we also develop a matheuristic, which allows us to solve larger instances. Specifically, our approach can solve problem instances with up to 1200 orders, while the largest instances of Xie et al. (2021) contain 250 orders.

To summarize, we distinguish our approach from the extant literature by considering the following points: The number of items on the pods is physically limited; the warehouse has multiple levels; we deal with a multi-criteria objective function with three workload-related objectives; and we develop a solution method beyond the straightforward application of a default solver, allowing us to solve instances of realistic size.

3 Order and pod assignment problem

The order and pod assignment problem (OPAP) is concerned with the following question: given a set of orders to be filled during the planning horizon, which picking station should handle which order, and which SKUs should be picked from which pod? When selecting pods to pick ordered SKUs at a station, it must be ensured that the total amount of picked items from each pod cannot exceed the number of stored SKUs. Moreover, considering the multi-level warehouse layout, a pod can only be assigned to a station if both of them are on the same level. Since orders can arrive at any time of the day, the order and pod assignment might be re-planned multiple times during the day. Therefore, at the beginning of the planning horizon, some stations may already have some orders and pods assigned from the previous planning run.

Note that OPAP is only a sub-problem in the whole order fulfillment process. It is outside the scope of OPAP to assign specific transport jobs to individual robots, or to track the order consolidation process at the packing stations, which also depend on factors other than order and pod assignment (e.g., the exact sequence of the processed orders and pod visits at the stations, replenishment request processing, current positions of the robots, etc.). Therefore, we achieve our workload-related goals by introducing the following substitute objectives:

- To balance the order-handling workload among all pickers, we aim to keep all picking stations about equally busy by minimizing the maximum workload at the busiest picking station. Note that processing of an order at a picking station includes not only the picking of items from the pods but also order preparation and follow-up tasks, like opening a new tote, packing, labeling, and placing it on the conveyor belt, etc. Therefore, our first objective is to minimize the maximum number of assigned orders over all stations.
- Our second goal is to minimize the order consolidation effort for packers, which can be achieved by avoiding unnecessary split orders. Therefore, our second objective aims to assign orders to as few stations as possible by minimizing the total number of order-to-station assignments.
- Finally, our third goal is to minimize the workload for the robots, which can be achieved by minimizing the total number of pod movements. The exact number of pod movements depends on the schedule of pod visits at the stations and may be influenced by many factors such as pod sequencing or routing policies (e.g., enabling pods to visit multiple stations one after each other or bringing pods to storage area after each pick process, etc.), which are outside the scope of OPAP. Therefore, our third substitute objective minimizes the total number of required pod-to-station assignments, which is the lower bound on the total number of pod movements.

3.1 Formal description

Let $S = \{1, \ldots, |S|\}$ denote the set of all SKUs stored in the warehouse (SKU index s), and $P = \{1, \ldots, |P|\}$ a set of all picking stations (station index p). SKUs are stored in a set of pods $R = \{1, \ldots, |R|\}$ (pod index r), so that for each pod $r \in R$ and SKU $s \in S$, we have a parameter $a_{rs} \in \mathbb{N}_0$ denoting the number of stored items of SKU s on pod r. Moreover, let L denote the set of warehouse levels (level index l), where each level $l \in L$ contains a subset of pods $R_l \subseteq R$ and picking stations $P_l \subseteq P$. We assume that the warehouse levels are isolated, meaning that robots are incapable of carrying pods between levels. Further, let $O = \{1, \ldots, |O|\}$ be a set of orders to be processed during the planning horizon (order index i), where each order $i \in O$ consists of several ordered items of different SKUs that have to be packed and shipped to a customer together. Hereby, parameter $o_{is} \in \mathbb{N}_0$, denotes the number of ordered items of SKU s in order i. Finally, at the beginning of the planning horizon, some stations $p \in P$ may already have some fixedly assigned pods $\bar{R}_p \subseteq R$ and (part of) orders $\bar{O}_p \subseteq O$ from a preceding planning run (or picking wave). This setting enables planning on rolling horizons, allowing us to update order and pod assignments multiple times throughout the day. The notation of all input parameters of OPAP is summarized in Table 2.

A solution Ω to the OPAP is defined by elements $\omega_{spri} \in \mathbb{N}_0$, denoting how many items of SKU $s \in S$ are picked at station $p \in P$ from pod $r \in R$ for order $i \in O$.

Table 2 Input parameters for OPAP

S	Set of SKUs (index s)
O	Set of orders (index i)
L	Set of warehouse levels (index l)
R	Set of pods (index r)
P	Set of picking stations (index p)
$\bar{O}_p$	Set of fixedly assigned orders to station p ($\bar{O}_p \subseteq O$)
$\bar{R}_p$	Set of fixedly assigned pods to station p ($\bar{R}_p \subseteq R$)
o_{is}	Number of ordered items of SKU s in order i
a_{rs}	Number of stored items of SKU s on pod r
R_l	Set of pods on level l ($R_l \subseteq R$)
P_l	Set of stations on level l ($P_l \subseteq P$)

Moreover, for notational convenience, we formally define

$$\eta(i, p) = \begin{cases} 1, & \text{if } \exists s \in S, r \in R : \omega_{spri} > 0 \\ 0, & \text{otherwise} \end{cases}$$

to indicate whether (part of) order $i \in O$ is processed at station $p \in P$. Similarly, we define

$$\rho(p, r) = \begin{cases} 1, & \text{if } \exists s \in S, i \in O : \omega_{spri} > 0 \\ 0, & \text{otherwise} \end{cases}$$

to indicate whether pod $r \in R$ is required at station $p \in P$.

We say that a solution is feasible, if and only if it satisfies the following conditions:

- Every order is fulfilled completely, i.e., for each order $i \in O$ and SKU $s \in S$, it must hold that $\sum_{p \in P} \sum_{r \in R} \omega_{spri} = o_{is}$.
- No more items can be picked from a pod than are actually stored on it, i.e., for each pod $r \in R$ and SKU $s \in S$, it must hold that $\sum_{p \in P} \sum_{i \in O} \omega_{spri} \leq a_{rs}$.
- A pod can only be assigned to a station if both of them are on the same level, i.e., for each pod $r \in R$ and station $p \in P$, it can hold that $\rho(p, r) = 1$ if and only if $\exists l \in L : r \in R_l, p \in P_l$.
- The fixed pod-to-station and order-to-station assignments from the previous planning run must be considered in the current planning horizon, i.e., for each station $p \in P$ and order $i \in O$ it must hold that $\eta(i, p) = 1$ if $i \in \bar{O}_p$. Similarly, for each station $p \in P$ and pod $r \in R$, it must hold that $\rho(p, r) = 1$ if $r \in \bar{R}_p$.

Among all feasible solutions, we seek the optimal solutions that minimize the following three objectives:

- *Maximum number of assigned orders over all stations:* $f_1 = \max_{p \in P} \left\{ \sum_{i \in O} \eta(i, p) \right\}$
- *Total number of order-to-station assignments:* $f_2 = \sum_{i \in O} \sum_{p \in P} \eta(i, p)$, and
- *Total number of pod-to-station-assignments:* $f_3 = \sum_{r \in R} \sum_{p \in P} \rho(p, r)$.

At its core, OPAP is a multi-objective optimization problem (MOOP). In the past forty years, there has been a vast amount of theoretical, methodological, and applied works dealing with MOOPs (for some methodological approaches see e.g., Zionts and Wallenius (1976); Steuer (1976); Ecker and Kouada (1978); for examples of meta-heuristic applications on MOOP problems see e.g., Hoseinpour et al. (2020, 2021)). Most of the existing MOOP approaches find the set of all *Pareto-optimal* solutions, from which decision-makers must choose the final solution based on their own preferences (Deb 2014; Cui et al. 2017; Gunantara 2018). This requires a careful judgement of the tradeoff between solutions from decision-makers if multiple Pareto optimal solutions are found. Our industry partners do not consider this desirable. Instead, we only have apriori high-level information about the priority order of these objectives. Therefore, we decided to deal with the multi-criteria objective function via the lexicographic approach (Isermann 1982), which is a commonly used approach to tackle multi-objective optimization problems (Al Chami et al. 2019). Particularly, we use the *lexicographic objective* from the multi-objective optimization package of ILOG CPLEX Optimization Solver, which receives a pre-defined order among the various objective functions as input (IBM 2021). Note that the lexicographic approach converts a MOOP into a single objective optimization problem, and provides optimal solutions only for the pre-defined lexicographic order, instead of finding a set of Pareto-optimal solutions.

After discussions with several industry representatives, we use the lexicographic order $f_1 \gg f_2 \gg f_3$ of the three objectives. This implies that we first seek solutions that minimize the primary objective f_1, leading to the minimal number of assigned orders to the busiest station, i.e., balancing the workload among the human pickers. Subsequently, we optimize the secondary objective f_2, i.e., among all solutions that are optimal regarding f_1, we look for the solutions which assign orders to as few stations as possible, i.e., minimizing split orders. Finally, among all optimal solutions with regard to the primary and the secondary objectives, we seek optimal solutions for the tertiary objective, which assign orders and pods to stations in such a way that the minimal number of pods is required over all stations, i.e., the workload for the robots is minimized.

Regarding computational complexity, OPAP is NP-hard even if the warehouse has only a single level, as we show in the following.

Theorem: OPAP is in the class of NP-hard optimization problems, even if only $|L| = 1$ level with only $|P| = 2$ stations exists.

The reduction is from the partition problem, which is well-known to be NP-complete (Garey and Johnson 1979), and can be described as follows:

Partition: Given a set S and weights $w(s) \in \mathbb{N}_0$ for all elements of S. Is there a subset $S' \subset S$ such that $\sum_{s \in S'} w(s) = \sum_{s \in S \setminus S'} w(s)$?

Proof: For an arbitrary instance of partition, we create a set of SKUs corresponding with set S, as well as one pod for each element $s \in S$, which solely and exclusively provides the SKU representing s. Additionally, for each SKU (representing an element $s \in S$), we create $w(s)$ orders which have a demand of the respective SKU only. Further, two picking stations are introduced to our problem instance. By searching for an optimal solution of the resulting OPAP instance with maximal station workload of

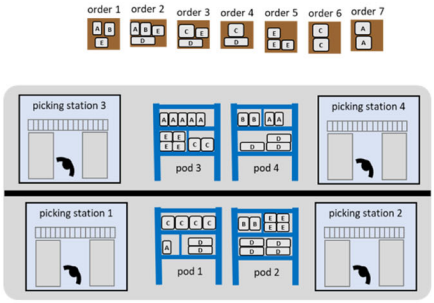
$f_1 = \frac{\sum_{s \in S} w(s)}{2}$, the total number of order-to-station-assignments of $f_2 = \sum_{s \in S} w(s)$, and the total number of required pods $f_3 = |S|$, we translate each problem instance of partition into an instance of OPAP in polynomial time and NP-hardness of OPAP is proven. □

3.2 Example OPAP and optimal schedules

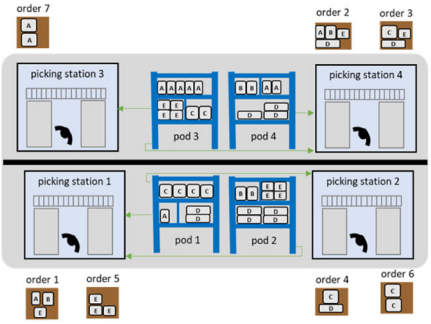
Consider an example OPAP instance with $|S| = 5$ SKUs $S = \{A, B, C, D, E\}$, stored on $|R| = 4$ pods, which are allocated on $|L| = 2$ warehouse floors. On the first level, there are pods 1 and 2 ($R_1 = \{1, 2\}$), where SKUs $\{A, C, C, C, C, D, D\}$ and $\{B, B, D, D, D, D, E, E, E, E\}$ are stored, respectively. Pods 3 and 4 stand on the second level ($R_2 = \{3, 4\}$) and contain SKUs $\{A, A, A, A, A, C, C, E, E, E, E\}$ and $\{A, A, B, B, D, D, D\}$, respectively. In the planning horizon, there are $|O| = 7$ orders with ordered SKUs $\{A, B, E\}$, $\{A, B, D, E\}$, $\{C, D, E\}$, $\{C, D\}$, $\{E, E, E\}$, $\{C, C\}$, and $\{A, A\}$. The orders can be picked at $|P| = 4$ picking stations, from which the stations 1 and 2 are located on the first floor ($P_1 = \{1, 2\}$), and the stations 3 and 4 on the second floor ($P_2 = \{3, 4\}$). For the sake of simplicity, we do not consider any fixed order-to-station and pod-to-station assignments from the previous planning runs, i.e., $\bar{R}_1 = \bar{R}_2 = \bar{R}_3 = \bar{O}_1 = \bar{O}_2 = \bar{O}_3 = \{\}$. The example problem instance with all the input parameters is depicted in Fig. 2a. To illustrate the importance of our lexicographic order of the three objectives, we present optimal solutions for two different objective priorities.

Figure 2b depicts an optimal solution for the lexicographic order $f_1 \gg f_2 \gg f_3$ (used in this paper), i.e., minimizing the workload of the busiest picking station f_1 is handled as a primary objective, the number of order-to-station-assignments f_2 as a secondary objective and the number of total required pods f_3 as a tertiary objective. In the illustration, the orders are placed next to the stations they are assigned to. In the optimal solution for this particular lexicographic order, orders 1 and 5 are assigned to station 1; orders 4 and 6, to station 2; order 7, to station 3; and orders 2 and 3, to station 4. The pod-to-station assignments are marked via (green) arrows. Specifically, station 1 must be visited by pods 1 and 2. Thereby, SKU $\{A\}$ for order 1 is picked from pod 1; SKUs $\{B, E\}$ for order 1 and SKUs $\{E, E, E\}$ for order 5, from pod 2. Station 2 is visited by pod 1, station 3 by pod 3, both of which cover all SKUs picked at those stations. Finally, pods 3 and 4 are required at station 4. SKUs $\{A, E\}$ for order 2 and SKUs $\{E, C\}$ for order 3 are picked from pod 3; SKUs $\{B, D\}$ for order 2 and $\{D\}$ for order 3, from pod 4. Regarding the objective values, this solution leads to the maximum number of assigned orders per station of $f_1^* = 2$, $f_2^* = 7$ order-to-station assignments (i.e., 0 split orders), and $f_3^* = 6$ pod-to-station assignments.

To illustrate the importance of the lexicographic order, Fig. 2c depicts an optimal solution for an alternative order $f_1 \ll f_2 \ll f_3$ with minimizing the number of pod-to-station-assignments f_3 as the primary objective; the number of order-to-station-assignments f_2 as the secondary objective; and the maximum number of assigned orders per station f_1 as the tertiary objective. This sequence of objectives leads to a different optimal solution. Namely, this time, orders 1, 3, and 4 are handled at station 1, and orders 2, 5, 6, and 7 at station 3, while stations 2 and 4 are idle without handling



(a) Example OPAP instance.



(b) opt. solution for lexicographic order: $f_1 \gg f_2 \gg f_3$.

(c) optimal solution for lexicographic order: $f_1 \ll f_2 \ll f_3$.

Fig. 2 Example OPAP instance with optimal solutions for two alternative objective priorities

any orders, and not being visited by any pod. Compared with the former solution, this solution leads to an improvement in the total number of required pod-to-station assignments $f_3^* = 4$ (handled as primary objective). However, it worsens the workload balance among the stations, leading to the maximal number of assigned orders at the busiest station $f_1^* = 4$ (handled as a tertiary objective), while the number of order assignments, handled as a secondary objective, remains at the same optimal value $f_2^* = 7$.

3.3 Model assumptions

Like all other models, our model is based on some simplifying assumptions, which we discuss below:

A1 : **No consideration of the replenishment** process In OPAP, we only focus on picking operations and assume that no pod replenishment takes place during the planning horizon of OPAP. Pod replenishment is outside the scope of OPAP and is assumed to be planned separately. While simultaneous planning of picking

and replenishment operations in an integrated manner could increase the total efficiency of the system, the resulting integrated problem would suffer from higher computational complexity. Moreover, due to the typical arrival times of inbound trucks at fulfillment centers, pods are usually batchwise refilled overnight, so that time-critical picking operations are executed during the day. Note that if pods are refilled during the day between the picking waves, this can be easily considered in OPAP by adjusting the pod inventory parameters a_{rs} for the subsequent picking wave. As the fill level parameters can be updated quite frequently, we believe that our assumption seems pardonable, if individual picking waves are short, which tends to be the case in many warehouses.

A2 : **Isolated warehouse levels** We do not allow robots and pods to switch warehouse levels, e.g., via an elevator. Such switching would undoubtedly add some flexibility to the order picking process because each picking station would have access to the whole warehouse, which would allow avoiding split orders. On the downside, it would cause additional coordination effort like balancing the number of robots and pods among different levels, scheduling elevators, etc. Moreover, although transporting whole pods between different levels via elevator could avoid order splitting, this would require additional space in the warehouse for elevators and consume more energy than sending smaller totes filled with partially picked orders via existing material handling system (e.g., conveyor belts). Therefore, we assume that all levels are isolated by restricting pod-to-station assignments if they are not on the same level. Note that in this way, a level can also be interpreted as an isolated picking zone, which is a common policy in the order-picking literature and practice Roy et al. (2019).

A3 : **Measuring picker workload as the number of processed orders** We use the number of assigned orders as a substitute measure of the picker workload. This is clearly an imperfect reflection of the actual workload. At first glance, the number of picked articles may seem to reflect the picker's workload in a better way. However, the total processing time of an order for a picker depends not only on the number of SKUs to be picked but also on other factors like the accessibility of individual SKUs on a given pod, their size, and weight, or even the experience of the picker. Moreover, processing an order includes not only the picking tasks themselves but also pre-processing (e.g., preparing a new tote) and follow-up steps (e.g., labelling and dispatching the tote) with a fixed duration, which is not affected by the number of picked SKUs. Therefore, after discussions with our industry partners, we agreed that minimizing the number of processed orders in f_1 is a good stand-in for minimizing the pickers' actual workload. This assumption is pardonable, especially in the e-commerce context, where most customer orders are quite small (mostly with only a single or a handful of items, Boysen et al. (2019a)) and the number of ordered SKUs per order does not vary too much.

A4 : **No consideration of the limited capacity of packing stations** Scheduling packing and consolidation stations is outside the scope of OPAP. The exact capacity limit of the packing stations at any given point in time is hard to keep track of, since a fulfillment center typically consists of multiple such stations, which are connected by conveyors. Therefore, we deal with this assumption by minimizing the total consolidation effort at packing in our secondary objective f_2. One

Table 3 Additional notation for the MIP-intuitive model

Parameters	
h_{rp}	1, if pod r and station p are on a same level; 0, otherwise
Variables	
x_{ip}	Binary variable: 1, if (part of) order i is picked at station p; 0, otherwise
$\mathcal{X}$	Set of x variables: $\mathcal{X} = \{x_{ip} \mid i \in O, p \in P\}$
y_{rp}	Binary variable: 1, if pod r visits station p; 0, otherwise
$\mathcal{Y}$	Set of y variables: $\mathcal{Y} = \{y_{rp} \mid r \in R, p \in P\}$
z_{spri}	Continuous variable: number of items of SKU s picked at station p from pod r for order i
$\mathcal{Z}$	Set of z variables: $\mathcal{Z} = \{z_{spri} \mid s \in S, p \in P, r \in R, i \in O\}$

industry representative described to us an alternative workflow of their multi-level warehouse, where partially picked orders are sequentially processed and consolidated at different levels (top-down) before being forwarded to packing stations. Note that minimizing the number of split orders appears to be meaningful also in such a case, since split orders are still associated with double-handling at different warehouse levels.

A5 : **No consideration of pod revisits at picking stations** In order to minimize the workload for robots, our tertiary objective f_3 minimizes the total number of pod-to-station assignments, which is only a lower bound on the actual number of pod visits (i.e., if pod re-visits at stations are relaxed). After solving the order and pod scheduling problem at picking stations in the following step, some pods may need to visit the same stations more than once. Depending on the order capacity at the picking stations, not every order can be processed simultaneously at the picking stations. Although our model is undoubtedly a simplification of reality, we believe that our assumption is pardonable because of the following reason: OPAP is typically solved for relatively short picking waves and re-optimized multiple times throughout the day. Therefore, the probability of such pod revisits is the lower, the shorter the actual planning horizon is. However, note that the actual workload for the robots and the energy consumption are influenced not only by OPAP, but also by the other planning problems, such as SKU storage assignment on pods or pod storage assignments (Li et al. 2020), or the traffic-management-related planning problems for robots.

3.4 Intuitive model

In this section, we will introduce an intuitive MIP model (denoted as MIP-intuitive) that can be applied to solve the OPAP via a default solver.

In the following, we will present the notation for the (additional) parameters and variables of the MIP-intuitive model, which are summarized in Table 3. First, for each pod $r \in R$ and station $p \in P$, we compute a binary parameter h_{rp} in the model pre-processing step, which has a value 1, if pod r and station p are on the same level, i.e., $\exists l \in L : r \in R_l \wedge p \in P_l$, 0 otherwise. Moreover, for each SKU $s \in S$, picking station $p \in P$, pod $r \in R$, and order $i \in O$, we define a continuous variable z_{spri} that

indicates the number of picked items of SKU s at station p from pod r for order i. Note that we do not explicitly define variables z_{spri} as integer variables. However, in optimal solutions, they will still always have integer values. Moreover, for each order $i \in O$ and station $p \in P$, we introduce a binary variable x_{ip} indicating whether (part) of order i is processed at station p ($x_{pi} = 1$) or not ($x_{rp} = 0$). Similarly, for each pod $r \in R$ and station $p \in P$, we define a binary variable y_{rp} which takes the value 1, if pod r is required at station p, 0 otherwise.

MIP-intuitive consists of objective function (1) and Constraints (5)-(14).

$$\text{Minimize } \mathcal{F}(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) = f_1 \text{ and } f_2 \text{ and } f_3 \tag{1}$$

$$f_1 = \max_{p \in P} \left\{ \sum_{i \in O} x_{ip} \right\} \tag{2}$$

$$f_2 = \sum_{i \in O} \sum_{p \in P} x_{ip} \tag{3}$$

$$f_3 = \sum_{r \in R} \sum_{p \in P} y_{rp} \tag{4}$$

subject to

$$\sum_{s \in S} \sum_{r \in R} z_{spri} \leq M \cdot x_{ip} \qquad \forall i \in O, p \in P \tag{5}$$

$$\sum_{s \in S} \sum_{i \in O} z_{spri} \leq M \cdot y_{rp} \qquad \forall r \in R, p \in P \tag{6}$$

$$\sum_{r \in R} \sum_{p \in P} z_{spri} = o_{is} \qquad \forall i \in O, s \in S \tag{7}$$

$$\sum_{i \in O} \sum_{p \in P} z_{spri} \leq a_{rs} \qquad \forall r \in R, s \in S \tag{8}$$

$$y_{rp} \leq h_{rp} \qquad \forall r \in R, p \in P \tag{9}$$

$$x_{ip} = 1 \qquad \forall p \in P, i \in \overline{O}_p \tag{10}$$

$$y_{rp} = 1 \qquad \forall p \in P, r \in \overline{R}_p \tag{11}$$

$$x_{ip} \in \{0; 1\} \qquad \forall i \in O, p \in P \tag{12}$$

$$y_{rp} \in \{0; 1\} \qquad \forall r \in R, p \in P \tag{13}$$

$$z_{spri} \in \mathbb{R}^+ \qquad \forall s \in S, r \in R, p \in P, i \in O \tag{14}$$

Multi-objective function (1) minimizes three objectives (2), (3), and (4) in this lexicographic order. The primary objective (2) minimizes the maximum number of the assigned orders over all stations; the secondary objective (3), the total number of the order-to-station assignments; the tertiary objective (4), the total number of required pods over all stations. Constraints (5) force the binary variables x_{ip} to be set to the value 1, if at least one ordered SKU for order i is picked at station p. Similarly, constraints (6) set binary variables y_{rp} on the value 1, if pod r is required at station p, i.e., if at

least one SKU is picked from pod r at station p. Equations (7) ensure that the orders are fulfilled completely, i.e., each ordered SKU must be picked from any pod at any station. Inequalities (8) guarantee that for each pod $r \in R$, the total number of picked items of SKU s never exceeds the number of available SKUs s on pod r. Constraints (9) ensure that pods $r \in R$ can be assigned to stations $p \in P$ only if they are on the same level. Constraints (10) and (11) take care, that each station $p \in P$ receives all fixedly assigned orders and pods from the previous planning step in the current planning. Finally, Constraints (12)–(14) define the domain of the variables.

4 Solution procedure

Our computational experiment reveals that the size of the MIP-intuitive model grows fast so that a modern default solver cannot be applied even for medium-sized problem instances. Keeping in mind that OPAP is an operational planning problem, which is re-optimized multiple times throughout the day, it is very important to have a fast solution approach that can handle realistically sized problem instances in short computational time. Therefore, we develop a specialized solution procedure that can provide acceptable solutions for large problem instances within short time. Particularly, our solution procedure is based on a transformation of OPAP into a type of multicommodity flow (MCF) problem. Therefore, in Sect. 4.1, we first show how OPAP can be represented as an MCF network. Later, in Sect. 4.2, we propose a path-based MIP model for the resulting MCF problem (denoted as MIP-MCFP). Finally, in Sect. 4.3, we propose a heuristic path-reduction scheme that pre-selects a subset of promising paths using decomposition of OPAP into two handier sub-problems, which enables users to apply the MIP-MCFP model as a heuristic.

4.1 Multicommodity flow network representation

In this section, we show how OPAP can be represented as an MCF network, which serves as a foundation for our second MIP model and the heuristic procedure.

In a classical multicommodity flow problem (MCFP), we are given a network, represented as a graph $G(V, E)$ with V as a set of vertices (or nodes), and E as a set of directed edges. Moreover, there is a set of commodities K, and each commodity $k \in K$ features an origin $O^k \subset V$ and a destination node $D^k \subset V$, as well as the demand quantity $d^k \in \mathbb{N}_0$. An edge $(i, j) \in E$ represents the freight asset from node $i \in V$ to node $j \in V$ and is associated with the capacity of the flow $\kappa_{(i,j)}$, fix-charge costs $c_{(i,j)}$, which occur when the edge is used, as well as variable costs $\phi_{(i,j)}$ per unit of flow. A feasible solution of MCFP consists of such a subset of edges and the amount of flow on these edges that all commodities $k \in K$ are shipped completely from the origin nodes O^k to the destination nodes D^k, and the amount of flow on each edge $(i, j) \in E$ does not exceed the edge capacity $\kappa_{(i,j)}$. Among all feasible solutions, we seek the optimal solution that minimizes the total costs (Barnhart et al. 2000).

In our case, the MCF network, representing OPAP, has a specific structure. First, the network consists of five layers, i.e., vertex set V is divided into five disjoint subsets

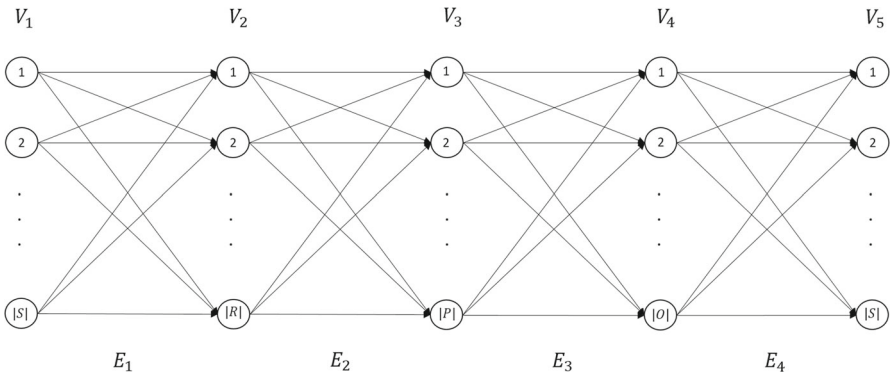
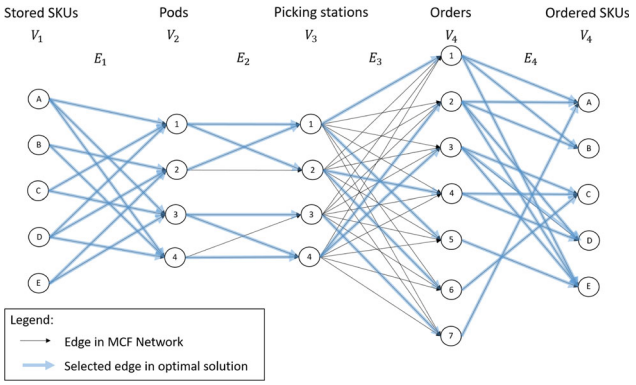


Fig. 3 Representation of OPAP as MCF network

$V_1, \ldots, V_5$ ($V = \cup_{i=1}^{5} V_i$), and edge set E into four disjoint subsets $E_1, \ldots, E_4$ ($E = \cup_{i=1}^{4} E_i$, where an edge from each subset E_j connects a node from V_j to a node in V_{j+1}. Second, some edges $(i, j) \in E$ have no capacity restrictions $\kappa_{(i,j)}$, while others have upper bounds $\overline{\kappa}_{(i,j)}$ or lower bounds $\underline{\kappa}_{(i,j)}$ on flows. Finally, the fixed charge costs for adding edges may have only values 0 or 1, and there are no variable flow costs, i.e., $c_{(i,j)} \in \{0, 1\}$, $\phi_{(i,j)} = 0, \forall (i, j) \in E$. Figure 3 schematically depicts an OPAP-related MCF network.

In the following, we describe how OPAP can be represented as an MCF network:

- Each SKU $s \in S$ in OPAP corresponds to a commodity k in MCFP and is associated with a source node $s \in V_1$ and a destination node $s \in V_5$. Thus, each of the subsets V_1 and V_5 contain $|S|$ vertices.
- Source nodes $s \in V_1$ are connected with pod nodes $r \in V_2$ in the second layer. Thereby, G contains an edge $(s, r) \in E_1$ if SKU s is stored on pod r. Selecting an edge $(s, r) \in E_1$ in the solution represents picking SKU s from pod r. Each edge $(s, r) \in E_1$ has a flow capacity of $\overline{\kappa}_{(r,s)} = a_{rs}$ that cannot be exceeded, meaning that we cannot pick more SKUs s from pod r than the number of stored items of SKU s on pod r a_{rs}. Finally, there are no fixed-charge costs for adding edges, i.e., $c_{(s,r)} = 0, \forall (s, r) \in E_1$.
- Pod nodes from the second layer $r \in V_2$ are connected with station nodes in the third layer $p \in V_3$. Selecting an edge $(r, p) \in E_2$ in a solution corresponds to an assignment of pod r to station p. Thereby, G contains an edge $(r, p) \in E_2$ between a pod node $r \in V_2$ and a station node $p \in V_3$, if and only if pod r can actually serve station p (i.e., if they are on the same level). Adding an edge $(r, p) \in E_2$ is related to a fixed-charge cost of $c_{(r,p)} = 1$, and there is no capacity restriction of the flow on these edges.
- Station nodes $p \in V_3$ are connecting with order nodes $i \in V_4$, and the edge $(p, i) \in E_3$ represents an assignment of order i to station p. There is an edge $(p, i) \in E_3$ between each station node $p \in V_3$ and an order node $i \in V_4$ with fixed-charge cost of $c_{(p,i)} = 1$ and no flow restrictions.
- Finally, each order node $i \in V_4$ is connected with a sink node $s \in V_5$, if in OPAP, order $i \in O$ contains SKU $s \in S$. Hereby, the flow on each edge $(i, s) \in E_4$ cannot be lower than the number of items of SKUs s in order i, i.e., $\underline{\kappa}_{(i,s)} = o_{is}$,



(a) MCF network for OPAP example.

Path	Flow
$A \to 1 \to 1 \to 1 \to A$	1
$B \to 2 \to 1 \to 1 \to B$	1
$E \to 2 \to 1 \to 1 \to E$	1
$A \to 3 \to 4 \to 2 \to A$	1
$B \to 4 \to 4 \to 2 \to B$	1
$D \to 4 \to 4 \to 2 \to D$	1
$E \to 3 \to 4 \to 2 \to E$	1
$C \to 3 \to 4 \to 3 \to C$	1
$D \to 4 \to 4 \to 3 \to D$	1
$E \to 3 \to 4 \to 3 \to E$	1
$C \to 2 \to 1 \to 4 \to C$	1
$D \to 2 \to 1 \to 4 \to D$	1
$E \to 2 \to 1 \to 5 \to E$	3
$C \to 1 \to 2 \to 6 \to C$	2
$A \to 3 \to 3 \to 7 \to A$	2

(b) Optimal paths.

Fig. 4 MCF Network for example OPAP instance with selected paths in the heuristic solution

meaning that orders must be fulfilled completely. The edges from the last layer do not have any fix-charge costs, i.e., $c_{(i,s)} = 0, \forall (i, s) \in E_4$.

In order to consider fixed pod-to-station and order-to-station assignments of OPAP from the previous planning run, we fix the corresponding edges and force them to be selected. Particularly, edges $(r, p) \in E_2$ (or $(p, i) \in E_3$) must be selected if $r \in \bar{R}_p$ (or $i \in \bar{O}_p$) holds.

Example (cont.): Fig. 4 visualizes the MCF network G and its optimal solution for the example problem instance from Sect. 3.2. The edge $(A, 1) \in E_1$ in G indicates that in the corresponding OPAP instance, pod 1 contains SKUs A; Accordingly, there is no link between vertices $B \in V_1$, and $1 \in V_2$, i.e., $(B, 1) \notin E_1$, meaning that pod 1 does not contain SKU B. The presence of edge $(1, 2) \in E_2$ in G indicates that pod 1 can be assigned to station 2, (i.e., they are on the same level), while the absence of edge $(1, 3) \notin E_2$ in G points out that pod 1 and station 3 are located on different levels. Since each order can be processed at any station, there is an edge in E_3 between each station node and order node. Finally, the edge $(1, A) \in E_4$ indicates that SKU A is ordered in order 1, and so on.

For this problem instance, the corresponding MCF Network consists of $\Pi = 72$ paths in total, from which 15 are selected in the optimal solution. The optimal paths are listed in the right-handed table of Fig. 4. The six selected (blue) edges in E_2 corresponds to $f_3^* = 6$ pod-to-station assignments in the optimal solution, and seven selected edges in E_3, to $f_2^* = 7$ order-to-station assignments.

4.2 MCF based MIP model

In this section, we propose the second MIP model (dubbed MIP-MCFP), which is based on the above-described MCFP representation of OPAP. Particularly, MIP-MCFP is based on the path-based model to solve the transformed MCFP (Barnhart et al. 2000).

Let Π denote a set of paths in G, whereas a path $\pi \in \Pi$ connects a source node $s \in V_1$ with a sink node $s \in V_5$ and, thus, consists of one node from each of the

Table 4 Additional notation for the MIP-MCFP model

Sets	
$G(V, E)$	Network, representing OPAP as MCFP, with V sets of nodes and E sets of edges
V	Set of nodes of network G
V_i	Set of nodes from i-th layer, $(V_i \subseteq V, \forall i = 1, \ldots, 5)$
E	Set of edges of network G
E_j	Set of edges, connecting nodes from set V_j to nodes in set V_{j+1}, $(E_j \subseteq E, \forall j = 1, \ldots, 4)$
Π	set of paths in G (index π)
Parameters	
k_{sr}^{π}	1, if path π, contains an edge $(s, r) \in E_1$; 0, otherwise
l_{rp}^{π}	1, if path π contains an edge $(r, p) \in E_2$; 0, otherwise
m_{pi}^{π}	1, if path π contains an edge $(p, i) \in E_3$; 0, otherwise
n_{is}^{π}	1, if path π contains an edge $(i, s) \in E_4$; 0, otherwise
Variables	
u_{rp}	Binary variable: 1, if edge $(r, p) \in E_2$, connecting node $r \in V_2$ with node $r \in V_3$, is used; 0, otherwise
$\mathcal{U}$	Set of u variables: $\mathcal{U} = \{u_{rp} \mid (r, p) \in E_2\}$
v_{pi}	Binary variable: 1, if edge $(p, i) \in E_3$, connecting node $p \in V_3$ with node $i \in V_4$, is used; 0, otherwise
$\mathcal{V}$	Set of v variables: $\mathcal{V} = \{v_{ip} \mid (i, p) \in E_3\}$
w_{π}	Continuous variable: flow on the path π
$\mathcal{W}$	Set of w variables: $\mathcal{W} = \{w_{\pi} \mid \pi \in \Pi\}$

five layers. A path π in G corresponds to a partial picking plan in OPAP and encodes which SKU $s \in S$ is picked from which pod $r \in R$ at which station $p \in P$ and for which order $i \in O$. In the model pre-processing step, we compute the following binary parameters for each path $\pi \in \Pi$:

- k_{sr}^{π} 1, if path π, contains an edge $(s, r) \in E_1$; 0, otherwise;
- l_{rp}^{π} 1, if path π contains an edge $(r, p) \in E_2$; 0, otherwise;
- m_{pi}^{π} 1, if path π contains an edge $(p, i) \in E_3$; 0, otherwise;
- n_{is}^{π} 1, if path π contains an edge $(i, s) \in E_4$; 0, otherwise.

For each path $\pi \in \Pi$, we define the amount of its flow as a continuous variable w_{π}, which corresponds to the number of picked items in the associated partial picking plan. Moreover, since we want to penalize adding edges from sets E_2 and E_3, we define the binary variables u_{rp} and v_{pi} to indicate whether an edge $(r, p) \in E_2$ and $(p, i) \in E_3$ are added ($u_{rp} = 1$ and $v_{pi} = 1$), or not ($u_{rp} = 0$ and $v_{pi} = 0$).

Using additional notation, summarized in Table 4, MIP-MCFP consists of the multi-objective function (15) subject to constraints (19) -(27).

$$\text{Minimize } \mathcal{G}(\mathcal{U}, \mathcal{V}, \mathcal{W}) = g_1 \text{ and } g_2 \text{ and } g_3 \tag{15}$$

$$g_1 = \max_{p \in V_3} \left\{ \sum_{i \in V_4 : (p,i) \in E_3} v_{pi} \right\} \tag{16}$$

$$g_2 = \sum_{(p,i) \in E_3} v_{pi} \tag{17}$$

$$g_3 = \sum_{(r,p) \in E_2} u_{rp} \tag{18}$$

subject to

$$\sum_{\pi \in \Pi} k_{sr}^{\pi} \cdot w_{\pi} \leq a_{rs} \qquad \forall (s,r) \in E_1 \tag{19}$$

$$\sum_{\pi \in \Pi} l_{rp}^{\pi} \cdot w_{\pi} \leq M \cdot u_{rp} \qquad \forall (r,p) \in E_2 \tag{20}$$

$$\sum_{\pi \in \Pi} m_{pi}^{\pi} \cdot w_{\pi} \leq M \cdot v_{pi} \qquad \forall (p,i) \in E_3 \tag{21}$$

$$\sum_{\pi \in \Pi} n_{is}^{\pi} \cdot w_{\pi} \geq o_{is} \qquad \forall (i,s) \in E_4 \tag{22}$$

$$u_{rp} = 1 \qquad \forall r \in V_2, p \in V_3 : r \in \overline{R}_p \tag{23}$$

$$v_{pi} = 1 \qquad \forall p \in V_3, i \in V_4 : i \in \overline{O}_p \tag{24}$$

$$u_{rp} \in \{0;1\} \qquad \forall (r,p) \in E_2 \tag{25}$$

$$v_{pi} \in \{0;1\} \qquad \forall (p,i) \in E_3 \tag{26}$$

$$w_{\pi} \in \mathbb{R}^{+} \qquad \forall \pi \in \Pi \tag{27}$$

Multi-objective function (15) minimizes three objectives (16), (17), and (18), which are ordered in a certain lexicographic order. To minimize the number of assigned orders at the busiest station in OPAP, the primary objective (16) minimizes the number of selected edges in E_3 originating from the busiest station-node $p \in V_3$, i.e., where most edges start at; secondary objective (17) minimizes the total number of active edges in E_3, i.e., the number of order-to-station assignment in OPAP; finally, the tertiary objective (18) minimizes the used edges in E_2, and thus, the total number of required pod-to-station assignments in OPAP. Inequalities (19) guarantee that the flow capacity restriction for each edge $(s,r) \in E_1$ is not violated, meaning that the total number of picked SKUs s from each pod $r \in R$ can never exceed the number of stored SKUs s on pod r. Constraints (20) ensure the edge $(r,p) \in E_2$ is used, if at least one partial schedule $\pi \in \Pi$, containing the edge $(r,p) \in E_3$, has a positive flow. Similarly, constraints (21) force the edge $(p,i) \in E_3$ to be added, if at least one partial schedule $\pi \in \Pi$ with edge $(p,i) \in E_2$ has a positive flow. Constraints (22) guarantee that the flow on each edge $(i,s) \in E_4$ is never less than o_{si}, meaning that every order must be fulfilled completely. Equations (23) take care that edges $(r,p) \in E_2$ are used for each fixedly assigned pod $r \in \overline{R}_p$ to each station $p \in P$. Similarly, constraints (24) forces each edge $(p,i) \in E_3$ to be added, if (part of) order i is fixedly assigned to

station p from the previous wave. Finally, constraints (25)–(27) define the domains of variables.

4.3 Heuristic path reduction

The number of paths in the MCF network grows exponentially with the size of the corresponding OPAP. However, the structure of the MIP-MCFP model offers a modeling advantage that enables us to employ it as a heuristic. Specifically, we can heuristically reduce Π to a smaller subset of paths $\Pi' \subseteq \Pi$ in the pre-processing step. In other words, instead of considering all partial picking plans, we can identify a subset of promising partial picking plans in the pre-processing step, and then find the optimal order and pod assignments for the reduced problem. As a result, the reduced MIP-MCFP becomes handier and can be solved in a shorter computational time. Similar approaches have proven to be very successful at finding near-to-optimal solutions for large problem instances in an acceptable computational time (Tadumadze et al. 2019, 2020).

Our heuristic path reduction scheme is based on the idea of decomposing OPAP into two handier sub-problems, **selecting pods for orders**, and **selecting stations for orders**, which can be solved individually in a sequential manner. Based on this decomposition, we can pre-select a sub-set of promising pods $R^{selected}(i) \subseteq R$ and stations $P^{selected}(i) \subseteq P$ for each order $i \in O$, and generate a reduced subset of paths $\Pi' \subseteq \Pi$ for the MIP-MCFP that contains only paths with the pre-selected subsets of pods $R^{selected}(i)$ and stations $P^{selected}(i)$ for the orders $i \in O$.

In the first sub-problem, we seek subsets of pods for orders from which the ordered SKUs can be picked. In particular, for each ordered SKU, we pre-select the μ^R most promising pods, with $\mu^R \in \mathbb{N}$ being a predefined positive integer parameter. The procedure of pre-selecting subsets of pods for the orders is outlined in Algorithm 1. For each order $i \in O$, we first compute the maximal hit rate, i.e., the maximal number of items that can be picked for order i from pod r as $\min\left\{\sum_{s\in S} o_{is}; \sum_{s\in S} a_{rs}\right\}$ for each pod $r \in R$ (see line 4 of Algorithm 1). Then, for each order $i \in O$, we sort the pods in a sorted list $R^{sorted}(i)$ according to the descending hit rate value (i.e., pods that contain more ordered items precede the pods with fewer ordered items). This is done to select μ^R "promising" pods that match the orders best (see lines 11-13 of Algorithm 1). Besides, to ensure that Π' enables at least one feasible pod selection such that the number of picked SKUs from each pod does not exceed the stored quantities, we initially select one pod per ordered SKU while taking the pods' limited stock into account (see line 6 of Algorithm 1). The procedure of selecting one feasible pod per ordered SKU is similar to Algorithm 1, with the exception that we select only one pod per ordered SKU (i.e., $\mu^R = 1$), and adjust the number of remaining SKUs on the selected pod after each pod selection to dynamically compute the hit rate values with the modified stocks of the pods.

Algorithm 1: Pre-selection of pods for orders

input: an instance of OPAP, parameter μ^R
output: $R^{selected}(i), \forall i \in O$

1 **for** $i = 1$ **to** $|O|$ **do**
2 $R^{selected}(i) := \emptyset$; // Set of pre-selected pods for order i
3 **for** $r = 1$ **to** $|R|$ **do**
4 $HIT\_RATE(r) := \min\left\{\sum_{s\in S} o_{is}; \sum_{s\in S} a_{rs}\right\}$;
5 $R^{sorted}(i) = SORT(R, HIT\_RATE)$;
6 $SELECT\_ONE\_FEASIBLE\_POD\_PER\_ORDERED\_SKU()$;
7 **for** $i = 1$ **to** $|O|$ **do**
8 **for** $s = 1$ **to** $|S|$ **do**
9 **if** $o_{is} > 0$ **then**
10 $r = 1$;
11 **while** $r < \min\{\mu^R; |R|\}$ **do**
12 $R^{selected}(i) := R^{selected}(i) \cup R^{sorted}(i)(r)$;
13 $r := r + 1$;

Once the potential pods are selected for the orders, we select the subset of potential stations for each order in the second sub-problem. Similarly to the first sub-problem, we seek μ^P stations for each order $i \in O$, with $\mu^P \in \mathbb{N}$ as a predefined positive integer parameter. The procedure of pre-selecting subsets of stations for the orders is outlined in Algorithm 2. Here, for each order $i \in O$, we are already given a fixed sub-set of the pre-selected pods $R^{selected}(i)$, which are determined in the first sub-problem. Depending on the distribution of the pre-selected pods $R^{selected}(i)$ on the warehouse levels, not every station might be a meaningful candidate for processing order $i \in O$. Therefore, for each order $i \in O$, we first identify the set of potential levels $L^{potential}(i) \subseteq L$ that contain at least one pre-selected pod $r \in R^{selected}(i)$ (see line 9 of Algorithm 2). Consequently, for each order $i \in O$, we derive a set of potential stations $P^{potential}(i)$, which have access to at least one pre-selected pod $r \in R^{selected}$ of the order i (see line 10 of Algorithm 1). We aim to pre-select the stations for the orders such that it is possible to evenly distribute the total workload among all stations. Therefore, for each order $i \in O$, we iteratively select the next least busy station with the least pre-assigned orders (see line 15 of Algorithm 2). Besides, for each order $i \in O$, we initially select one station from each potential level $L^{potential}(i)$, enabling every pre-selected pod $r \in R^{selected}(i)$ of each order $i \in O$ to be a candidate in the MIP-MCFP (i.e., for Π' to contain at least one partial picking plan π where pod r serves order i) (see line 15 of Algorithm 2).

Algorithm 2: Pre-selection of stations for orders

input: an instance of OPAP, $R^{selected}(i), \forall i \in O$, parameter μ^P
output: $P^{selected}(i), \forall i \in O$

$O^{selected}(p) := \emptyset \; \forall p \in P$; // Set of pre-selected orders at station p
for $i = 1$ **to** $|O|$ **do**
  $P^{selected}(i) := \emptyset$; // Set of pre-selected stations for order i
  $L^{potential}(i) := \emptyset$; // Set of potential levels with at least one $r \in R^{selected}(i)$
  $P^{potential}(i) := \emptyset$; // Set of potential stations for order i
  for $l = 1$ **to** $|L|$ **do**
    for each $r \in R^{selected}(i)$ **if** $r \in R_l$ **then**
      $L^{potential}(i) := L^{potential}(i) \cup l$;
      $P^{potential}(i) := P^{potential}(i) \cup P_l$;
    $p^* := P_l(i \bmod |P_l|)$;
    $P^{selected}(i) := P^{selected}(i) \cup p^*$;
    $O^{selected}(p) := O^{selected}(p) \cup i$;
  while $|P^{selected}(i)| < \min\{\mu^P; |P^{potential}(i)|\}$ **do**
    $p^* := argmin_{p \in P^{potential}(i)}\{|O^{selected}(p)|\}$;
    $P^{selected}(i) := P^{selected}(i) \cup p^*$;
    $O^{selected}(p) := O^{selected}(p) \cup i$;

We will denote the whole procedure of selecting μ^R pods per ordered SKU, and μ^P stations per order, generating the sub-set of the reduced paths Π', as well as building and solving the corresponding reduced MIP-MCFP model as "MCFP-h". An outline of the MCFP-h approach is summarized in Fig. 5.

Example (cont.): In our example OPAP instance, using MCFP-h with the values $\mu^R = 1$ and $\mu^P = 1$ reduces the total number of paths from $|\Pi| = 72$ to $|\Pi'| = 22$, and leads to a solution with the objective values $f_1 = 3$, $f_2 = 9$, and $f_3 = 6$. Incrementing the values of μ^R and μ^P to 2 leads to $|\Pi'| = 30$ paths, and a solution with $f_1 = 2$, $f_2 = 7$, and $f_3 = 7$. Moreover, for this example, MCFP-h with $\mu^R = 3$

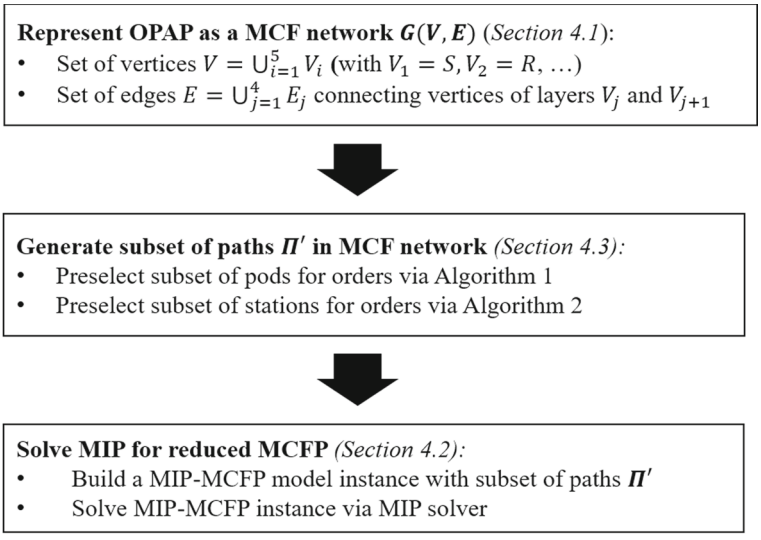


Fig. 5 Outline of the MCFP-h procedure

and $\mu^P = 3$ leads to $|\Pi'| = 53$ paths, and finds the solution with $f_1 = 2$, $f_2 = 7$, and $f_3 = 6$.

It stands to reason that the sets $R^{selected}(i)$ and $P^{selected}(i)$ have a substantial effect on the performance of the solution procedure. By varying the parameter values for μ^R and μ^P, we can reduce the solution space: lower values of μ^R and μ^P will reduce the number of paths $|\Pi'|$ so that the corresponding MIP-MCFP models become easier to solve. On the other hand, fewer paths increase the risk that the optimal partial picking plans are not in Π'. We further investigate this trade-off while looking for the best values for the parameters μ^R and μ^P in our computational study (Sect. 5.2.2).

5 Computational study

To explore the computational performance of the proposed MIP models and the MCFP-h algorithm, we implemented them in C# 7.0, applying the commercial solver IBM ILOG CPLEX Optimizer V12.9.0 to solve the MIP model instances and test them on newly generated OPAP instances. Thereby, the time limit to generate and solve the MIP models was set at 1800 CPU seconds. All tests were executed on an x64 PC with an Intel Core i7-8700K 3.70 GHz CPU and 64 GB of RAM. In Sect. 5.1, we describe how the new OPAP instances were generated. In Sect. 5.2, we examine the computational performance of the proposed solution methods. Finally, in Sect. 5.3, we derive some managerial insights, such as the impact of the number of warehouse levels and the length of the planning horizons (or waves) on the objective values.

5.1 Instance generation

In this section, we describe how the new OPAP instances were generated, which we use for our computational experiments. We generated two datasets of problem instances (dubbed $\mathcal{S}$ and $\mathcal{L}$) for the algorithmic performance tests (Sect. 5.2) and another two datasets (dubbed $\mathcal{M}_1$ and $\mathcal{M}_2$) for the managerial study (Sect. 5.3). Particularly, to explore the impact of instance size on the computational performance of the solution approaches, datasets $\mathcal{S}$ and $\mathcal{L}$ contain different-sized OPAP instances. Thereby, the size of an OPAP instance is defined by the number of the SKUs $|S|$, picking stations $|P|$, pods $|R|$, and orders $|O|$, which our instance generator receives as input parameters. Another important parameter, which defines the layout of the warehouse, is the number of levels $|L|$. Note that apart from the instances in dataset $\mathcal{M}_1$, all generated OPAP instances have a constant number of $|L| = 3$ levels, while dataset $\mathcal{M}_1$ contains OPAP instances with a varying value of $|L|$. In e-commerce fulfillment centers, orders typically arrive progressively throughout the day, so that OPAP is typically solved in small waves and is re-planned multiple times during the day. All generated OPAP instances, apart from the ones in dataset $\mathcal{M}_2$, are solved in one single wave ($|T| = 1$). In OPAP, the length of the planning horizon of a wave is reflected by the number of orders $|O|$ in the planning horizon. To observe the impact of the length of the planning horizon on the problem complexity, in datasets $\mathcal{S}$ and $\mathcal{L}$, the number of orders $|O|$ is varied in four different values $|O| \in \{50, 100, 150, 200\}$ for dataset

Table 5 Dimensions of the used OPAP instances

Dataset	Algorithmic study (Sect. 5.2)		Managerial study (Sect. 5.3)	
	$\mathcal{S}$	$\mathcal{L}$	$\mathcal{M}_1$	$\mathcal{M}_2$
$\|S\|$	200	500	200	200
$\|P\|$	9	90	36	18
$\|R\|$	90	300	300	300
$\|O\|$	(100, 150, 200, 250)	(600, 800, 1000, 1200)	360	600
$\|L\|$	3	3	(1, 2, 3, 4)	3
$\|T\|$	1	1	1	(1, 2, 3, 4, 5, 6)

$\mathcal{S}$ and $|O| \in \{600, 800, 1000, 1200\}$ for dataset $\mathcal{L}$, respectively. Note that in OPAP, the set of SKUs S and pods R represent only sub-sets of all SKUs and pods of the warehouse. This is because, for an OPAP instance, it is sufficient to consider only ordered SKUs in the current wave, and only such pods, which contain at least one item of ordered SKU type. Table 5 summarizes the parameters defining the size and characteristics of the four datasets.

Note that our instance generator is a result of intensive discussions and consultations with practitioners. We aimed to generate realistic OPAP instances, which represent a typical mid-sized warehouse, in the best manner. Therefore, all the input parameters (including probability distributions) used to generate OPAP instances are checked with our industry partners.

We generate the orders similarly as proposed by Xie et al. (2021). Particularly, for each order $i \in O$, we first randomly draw its size, i.e., the number of ordered items, and then randomly select the SKU types for each ordered item. According to discussions with our industry partners, a typical order size in an e-commerce warehouse contains 1.4 items on average. Moreover, orders with a few (or even single) items arrive more frequently than larger orders. Therefore, for each order $i \in O$, we randomly draw its size from a geometric distribution with parameter $p = 1/1.4$, leading to an expected order size of 1.4 items per order. Regarding the selection of ordered SKU types, according to our industry partners, a typical ABC curve in a warehouse reflects the so-called 20/80 rule. In other words, about 20% of the most popular articles in the warehouse are responsible for about 80% of the total inventory movements (Bender 1981). To reflect this, the SKU type for each ordered item is chosen by sampling from the set S using the probability mass function of exponential distribution with such a value for the parameter λ that leads to selecting the first $0.2 \cdot |S|$ SKU types with a probability rate of 0.8. Particularly, for instances with $|S| = 200$ SKU types, we use $\lambda = 0.04$, and for instances with $|S| = 500$ SKU types, we use $\lambda = 0.016$. Finally, for each order $i \in O$ and SKU $s \in S$, we compute the number of ordered items o_{is} of SKU s in order i, which OPAP receives as an input parameter. Note that in contrast to the instance generation scheme proposed by Xie et al. (2021), our instance generator allows more than one item of the same SKU type to be ordered in the same order (i.e., $o_{is} > 1$ may occur).

After generating the order-related parameters, we distribute the items of ordered SKU types on the set of pods R. Specifically, we generate the pod-related parameters

similarly, as proposed by Merschformann et al. (2019). First, for each SKU type $s \in S$, the required space $Size_s$ to store one item is drawn from a uniform distribution between 2 and 8 slots, while the storage capacity of each pod $r \in R$ is assumed to be 500 slots. Then, for each SKU $s \in S$, we compute the total number of ordered items as $\#ordered_s = \sum_{i \in O} o_{is}$. Assuming that the required space to store all ordered SKU types on the given pods P is 30%, we derive the total number of stored items $\#stored_s$ for each SKU $s \in S$ as:

$$\#stored_s = \#ordered_s \times \frac{\sum_{s \in S} \#ordered_s \cdot Size_s}{500 \cdot |P| \cdot 30\%}.$$

After determining the total number of stored items for each SKU $s \in S$, we distribute them on the pods, by randomly selecting the next free pod from R, filling them with $rand(10; 20)$ items of SKU s, and adjusting the remaining storage capacity of the pod accordingly. We repeat this step until all items are assigned to pods. Finally, we derive the number of stored items a_{rs} of each SKU $s \in S$ on each pod $r \in R$.

Regarding the allocation of the pods and the picking stations on the different warehouse levels, we assume that they are evenly distributed on the warehouse levels (apart from the last level, which may receive fewer pods or stations if the total number of pods or stations are not divisible by the number of levels). Particularly, on the first level $l = 1$, we allocate $R_1 = \{1, \ldots, \lfloor \frac{|R|}{|L|} \rfloor\}$ pods and $P_1 = \{1, \ldots, \lfloor \frac{|P|}{|L|} \rfloor\}$ stations; on the second level $l = 2$, the next $R_2 = \{\lfloor \frac{|R|}{|L|} \rfloor + 1, \ldots, 2 \cdot \lfloor \frac{|R|}{|L|} \rfloor\}$ pods and $P_2 = \{\lfloor \frac{|P|}{|L|} \rfloor + 1, \ldots, 2 \cdot \lfloor \frac{|P|}{|L|} \rfloor\}$, and so on to the last level $l = |L|$, which receives the last $R_{|L|} = \{((|L| - 1) \cdot \lfloor \frac{|R|}{|L|} \rfloor) + 1, \ldots, |R|\}$ pods and $P_{|L|} = \{((|L| - 1) \cdot \lfloor \frac{|P|}{|L|} \rfloor) + 1, \ldots, |P|\}$ stations.

5.2 Algorithmic performance

In this section, we test the computational performance of both proposed models, MIP-intuitive and MIP-MCFP, as well as heuristic MCFP-h at solving various OPAP instances. Specifically, in Sect. 5.2.1, we compare the algorithmic performance of proposed MIP formulations, i.e., MIP-intuitive and MIP-MCFP. Section 5.2.2 deals with the proposed heuristic MCFP-h. Specifically, we first tune the parameters for MCFP-h identifying the best values for the parameters μ^R and μ^P. Thereafter, we compare the performance of tuned MCFP-h with two extreme versions of MCFP-h: i.e., $\mu^R = \mu^P = +\infty$, when MCFP-h becomes an exact solution approach, and $\mu^R = \mu^P = 1$, transforming MCFP-h to a simple constructive heuristic. For the algorithmic experiments, we use the newly generated OPAP instances from the dataset $\mathcal{S}$ and $\mathcal{L}$, generated as described in Sect. 5.1.

5.2.1 Comparison of the MIP models

In this section, we compare the performance of the two proposed MIP formulations solving small OPAP instances from dataset $\mathcal{S}$. Unfortunately, CPLEX runs out of

Table 6 MIP-intuitive vs MIP-MCFP

	MIP-intuitive							MIP-MCFP						
Size	Cols	Rows	Opt	Time	Gap^{f_1}	Gap^{f_2}	Gap^{f_3}	Cols	Rows	Opt	Time	Gap^{f_1}	Gap^{f_2}	Gap^{f_3}
$9 \times 90 \times 100$	4,610,611	13,340	10	121.1	0.0	0.0	0.0	3904.6	6980.6	9	529.8	0.0	0.0	0.6
$9 \times 90 \times 150$	8,069,761	18,915	10	187.3	0.0	0.0	0.0	5189.2	8352.5	5	1102.3	0.0	0.0	2.9
$9 \times 90 \times 200$	12,136,411	25,150	9	896.6	0.0	0.0	0.6	6401.8	9633.1	1	1735.6	0.0	0.0	7.6
$9 \times 90 \times 250$	16,122,061	30,943	5	1276.4	0.0	0.0	3.3	7726.9	10580.9	1	1743.7	0.0	0.0	8.8
Mean	*10,234,711*	*22,087*	*8.5*	*620.3*	*0.0*	*0.0*	*1.0*	*5806.6*	*8886.8*	*4*	*1277.8*	*0.0*	*0.0*	*5.0*

memory (or exceeds the array limits) even before generating the corresponding MIP-intuitive models for large instances from dataset $\mathcal{L}$. This is due to the abundant number of variables z_{spri}, stored in a sizable 4-dimensional matrix, whose size grows very fast with the growth of the OPAP instance. Hence, for this comparison, we only use the small OPAP instances from dataset $\mathcal{S}$.

Table 6 summarizes the results of the comparison, where each cell represents the aggregated computational results of the 10 OPAP instances of the same size. The first column "Size" indicates the size of the OPAP instances, reflected by $|P| \times |R| \times |O|$. To have a better idea of the size and the structure of the MIP models, in columns "Cols" and "Rows", we report the average number of columns and rows in each MIP model. Further, for each MIP model, the column "Opt" summarizes the number of instances solved to proven optimality within the given time limit, while the average computational time (in CPU seconds) is reported in the column "Time". Besides, we report the average relative gaps between the best found upper bounds and the lower bounds. Namely, if an instance is not solved to proven optimality within the time limit, we store the upper bounds ($\overline{f}_1$, $\overline{f}_2$, $\overline{f}_3$) and the lower bounds ($\underline{f}_1$, $\underline{f}_2$, $\underline{f}_3$) of the objective values f_1, f_2 and f_3, respectively. Consequently, for each objective $i \in \{1, 2, 3\}$, we calculate the relative gap between the upper bound $\overline{f}_i$ and lower bound $\underline{f}_i$ (in %) as $Gap^{f_i} = \frac{\overline{f}_i - \underline{f}_i}{\underline{f}_i} \cdot 100$, which are reported in columns "Gap^{f_1}", "Gap^{f_2}", and "Gap^{f_3}".

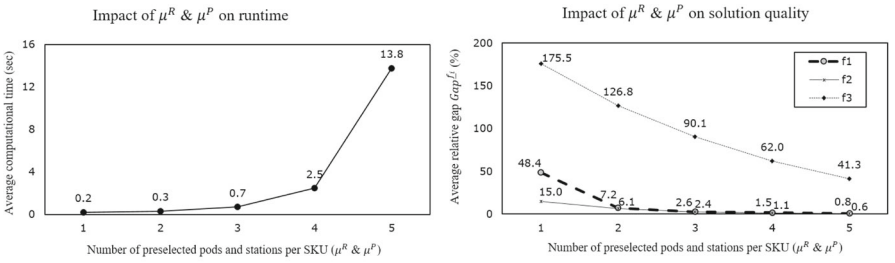
As can be seen from the results, MIP-intuitive performs quite well at solving small OPAP instances from dataset $\mathcal{S}$. Particularly, it solves 34 out of the small 40 OPAP instances to proven optimality. However, the computational effort grows with rising instance size, reflected by fewer optimally solved instances for the larger instances with more orders. In comparison, the computational performance of MIP-MCFP lags behind the performance of MIP-intuitive, being capable of finding a proven optimal solution for 16 out of 40 small instances. In the remaining 24 cases, it finds an optimal solution with regard to the primary and secondary objectives, but struggles with the tertiary objective, leading to the average relative gap Gap^{f_3} of 5%. An important factor besides the relative gap is the computational time required to solve the OPAP instances. The average computational time for the MIP-intuitive model over 40 small OPAP instances is 620 s, while MIP-MCFP requires approximately twice as much (i.e., 1277 s).

Note that although MIP-intuitive shows better performance on small instances than its competitor, suffering from the large number of w_{spri} variables, it scales rather poorly as the model size increases very fast (see column "Cols"). As a result, it cannot handle large OPAP instances from dataset $\mathcal{L}$. Even for the smallest instances in the dataset $\mathcal{L}$ (i.e., with $|O| = 600$ orders), the MIP-intuitive model instances become too large and require high memory usage, leading to termination of the procedure in the model generation phase. In contrast, MIP-MCFP tends to scale better, being able to generate and solve (at least to feasibility) the corresponding MIP models for all OPAP instances from both datasets (the detailed results on large instances are discussed in Sect. 5.2.2).

5.2.2 Evaluation of the heuristic procedure

Due to the operational character of OPAP, which must be solved multiple times during the day for quite short planning horizons (e.g., every 10 min in e-commerce warehouses), neither of the MIP models can be applied in practice, so that a fast heuristic algorithm is needed. In this section, we evaluate the computational performance of the proposed heuristic algorithm MCFP-h. First, we will calibrate the values of parameters μ^R and μ^P on small OPAP instances from dataset S. Consequently, we observe the performance of the tuned MCFP-h on large instances from dataset $\mathcal{L}$ and compare it with an exact solution approach and a greedy constructive heuristic.

The algorithmic performance of the MCFP-h strongly depends on the values of the parameters μ^R and μ^P: low values of μ^R and μ^P lead to a low number of preselected pods and stations in the model pre-processing step, i.e., fewer paths for the corresponding MIP-MCFP model. This kind of reduction of the MIP-MCFP model can be beneficial in terms of the required computational time to solve it. On the downside, it can be counterproductive in terms of solution quality, since it enlarges the risk that optimal (or promising) paths are not included in the resulting reduced MIP-MCFP. The preliminary tests have shown that setting μ^R, and μ^P on the same values leads to better solutions, rather than selecting only a few pods (i.e., low μ^R) and a lot of stations (high μ^P) or vice versa. Therefore, in parameter tuning tests, we observe the behavior of MCFP-h when both parameters μ^R and μ^P have the same values, and vary them in the range of five values $\mu^R, \mu^P \in \{1, 2, 3, 4, 5\}$.



(a) Impact of μ^R & μ^P on computational runtime.

(b) Impact of μ^R & μ^P on relative gaps to lower bounds.

Fig. 6 Parameter tuning of MCFP-h

Table 7 Comparison of MCFP-h$^{\infty}$, MCFP-h* and MCFP-h^{1}

	MCFP-h$^{\infty}$				MCFP-h*				MCFP-h^{1}			
Size	Time	$\overline{f}_1$	$\overline{f}_2$	$\overline{f}_3$	Time	$Gap^{\overline{f}_1}$	$Gap^{\overline{f}_2}$	$Gap^{\overline{f}_3}$	Time	$Gap^{\overline{f}_1}$	$Gap^{\overline{f}_2}$	$Gap^{\overline{f}_3}$
$90 \times 300 \times 600$	1800.6	**7**	**609**	1145	57.8	1.4	2.8	−11.4	37.9	90.0	17.1	5.7
$90 \times 300 \times 800$	1800.4	**10**	**817**	1262	82.8	0.0	3.0	−12.2	56.5	80.0	18.3	9.4
$90 \times 300 \times 1000$	1800.7	**12**	**1020**	775	120.5	0.0	3.0	17.3	73.9	75.8	17.4	52.7
$90 \times 300 \times 1200$	1801.0	**14**	**1227**	3129	206.8	5.0	2.9	−19.9	95.4	71.4	17.4	8.2

Bold typesetting of some objective values indicates that for all 10 instances of this size the objective values found are optimal

Figure 6 visualizes the aggregated computational results of MCFP-h for each observed value of parameters μ^R and μ^P. Namely, each dot in graphics summarizes the average results over 40 OPAP instances from dataset $\mathcal{S}$. The left-handed graphic Fig. 6a depicts the average computational time (in CPU seconds) and shows an apparently exponential increase in computational time with rising values of μ^R and μ^P. Recall that rising the values of these parameters leads to an exponential rise in the number of paths in MIP-MCFP, which requires longer solution times. The right-hand Fig. 6b depicts the relative gaps between the objective values f_1, f_2 and f_3 of MCFP-h solutions, and the best known lower bounds on these objectives $\underline{f}_1$, $\underline{f}_2$ and $\underline{f}_3$, obtained by MIP-intuitive. It can be observed that higher values of μ^R and μ^P lead to lower relative gaps for all objectives, i.e., an improvement of solution quality. In other words, lower values of μ^R and μ^P, on the one hand, lead to shorter computational times (see Fig. 6a). On the other hand, it reduces the probability that the reduced MIP-MCFP contains potentially good paths, which leads to higher optimality gaps of the reduced problem's solution (see Fig. 6b).

In practice, the exact choice of values for parameters μ^R and μ^P depends on many factors, such as the size of the warehouse, the number of orders, the desired level of the solution quality, the available time, and the hardware, etc. For our study, $\mu^{*R} = \mu^{*P} = 3$ turns out to be the most promising compromise between solution quality and computational time. Therefore, for the next computational experiments, we set parameters μ^R and μ^P to the value 3 and refer to MCFP-h with this parameter constellation as *MCFP-h**.

In the following, we observe the computational performance of tuned algorithm *MCFP-h** on large OPAP instances from dataset $\mathcal{L}$ and compare it with the following two special cases:

MCFP-h$^{\infty}$ when parameters μ^R and μ^P are set to ∞;
MCFP-h^{1} when both parameters μ^R and μ^P have a value of 1.

Note that in MCFP-h$^{\infty}$, the resulting MCFP contains all paths (i.e., $\Pi' = \Pi$), and the solution approach becomes exact. MCFP-h^{1} contains only one pod per ordered SKU and one station per order. As a result, the procedure becomes similar to a greedy constructive heuristic, where both sub-problems are solved sequentially via simple priority rule-based heuristics, e.g., like priority-rule-based decision rules, proposed by

Merschformann et al. (2019). However, in MCFP-h^1, the initial constructive-heuristic solution can be improved after solving the resulting reduced MIP-MCFP instance.

Table 7 summarizes the aggregated results of the comparison of the exact approach MCFP-h$^\infty$ (used as a benchmark approach) with the two heuristics, MCFP-h* and MCFP-h^1. More specifically, for each approach, the column "Time" reports the required average computational time. Besides, we report the upper bounds $\overline{f}_1, \overline{f}_2, \overline{f}_3$ of optimal objective values f_1, f_2, and f_3, obtained by the exact approach MCFP-h$^\infty$ within the time limit of 1800 s, where the values that are proven to be optimal are marked by **bold typesetting**. The average relative gaps (in %) of the MCFP-h* and MCFP-h^1-solutions to the upper bounds $\overline{f}_1, \overline{f}_2, \overline{f}_3$ of the MCFP-h$^\infty$-solutions are reported in columns $Gap^{\overline{f}_1}$, $Gap^{\overline{f}_2}$ and $Gap^{\overline{f}_3}$. Thereby, a negative value under these columns indicates an improvement of MCFP-h$^\infty$-solutions, found within a given time limit, by a heuristic approach.

As can be seen from the results, none of the large instances from dataset $\mathcal{L}$ can be solved to proven optimality by the exact solution approach (i.e., MCFP-h$^\infty$) within 1800 s. However, for all 40 large instances, the found solutions are optimal in terms of the primary and the secondary objectives, and the solver struggles with the tertiary objective, which in most cases shows very poor lower bounds. As a result, CPLEX does not terminate before reaching the given time limit of 1800 s. In contrast, both heuristic approaches (MCFP-h* and MCFP-h^1) manage to terminate before reaching the time limit and provide solutions in a reasonable runtime. The average computational time over 40 large instances is 117 s for the MCFP-h* and 66 s for the MCFP-h^1. Regarding the solution quality, MCFP-h*-solutions show a slight deterioration of the optimal objective values for the primary and secondary objectives, with the average optimality gaps of $Gap^{\overline{f}_1} = 1.6\%$ and $Gap^{\overline{f}_2} = 2.9\%$, and the improvement of the MCFP-h$^\infty$ upper bounds $\overline{f}_3$ by 6.5% on average. In contrast, constructive heuristic solutions (i.e., found by MCFP-h^1) show deterioration of the benchmark MCFP-h$^\infty$-solutions in terms of all three objectives, with average relative gaps of $Gap^{\overline{f}_1} = 79.3\%$, $Gap^{\overline{f}_2} = 17.6\%$ and $Gap^{\overline{f}_3} = 19.0\%$.

To conclude, MCFP-h* turns out to be superior among its competitors and the tuned parameter values $\mu^R = \mu^P = 3$ seem to be a good compromise between the computational time and the solution quality also for the large OPAP instances.

5.3 Managerial insights

In this section, we explore some practical insights, such as the impact of the warehouse layout (i.e., the number of warehouse levels) and the number of waves (under usage of the rolling planning environment) on our objective values. The results of the former observation can be helpful from a managerial viewpoint at the strategic planning level when deciding on the facility layout of the warehouse. The latter results provide insights for the proper design of the rolling planning environment and evaluate the performance of the division of a planning horizon into smaller waves. For the experiments of the managerial study, we use two datasets $\mathcal{M}_1$ and $\mathcal{M}_2$. Each of the datasets $\mathcal{M}_1$ and $\mathcal{M}_2$ contains 10 benchmark OPAP instances, which are then modified by

changing solely one parameter (the number of Levels $|L|$ or the number of waves $|T|$) while the remaining parameters are the same in the benchmark instances.

5.3.1 Impact of the number of warehouse levels on the objective values

A decision on how many levels the warehouse facility must have is typically made on the strategic level and has a crucial impact on the performance of the picking process. Running a warehouse in a tight, but multi-level facility instead of in a wide and few-level facility may save initial investment costs, especially in areas where land is expensive, e.g, in (or next to) big cities. However, having multiple levels comes at the cost of operational productivity of the warehouse, which in the long term may counterbalance the investment savings.

To explore the impact of the number of warehouse levels $|L|$ on our workload-related objectives, we generate OPAP instances with varying warehouse levels. In particular, we first generate 10 benchmark OPAP instances with $|S| = 200$ SKUs, $|P| = 36$ stations, $|R| = 300$ pods, $|O| = 360$ orders and single warehouse level, i.e., $|L| = 1$. Then, we modify each benchmark instance by resetting the number of levels to $|L| = \{2, 3, 4\}$ and evenly distributing the pods and stations on the resulting $|L|$ levels. This leads to changes in parameters R_l and P_l for $l = 1, \ldots, |L|$, while the remaining input parameters (SKUs, orders, and pods) are the same as in the benchmark instances. We solve the resulting 40 OPAP instances using our tuned algorithm MCFP-h* and observe the maximal number of processed orders per station, the number of split orders, and the total number of required pod-to-station assignments of the found solutions.

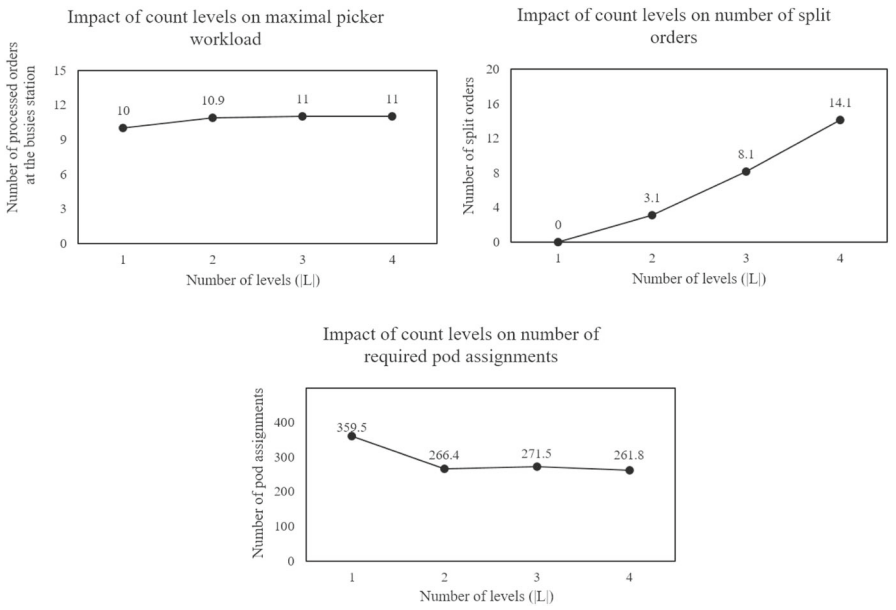


Fig. 7 Impact of the warehouse levels on the objective values

Figure 7 displays the aggregated results of our study. Each dot in the figure represents the average results of the 10 OPAP instances with the same number of levels. The results of our study reveal that the number of levels has only a marginal impact on the maximum number of handled orders per station (i.e., f_1) and the number of pod-to-station assignments (i.e., f_3). However, the approximately linearly increasing curve of the chart in the middle graphic indicates the raising number of split orders when the number of warehouse levels grows. In other words, multi-level warehouse layouts lead to a higher number of split orders, which comes with the cost of additional consolidation effort for packers. As a result of our tests, we can conclude that warehouse planners need to consider the operational costs for order consolidation when reducing investment costs via a multi-level setup.

5.3.2 Impact of the number of waves on the objective values

In the previous sections, we assumed that each OPAP instance was solved in $|T| = 1$ wave. In practice, OPAP is solved in multiple waves using a rolling planning environment. Dividing a long planning horizon into smaller waves will lead to a loss of order consolidation effects of OPAP, which can negatively affect the effectiveness of the picking process. However, in practice, it is often impossible to plan for long planning horizons due to a dynamically changing order information during the day as well as a high complexity of the corresponding large OPAP problems, as shown in Sect. 5.2. In this section, we want to observe how the division of the planning horizon into smaller waves affects the objective functions.

For this observation, we generate another 10 benchmark instances with $|S| = 200$ SKUs, $|P| = 18$ stations, $|R| = 300$ pods, $|O| = 600$ orders, $|L| = 3$ levels and $|T| = 1$ wave. Consequently, we divide each benchmark instance into $|T| = \{2, 3, 4, 5, 6\}$ waves, whereas in each wave we consider only $\frac{|O|}{|T|}$ orders (assuming that orders arrive in a sequence with increasing index). Finally, we solve the resulting OPAP instances via MCFP-h* algorithm for each wave $t = 1, \ldots |T|$, and adjust the inventory for the next wave $t + 1$ by subtracting from the pod stock the picked SKUs from the previous wave t, i.e., $a'_{rs} = a_{rs} - \sum_{p \in P} \sum_{i \in O} \omega_{srpi}$. Finally, we aggregate the maximal number of assigned orders per station, the number of split orders, and the total number of required pod-to-station assignments for the whole planning horizon by summing up the corresponding values from each wave over the waves $t = 1, \ldots |T|$.

The aggregated results of this study are depicted in Fig. 8. We would expect that dividing a planning horizon into smaller waves would harm the picking operating performance. And indeed, this is what the results indicate. However, here as well, the number of waves does not affect all three objectives in the same manner. While $|T|$ turns out to have rather a smaller impact on the picker and packer workload, i.e., f_1 and the number of split orders f_2, it seems to have a greater impact on the number of pod-to-station assignments, which is reflected in the increasing curve with a higher slope of the right-handed chart. This can be explained by the loss of consolidation effects when OPAP is solved in multiple waves separately. Our results imply that terminal managers should take care not to plan OPAP in extremely short waves, which would weaken the consolidation effect of similar orders at stations by allowing pickers to pick only a few SKUs per pod visit so that the mobile robots must transport more pods

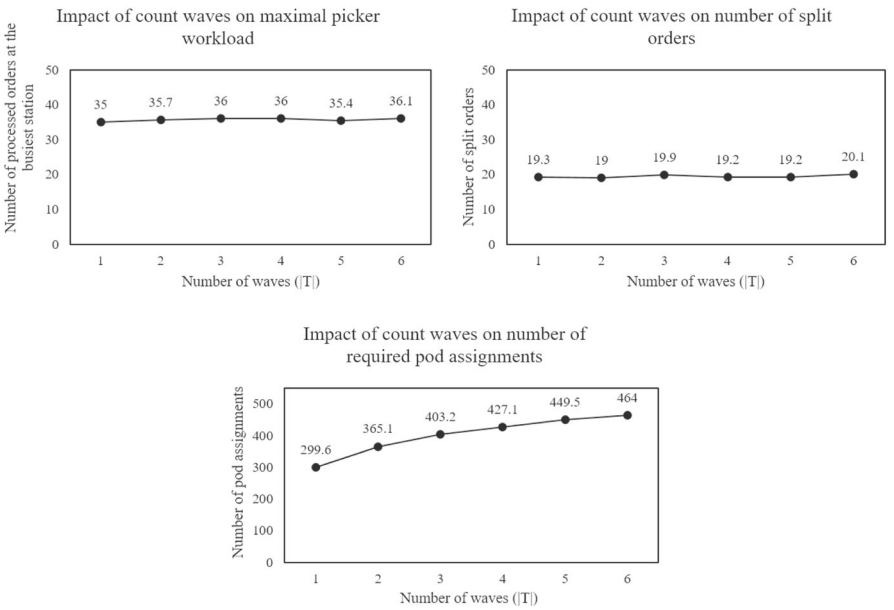


Fig. 8 Impact of the number of waves on the objective values

to the stations to cover all orders. The good news for practitioners is that the effects of a shorter wave duration can be neglected if the size of the robot fleet is large enough. On the other hand, obviously, robots, and therefore investment costs, can be reduced if wave duration can be prolonged.

6 Conclusion

In this paper, we investigate the order and pod assignment problem (OPAP) in a multi-level robotic mobile fulfillment system. The problem integrates the following two interrelated decisions: which order must be handled at which picking station and from which pods must the ordered items be picked, considering the limited stock of SKUs of the pods as well as the warehouse levels where the pods and stations are located? In terms of the optimization criteria, we consider three workload-related goals, namely balancing the order handling workload among all pickers, minimizing the order consolidation workload for packers, and minimizing the pod-movement workload for robots. We formalize the problem as a multi-criteria optimization problem by introducing a surrogate objective for each observed goal and design an intuitive mixed-integer linear programming model. Moreover, we transform the problem as a type of multicommodity flow problem (MCFP), develop a path-based mixed-integer linear programming model, and propose a heuristic path reduction scheme that allows us to use the latter model as a heuristic. We compare the computational performance of the proposed models and the heuristic solution procedure. After reducing paths to the

proper level, the proposed MCFP-h* manages to find reasonable solutions in a short computational time.

In summary, we conduct a comparative study with all three approaches, namely intuitive mixed-integer linear programming model (MIP-intuitive), multicommodity flow problem MIP (MIP-MCFP), and reduced-path heuristic (MCFP-h*). Our numerical results show that the MIP-MCFP outperforms MIP-intuitive, but neither model is suitable for realistic instances, using a state-of-the-art default solver. Consequently, we develop the heuristic MCFP-h*, which exhibits the fastest solution times (less than five minutes even in the largest instances) while delivering acceptable solution quality (at worst a 5% optimality gap for the large instances where the optimal objective value is available). For practical application of RMFS in, e.g., e-commerce warehouses, MCFP-h* is hence the most relevant solution method. Moreover, we derive the following practical insights, which might be interesting from a managerial perspective:

- While using a multi-level facility for an RMFS warehouse improves space utilization, it comes at the cost of productivity of operative picking processes. Especially the packers at consolidation stations will suffer from multiple levels since it increases the consolidation effort of partially picked orders, which, in a multi-level warehouse, are spread among different levels.
- Division of a planning horizon into smaller waves eases the forecasting and solving of short-term planning problems. However, waves that are too short harm the consolidation effect by not considering the similarities of all orders, so that a higher number of pods are required to cover all orders. This could increase the workload for the mobile robots, which are responsible to move the required pods to the stations.

Due to the model assumptions listed in Sect. 3.3, our study and results consist of some limitations, which can be further explored in future research. Future research should focus on integrating OPAP with the related planning problems, like order processing, SKU and pod storage assignment, or traffic management. Moreover, combining OPAP with the pod replenishment problem seems a challenging task for future research. Due to the high complexity of these operative processes, it is suggested to integrate the planning problems into a simulation model, which would enable exploring the impact of the proposed surrogate objectives on various practical KPIs, such as cycle times, required resources, or energy consumption. On the algorithmic side, solving OPAP with another multi-objective-approach, such as weighting objectives (e.g., using the analytic hierarchy process) or combining objectives, could be another interesting and challenging stream for future research as the proposed lexicographic approach is only one of many possible ways to deal with multi-objective optimization problems.

Acknowledgements This work is funded by the Federal Ministry for Economic Affairs and Climate Action based on a resolution of the German Bundestag, specifically, by the German Federation of Industrial Research Associations (AiF), Industrial Collective Research (IGF): Project number IGF-20351 N, further project results are available via IGF. The authors would also like to thank the project partners (logistics providers and experts from the industry) for their insightful comments and suggestions, who declare they have no financial interests.

Funding Open Access funding enabled and organized by Projekt DEAL. This work is funded by the Federal Ministry for Economic Affairs and Climate Action based on a resolution of the German Bundestag (specifically, of the German Federation of Industrial Research Associations (AiF), Industrial Collective Research (IGF), Project number IGF-20351 N).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Al Chami Z, Manier H, Manier MA (2019) A lexicographic approach for the bi-objective selective pickup and delivery problem with time windows and paired demands. Ann Oper Res 273(1):237–255

Azadeh K, De Koster R, Roy D (2019) Robotized and automated warehouse systems: review and recent developments. Transp Sci 53(4):917–945

Barnhart C, Hane CA, Vance PH (2000) Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Oper Res 48(2):318–326

Bender PS (1981) Mathematical modeling of the 20/80 rule: theory and practice. J Bus Logist 2(2):139–157

Boysen N, Briskorn D, Emde S (2017) Parts-to-picker based order processing in a rack-moving mobile robots environment. Eur J Oper Res 262(2):550–562

Boysen N, de Koster R, Weidinger F (2019a) Warehousing in the e-commerce era: a survey. Eur J Oper Res 277(2):396–411

Boysen N, Stephan K, Weidinger F (2019b) Manual order consolidation with put walls: the batched order bin sequencing problem. EURO J Transp Logist 8(2):169–193

Cui Y, Geng Z, Zhu Q, Han Y (2017) Multi-objective optimization methods and application in energy saving. Energy 125:681–704

De Koster R, Le-Duc T, Roodbergen KJ (2007) Design and control of warehouse order picking: a literature review. Eur J Oper Res 182(2):481–501

Deb K (2014) Multi-objective optimization. In: Search methodologies. Springer, Boston, MA, pp 403–449

Ecker JG, Kouada IA (1978) Finding all efficient extreme points for multiple objective linear programs. Math Program 14(1):249–261

Fragapane G, de Koster R, Sgarbossa F, Strandhagen JO (2021) Planning and control of autonomous mobile robots for intralogistics: literature review and research agenda. Eur J Oper Res 294(2):405–426

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York

Guan M, Li Z (2018) Genetic algorithm for scattered storage assignment in kiva mobile fulfillment system. Am J Oper Res 8(6):474–485

Gunantara N (2018) A review of multi-objective optimization: methods and its applications. Cogent Eng 5(1):1502242

Hoseinpour Z, Kheirkhah AS, Fattahi P, Taghipour M (2020) The problem solving of bi-objective hybrid production with the possibility of production outsourcing through meta-heuristic algorithms. Management 4(2):1–17

Hoseinpour Z, Taghipour M, Beigi JH, Mahboobi M (2021) The problem solving of bi-objective hybrid production with the possibility of production outsourcing through imperialist algorithm, NSGA-II, GAPSO hybrid algorithms. Turk J Comput Math Educ (TURCOMAT) 12(13):8090–8111

IBM (2021) Multiobjective optimization. https://www.ibm.com/docs/en/icos/12.10.0?topic=optimization-multiobjective. Accessed July 2022
Isermann H (1982) Linear lexicographic optimization. OR Spektrum 4(4):223–228
Jaghbeer Y, Hanson R, Johansson MI (2020) Automated order picking systems and the links between design and performance: a systematic literature review. Int J Prod Res 58(15):4489–4505
Kim HJ, Pais C, Shen ZJM (2020) Item assignment problem in a robotic mobile fulfillment system. IEEE Trans Autom Sci Eng 17(4):1854–1867
Lamballais T, Roy D, De Koster RBM (2017) Estimating performance in a robotic mobile fulfillment system. Eur J Oper Res 256(3):976–990
Lamballais Tessensohn T, Roy D, De Koster RBM (2020) Inventory allocation in robotic mobile fulfillment systems. IISE Trans 52(1):1–17
Li X, Hua G, Huang A, Sheu J-B, Cheng TCE, Huang F (2020) Storage assignment policy with awareness of energy consumption in the Kiva mobile fulfilment system. Transp Res Part E Logist Transp Rev 144:102–158
Merschformann M, Xie L, Erdmann D (2017) Path planning for robotic mobile fulfillment systems. Preprint arXiv:1706.09347
Merschformann M, Xie L, Li H (2018) RAWSim-O: a simulation framework for robotic mobile fulfillment systems. Logistics Research. Advance online publication. https://doi.org/10.23773/2018_8
Merschformann M, Lamballais T, De Koster MBM, Suhl L (2019) Decision rules for robotic mobile fulfillment systems. Oper Res Persp 6:100–128
Roy D, Nigam S, de Koster R, Adan I, Resing J (2019) Robot-storage zone assignment strategies in mobile fulfillment systems. Transp Res Part E Logist Transp Rev 122:119–142
Steuer RE (1976) Multiple objective linear programming with interval criterion weights. Manage Sci 23(3):305–316
Tadumadze G, Boysen N, Emde S, Weidinger F (2019) Integrated truck and workforce scheduling to accelerate the unloading of trucks. Eur J Oper Res 278(1):343–362
Tadumadze G, Emde S, Diefenbach H (2020) Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines. OR Spectrum 42(2):461–497
Valle CA, Beasley JE (2021) Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment. Comput Oper Res 125:105090
Wang B, Yang X, Qi M (2022) Order and rack sequencing in a robotic mobile fulfillment system with multiple picking stations. Advance online publication, Flexible Services and Manufacturing Journal
Weidinger F, Boysen N, Briskorn D (2018) Storage assignment with rack-moving mobile robots in KIVA warehouses. Transp Sci 52(6):1479–1495
Wurman PR, D'Andrea R, Mountz M (2008) Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI Mag 29(1):9–9
Xie L, Thieme N, Krenzler R, Li H (2021) Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems. Eur J Oper Res 288(1):80–97
Zionts S, Wallenius J (1976) An interactive programming method for solving the multiple criteria problem. Manage Sci 22(6):652–663
Zou B, Gong Y, Xu X, Yuan Z (2017) Assignment rules in robotic mobile fulfilment systems for online retailers. Int J Prod Res 55(20):6175–6192

Giorgi Tadumadze is a data scientist and optimization engineer at Techgroup Schweiz AG - Maison du Software. During working on this article, he was a postdoctoral research associate at the Chair of Management Science/Operations Research at the Technical University of Darmstadt, where he received his PhD, researching on mathematical optimization and scheduling problems in production and logistics. At Techgroup Schweiz AG - Maison du Software, he develops optimization algorithms and codes the future of logistics with the latest technologies and full-blooded engineers.

Julia Wenzel is a research assistant at the Chair of Management and Logistics at the Technical University of Darmstadt. She holds a Master's degree in Mechanical and Process Engineering at the Technische Universität Darmstadt. Her research focuses on the simulation of warehouse processes, especially order picking processes.

Simon Emde is Professor of Business Informatics, esp. Business Intelligence, at Friedrich-Schiller-Universität Jena, Germany. He teaches courses on business intelligence, machine learning, and decision science. His research interests include scheduling, combinatorial optimization, and business analytics.

Felix Weidinger is professor for Operations Research at the Technical University of Darmstadt. He received his PhD from the Friedrich-Schiller-University in Jena, where he worked on e-commerce warehousing. His research focuses on quantitative problems in the logistics sector and the sharing economy. Based on techniques of Operations Research and Machine Learning, his research helps to improve productivity and save resources.

Ralf Elbert is Professor and Head of the Chair of Management and Logistics at the Technical University of Darmstadt. He has extensive experience in transportation logistics, especially intermodal freight transportation and intralogistics. His research focuses on method-based and instrumentally-applicable support of planning and analysis of transport and logistics systems. He is a member of various associations such as Transportation Research Board (TRB) and World Conference on Transport Research Society (WCTRS).