A two-stage approach for order and rack allocation in a mobile rack environment

Cristiano Arbex Valle1 John E Beasley2

1Departamento de Ciência da Computação,
Universidade Federal de Minas Gerais,
Belo Horizonte, MG 31270-010, Brasil
arbex@dcc.ufmg.br

2Brunel University
Mathematical Sciences, UK
john.beasley@brunel.ac.uk

Abstract
In this paper we investigate a problem associated with operating a robotic mobile fulfilment system (RMFS). This is the problem of allocating orders and mobile storage racks to pickers.

We present a two-stage formulation of the problem. In our two-stage approach we, in the first-stage, deal with the orders which must be definitely fulfilled (picked), where the racks chosen to fulfil these first-stage orders are chosen so as to (collectively) contain sufficient product to satisfy all orders. In the second-stage we restrict attention to those racks chosen in the first-stage solution in terms of allocating second-stage orders.

We present three different strategies for first-stage order selection; one of these strategies minimises the requirement to make decisions as to the rack sequence (i.e. the sequence in which racks are presented to each picker).

We present a heuristic procedure to reduce the number of racks that need to be considered and too present a heuristic for order and rack allocation based on partial integer optimisation that makes direct use of our two-stage formulation.

Extensive computational results are presented for test problems that are made publicly available.

Keywords: integer programming, inventory management, mobile storage racks, order picking, partial integer optimisation, robotic mobile fulfilment systems

1 Introduction
In a facility operating as a Robotic Mobile Fulfilment System (RMFS) mobile robots bring moveable racks of shelves containing inventory to static pickers, so that these pickers can pick the items needed for customer orders. Such a facility is an example of a parts-to-picker system. This contrasts with the more traditional picker-to-parts systems, in which pickers walk/ride through the facility collecting requested items. In operating a RMFS decisions need to be made as to the batching of orders (i.e. the set of orders to be assigned to each picker), as well as which racks should be allocated (moved) to which picker.

Robotic mobile fulfilment systems are increasingly common in the B2C (business to consumer) market. Moreover their operation is an increasing topic of research within the academic community. In this section we will not repeat material relating to the operation of a RMFS most likely familiar to the majority of readers. Rather we give here just a short description of the operation of a RMFS relevant to the problem we examine in this paper. Readers totally unfamiliar with the operation of a RMFS are referred to Azadeh et al. [2019], Boysen et al. [2017, 2019].

In a RMFS product to satisfy customer demand is stored on racks, also referred to as pods, or inventory pods. These racks are moved by small mobile robots to static pickers, who pick items from the rack to satisfy customer orders. Racks are placed in a queue leading past each picker and successively presented to the picker. The picker picks items from the presented rack and places them in one or more bins, each bin being associated with a single customer order (in some contexts these bins are known as crates or totes). Once all the items in an order have been picked the bin containing the completed order is put onto a conveyor, being carried away for further processing, and a new empty bin is positioned in the space so created. Often a scattered storage policy is adopted, so that the same product is stored in multiple racks in order to try and ensure that there is a rack containing required product free when needed [Weidinger and Boysen] [2018].
This paper deals with two related problems, namely given a fixed number of pickers which orders should be allocated to which picker? Also which racks should be used so that the orders allocated to a picker can be fulfilled? In this paper we present a two-stage procedure to answer these questions.

Our two-stage approach recognises that in a RMFS typically new orders appear relatively rapidly, and so for efficient operation some orders need not be fulfilled (picked) immediately, rather they can be delayed until a more opportune time. For example delayed until one or more orders that can utilise the same rack as the delayed order appear. Clearly customer orders cannot be delayed indefinitely, but within limits orders need not necessarily be dealt with immediately. For example in a RMFS there is typically no necessity to fulfil orders on a first-come first-served basis.

In our two-stage approach we have that in the first-stage the user can specify the orders which must be definitely fulfilled, e.g. if they have been delayed too long, or are a priority order, or satisfy some other user-imposed criteria. The racks chosen to fulfil these first-stage orders are chosen so as to (collectively) contain sufficient product to satisfy all orders. In the second-stage we restrict attention to those racks chosen in the first-stage solution in terms of allocating second-stage orders. We believe that this two-stage approach, which provides the user with explicit control over the fulfilment of orders, enables a RMFS to be better managed.

In this paper we also present a reduction procedure to reduce the number of racks considered. We also present a partial integer optimisation heuristic for the problem based upon our formulation. Extensive computational results are given for publicly available test problems involving up to 500 products, 500 orders, 200 racks and 10 pickers.

The structure of this paper is as follows. In Section 2 we review the relevant literature on the operation of a robotic mobile fulfilment system. In the light of this review we also state what we believe to be the contribution of this paper to the literature. In Section 3 we present our two-stage formulation and outline three different strategies that can be used with the formulation. One of these strategies minimises the requirement to make decisions as to the rack sequence (i.e. the sequence in which racks are presented to each picker). In Section 4 we present our procedure to reduce the number of racks that need to be considered. In Section 5 we present a partial integer optimisation heuristic for the order and rack allocation problem based upon our formulation. Section 6 presents our computational results for the test problems we examined. Finally in Section 7 we present our conclusions.

2 Literature review

In this section we first review recent literature on the operation of a robotic mobile fulfilment system. In the light of this review we also state what we believe to be the contribution of this paper to the literature.

2.1 Recent literature

Azadeh et al. [2019] and Boysen et al. [2019] have recently presented work discussing the literature regarding warehousing systems involving automated and robotic handling. Jagheer et al. [2020] have presented a survey focusing on automated order picking systems which includes parts-to-picker systems involving mobile robots such as considered in this paper. da Costa Barros and Nascimento [2021] have presented a survey on recent developments and research opportunities with respect to robotic mobile fulfilment systems. They discussed a number of problem areas associated with such systems and identified many research challenges. The literature survey in Valle and Beasley [2021] included additional papers beyond those considered in Azadeh et al. [2019], Boysen et al. [2019]. Accordingly, for space reasons, we only consider in this section work additional to that discussed there.

Merschformann et al. [2019] used discrete-event simulation to investigate the impact of a large number of different decision rules in the operation of a RMFS. In terms of allocation of orders to pickers each picker chose an order from those available as and when needed, so unlike the work considered in this paper they did not consider simultaneous allocation of orders to pickers. They also considered rack replenishment and rack storage (i.e. where in the facility a rack should remain whilst unused). One conclusion from their work was that the choice of order for each picker had a significant effect on throughput.

Ouzidan et al. [2020] considered the problem of a single picker and how to sequence the orders dealt with by that picker, as well as the racks to be brought to the picker. The problem they consider has also been considered in Boysen et al. [2017], Fussler and Boysen [2019], Valle and Beasley [2021]. In their paper all racks contain just a single item and sufficient to satisfy all orders, rack revisits to the picker are allowed. They presented a zero-one integer program for the problem, minimising the number of racks brought to the picker. They also presented a constructive heuristic for the problem which is used to generate a starting solution for a general variable neighbourhood search heuristic. Computational results were presented. They compared the performance of their formulation with that given in Fussler and Boysen [2019] and concluded that their formulation performed considerably better. Heuristic
results were also presented both on artificial (synthetic) instances and real-world instances both for their general variable neighbourhood search heuristic and for the heuristics presented in Boysen et al. [2017], Fussler and Boysen [2019].

Cai et al. [2021] considered two related RMFS issues, storage location assignment and robot path planning. They presented a mathematical model to address these problems. Simulation was used to gain insight into the effectiveness of their approach. Duan et al. [2021] considered a RMFS and used queueing theory to examine the performance of the system with time-varying arrivals. Simulation was used to validate their analytic estimates.

Gheliči and Kilani [2021] considered the situation where mobile robots are involved, but they interact with human pickers who walk fixed storage aisles (within an assigned picker-dependent zone) collecting items for customer orders. In their paper robots can take full carts from the end of aisles, where the carts contain items collected by a picker. Alternatively the robots can accompany pickers as they move down the aisles and act as a mobile cart into which pickers place picked items. The robots move picked items to a consolidation area to consolidate customer orders. They presented a number of analytical models for estimating the performance of the picking system and compared these models with simulation results. The situation considered in their paper contrasts with the situation considered in this paper where the pickers operate at fixed locations and items for customer orders are brought to them on storage racks moved by mobile robots. Moreover in our work customer order consolidation takes place naturally as a separate bin is used at each picker for each customer order, rather than as a separate process after items have been collected.

Li et al. [2021] presented a simulation study dealing with a RMFS with high-density storage. Here high-density storage means that not all racks are immediately accessible, rather in some cases other racks must first be moved aside in order to access a required rack. They assume that orders have been preassigned to pickers and racks, so they focus on assigning movement tasks to robots, robot path planning and traffic control. Simulation was used to compare traditional and high-density storage systems.

Mirzaei et al. [2021] considered the problem of product storage in a RMFS, with the aim being to store product so as to minimise expected robot travel time. A mixed-integer linear program for storage allocation was presented, as well as a construction and improvement heuristic. Computational results were presented for a problem based upon real-world data.

Rimelé et al. [2021] modelled the operational decisions in a RMFS as a stochastic dynamic program. They considered a number of elements, such as stochastic demand (orders are revealed over time) and stochastic travelling and picking times. Computational results, based on simulation, illustrating their model in the case of one picker when using storage allocation decision rules from the literature were given.

Shi et al. [2021] presented a two-stage hybrid heuristic algorithm for the problem of simultaneously allocating orders and racks to pickers. In the first stage dynamic programming together with a beam search heuristic was used to find a critical rack set, the set of most promising racks to use. In the second stage a constructive heuristic and an adaptive neighbourhood search heuristic were used to find an allocation of orders and racks to pickers. Computational results were presented for a number of randomly generated instances and for a problem based upon real-world data.

Xie et al. [2021] considered the problem of assigning orders and racks to pickers. In their paper they do not explicitly consider the number of units of each product stored on a rack, rather they assume that if a rack contains a product then it has sufficient of that product to satisfy all orders requiring that product. In addition they assume that each order requires just one unit of each product associated with the order. They presented a formulation of the problem as a zero-one integer program that minimises a weighted sum of the number of racks used and unused picker capacity. They do not impose a constraint which requires orders to be picked, rather the orders to be picked arise as a result of the term in the minimisation objective associated with unused picker capacity. They extended the formulation to account for the splitting of orders, i.e. an order could be split so that products for the order were picked by two or more pickers. They also extend their formulation to allow orders to be split across different time periods. They presented proofs that all of the problems considered are NP-hard. Computational results, based on simulation, were given for 24 instances, where Gurobi was used as the optimiser. They also amended their approach to act as a heuristic for a real-world problem.

Yang et al. [2021a] considered the problem as to how, for a single picker and a given set of orders, to sequence the processing of the orders as well as how to decide the allocation and arriving sequence of racks. They do not explicitly consider the number of units of each product stored on a rack, rather they assume that if a rack contains a product then it has sufficient of that product to satisfy all orders requiring that product. They formulated the problem as a mixed-integer linear program and presented a two-stage solution procedure consisting of an initial greedy heuristic stage followed by an improvement stage. Computational results were presented.
Yang et al. [2021b] considered the problem of where to locate picker workstations in a RMFS and presented an
integer programming model to determine the location of workstations to minimise total robot travelling distance.
They considered both traditional and flying-V layouts. Computational results were presented.

Zhuang et al. [2021] considered the problem of assigning orders and racks to pickers. They do not explicitly
consider the number of units of each product stored on a rack, rather they assume that if a rack contains a
product then it has sufficient of that product to satisfy all orders requiring that product. In their paper they
considered workload balancing between pickers and rack conflicts. A rack conflict occurs when two or more pickers
both require the same rack simultaneously. They presented a mixed-integer programming model for the problem.
They also presented an adaptive large neighbourhood search heuristic that included a simulated annealing phase.
Computational results were presented both for small randomly generated problems and for a large problem based
upon real-world data.

We should also mention here our previous work Valle and Beasley [2021]. In that paper we consider the problem
of order and rack allocation, but also consider the problem of how to sequence racks for presentation to a picker.
In the approach given in that paper we split the decision as to order and rack allocation from rack sequencing. In
other words we first decide the order and rack allocation. Then, given the orders and racks allocated to each picker,
we decide how to sequence the racks. In that paper, as well as deciding the sequence of racks at a picker, we show
that the approach given to decide the rack sequence explicitly implies the sequence in which orders are processed.

With regard to order and rack allocation two major differences between the work given in that paper and the
work presented here are that Valle and Beasley [2021] is: (a) a single stage approach; where (b) all orders are treated
equally. By contrast in this paper we have: (a) a two-stage approach; in which (b) we explicitly split orders into
two sets, one set which must be picked and a second set which might be picked if appropriate, so treating orders
unequally.

Our judgement is that a two-stage approach better represents the decision with respect to order and rack
allocation faced by the user within a RMFS. Our two-stage approach allows the user flexibility to distinguish
between orders, picking some now, allowing others (if advantageous) to be held over for later fulfilment.

2.2 Contribution

In light of the literature review above we believe that the contribution of this paper to the literature is:

- to present an innovative two-stage optimisation based approach for simultaneously allocating both orders and
  racks to multiple pickers that better represents the decision faced by the user of a RMFS with respect to order
  and rack allocation
- to explore use of our two-stage approach with three different strategies for first-stage order selection; one of
  these strategies minimises the requirement to make decisions as to the rack sequence (i.e. the sequence in
  which racks are presented to each picker)
- to present a heuristic procedure to reduce the number of racks that need be considered; this procedure is
  independent of our two-stage approach and hence could easily be incorporated into any other solution
  approach (heuristic or optimal) for order and rack allocation
- to present a heuristic for order and rack allocation that makes direct use of our two-stage mathematical
  formulation
- to present extensive computational results for test problems that are made publicly available for use by other
  researchers

3 First-stage and second-stage formulations

Suppose that there are in total $N$ products within a facility that can be ordered by customers. We currently have
$O$ orders to be allocated to pickers where each order $o$ requires $q_{io}$ units of product $i$. In a practical setting it is
very likely that the set of orders currently being considered does not involve all of the $N$ products, so let the set of
active products $I$ be defined using $I = \{ i \mid \sum_{o=1}^{O} q_{io} \geq 1 \ i = 1, \ldots, N \}$. Let $Q$ be the set of orders comprising
just a single unit of one product, so $Q = \{ o \mid \sum_{i \in I} q_{io} = 1 \ o = 1, \ldots, O \}$ since if we have some order $o$ for which
$\sum_{i \in I} q_{io} = 1$ then this can only occur if the order is just for a single unit of one product.

There are $P$ pickers, where picker $p$ has a capacity for at most $C_p$ orders. Without loss of generality assume
that the pickers are indexed in decreasing $C_p$ order (so $C_p \geq C_{p+1} \ p = 1, \ldots, (P - 1)$).
We have $R$ mobile racks which can be allocated (moved) to pickers. Rack $r$ contains $s_{ir}$ units of product $i$. Let $\Gamma(i)$ be the set of racks that contain one or more units of product $i$, so $\Gamma(i) = \{r \mid s_{ir} \geq 1 \text{ } r = 1, \ldots, R\}$. Let $\gamma_i$ be the minimum number of racks that are needed to supply all orders involving product $i$. Let $\Delta(o) = \{r \mid r \in \Gamma(i) \text{ } i \in I \text{ } q_{io} \geq 1\}$ be the set of racks that involve one (or more) products in order $o$. Let $\delta_o$ be the minimum number of racks that are needed to supply order $o$. We indicate later below how to find values for $\gamma_i$ and $\delta_o$. We assume without loss of generality that any orders that cannot be satisfied at all, even by using the entire set of racks, have been identified and eliminated from the problem.

Note here that in our work we explicitly consider rack inventory positions and the number of units of each product required by each order. This contrasts with previous work in the literature [Boysen et al. 2017, Fussler and Boysen 2019, Ouzidan et al. 2020, Xie et al. 2021, Yang et al. 2021a, Zhuang et al. 2021] which does not explicitly consider the number of units of each product stored in a rack, rather such work assumes that if a rack contains a product then it has sufficient of that product to satisfy all orders requiring that product. Note also here that our work considers multiple pickers simultaneously, in contrast to previous work that just considers one picker.

In our approach we partition the set of orders into two disjoint sets $F$ and $S$ where $F \cap S = \emptyset$ and $F \cup S = \{o \mid o = 1, \ldots, O\}$. $F$ is the set of orders which must be allocated to pickers whilst respecting picker capacity constraints, and $S$ is the set of orders which (potentially) may involve some orders that cannot be allocated to pickers (e.g. because to do so would violate picker capacity constraints). Our approach involves:

- a first-stage formulation allocating both orders $o \in F$ and racks to pickers whilst maintaining inventory levels sufficient for orders $o \in S$
- a second-stage formulation allocating (if possible) orders $o \in S$ to pickers, but restricting attention to just the set of racks assigned to pickers at the first-stage

### 3.1 First-stage formulation

Our first-stage formulation for allocating orders and racks to pickers involves the following zero-one decision variables:

- $x_{op} = 1$ if order $o$ is allocated to picker $p$, zero otherwise
- $u_r = 1$ if rack $r$ is used, so allocated to some picker, zero otherwise
- $y_{rp} = 1$ if rack $r$ is allocated to picker $p$, zero otherwise
Our formulation is:

\[
\begin{align*}
\text{min} & \quad M \sum_{r=1}^{R} u_r - \sum_{i \in I} \left( \sum_{r=1}^{R} s_{ir} y_{rp} - \sum_{o=1}^{O} q_{io} \right) / \sum_{o=1}^{O} q_{io} \\
\text{subject to:} & \quad \sum_{o \in F} x_{op} \leq C_p \quad \forall p \in \{1, \ldots, P\} \\
& \quad \sum_{p=1}^{P} x_{op} = 1 \quad \forall o \in F \\
& \quad \sum_{p=1}^{P} y_{rp} = u_r \quad \forall r \in \{1, \ldots, R\} \\
& \quad \sum_{r=1}^{R} s_{ir} y_{rp} \geq \sum_{o \in F} q_{io} x_{op} \quad \forall i \in I; p \in \{1, \ldots, P\} : \sum_{o \in F} q_{io} \geq 1 \\
& \quad \sum_{r=1}^{R} s_{ir} u_r \geq \sum_{o=1}^{O} q_{io} \quad \forall i \in I \\
& \quad \sum_{r \in \Gamma(i)} u_r \geq \gamma_i \quad \forall i \in I \\
& \quad \sum_{r \in \Delta(o)} u_r \geq \delta_o \quad \forall o \in F \\
& \quad u_r = 0 \quad \forall r \in \{1, \ldots, R\} : \sum_{o=1}^{O} \sum_{i \in I} s_{ir} q_{io} = 0 \\
& \quad x_{op} \in \{0, 1\} \quad \forall o \in F; p \in \{1, \ldots, P\} \\
& \quad u_r \in \{0, 1\} \quad \forall r \in \{1, \ldots, R\} \\
& \quad y_{rp} \in \{0, 1\} \quad \forall r \in \{1, \ldots, R\}; p \in \{1, \ldots, P\}
\end{align*}
\]

Equation (2) ensures that we allocate orders \( o \in F \) to each picker so as to respect picker capacity. Equation (3) ensures that each order \( o \in F \) is allocated to one picker. Note here that in this first-stage formulation we are not allocating orders \( o \in S \) to pickers. Equation (4) ensures that each rack is either allocated to one picker, or not used at all.

Equation (5) is the product supply constraint and ensures that, for each product \( i \) and each picker \( p \), the number of units of that product available from the racks assigned to that picker (so \( \sum_{r=1}^{R} s_{ir} y_{rp} \)) is sufficient to meet the required number of units of product \( i \) at picker \( p \) given the orders in \( F \) allocated to the picker (so \( \sum_{o \in F} q_{io} x_{op} \)). Hence Equation (5) ensures that appropriate racks are assigned to picker \( p \) so as to enable the picking of all of the products associated with the orders \( o \in F \) assigned to the picker.

Equation (6) ensures that, over the entire set of racks allocated to pickers, the total number of units of product \( i \) available in those racks (so \( \sum_{r=1}^{R} s_{ir} u_r \)) is sufficient to satisfy all orders (where the orders require in total \( \sum_{o=1}^{O} q_{io} \) units of product \( i \)). This constraint links the racks chosen at the first-stage with the allocation of orders \( o \in S \) to these chosen racks at the second-stage in terms of having sufficient of each product available (after allocation of orders \( o \in F \) at the first-stage) for orders \( o \in S \) at the second-stage (as in the second-stage formulation presented later below).

Equation (7) means that we choose at least one rack for each picker. The assumption behind this constraint is that since we have \( P \) pickers available we wish to make use of them. Conceptually, assuming pickers have unlimited picker order capacity for the sake of illustration, using just one picker would minimise the overall number of racks needed. However clearly, in practice, one picker picking all orders would slow order throughput. Having \( P \) pickers available implicitly implies that we wish to make use of them, and we represent this using Equation (7).

Equation (8) ensures that for each product \( i \) we use at least the minimum number \( (\gamma_i) \) of racks needed. Equation (9) ensures that for each order \( o \in F \), which we know (from Equation (3)) must be allocated to some picker, we use at least the minimum number \( (\delta_o) \) of racks needed.
Given the $O$ orders then we know the products needed. If a rack contains none of the required products then clearly it is irrelevant and will never be assigned to a picker, and so can be removed from the problem. Equation (10) ensures that any such racks are eliminated since the term $\sum_{o=1}^{O} \sum_{i \in I} s_{ir} q_{io}$ is zero if and only if rack $r$ does not contain any of a required product. Note here that clearly any rack $r$ for which $u_r = 0$ implies from Equation (4) that $y_{rp} = 0$ $p = 1, \ldots, P$. The solver we used (Cplex \cite{CPLEX2019}) contains procedures to automatically identify this and so we have not included an explicit constraint relating to the elimination of these variables in the formulation presented above. Equations (11)-(13) are the integrality constraints.

In our objective, Equation (1), $M$ is a large positive constant and the first term in that objective ensures that we minimise the total number of racks allocated to pickers. Using an objective related to rack minimisation is common in the literature \cite{Boysen2017, Hanson2018, Valle2021, Xie2021} as it contributes to less movement of racks from the storage area to the picking area, and too encourages the picking of multiple products for different orders from the same rack.

Given that there may be alternative optimal solutions involving the same (minimum) number of racks we make use of the second term in the objective. This term is designed to maximise the number of units of products $i \in I$ associated with the racks chosen. In this objective term ($\sum_{r=1}^{R} s_{ir} u_r - \sum_{o=1}^{O} q_{io}$) is the total number of units of product $i$ on the racks chosen ($\sum_{r=1}^{R} s_{ir} u_r$) in excess of those needed ($\sum_{o=1}^{O} q_{io}$), where this excess is proportionally scaled by the total number needed ($\sum_{o=1}^{O} q_{io}$). This term, with its associated negative sign in Equation (1), means that we maximise the number of units of products $i \in I$ associated with the racks chosen. This gives us additional flexibility in terms of allocation of orders $o \in S$ at the second-stage.

Rearranging Equation (1), and dropping the constant term, we have that our objective is:

$$\min \sum_{r=1}^{R} (M - \sum_{i \in I} (s_{ir}/\sum_{o=1}^{O} q_{io})) u_r$$

With this objective a suitable value for $M$ is any value that exceeds $\sum_{r=1}^{R} \sum_{i \in I} (s_{ir}/\sum_{o=1}^{O} q_{io})$.

With respect to a minor technical issue here note that we need to assume that $\sum_{p=1}^{P} C_p \geq |F|$ in order for Equations (2), (3) not to lead to infeasibility. This assumption that the pickers (in total) have capacity to deal with $|F|$ orders is not a limiting one. If it is not automatically satisfied then we can add a single “artificial picker” whose capacity is set so that in total the pickers can deal with $|F|$ orders.

### 3.2 Second-stage formulation

Once the first-stage formulation presented above, Equations (1)-(13), has been solved then we will have decided the racks to be allocated to each picker. In addition, for all orders $o \in F$, the picker to which they have been allocated will also have been decided. Note here however that it is possible that some of the chosen racks will not have any orders allocated to them. For example, racks without any allocated orders may have been chosen to ensure that we satisfy Equation (9), so have sufficient inventory for all $O$ orders. For such racks the picker to which they have been allocated (via Equation (4)) will be arbitrary.

In our approach the racks chosen at the first-stage are not changed when we come to the second-stage. Allocation of all orders at the second-stage then occurs only considering the set of racks chosen at the first-stage (rather than considering the complete set of racks). The logic behind this is that we know from the first-stage solution that with this chosen set of racks we can feasibly allocate all orders $o \in F$ (from Equation (3)). However, by judicious reallocation of orders to racks, and racks to pickers, we may be better able to deal with allocating orders $o \in S$ to pickers, as these orders have yet to be allocated.

Equation (6) ensures that collectively the racks chosen in the first-stage formulation solution have sufficient inventory left (after dealing with orders $o \in F$) to deal with all orders $o \in S$. However, this is not sufficient to ensure that we can always feasibly allocate all orders $o \in S$ to pickers. This may, for example, be due to the fact the racks allocated to a picker do not contain sufficient product to satisfy an order, even if the racks in total have sufficient product available (recall here that orders must be supplied by a single picker). Alternatively allocation of all orders $o \in S$ to pickers whilst respecting rack inventory positions may require violation of picker capacity.

To account for these possibilities we, in allocating orders $o \in S$ to pickers, aim to maximise the number of allocated orders. Note here that use of the first-stage objective function term associated with maximising the number of units of products $i \in I$ was designed to enable more orders to be allocated to pickers at the second-stage.

Let $\Theta = \{ r \mid u_r = 1, r = 1, \ldots, R \}$ be the set of racks chosen in the first-stage solution. Let $v_o = 1$ if order $o \in S$
is allocated to some picker in the second-stage solution, zero otherwise. Then our second-stage formulation is:

\[
\max \sum_{o \in S} v_o \tag{15}
\]

subject to:

\[
\sum_{o \in S} x_{op} \leq C_p \quad \forall p \in \{1, \ldots, P\} \tag{16}
\]

\[
\sum_{p=1}^{P} x_{op} = 1 \quad \forall o \in F \tag{17}
\]

\[
\sum_{p=1}^{P} x_{ap} = v_o \quad \forall o \in S \tag{18}
\]

\[
\sum_{p=1}^{P} y_{rp} = 1 \quad \forall r \in \Theta \tag{19}
\]

\[
\sum_{r \in \Theta} s_{ir} y_{rp} \geq \sum_{o=1}^{O} q_{io} x_{op} \quad \forall i \in I; p \in \{1, \ldots, P\} \tag{20}
\]

\[
v_o \in \{0, 1\} \quad \forall o \in S \tag{21}
\]

\[
x_{ap} \in \{0, 1\} \quad \forall o \in \{1, \ldots, O\}; p \in \{1, \ldots, P\} \tag{22}
\]

\[
y_{rp} \in \{0, 1\} \quad \forall r \in \Theta; p \in \{1, \ldots, P\} \tag{23}
\]

Equation (15) maximises the number of orders \(o \in S\) allocated to pickers. Equation (16) ensures that for picker \(p\) the capacity \(C_p\) is respected. Equation (17) ensures that each order \(o \in F\) is allocated to one picker and Equation (18) ensures that each order \(o \in S\) is either allocated to one picker (or not allocated).

Equation (19) ensures that each rack \(r \in \Theta\) is allocated to a picker. Equation (20) is the product supply constraint and ensures that, for each product \(i\) and each picker \(p\), the number of units of that product available from the racks assigned to that picker is sufficient to meet the required number of units of product \(i\) at picker \(p\) given the orders in allocated to the picker. Equations (21)-(23) are the integrality constraints.

In practice in an environment such as Amazon [Weidinger, 2018], where new orders arrive frequently, if we have orders \(o \in S\) associated with a solution involving \(v_o = 0\) (i.e. the orders are not allocated to a picker) then such orders can be left for fulfilment in the near future together with new orders that have yet to appear.

### 3.3 Finding \(\gamma_i\) and \(\delta_o\)

Identifying a value for \(\gamma_i\) for each product \(i\) can be done by solving a simple zero-one integer program involving just a single constraint. For product \(i \in I\) we have that \(\gamma_i\) corresponds to the optimal value of:

\[
\min \sum_{r \in \Gamma(i)} u_r \tag{24}
\]

subject to:

\[
\sum_{r \in \Gamma(i)} s_{ir} u_r \geq \sum_{o=1}^{O} q_{io} \tag{25}
\]

\[
u_r \in \{0, 1\} \quad \forall r \in \Gamma(i) \tag{26}
\]

Equation (24) minimises the number of racks used whilst Equation (25) ensures that for the product considered sufficient racks are chosen to be able to supply all orders with that product. Equation (26) is the integrality constraint. Equations (24, 26) is a simple knapsack problem that can be easily solved. Computationally there is no need to solve this problem if \(\max_{\Gamma(i)} s_{ir} \geq \sum_{o=1}^{O} q_{io}\) since in such a case we know that one rack can supply all orders requiring product \(i\) and so \(\gamma_i = 1\).

Note here that, as the right-hand side of Equation (25) involves summation over all \(O\) orders, the value for \(\gamma_i\) that is found will be based upon the assumption that all orders for product \(i\) are fulfilled (allocated to a picker). In our two-stage approach this assumption is always true for any order \(o \in F\), due to Equation (3). However in the second-stage for orders \(o \in S\) we may arrive at a solution in which some orders are not allocated to any picker, see Equation (18).
Recall here that the objective in the second-stage is to maximise the number of orders allocated to a picker. In our judgement therefore adding the constraint associated with \( \gamma_i \) at the first-stage (so Equation (8)) seemed appropriate in order to try and ensure that the racks chosen at the first-stage contribute to the allocation of orders at the second-stage.

A further advantage of the constraint associated with \( \gamma_i \) (so Equation (8)) is that it contributes to strengthening the linear programming relaxation of our formulation (Equations (1)-(13)). For example suppose that we have some order \( o \in F \) that requires a single unit of a particular product \( i \) and that product is only stored on one of the \( R \) racks, rack \( r \) with inventory level \( s_{ir} \). Then we will have \( \gamma_{oi} = 1 \), \( \Gamma(i) = r \) and Equation (8) will ensure that \( u_r = 1 \). However if Equation (8) was not included in the formulation then from Equation (6) the linear programming relaxation solution would have \( u_r = 1/s_{ir} \), e.g. \( u_r = 0.5 \) if \( s_{ir} = 2 \). This example clearly indicates how Equation (8) can contribute to improving the linear programming relaxation solution of our first-stage formulation.

In a similar fashion to the approach given above for finding the value of \( \gamma_i \) we have that \( \delta_o \), the minimum number of racks needed to fulfil order \( o \in F \), corresponds to the optimal value of:

\[
\min \sum_{r \in \Delta(o)} u_r \\
\text{subject to: } \sum_{r \in \Delta(o)} s_{ir} u_r \geq q_{io} \quad \forall i \in I : q_{io} \geq 1
\]

Equation (27) minimises the number of racks used whilst Equation (28) ensures that for each product involved in order \( o \) we have sufficient inventory on the racks used to supply that product. Equation (29) is the integrality constraint. Equations (27)-(29) is a simple zero-one problem that can be easily solved. Computationally there is no need to solve this problem if \( \sum_{i \in I} q_{io} = 1 \), i.e. the order is just for a single unit of one product, since in such a case we must have \( \delta_o = 1 \).

### 3.4 Choice of \( F \) and \( S \)

Our two-stage approach requires us to define the set of orders \( F \) that are allocated to a picker at the first-stage, or the set of orders \( S \) dealt with at the second-stage (recall here that \( F \cap S = \emptyset \) and \( F \cup S = \{ o \mid o = 1, \ldots, O \} \), so defining one of these two sets automatically defines the other). Clearly in any individual practical RMFS application the user would specify how \( F \) and \( S \) are defined having regard to their own unique circumstances. In this paper, in an academic context, we consider three different strategies for choice of these sets:

- **Strategy 1**: \( S = \emptyset \).
- **Strategy 2**: \( S = Q \), so \( S \) is the set of orders comprising just a single unit of one product
- **Strategy 3**: \( F = \{ o \mid \delta(o) = 1 \} o = 1, \ldots, O \)\), so \( F \) is the set of orders which only require one rack

#### 3.4.1 Strategy 1

In Strategy 1 since \( S = \emptyset \) we allocate all orders to pickers in the first-stage. As such there are no decisions to be made at the second-stage. We would expect this strategy to be computationally the most demanding strategy as it involves explicit consideration of all orders \( F = \{ o \mid o = 1, \ldots, O \} \) in the first-stage.

Note here that since for this strategy we have no second-stage the second term in the objective (Equation (1)) becomes irrelevant and so for this strategy we simply minimise \( \sum_{r=1}^{R} u_r \).

#### 3.4.2 Strategy 2

With regard to Strategy 2 then in our previous work \cite{Valle and Beasley 2021} we showed that when formulating the problem of rack sequencing at each picker advantage was gained by explicit and separate consideration of orders comprising a single unit of one product (subject to certain conditions being satisfied). Distinguishing such orders seems appropriate for environments, such as Amazon \cite{Weidinger 2018}, where the vast number of orders are for only one or two items.

Recall here that, as discussed above, order and rack allocation is followed (for each picker individually) by the sequencing of racks so that the picker can fulfil the orders assigned to them. If an order involves just a single unit of one product then we automatically know that it can be satisfied by just a single rack. All other orders, including
any requiring two or more units of a single product, **may** require two or more racks. For example an order just for two units of the same product may take one unit from two different racks. An order for one unit of each of two different products may also take one unit for these two products from two different racks.

Hence the logic behind Strategy 2 was that it seemed appropriate to consider orders \( F = \{ o \mid o \notin Q \land o = 1, \ldots, O \} \) (where \( Q \) is the set of single unit, single product, orders) potentially involving the use of two or more racks.

In general terms this strategy aims to streamline picker rack sequencing. Recall here, from the description of the problem given above, that the racks allocated to a picker must be sequenced such that all orders allocated to a picker are picked using the bins available for assembling orders at a picker. If it is known that all of the products (and their appropriate quantities) for an order can be fully supplied (picked) from a single rack then this helps simplify the sequencing of picking operations.

Clearly all orders \( o \in Q \) can be fully supplied by a single rack by definition (such any order \( o \in Q \) is for a single unit of one product). However to ensure that orders \( o \in F \setminus Q \) can be fully supplied using just a single rack we need to add additional constraints to the formulation. Let \( \Omega(o) \) be the set of racks that can fully supply order \( o \), i.e. \( \Omega(o) = \{ r \mid r \in \Delta(o) \land q_{io} \leq s_{ir} \forall i \in I \} \). Let \( z_{or} = 1 \) if order \( o \in F \setminus Q \) is fully supplied by rack \( r \in \Omega(o) \), zero otherwise. Then we add to the formulation:

\[
\sum_{r \in \Omega(o)} z_{or} = 1 \quad \forall o \in F \setminus Q \tag{30}
\]

\[
\sum_{o \in F \setminus Q, r \in \Omega(o)} q_{io}z_{or} \leq s_{ir}u_r \quad \forall i \in I; r \in \{ 1, \ldots, R \} \tag{31}
\]

\[
z_{or} - 1 \leq x_{op} - y_{rp} \leq 1 - z_{or} \quad \forall o \in F \setminus Q; r \in \Omega(o); p \in \{ 1, \ldots, P \} \tag{32}
\]

\[
\sum_{r \in \Omega(o)} u_r \geq 1 \quad \forall o \in F \setminus Q \tag{33}
\]

Equation (30) ensures that each order \( o \in F \setminus Q \) is allocated to a single rack \( r \in \Omega(o) \). Equation (31) ensures that there is sufficient inventory on a rack to deal with all orders which must be fully supplied from that rack. Equation (32) ensures that if \( z_{or} = 1 \) then \( x_{op} = y_{rp} p = 1, \ldots, P \). In other words if order \( o \) is fully supplied from rack \( r \) then the picker to which order \( o \) is (uniquely) allocated corresponds to the picker to which rack \( r \) is (uniquely) allocated. If \( z_{or} = 0 \) then this constraint has no effect. Equation (33) ensures that at least one of the racks in \( \Omega(o) \) is used for each order \( o \in F \setminus Q \). Note here that Equation (33) replaces Equation (9) in the first-stage formulation given above.

To amend the second-stage formulation for Strategy 3 we add:

\[
\sum_{r \in \Omega(o) \cap \Theta} z_{or} = 1 \quad \forall o \in F \setminus Q \tag{34}
\]

\[
\sum_{o \in F \setminus Q, r \in \Omega(o) \cap \Theta} q_{io}z_{or} \leq s_{ir} \quad \forall i \in I; r \in \Theta \tag{35}
\]

\[
z_{or} - 1 \leq x_{op} - y_{rp} \leq 1 - z_{or} \quad \forall o \in F \setminus Q; r \in \Omega(o) \cap \Theta; p \in \{ 1, \ldots, P \} \tag{36}
\]

As a consequence of Equations (30)-(36) we, in Strategy 3, choose racks so that all first-stage orders \( F = \{ o \mid \delta(o) = 1 o = 1, \ldots, O \} \), the set of orders which only require one rack, are all fully supplied by a single rack, where \( Q \subseteq F \). Hence under Strategy 3 we **only** have orders \( o \notin F \) being satisfied by two (or more) racks, as it is impossible for them to be fully satisfied by a single rack. In other words Strategy 3 minimises (as far as possible) the requirement for orders to be picked from two or more racks.

Clearly we would expect that using Strategy 3 will require more racks than if we are allowed (as in Strategies 1, 2 and Valle and Beasley [2021]) to satisfy all orders using products picked from a number of different racks. Strategy 3 might appropriate however for some environments.
For example Strategy 3 might be adopted in an environment where each picker has a very limited space to consolidate products for the same order, products for the same order being consolidated in a single assigned bin as they are picked. Indeed under Strategy 3 in the best possible case (where $F = \{ o \mid o = 1, \ldots, O \}$, so all orders can be fully satisfied by just a single rack, each picker would only require one bin, because each order would be completed as it is picked (from a single rack).

Another potentially important advantage of Strategy 3 is that it minimises the requirement to make decisions as to the rack sequence (i.e. the sequence in which racks are presented to each picker). This is because racks can arrive in any (random) sequence for all orders fully supplied by a single rack. Hence rack sequencing is only required for racks involved in satisfying orders which require two or more racks.

Minimising decisions as to the rack sequence in turn implies minimising the decisions related to coordinating the arrival of racks at the picker to achieve a specified rack sequence. Coordinating rack arrival to achieve a specified rack sequence is (in practice) a complex decision process involving a number of different elements: racks at different locations in the facility; rack to picker travel times; robot availability. Minimising the requirement to decide and achieve the rack sequence, thereby simplifying operations, may be an important consideration in some RMFS environments.

With regard to Strategy 3 and choosing racks to fully satisfy individual orders and hence minimise the requirement for rack sequencing note here that a recent paper (Xie et al. [2021]) adopted an opposite approach. In that work an order could be split so that products for the order were picked by two or more pickers. Clearly splitting an order to allow the products involved in that order to be picked by different pickers might well reduce the number of racks needed. However such splitting complicates the downstream operation needed to assemble and ship the order to the customer. For example if the order is shipped as one (e.g. in a cardboard box) then a process is needed to assemble the different picked products back into the single order for shipping. Alternatively if the single customer order is treated in terms of shipping as two (or more) separate orders (so shipped in two or more cardboard boxes) to the same customer then extra shipping and packaging cost will be incurred.

### 3.5 Other constraints

Suppose we have $x_{op} = 1$, so order $o$ is allocated to picker $p$. Then it is clear that at least $\delta_o$ of the racks $r \in \Delta(o)$ must be allocated to picker $p$ to fulfil the order. So we have the constraint:

$$\sum_{r \in \Delta(o)} y_{rp} \geq \delta_o x_{op} \quad \forall o \in \{1, \ldots, O\}; p \in \{1, \ldots, P\}$$

Equation (37) ensures that if order $o$ is allocated to picker $p$ sufficient racks are also allocated to that picker to supply the order.

Moreover for any product $i$ in order $o$ for which $q_{io} \geq 1$ there must be sufficient inventory of that product on the racks $r \in \Delta(o)$ allocated to picker $p$. So we have the constraint:

$$\sum_{r \in \Delta(o)} s_{ir} y_{rp} \geq q_{io} x_{op} \quad \forall o \in \{1, \ldots, O\}; p \in \{1, \ldots, P\}; i \in I : q_{io} \geq 1 \quad (38)$$

In terms of the first-stage Equations (37),(38) are amended to restrict attention to orders $o \in F$. in terms of the second-stage Equation (38) is amended to restrict attention to racks $r \in \Delta(o) \cap \Theta$.

It is possible to set a lower bound on the number of orders dealt with by picker $p$ using:

$$\sum_{o \in F} x_{op} \geq |F| - \sum_{k=1}^{P} C_k \quad \forall p \in \{1, \ldots, P\} \quad (39)$$

Equation (39) follows directly from Equations (2),(3) where the summation on the right-hand side is the maximum number of orders that can be dealt with excluding picker $p$, leaving at least $|F| - \sum_{k=1}^{P} C_k$ which must be dealt with using picker $p$. This constraint applies at both stages. Clearly (for picker $p$) it can only be of benefit if $|F| - \sum_{k=1}^{P} C_k \geq 1$.

We can add some constraints relating to orders which involve one (or more) products that can only be supplied by one rack. Recall that $\Gamma(i)$ is the set of racks that contain product $i$, so $\Gamma(i) = \{ r \mid s_{ir} \geq 1 \forall r = 1, \ldots, R \}$. In terms of the first-stage if we have some order $o \in F$ containing a product $i$ for which $|\Gamma(i)| = 1$ then it is clear that the (unique) rack in $\Gamma(i)$ must be allocated to the same picker as that order. Let $\alpha(i)$ be the first rack in the set $\Gamma(i)$, ties broken arbitrarily. Then we have:

$$x_{op} = y_{\alpha(i)p} \forall o \in F; p \in \{1, \ldots, P\} : \exists i \in I \text{ s.t. } |\Gamma(i)| = 1, q_{io} \geq 1 \quad (40)$$
Note here that there is no need to set \( u_{a(i)} = 1 \) since that will be enforced by Equation (8). We would anticipate that at the first-stage Equation (40) might be relatively ineffective since in that stage we consider the complete set of racks and so there will be relatively few products \( i \in I \) which can only be supplied by one rack. However at the second-stage, where we only consider a subset \( \Theta \) of racks, this constraint might be more useful. In addition in the second-stage we can extend Equation (40) to orders \( o \in S \).

In more detail, and to ease the notation, suppose that we are considering the second-stage and consequently have set \( \Gamma(i) \leftarrow \Gamma(i) \cap \Theta \forall i \in I \). Then in this second-stage we have that Equation (40) applies, and in addition we have

\[
v_o - 1 \leq x_{op} - y_{a(i)p} \leq 1 - v_o \quad \forall o \in S; p \in \{1, \ldots, P\} : \exists i \in I \text{ s.t. } \Gamma(i) = 1, \quad q_{io} \geq 1
\]  

Equation (41) is the equivalent of Equation (40) for orders \( o \in S \) and ensures that if \( v_o = 1 \) then \( x_{op} = y_{a(i)p} = 1, \ldots, P \). If \( v_o = 0 \) then this constraint has no effect.

4 Rack reduction

In our first-stage formulation we potentially consider all racks (although Equation (10) eliminates from consideration any racks that cannot contribute any product items for the orders considered). In order to improve computational performance we can reduce the number of racks that we consider in the first-stage, selecting for consideration only a small set of racks. The procedure we adopted for this is given below.

In terms of rack selection for the first-stage formulation (Equations (1)–(13)) we aim to include sufficient racks to enable us to fulfil orders \( o \in F \) (Equation (5)). In addition we aim to include racks to enable the fulfilment of all orders (Equation (6)).

In order to present our rack selection procedure for the order and rack allocation problem let \( \Phi \) be the set of possible racks for selection, where initially \( \Phi \leftarrow \{1, \ldots, R\} \). Define \( T_r \) as the total number of orders that it is possible to (potentially) fulfil from rack \( r \), but allowing for fractional fulfilment of an order. So:

\[
T_r = \sum_{o=1}^{O} \left( \sum_{i \in I} \min[s_{ir}, q_{io}]/(\sum_{i \in I} q_{io}) \right) \forall r \in \Phi
\]  

In Equation (42) the term \( \left( \sum_{i \in I} \min[s_{ir}, q_{io}]/(\sum_{i \in I} q_{io}) \right) \) is the proportion of order \( o \) that can be supplied by rack \( r \).

Let \( \Lambda \) be an ordered (sequenced) set of customer orders, with the first orders in this list being customer orders \( o \in F \) (in descending \( \sum_{i \in I}(q_{io}/\sum_{r \in \Phi} s_{ir}) \) order, ties broken arbitrarily) and the remaining orders in this list being customer orders \( o \notin F \) (also in descending \( \sum_{i \in I}(q_{io}/\sum_{r \in \Phi} s_{ir}) \) order, ties broken arbitrarily). Here the term \( (q_{io}/\sum_{r \in \Phi} s_{ir}) \) represents for order \( o \) the fraction that \( q_{io} \) is of the total inventory available for product \( i \) on the racks considered, \( \sum_{r \in \Phi} s_{ir} \).

Let \( \pi_{ip} \) be the number of units of product \( i \) available at picker \( p \). Set \( \pi_{ip} \leftarrow 0, \forall i \in I, \ p = 1, \ldots, P \) as initially no racks are allocated to any picker. Let \( \Phi \) be the set of racks that we select, where initially \( \Phi \leftarrow \emptyset \). Then our procedure for selecting racks is:

(a) Consider the racks in descending \( T_r \) order (ties broken arbitrarily) and:

- Allocate the rack \( r \) currently being considered to the picker \( p \) with maximum \( C_p \) value (ties broken arbitrarily). Set \( \pi_{ip} \leftarrow \pi_{ip} + s_{ir}, \forall i \in I \).
- Consider the customer orders \( o \in \Lambda \) in turn and allocate each such order to the current picker \( p \) if the order can be completely fulfilled using the racks now allocated to picker \( p \), i.e., if \( C_p \geq 1 \) and \( q_{io} \geq 1 \). If the order is allocated to picker \( p \) then set \( C_p \leftarrow C_p - 1; \pi_{ip} \leftarrow \pi_{ip} - q_{io}, \forall i \in I; \ q_{io} \geq 1; \ \Lambda \leftarrow \Lambda \setminus \{o\}; \ \Phi \leftarrow \Phi \cup \{r\} \).
- If after considering all customer orders \( o \in \Lambda \) in turn no order has been allocated to picker \( p \) then it is clear that rack \( r \) has made no contribution to the fulfilment of orders so we remove it from picker \( p \), i.e. set \( \pi_{ip} \leftarrow \pi_{ip} - s_{ir}, \forall i \in I \).

(b) Consider all products \( i \in I \) in turn and if there exists any product \( i \) for which there is insufficient of the product on the racks selected to supply total demand (i.e. if \( \sum_{r \in \Phi} s_{ir} < \sum_{o=1}^{O} q_{io} \)) then consider the racks in descending \( T_r \) order and add racks \( r \notin \Phi \) for which \( s_{ir} \geq 1 \) to \( \Phi \) until there is sufficient available.
Formally our PIO heuristic is: fixed to the integer values that they were assigned at previous iterations, or are regarded as continuous variables. at each iteration as integer variables and decides the values for these integer variables. Other variables are either whether we are considering the first-stage or the second-stage. in our two-stage approach discussed above, when we solve the first-stage to proven optimality we pass the set of racks used in the first-stage PIO heuristic solution to the second-stage for use in the second-stage PIO heuristic. In the special case of Strategy 3 we add a rack for each order than does not has a rack available to fully supply it. We would stress here that the rack reduction procedure we have presented above could easily be incorporated into any other solution approach (heuristic or optimal) for order and rack allocation, e.g. by applying it with $F = \{ o \mid o = 1, \ldots, O \}$, so $F$ is the entire set of orders. To incorporate our rack reduction procedure into our two-stage approach it is clear that we need in the first-stage formulation to restrict attention to the set $\Upsilon$ of racks selected. This can be easily done by adding the constraint $u_r = 0 \forall r \notin \Upsilon$ to the first-stage formulation. However excluding from consideration all racks $r \notin \Upsilon$ may lead to infeasibility, and so we do include all racks, but penalise in the objective function use of any rack $r \notin \Upsilon$. Note here that (ignoring the special case of Strategy 3) as this rack selection procedure sequences customer orders using the set $F$ of orders that must be fulfilled in the first-stage formulation, Equation (3), the results from it may differ depending upon the strategy adopted for choice of $F$.

5 Partial integer optimisation (PIO) heuristic

We developed a heuristic for the order and rack allocation problem based upon partial integer optimisation, which is a relatively new approach in the literature. In general terms in a partial integer optimisation (PIO) heuristic we start with a mathematical formulation of the optimisation problem under consideration. From this formulation a subset of the integer variables are declared as integer, with all of the remaining integer variables being declared as continuous. The resulting mixed-integer problem is then solved. The integer variables are then fixed at the values that they have in this solution and the process repeats with a new subset of integer variables being declared as integer. Our partial integer optimisation approach is a matheuristic, as it works directly from a mathematical formulation of the problem under consideration [Boschetti et al., 2009]. A related approach to our partial integer optimisation approach is kernel search [Angeelli et al., 2012, Guastaroba et al., 2017].

As we have both a first-stage and a second-stage in our approach we need PIO heuristics for both stages. As in our two-stage approach discussed above, when we solve the first-stage to proven optimality we pass the set of racks used in the first-stage PIO heuristic solution to the second-stage for use in the second-stage PIO heuristic. In the interests of brevity, we shall just present a single POI heuristic and indicate how it is amended depending on whether we are considering the first-stage or the second-stage.

Our PIO heuristic for the order and rack allocation problem considers the variables associated with $\tau$ pickers at each iteration as integer variables and decides the values for these integer variables. Other variables are either fixed to the integer values that they were assigned at previous iterations, or are regarded as continuous variables.

Let $\Lambda$ be a set of pickers, where initially $\Lambda \leftarrow \emptyset$. Let $\tau$ be the number of pickers we consider at each iteration. Formally our PIO heuristic is:

(a) Add $\tau$ pickers currently not in $\Lambda$ to $\Lambda$.

(b) Amend the formulation (first-stage or second-stage formulation), so that although the order allocation variables ($x_{op}$) and the rack allocation variables ($y_{rp}$) associated with pickers $p \in \Lambda$ remain as binary variables, the order allocation and rack allocation variables associated with pickers $p \notin \Lambda$ become continuous variables lying between zero and one.

In terms of the first-stage formulation the rack usage variables ($u_r$) become continuous variables lying between zero and one. In terms of the second-stage formulation the order fulfilment variables ($v_o$) become continuous variables lying between zero and one. If we are considering Strategy 3 then the $z_{op}$ variables, indicating whether order $o$ is fully supplied by rack $r$, remain as binary variables.

(c) Solve the amended order and rack allocation formulation. In this solution we will have a (integer variable) feasible allocation of orders and racks for all pickers $p \in \Lambda$. Let the solution values for the order and rack allocation variables ($x_{op}$ and $y_{rp}$) be $X_{op}$ and $Y_{rp}$.
(d) If all of the binary variables that have been relaxed to continuous variables are naturally integer in the current amended formulation solution then stop. If this occurs then the solution which has been found is an integer feasible solution to the original formulation considered.

(e) If there are any pickers \( p \notin \Lambda \) for which the values \( X_{op} \forall o \) and \( Y_{rp} \forall r \) are naturally integer then add \( p \) to \( \Lambda \).

Amend the formulation further by adding constraints setting \( x_{op} = X_{op} \forall o \) and \( y_{rp} = Y_{rp} \forall r \in \Lambda \). If we are considering the first-stage formulation then the constraints to be added are \( x_{op} = X_{op} \forall o \in F; \forall p \in \Lambda \) and \( y_{rp} = Y_{rp} \forall r \in \Lambda \). If we are considering the second-stage formulation then the constraints to be added are \( x_{op} = X_{op} \forall o = 1, \ldots, O; \forall p \in \Lambda \) and \( y_{rp} = Y_{rp} \forall r \in \Theta; \forall p \in \Lambda \).

(f) If \( |\Lambda| \neq P \) set \( \tau \leftarrow \min[P - |\Lambda|, \tau] \) and go to step (a).

(g) If we are considering the first-stage then the rack usage variables \( (u_r) \) in the amended formulation now revert to binary variables and we resolve. We can reach this point in the heuristic if all of the \( x_{op} \) and \( y_{rp} \) variables are binary, but the values for \( u_r \) adopted (e.g. to satisfy Equation \( (\ref{pin}) \)) are fractional.

In this heuristic \( \Lambda \) is the set of pickers for which the order and rack allocation variables will be binary. Given a binary (integer feasible) solution for pickers \( p \in \Lambda \) we (at step (e) above) constrain all future solutions to retain these integer values. We then add a further \( \tau \) pickers to \( \Lambda \) (steps (f) and (a) above), and repeat the process. The heuristic terminates when all order and rack allocation variables are binary, whereupon we resolve (at step (g) above) any continuous \( u_r \) variables.

The PIO heuristic given above solves a succession of mixed-integer programs, successively fixing binary variables to either zero or one. As such there is no guarantee that it will always terminate with a feasible solution, i.e. it may be that this heuristic cannot find a feasible solution.

In the PIO heuristic presented above we have not specified any rule for choosing the \( \tau \) pickers to add to \( \Lambda \) at each iteration. In the computational results presented later below we choose the \( \tau \) pickers which had values for associated order and rack allocation variables closest to integer.

In more detail for each picker \( p \notin \Lambda \) compute a score \( S_p \) defined using \( S_p = \sum_o \min[X_{op}, 1-X_{op}] + \sum_r \min[Y_{rp}, 1-Y_{rp}] \). Here the summation over \( o \) and \( r \) differs depending upon the stage considered. The summation over \( o \) is over \( o \in F \) for the first-stage formulation and \( o = 1, \ldots, O \) for the second-stage formulation. The summation over \( r \) is over \( r = 1, \ldots, R \) for the first-stage formulation and \( r \in \Theta \) for the second-stage formulation.

Then if \( \tau = 1 \) we choose the single picker \( p \) with the minimum value of \( S_p \) (ties broken arbitrarily) to add to \( \Lambda \). In a similar fashion if \( \tau = 2 \) (for example) we choose the two pickers \( p, k \) (\( p \neq k \)) with the minimum value of \( S_p + S_k \) (ties broken arbitrarily) to add to \( \Lambda \). Initially, at step (a) when \( \Lambda \) is empty, we base the choice of \( \tau \) pickers on the solution to the linear programming (LP) relaxation of the formulation (so the values for \( X_{op} \) and \( Y_{rp} \) used in calculating \( S_p \) are the values in the LP relaxation solution).

6 Computational results

In this section we present results for the optimal and heuristic solution of our two-stage approach to order and rack allocation.

6.1 Test problem instances

As discussed above our two-stage approach (using Strategies 2 and 3) does not require all orders to be picked, rather some can be left unpicked. We would expect that if total picker capacity is greater than the total number of orders then it would be easier to fulfil all orders. Hence in the computational results presented below we only consider instances in which total picker capacity is equal to the total number of orders, i.e. \( \sum_{p=1}^{P} C_p = O \). The test instances examined, for \( O = 50, 100, 150, 200, 500 \), were generated in the manner described in our previous work Valle and Beasley [2021] and are publicly available at [https://www.dcc.ufmg.br/~arbex/mobileRacks.html](https://www.dcc.ufmg.br/~arbex/mobileRacks.html).

In the computational results presented below we used an Intel Core i7-7500U @ 2.70GHz with 8GB of RAM and Linux as the operating system. The code was written in C++ and Cplex 12.10 [CPLEX Optimizer [2019] was used as the mixed-integer solver. A maximum time limit of 5 CPU minutes (300 seconds) was imposed. This is the same time limit as in our previous work Valle and Beasley [2021]. As stated there the logic for imposing a 5 minute time limit was that we might reasonably expect that in a B2C environment decisions as to order and rack allocation have to be made relatively quickly.
6.2 Order and rack allocation: no rack reduction

Tables 1-5 show the results for $O = 50, 100, 150, 200, 500$ when we apply our varying strategies for order and rack allocation with no rack reduction. We also give here results for directly applying the approach given in our previous work (Valle and Beasley [2021]) to the same set of test problems.

In the computational results tables shown below, $T(s)$ denotes the total computation time in seconds and $UB$ is the best solution (upper bound) obtained at the end of the search, either when the instance was solved to proven optimality or when the time limit was reached. $GAP$ is defined as $100(UB - LB)/LB$ where $LB$ is the best lower bound obtained at the end of the search, either when the instance was solved to proven optimality or when the time limit was reached. If the time limit is reached then this is indicated by $TL$ in the tables below.

For some of the problems reported below no upper bound (feasible solution) was found before the problem terminated at time limit, and consequently for these problems no GAP value can be reported. Problems where this occurs are signified by $NF$ for No Feasible solution.

In the results for Strategies 1, 2 and 3 $|\Theta|$ is the number of racks chosen in the first-stage solution and $|S|$ is the number of orders that are considered for picking in the second-stage. $NPO$ is the number of Non-Picked Orders, so orders not picked at the second-stage.

To illustrate the results consider Table 1 and the results for $O = 50$ orders; $N = 500$ products; $R = 75$ racks; $P = 10$ pickers; each with a capacity $C_p$ of 5 orders (so the last line in the main body of Table 1). This table deals with the case where there is no rack reduction.

The approach given in our previous work (Valle and Beasley [2021]), when run on the same hardware with the same version of Cplex as used in this new work, requires 19.3 seconds, has a zero GAP (so solves the problem to proven optimality) with the optimal solution requiring 23 racks. This means that 23 racks was the minimum number of racks needed to deal with the total of $\sum_{p=1}^{P} C_p = 50$ orders allocated to the pickers.

Strategy 1, which requires all orders to be picked, also solves all problems to the same proven optimal solution, requiring 70.3 seconds. For Strategy 1 the second-stage is irrelevant as all orders are picked at the first-stage.

Strategy 2, which leaves orders requiring just a single unit of one product to the second-stage, also solves all problems to proven optimality. For the problem in the last line of the main body of Table 1 the first-stage required 17.5 seconds and the second-stage 0.3 seconds. 21 racks were needed, with the number of second-stage orders being 28. 7 of these 28 orders were left unpicked. This is because in the second-stage we restrict ourselves to only using the racks associated with the first-stage solution. So here we arrive at a solution involving fewer racks faster than either the approach of Valle and Beasley [2021] or Strategy 1, but with the solution leaving 7 orders unpicked.

Strategy 3 requires orders which can be supplied from a single rack to be picked at the first-stage. For the problem in the last line of the main body of Table 1 the first-stage required 0.1 seconds and the second-stage 0.4 seconds. 21 racks were needed, with the number of second-stage orders being 14. 3 of these 14 orders were left unpicked. As for Strategy 2 these orders are left unpicked because in the second-stage we restrict ourselves to only using the racks associated with the first-stage solution. So here we arrive at a solution faster than either the approach of Valle and Beasley [2021] or Strategy 1 or Strategy 2, but with the solution leaving 3 orders unpicked.

Considering Tables 1-5 then, as we would expect, as problem size (in terms of the number of orders) increases we have an increasing number of instances terminating due to reaching the time limit.

Both Valle and Beasley [2021] and Strategy 1 require all orders to be fulfilled in the first-stage. Referring to the averages seen at the foot of each of these tables then for $O \leq 150$ the approach of Valle and Beasley [2021] appears to perform better than Strategy 1, given the time limit of 300 seconds imposed. It requires the same (or less) time whilst returning a solution involving the same, or fewer, racks. However that performance advantage disappears for $O = 200$ and $O = 500$. For these problems (all of which went to time limit) we have that Strategy 1 required (on average) fewer racks, indeed significantly fewer racks when $O = 500$.

Strategy 2, which leaves orders requiring only a single unit of one product to the second-stage, appears to be increasingly effective as the number of orders increases. Indeed for $O = 500$ it (on average) gave solutions requiring far fewer racks than either Valle and Beasley [2021] or Strategy 1, and over all ten cases in Table 5 left no orders unpicked.

Strategy 3, which in the first-stage deals with the set of orders that can be satisfied (fully supplied) by a single rack, on average requires more racks than Strategy 2. However it is important to be clear here this result, requiring more racks, is an inherent feature of Strategy 3. Recall that Strategy 3 chooses racks such that all first-stage orders can be fully satisfied by a single rack. Clearly having sufficient racks such that all first-stage orders can be fully satisfied by just one of the chosen racks will be expected to require more racks than if we are allowed (as in Strategies 1, 2 and Valle and Beasley [2021]) to satisfy orders using products picked from a number of different racks.

With regard to the largest problem with $O = 500$ orders in Table 5 we see some instances with NF, for no feasible solution. Some of these will be due to the time limit being reached before any feasible solution is found.
However for $N = 300$ in that table and Strategy 3 the first-stage is solved to proven optimality very quickly (GAP is zero), yet no feasible solution is found. This Strategy 3 case is from an instance (both for $P = 5$ and $P = 10$) where there is genuinely no feasible solution. This is because in Strategy 3 we require all orders that are capable of being picked from a single rack be so picked in the first-stage. In this ($O = 500, N = 300$) problem, which was randomly generated, infeasibility occurs because there are two orders each of which can only be picked from a single rack. But for these two orders the single rack from which they can be picked is the same, and that rack does not have sufficient product inventory to allow both orders to be picked from it.

Given our two-stage approach and the variables and constraints as described above it is easy to think of conceptual solutions to this issue. For example introduce a pre-first-stage which for Strategy 3 first maximises the number of orders in $F$ (which must be picked in the first-stage) and any orders in $F$ that cannot be picked in that solution are left unpicked when we come to the first-stage. However, since only two of our fifty instances (without rack reduction, Strategy 3) led to a genuinely infeasible case, we have not implemented such a procedure in the results given here.

### Table 1: $O = 50$, no rack reduction

| $N$ | $R$ | $P$ | $C_p$ | Valle and Beasley [2021] | Strategy 1 | Strategy 2 | Strategy 3 |
|-----|-----|-----|-------|--------------------------|------------|------------|------------|
|     |     |     |       | T(s) GAP | T(s) GAP | 1st stage | 2nd stage | T(s) GAP | T(s) GAP | 1st stage | 2nd stage | T(s) GAP | T(s) GAP | 1st stage | 2nd stage |
| 100 | 5   | 10  | 64.4  | 7  | 8.2 | 0.1 | 7 | 34 | 5 | 0.3 | 0.0 | 8 | 0 | 0 |
|     | 10  | 5   | 1.3   | 10 | 3.8 | 2.0 | 0.2 | 10 | 34 | 0 | 0.3 | 0.1 | 10 | 0 | 0 |
| 200 | 5   | 10  | 2.8   | 12 | 2.7 | 0.3 | 0.1 | 11 | 30 | 5 | 0.1 | 0.1 | 13 | 8 | 0 |
|     | 10  | 5   | 95.8  | 15 | 108.1 | 33.9 | 3.6 | 15 | 30 | 7 | 0.6 | 0.9 | 13 | 8 | 4 |
| 300 | 5   | 10  | 0.7   | 16 | 1.1 | 0.1 | 0.1 | 15 | 31 | 6 | 0.1 | 0.1 | 17 | 9 | 1 |
|     | 10  | 5   | 45.4  | 18 | 134.2 | 40.2 | 0.6 | 17 | 31 | 9 | 0.2 | 1.3 | 17 | 9 | 2 |
| 400 | 5   | 10  | 0.5   | 19 | 0.0 | 0.0 | 0.0 | 19 | 34 | 0 | 0.1 | 0.0 | 19 | 10 | 0 |
|     | 10  | 5   | 2.5   | 20 | 7.4 | 3.5 | 1.1 | 19 | 34 | 5 | 0.1 | 1.7 | 19 | 10 | 2 |
| 500 | 5   | 10  | 0.5   | 21 | 0.1 | 0.1 | 0.1 | 20 | 28 | 9 | 0.0 | 0.1 | 21 | 14 | 1 |
|     | 10  | 5   | 19.3  | 23 | 70.3 | 17.5 | 0.3 | 21 | 28 | 7 | 0.1 | 0.4 | 21 | 14 | 3 |

Average: 23.3 0.0 16.0 40.6 0.0 16.0 25.6 0.0 15.4 31.4 5.3 0.2 0.0 15.8 31.4 5.3 0.2 0.0 0.5 0.0 15.8 6.2 1.3

### Table 2: $O = 100$, no rack reduction

| $N$ | $R$ | $P$ | $C_p$ | Valle and Beasley [2021] | Strategy 1 | Strategy 2 | Strategy 3 |
|-----|-----|-----|-------|--------------------------|------------|------------|------------|
|     |     |     |       | T(s) GAP | T(s) GAP | 1st stage | 2nd stage | T(s) GAP | T(s) GAP | 1st stage | 2nd stage | T(s) GAP | T(s) GAP | 1st stage | 2nd stage |
| 100 | 5   | 20  | TL 31.16 | 9 | TL 40.00 | 10 | 2.4 | 0.1 | 11 | 0 | 0 |
|     | 10  | 10  | TL 9.09  | 11 | TL 23.08 | 13 | 38.2 | 0.2 | 11 | 0 | 0 |
| 200 | 5   | 20  | TL 32.96 | 17 | TL 30.16 | 17 | 0.3 | 0.1 | 20 | 8 | 0 |
|     | 10  | 10  | TL 44.71 | 20 | TL 47.00 | 21 | 0.8 | 0.4 | 20 | 8 | 0 |
| 300 | 5   | 20  | TL 16.11 | 20 | TL 13.85 | 20 | 123.7 | 0.2 | 18 | 52 | 23 | 0.5 | 0.1 | 23 | 14 | 0 |
|     | 10  | 10  | TL 31.77 | 24 | TL 38.99 | 27 | 1.5 | 1.3 | 23 | 14 | 1 |
| 400 | 5   | 20  | TL 98.3  | 22 | TL 263.9 | 22 | 41.7 | 0.3 | 22 | 60 | 27 | 0.5 | 0.1 | 25 | 16 | 0 |
|     | 10  | 10  | TL 30.87 | 29 | TL 29.59 | 29 | 2.0 | 3.1 | 25 | 16 | 2 |
| 500 | 5   | 20  | TL 75.3  | 27 | TL 65.6 | 27 | 7.8 | 0.3 | 26 | 61 | 20 | 0.9 | 0.4 | 29 | 23 | 0 |
|     | 10  | 10  | TL 21.50 | 33 | TL 15.65 | 31 | 1.1 | 16.0 | 29 | 23 | 2 |

Average: 257.4 21.8 21.2 272.9 23.8 21.7 227.3 13.2 3.5 0.0 19.4 60.6 13.3 4.8 0.0 2.2 0.0 21.6 12.2 0.5

### 6.3 Order and rack allocation, rack reduction

Tables 6-10 give the results for the same problems as considered in Tables 1-5, but now with rack reduction. Recall from above that our rack reduction procedure reduces the number of racks that need to be considered and so we would expect a decrease in computation time as a result of this reduction.

Comparing the results for Valle and Beasley [2021] and Strategy 1 we can again conclude that as problem size increases Strategy 1 performs better than the approach of Valle and Beasley [2021]. Here for $O \geq 150$ the results from Strategy 1 are better (on average) than those from Valle and Beasley [2021]. For Strategies 2 and 3 we do see a decrease in computation time (considering both stages together).
Table 3: $O = 150$, no rack reduction

| $N$ | $R$ | $P$ | $C_p$ | Valle and Beasley [2021] | Strategy 1 | Strategy 2 | Strategy 3 |
|-----|-----|-----|-------|-------------------------|------------|------------|------------|
|     |     |     |       |                         | 1st stage  | 1st stage  | 1st stage  |
|     |     |     |       |                         | $T(s)$ GAP | $T(s)$ GAP | $T(s)$ GAP |
|     |     |     |       |                         | UB        | UB         | UB         |
|     |     |     |       |                         | $[\theta]$ | $[\theta]$ | $[\theta]$ |
| 100 | 100 | 5   | 30    | TL 37.22                | TL 37.13   | TL 35.60   | 31.0       |
|     |     |     |       |                         | 13        | 85         | 0          |
| 200 | 100 | 5   | 30    | TL 40.09                | TL 33.66   | TL 24.06   | 0.4        |
|     |     |     |       |                         | 20        | 93         | 0          |
| 300 | 100 | 5   | 30    | TL 26.46                | TL 25.44   | TL 12.90   | 0.7        |
|     |     |     |       |                         | 27        | 89         | 0          |
| 400 | 150 | 5   | 30    | TL 24.43                | TL 26.86   | TL 11.16   | 3.5        |
|     |     |     |       |                         | 28        | 97         | 0          |
| 500 | 150 | 5   | 30    | TL 25.29                | TL 26.02   | TL 9.74    | 12.7       |
|     |     |     |       |                         | 35        | 88         | 0          |

Average: 300.0 34.8 26.8 300.0 35.4 27.5 300.0 25.2 13.4 0.0 24.3 90.4 15.1 49.8 0.0 47.6 1.2 25.8 17.6 1.3

Table 4: $O = 200$, no rack reduction

| $N$ | $R$ | $P$ | $C_p$ | Valle and Beasley [2021] | Strategy 1 | Strategy 2 | Strategy 3 |
|-----|-----|-----|-------|-------------------------|------------|------------|------------|
|     |     |     |       |                         | 1st stage  | 1st stage  | 1st stage  |
|     |     |     |       |                         | $T(s)$ GAP | $T(s)$ GAP | $T(s)$ GAP |
|     |     |     |       |                         | UB        | UB         | UB         |
|     |     |     |       |                         | $[\theta]$ | $[\theta]$ | $[\theta]$ |
| 100 | 200 | 5   | 40    | TL 47.85                | TL 51.04   | TL 42.29   | TL 11.53   |
|     |     |     |       |                         | 14        | 16.1       | 0.4        |
| 200 | 200 | 5   | 40    | TL 49.08                | TL 42.63   | TL 38.83   | TL 75.67   |
|     |     |     |       |                         | 23        | 1.5        | 0.3        |
| 300 | 200 | 5   | 40    | TL 55.35                | TL 56.45   | TL 53.55   | TL 186.7   |
|     |     |     |       |                         | 26        | 16.9       | 0.3        |
| 400 | 200 | 5   | 40    | TL 55.35                | TL 42.63   | TL 34.55   | TL 32.3    |
|     |     |     |       |                         | 28        | 1.2        | 0.3        |
| 500 | 200 | 5   | 40    | TL 30.70                | TL 32.01   | TL 27.98   | TL 29.8    |
|     |     |     |       |                         | 40        | 3.0        | 0.4        |
|     |     |     |       |                         | 45        | 121        | 0          |

Average: 300.0 47.2 34.4 300.0 47.0 33.2 300.0 37.2 94.0 2.6 27.3 122.4 4.3 97.8 9.0 4.0 0.0 36.7 17.0 0.0

Table 5: $O = 500$, no rack reduction

| $N$ | $R$ | $P$ | $C_p$ | Valle and Beasley [2021] | Strategy 1 | Strategy 2 | Strategy 3 |
|-----|-----|-----|-------|-------------------------|------------|------------|------------|
|     |     |     |       |                         | 1st stage  | 1st stage  | 1st stage  |
|     |     |     |       |                         | $T(s)$ GAP | $T(s)$ GAP | $T(s)$ GAP |
|     |     |     |       |                         | UB        | UB         | UB         |
|     |     |     |       |                         | $[\theta]$ | $[\theta]$ | $[\theta]$ |
| 100 | 200 | 5   | 100   | TL 37.05                | TL 33.70   | TL 22.78   | TL 74.18   |
|     |     |     |       |                         | 23        | 5.5        | 4.5        |
| 200 | 200 | 5   | 100   | TL 44.19                | TL 43.37   | TL 35.86   | TL 38.99   |
|     |     |     |       |                         | 35        | 12.8       | 31.5       |
| 300 | 200 | 5   | 100   | TL 74.94                | TL 64.11   | TL 39.66   | TL 54.08   |
|     |     |     |       |                         | 93        | 9.2        | 58.6       |
| 400 | 200 | 5   | 100   | TL 63.85                | TL 55.66   | TL 39.07   | TL 45.77   |
|     |     |     |       |                         | 84        | 12.9       | 110.0      |
| 500 | 200 | 5   | 100   | TL 52.04                | TL 43.03   | TL 31.99   | TL 49.77   |
|     |     |     |       |                         | 77        | 11.6       | 110.0      |

Average: 300.0 56.0 58.7 300.0 49.7 50.2 300.0 39.6 60.4 0.0 43.9 292.4 0.0 110.5 10.9 14.3 0.0 68.3 39.8 0.0
### Table 6: $O = 50$, rack reduction

| $N$ | $R$ | $P$ | $C_P$ | Strategy 1 | Strategy 2 | Strategy 3 |
|-----|-----|-----|-------|------------|------------|------------|
|     |     |     |       | 1st stage | 2nd stage | 1st stage | 2nd stage | 1st stage | 2nd stage |
|     |     |     | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ |
| T(s) | GAP | UB | T(s) | GAP | | T(s) | GAP | T(s) | GAP | T(s) | GAP |
| 100 | 50  | 5   | 0.1  | 8   | 0.0  | 8   | 0.0  | 0.1  | 8   | 34  | 0   | 0.1  | 0.0  | 9   | 0   |
| 5   | 0.4 | 11  | 0.5  | 11  | 0.1  | 0.3  | 11  | 34  | 0   | 0.3  | 0.1  | 10  | 0   | 0   |
| 7   | 5   | 3   | 14.1 | 16  | 1.5  | 5.0  | 16  | 30  | 0   | 0.1  | 1.5  | 14  | 8   | 2   |
| 300 | 50  | 5   | 0.2  | 18  | 0.1  | 19  | 0.0  | 0.1  | 19  | 31  | 0   | 0.0  | 0.0  | 19  | 9   |
| 10  | 5   | 1.3 | 20   | 30  | 0.1  | 10   | 19  | 31  | 2   | 0.5  | 1.4  | 19  | 9   | 1   |

Average: 1.1 0.0 17.7 3.0 0.0 17.7 0.3 0.0 0.8 0.0 16.9 31.4 2.2 0.1 0.0 0.4 0.0 17.1 8.2 0.6

### Table 7: $O = 100$, rack reduction

| $N$ | $R$ | $P$ | $C_P$ | Strategy 1 | Strategy 2 | Strategy 3 |
|-----|-----|-----|-------|------------|------------|------------|
|     |     |     |       | 1st stage | 2nd stage | 1st stage | 2nd stage | 1st stage | 2nd stage |
|     |     |     | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ |
| T(s) | GAP | UB | T(s) | GAP | | T(s) | GAP | T(s) | GAP | T(s) | GAP |
| 100 | 100 | 5   | 0.4  | 9   | 0.2  | 9   | 0.0  | 0.3  | 9   | 71  | 0   | 0.4  | 0.1  | 13  | 0   |
| 10  | 15  | 5   | 1.6  | 12  | 8.9  | 12  | 0.8  | 9.0  | 12  | 71  | 0   | 1.6  | 0.3  | 13  | 0   |
| 200 | 100 | 5   | 3.7  | 22  | 1.9  | 22  | 0.1  | 0.3  | 22  | 52  | 0   | 0.1  | 0.1  | 26  | 14  |
| 10  | 20  | 5   | 206.4| 24  | 241.3| 24  | 259.7| 12.8 | 23  | 52  | 5   | 0.4  | 5.2  | 26  | 14  |
| 300 | 100 | 5   | 0.7  | 25  | 0.2  | 25  | 0.1  | 0.2  | 25  | 60  | 0   | 0.1  | 0.1  | 26  | 14  |
| 10  | 15  | 5   | 13.64| 22  | 13.64| 22  | 108.3| 79.3 | 13  | 52  | 4   | 3.1  | 1.6  | 29  | 16  |
| 400 | 100 | 5   | 0.4  | 29  | 0.3  | 29  | 0.1  | 0.3  | 29  | 61  | 0   | 0.1  | 0.1  | 31  | 23  |
| 10  | 15  | 5   | 8.99 | 32  | 6.25 | 32  | 2.3  | 11.8 | 29  | 61  | 2   | 0.4  | 5.1  | 31  | 23  |

Average: 104.4 2.9 21.6 100.8 2.0 21.5 56.3 0.4 7.5 0.0 20.9 60.6 2.7 0.4 0.0 1.2 0.0 24.4 12.2 0.2

### Table 8: $O = 150$, rack reduction

| $N$ | $R$ | $P$ | $C_P$ | Strategy 1 | Strategy 2 | Strategy 3 |
|-----|-----|-----|-------|------------|------------|------------|
|     |     |     |       | 1st stage | 2nd stage | 1st stage | 2nd stage | 1st stage | 2nd stage |
|     |     |     | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ | $\Theta$ |
| T(s) | GAP | UB | T(s) | GAP | | T(s) | GAP | T(s) | GAP | T(s) | GAP |
| 100 | 150 | 5   | 5.1  | 29  | 1.0  | 29  | 0.1  | 0.3  | 29  | 97  | 0   | 0.3  | 0.2  | 33  | 23  |
| 10  | 15  | 5   | 96.73| 33  | 28.4 | 25.8 | 35  | 88  | 22  | 0.5  | 108.6| 37  | 30  | 1   |

Average: 146.6 6.4 26.4 144.1 13.6 26.3 104.9 0.7 48.0 2.2 24.3 90.4 8.3 0.8 0.0 13.0 0.0 29.8 17.6 0.1
Table 9: \( O = 200 \), rack reduction

| \( N \) | \( R \) | \( P \) | \( C_P \) | Valle and Beasley [2021] | Strategy 1 | Strategy 2 | Strategy 3 |
|-------|------|------|------|----------------|----------|----------|----------|
|       |      |      |      | T(s) GAP UB | T(s) GAP | 1st stage | 1st stage |
|       |      |      |      |              | [\( \Theta \)] |          | [\( \Theta \)] |
| 100   | 200  | 5    | 40   | 84.3 – 12   | 4.7 – 12 | 0.3 – 1.4 – 12 | 121 | 0 |
|       |      | 10   | 20   | TL 15.08 15 | TL 18.19 16 | 16 | 121 | 0 |
| 200   | 200  | 5    | 40   | 109.4 – 20  | 103.3 – 20 | 0.5 – 3.5 – 20 | 119 | 0 |
|       |      | 10   | 20   | TL 34.48 29 | TL 29.52 27 | 24 | 119 | 1 |
| 300   | 200  | 5    | 40   | 48.6 – 26   | 8.7 – 26 | 0.2 – 1.7 – 26 | 123 | 21 |
|       |      | 10   | 20   | TL 20.59 34 | TL 18.18 33 | 31 | 123 | 21 |
| 400   | 200  | 5    | 40   | 4.0 – 31    | 2.9 – 31 | 0.2 – 1.2 – 31 | 128 | 0 |
|       |      | 10   | 20   | TL 15.22 38 | TL 5.71 35 | 4.69 | 128 | 6 |
| 500   | 200  | 5    | 40   | 12.1 – 42   | 3.4 – 42 | 0.2 – 0.6 – 42 | 121 | 0 |
|       |      | 10   | 20   | TL 17.67 49 | TL 8.89 45 | 14.66 | 121 | 20 |

Average: 175.9 10.3 29.6 162.3 8.0 28.7 100.6 4.3 122.8 3.7 27.3 122.4 4.8 2.1 0.0 1.7 0.0 37.5 17.0 0.0

Table 10: \( O = 500 \), rack reduction

| \( N \) | \( R \) | \( P \) | \( C_P \) | Valle and Beasley [2021] | Strategy 1 | Strategy 2 | Strategy 3 |
|-------|------|------|------|----------------|----------|----------|----------|
|       |      |      |      | T(s) GAP UB | T(s) GAP | 1st stage | 1st stage |
|       |      |      |      |              | [\( \Theta \)] |          | [\( \Theta \)] |
| 100   | 200  | 5    | 40   | 187.3 – 21  | 5.9 – 21 | 0.3 – 6.0 – 21 | 303 | 0 |
|       |      | 10   | 50   | TL NF       | TL 9.44 22 | 279.6 – 28.7 | 21 | 303 | 0 |
| 200   | 200  | 5    | 40   | 22.56 36   | 247.3 – 29 | 8.8 – 127.2 – 29 | 285 | 0 |
|       |      | 10   | 50   | TL NF       | TL 32.16 40 | 27.30 79.4 – 38 | 285 | 0 |
| 300   | 200  | 5    | 40   | 16.86 44   | 29.6 – 38 | 1.4 – 10.8 – 38 | 292 | 0 |
|       |      | 10   | 50   | TL NF       | TL 34.43 54 | 15.40 18.49 | 292 | 0 |
| 400   | 200  | 5    | 40   | 10.07 56   | 45.4 – 51 | 2.4 – 17.8 – 51 | 275 | 0 |
|       |      | 10   | 50   | TL NF       | TL 18.05 63 | 18.05 133.6 – 56 | 275 | 0 |
| 500   | 200  | 5    | 40   | 9.96 58    | 244.5 – 53 | 2.4 – 12.4 – 52 | 307 | 0 |
|       |      | 10   | 50   | TL NF       | TL 18.46 65 | 5.36 10.75 | 307 | 0 |

Average: 288.7 11.9 43.0 207.3 10.5 41.4 148.6 6.6 101.6 2.9 49.4 292.4 8.7 38.0 0.0 13.9 0.0 65.9 39.8 0.0

Table 9: \( O = 200 \), rack reduction

Table 10: \( O = 500 \), rack reduction
6.4 Comparison: order and rack allocation

Table 11 combines the average row from each of the order and rack allocation results tables (Tables 10) given above into a single table for comparison purposes. The first set of rows in this table refer to results without rack reduction, the second set of rows refer to results with rack reduction. The average row gives the average value over all values of $O$. These are the averages over five different values for $O$, with ten cases being considered for each value of $O$. Considering Table 11 it seems reasonable to conclude that:

- For both Valle and Beasley [2021] and Strategy 1, where we fulfil all orders at the first-stage, rack reduction has a very positive effect resulting (on average) in finding solutions requiring less racks quicker.

- For Strategy 2 and Strategy 3 where we fulfil some orders at the first-stage, some at the second-stage, we do not see such a marked improvement in performance in terms of the number of racks and number of non-picked orders as a result of rack reduction. However computation time (taking both stages into account) is (approximately) reduced by a factor of two for Strategy 2 and by a factor of five for Strategy 3.

- Comparing Valle and Beasley [2021] and Strategy 1 with Strategy 2 is complicated by the fact that we deliberately allow for non-picked orders in Strategy 2. Here, when we have rack reduction, we have that Strategy 2 gives solutions requiring on average 26.0 racks, with 5.3 orders left unpicked. This is fewer racks than both Valle and Beasley [2021] and Strategy 1 (27.7 and 27.1 respectively) and we would note here that any non-picked (second-stage) orders in Strategy 2 will be orders that only require a single unit of one product, so might easily be satisfied in a future order and rack allocation as orders come in over time.

- As discussed above Strategy 3 does require more racks than the other approaches. However it is by far the fastest of the approaches with a very low number of non-picked orders. Moreover, when rack reduction is used all instances are solved to proven optimality, with for the larger problems ($O = 200, 500$) no unpicked orders.

We would stress here that the rack reduction procedure we have presented above could easily be incorporated into any other solution approach (heuristic or optimal) for order and rack allocation.

| Case | $O$ | Valle and Beasley [2021] | Strategy 1 | Strategy 2 | Strategy 3 |
|------|-----|-------------------------|------------|------------|------------|
|      | T(s) | GAP | UB | 1st stage T(s) | GAP | UB | 1st stage T(s) | GAP | $|\Theta|_1$ | | 2nd stage T(s) | GAP | $|\Theta|_2$ | | $|\Theta|_3$ | NPO | $|\Theta|_4$ | $|\Theta|_5$ | NPO |
| Rack reduction | 50 | 23.3 | 0.0 | 16.8 | 40.6 | 0.0 | 16.0 | | 10.8 | 0.0 | 0.6 | 0.0 | 15.4 | 31.4 | 5.3 | 0.2 | 0.0 | 0.5 | 0.0 | 15.8 | 8.2 | 1.3 |
| | 100 | 257.4 | 21.8 | 21.2 | 272.9 | 23.8 | 21.7 | | 227.3 | 13.2 | 3.5 | 0.0 | 19.4 | 60.6 | 13.3 | 4.8 | 0.0 | 2.2 | 0.0 | 21.6 | 122.2 | 0.5 |
| | 150 | 300.0 | 34.8 | 28.8 | 300.0 | 35.4 | 27.5 | | 300.0 | 25.2 | 13.4 | 0.0 | 24.3 | 90.4 | 15.1 | 49.8 | 0.0 | 47.6 | 1.2 | 25.8 | 17.6 | 1.3 |
| | 200 | 300.0 | 47.2 | 34.4 | 300.0 | 47.0 | 33.2 | | 300.0 | 37.2 | 94.0 | 2.6 | 27.3 | 122.4 | 4.3 | 97.8 | 9.0 | 4.0 | 0.0 | 36.7 | 17.0 | 0.0 |
| | 500 | 300.0 | 56.7 | 58.7 | 300.0 | 49.7 | 50.2 | | 300.0 | 30.0 | 60.4 | 0.0 | 43.9 | 292.4 | 0.0 | 110.5 | 10.9 | 14.3 | 0.0 | 68.3 | 39.8 | 0.0 |
| Average | 236.1 | 32.0 | 31.4 | 242.7 | 31.2 | 29.7 | | 227.6 | 23.0 | 34.4 | 0.5 | 26.1 | 119.4 | 7.6 | 52.6 | 4.0 | 13.7 | 0.0 | 33.6 | 19.0 | 0.0 |

Table 11: Order and rack allocation: comparison of average results

6.5 PIO

Table 12 gives the results for our PIO heuristic for the same problems as considered in Table 1. To illustrate the results consider the last line in the main body of Table 12. For Strategy 1 and $\tau = 1$ we solve the problem in 0.9 seconds requiring 26 racks, with $\tau = 2$ we solve the problem in 0.8 seconds requiring 25 racks. For Strategy 1 the second-stage is irrelevant as all orders are picked at the first-stage.

For Strategy 2 $|\Theta|_1$ is 28 (both for $\tau = 1$ and $\tau = 2$). For $\tau = 1$ the first-stage requires 0.5 seconds and the second stage 0.4 seconds (both problems being solved to proven optimality). We automatically know that these problems have been solved to proven optimality because any first-stage or second-stage result with an associated computation time less then the time limit must have been solved to proven optimality. 23 racks are required with 4 orders being left unpicked. For $\tau = 2$ the first-stage requires 0.4 seconds and the second stage 0.5 seconds (both problems being solved to proven optimality); 22 racks are required with 2 orders being left unpicked.
For Strategy 3 $|S|$ is 14 (both for $\tau = 1$ and $\tau = 2$). For $\tau = 1$ the first-stage requires 0.3 seconds and the second stage 0.6 seconds (both problems being solved to proven optimality); 21 racks are required with 3 orders being left unpicked. For $\tau = 2$ the first-stage requires 0.3 seconds and the second stage 0.6 seconds (both problems being solved to proven optimality); 21 racks are required with 3 orders being left unpicked.

Tables 13 and 16 show the results for our PIO heuristic with no rack reduction for $O = 100, 150, 200, 500$ respectively. Tables 17 and 21 show the results for our PIO heuristic with rack reduction.

| N  | R  | P     | $C_p$ | $t_0$ | $t_1$ | $t_2$ |
|----|----|-------|-------|-------|-------|-------|
| 100| 5  | 10    | 8     | 3.4   | 7     | 0     |
| 200| 5  | 10    | 0.6   | 13    | 5.2   | 13    |
| 300| 5  | 10    | 0.4   | 17    | 2.6   | 17    |
| 400| 5  | 10    | 0.2   | 20    | 0.2   | 19    |
| 500| 5  | 10    | 0.3   | 21    | 0.6   | 21    |

| N  | R  | P     | $C_p$ | $t_0$ | $t_1$ | $t_2$ |
|----|----|-------|-------|-------|-------|-------|
| 100| 5  | 10    | 1.0   | 3.4   | 7     | 0     |
| 200| 5  | 10    | 0.7   | 12    | 0.5   | 13    |
| 300| 5  | 10    | 1.1   | 17    | 2.2   | 15    |
| 400| 5  | 10    | 0.4   | 17    | 2.6   | 17    |
| 500| 5  | 10    | 0.2   | 20    | 0.2   | 19    |

Table 12: $O = 50$, PIO, no rack reduction

Table 13: $O = 100$, PIO, no rack reduction

Table 14: $O = 150$, PIO, no rack reduction
| N | R | P | C_p | T(s) | NPO |
|---|---|---|-----|-----|-----|
| 100 | 200 | 5 | 40 | TL 14 | TL 13 |
| 10 | 20 | 144.9 | 17 | TL 16 |
| 200 | 200 | 5 | 40 | TL 21 | TL 21 |
| 10 | 20 | TL 24 | TL 27 |
| 300 | 200 | 5 | 40 | TL 26 | TL 29 |
| 10 | 20 | TL 31 | TL 32 |
| 400 | 200 | 5 | 40 | TL 31 | TL 33 |
| 10 | 20 | TL 35 | TL 37 |
| 500 | 200 | 5 | 40 | TL 39 | TL 42 |
| 10 | 20 | TL 45 | TL 46 |

Average: 284.5 28.3 300.0 29.6 122 214.1 125.1 25.8 11.8 277.1 125.3 25.6 15.0 1 17 57.0 2.3 32.7 0.0 65.0 4.3 32.9 0.0

| N | R | P | C_p | T(s) | NPO |
|---|---|---|-----|-----|-----|
| 100 | 200 | 5 | 100 | TL 20 | TL 22 |
| 10 | 50 | TL 22 | TL 25 |
| 200 | 200 | 5 | 100 | TL 35 | TL 35 |
| 10 | 50 | TL 39 | TL 78 |
| 300 | 200 | 5 | 100 | TL 41 | TL 43 |
| 10 | 50 | TL 47 | TL 57 |
| 400 | 200 | 5 | 100 | TL 54 | TL 53 |
| 10 | 50 | TL 65 | TL 68 |
| 500 | 200 | 5 | 100 | TL 59 | TL 61 |
| 10 | 50 | TL 67 | TL 68 |

Average: 300.0 44.9 300.0 51.0 292 294.0 184.5 38.3 65.7 300.0 107.4 41.1 1.1 32 68.8 20.7 67.2 0.0 71.6 23.0 67.2 0.0

| N | R | P | C_p | T(s) | NPO |
|---|---|---|-----|-----|-----|
| 100 | 50 | 5 | 10 | 0.1 NF | 0.1 NF |
| 10 | 5 | 0.3 NF | 0.3 NF |
| 200 | 50 | 5 | 10 | 0.1 NF | 0.1 NF |
| 10 | 5 | 0.4 NF | 0.4 NF |
| 300 | 50 | 5 | 10 | 0.2 NF | 0.2 NF |
| 10 | 5 | 0.6 NF | 0.6 NF |
| 400 | 75 | 5 | 10 | 0.2 NF | 0.2 NF |
| 10 | 5 | 0.7 NF | 0.7 NF |
| 500 | 75 | 5 | 10 | 0.2 NF | 0.2 NF |
| 10 | 5 | 0.7 NF | 0.7 NF |

Average: 0.4 22.4 0.3 21.6 31 0.2 0.1 18.7 0.2 0.2 19.4 0.2 8 0.3 0.6 17.3 0.6 0.2 0.5 17.2 0.6

### Table 15: O = 200, PIO, no rack reduction

### Table 16: O = 500, PIO, no rack reduction

### Table 17: O = 50, PIO, rack reduction

22
Table 18: $O = 100$, PIO, rack reduction

| $N$ | $R$ | $P$ | $C_p$ | $1st$ T(s) | $2nd$ T(s) | $\Theta$ | $\tau_T$ | $\tau_{0}$ | NPO | $1st$ T(s) | $2nd$ T(s) | $\Theta$ | $\tau_T$ | $\tau_{0}$ | NPO |
|-----|-----|-----|-------|-----------|-----------|-------|-------|-------|-----|-----------|-----------|-------|-------|-------|-----|
| 100 | 100 | 5   | 20    | 0.2 NF   | 0.2 NF    |       |       |       |     | 0.1      | 0.1      |       |       |       |     |
|     |     | 10  | 1.1 NF | 1.1      | 1.0       |       |       |       |     | 0.7      | 1.2      |       |       |       |     |
| 200 | 100 | 10  | 2.5   | 21       | 4.7       | 22    |       |       |     | 59       | 1.4      | 3.0   | 16    | 8    |     |
|     |     | 10  | 2.0   | 29       | 2.1       | 28    |       |       |     | 52       | 1.4      | 2.6   | 22    | 14   |     |
| 300 | 100 | 10  | 2.0   | 31       | 1.5       | 30    |       |       |     | 60       | 1.1      | 1.2   | 28    | 16   |     |
|     |     | 10  | 2.1   | 34       | 1.5       | 30    |       |       |     | 60       | 1.1      | 1.2   | 28    | 16   |     |
| 400 | 100 | 10  | 2.0   | 6.0     | 0.6      | 0.7   | 25    |       |     | 60       | 0.3      | 0.2   | 25    | 16   |     |
|     |     | 10  | 2.1   | 34       | 1.5       | 30    |       |       |     | 60       | 1.1      | 1.2   | 28    | 16   |     |
| 500 | 100 | 10  | 2.0   | 30       | 0.6     | 0.3   | 29    |       |     | 61       | 0.3      | 0.3   | 30    | 23   |     |
|     |     | 10  | 2.2   | 16       | 1.6      | 36    |       |       |     | 61       | 0.3      | 0.3   | 30    | 23   |     |

Average: 1.3 27.5 1.4 24.9 61 0.7 23.6 22.3 0.1 0.6 6.6 22.1 0.1 12 1.0 1.5 24.6 0.2 0.7 15.4 24.5 0.2

Table 19: $O = 150$, PIO, rack reduction

| $N$ | $R$ | $P$ | $C_p$ | $1st$ T(s) | $2nd$ T(s) | $\Theta$ | $\tau_T$ | $\tau_{0}$ | NPO | $1st$ T(s) | $2nd$ T(s) | $\Theta$ | $\tau_T$ | $\tau_{0}$ | NPO |
|-----|-----|-----|-------|-----------|-----------|-------|-------|-------|-----|-----------|-----------|-------|-------|-------|-----|
| 100 | 100 | 5   | 30    | 0.6 15   | 0.9      | 15    |       |       |     | 85       | 0.4      | 0.6   | 15    | 5     |     |
|     |     | 10  | 2.4   | 17       | 2.9      | 17    |       |       |     | 85       | 1.6      | 2.5   | 15    | 5     |     |
| 200 | 100 | 5   | 30    | 0.8 21   | 1.1      | 21    |       |       |     | 93       | 0.5      | 0.5   | 21    | 5     |     |
|     |     | 10  | 3.0   | 24       | 4.7      | 23    |       |       |     | 93       | 1.6      | 1.5   | 21    | 5     |     |
| 300 | 100 | 5   | 30    | 1.7 30   | 17.6     | 27    |       |       |     | 89       | 1.0      | 0.9   | 27    | 30    |     |
|     |     | 10  | 5.0   | 33       | 44.6     | 33    |       |       |     | 89       | 5.5      | 16.8  | 30    | 30    |     |
| 400 | 150 | 5   | 30    | 1.3 31   | 1.9      | 31    |       |       |     | 97       | 0.7      | 0.2   | 31    | 23    |     |
|     |     | 10  | 4.2   | 30       | 7.0      | 35    |       |       |     | 97       | 2.3      | 21.5  | 30    | 23    |     |
| 500 | 150 | 5   | 30    | 1.2 37   | 1.6      | 37    |       |       |     | 88       | 0.6      | 0.2   | 34    | 30    |     |
|     |     | 10  | 4.7   | 44       | 4.4      | 44    |       |       |     | 88       | 2.7      | 3.2   | 44    | 30    |     |

Average: 2.5 28.8 8.7 27.9 90 1.5 68.2 6.3 0.3 6.9 15.0 25.9 0.0 18 1.8 8.4 29.9 0.1 1.2 16.6 30.1 0.1

Table 20: $O = 200$, PIO, rack reduction

| $N$ | $R$ | $P$ | $C_p$ | $1st$ T(s) | $2nd$ T(s) | $\Theta$ | $\tau_T$ | $\tau_{0}$ | NPO | $1st$ T(s) | $2nd$ T(s) | $\Theta$ | $\tau_T$ | $\tau_{0}$ | NPO |
|-----|-----|-----|-------|-----------|-----------|-------|-------|-------|-----|-----------|-----------|-------|-------|-------|-----|
| 100 | 200 | 5   | 40    | 1.1 14   | 5.8      | 14    |       |       |     | 121      | 1.0      | 0.4   | 14    | 0     |     |
|     |     | 10  | 2.0   | 21       | 7.0      | 18    |       |       |     | 121      | 1.4      | 2.7   | 16    | 0     |     |
| 200 | 200 | 5   | 40    | 3.7 22   | 30.3     | 22    |       |       |     | 119      | 1.1      | 1.7   | 21    | 0     |     |
|     |     | 10  | 2.7   | 28       | 27.8     | 26    |       |       |     | 119      | 4.8      | 156.7 | 25    | 0     |     |
| 300 | 200 | 5   | 40    | 3.3 28   | 9.6      | 26    |       |       |     | 123      | 0.9      | 0.9   | 27    | 0     |     |
|     |     | 10  | 12.5  | 34       | 55.1     | 32    |       |       |     | 123      | 1.2      | 13.6  | 33    | 0     |     |
| 400 | 200 | 5   | 40    | 2.9 35   | 18.3     | 32    |       |       |     | 128      | 1.1      | 1.6   | 32    | 0     |     |
|     |     | 10  | 7.8   | 39       | 43.3     | 41    |       |       |     | 128      | 5.6      | TL 35 | 2     | 0     |     |
| 500 | 200 | 5   | 40    | 2.8 45   | 9.1      | 42    |       |       |     | 121      | 1.1      | 0.6   | 43    | 0     |     |
|     |     | 10  | 12.6  | 48       | 144.7    | 48    |       |       |     | 121      | 7.3      | 114.0 | 44    | 0     |     |

Average: 6.0 31.3 35.1 30.1 122 3.3 59.2 29.1 0.2 26.5 36.9 28.5 0.2 17 3.8 1.7 37.8 0.0 2.7 1.8 37.8 0.0

23
6.6 Comparison: PIO

Table 22 combines the average row from each of the PIO heuristic results tables (Tables 12-21) given above into a single table for comparison purposes. The first set of rows in this table refer to results without rack reduction, the second set of rows refer to results with rack reduction. Table 22 also shows (under $\tau = 1$ for ease of presentation) appropriate summary figures taken directly from Table 11.

Considering just the PIO results in Table 22 it seems reasonable to conclude that:

- For Strategy 1, although rack reduction results in a reduction in computation time the number of racks used increases.

- For Strategy 2 with no rack reduction although moving from $\tau = 1$ to $\tau = 2$ increases the computation time (taking both strategies together) and (slightly) the number of racks used there is a significant decrease in the number of non-picked orders.

- Rack reduction for Strategy 2 seems (on average) to be beneficial in terms of the number of non-picked orders when $\tau = 1$.

- Based on the problem instances examined if we choose to use Strategy 3 then Strategy 3 with rack reduction and $\tau = 1$ seems to be the best option to choose out of those examined.

- Based on the problem instances examined if we choose to use Strategy 3 then Strategy 3 with rack reduction and $\tau = 1$ seems to be the best option to choose out of those examined.

Comparing the results in Table 22 for our PIO heuristic with the results taken from Table 11 as reproduced in Table 22 we would conclude:

- With no rack reduction:
  - our PIO heuristic when used with Strategy 1 ($\tau = 1$ or $\tau = 2$) performs better (both in terms of computation time and in terms of the number of racks used) than Strategy 1 on its own
  - our PIO heuristic when used with Strategy 2 ($\tau = 2$) performs better (both in terms of computation time and in terms of the number of racks used and the number of non-picked orders) than Strategy 2 on its own
  - our PIO heuristic when used with Strategy 3 ($\tau = 1$ or $\tau = 2$) performs better (both in terms of computation time and in terms of the number of racks used and the number of non-picked orders) than Strategy 3 on its own

- With rack reduction:
– our PIO heuristic when used with Strategy 1 is faster, but requires more racks, than Strategy 1 on its own
– our PIO heuristic when used with Strategy 2 ($\tau = 1$) is faster and has fewer non-picked orders, but requires more racks, than Strategy 2 on its own
– our PIO heuristic when used with Strategy 3 ($\tau = 1$ or $\tau = 2$) performs better (both in terms of computation time and in terms of the number of racks used and the number of non-picked orders) than Strategy 3 on its own

| Case O | Strategy 1 | Strategy 2 | Strategy 3 |
|--------|------------|------------|------------|
|        | $\tau = 1$ | $\tau = 2$ | $\tau = 1$ | $\tau = 2$ |
|        | 1st | 2nd | NPO | 1st | 2nd | NPO | 1st | 2nd | NPO |
| No rack reduction 50 | 0.7 17.7 | 1.7 17.3 | 31 | 0.3 0.9 16.4 | 1.4 | 0.5 0.7 16.2 | 1.8 | 0.6 0.8 16.1 | 1.3 | 0.2 0.6 15.9 | 1.3 |
| 100 | 9.2 22.1 | 189.3 21.8 | 61 | 6.0 5.6 20.8 | 2.8 | 65.0 6.1 20.3 | 3.3 | 12.1 5.4 22.1 | 0.4 | 1.3 4.6 21.9 | 0.7 |
| 150 | 77.3 25.8 | 288.2 25.8 | 90 | 66.8 36.8 24.1 | 5.0 | 165.3 54.9 23.7 | 10.2 | 18.3 41.2 26.2 | 0.9 | 5.6 40.6 26.3 | 0.7 |
| 200 | 284.5 28.3 | 300.0 29.6 | 122 | 214.1 125.1 25.8 | 11.8 | 277.1 125.3 25.6 | 15.0 | 17.5 23.2 27.0 | 0.0 | 65.0 4.3 32.9 | 0.0 |
| 500 | 300.0 44.9 | 300.0 51.0 | 292 | 294.0 184.5 38.3 | 65.7 | 300.0 107.4 41.1 | 1.1 | 32.3 20.7 67.2 | 0.0 | 71.6 23.0 67.2 | 0.0 |
| Average | 134.3 27.8 | 215.8 29.1 | 116.2 70.6 25.1 | 17.3 | 161.6 58.9 25.4 | 6.3 | 26.4 13.5 32.9 | 0.5 | 28.7 14.6 32.8 | 0.5 |

| Rack reduction 50 | 0.4 22.4 | 0.3 21.6 | 31 | 0.2 0.1 18.7 | 0.2 | 0.2 0.1 19.4 | 0.2 | 8.3 0.6 17.3 | 0.6 | 0.2 0.5 17.2 | 0.6 |
| 100 | 1.3 27.5 | 1.4 24.9 | 61 | 0.7 23.6 22.3 | 0.2 | 0.6 66.2 22.1 | 0.1 | 12.1 1.5 24.5 | 0.2 | 0.7 1.5 24.5 | 0.2 |
| 150 | 2.5 28.8 | 8.7 27.9 | 90 | 1.5 6.8 26.3 | 0.3 | 6.9 15.9 25.9 | 0.0 | 18.4 8.4 29.9 | 0.1 | 1.2 16.6 30.1 | 0.1 |
| 200 | 6.0 31.3 | 35.1 30.1 | 122 | 3.3 59.2 29.1 | 0.2 | 26.5 36.9 28.5 | 0.2 | 17.5 3.8 37.8 | 0.0 | 2.7 1.8 37.8 | 0.0 |
| 500 | 96.2 44.7 | 238.1 44.4 | 292 | 59.3 84.8 41.4 | 1.7 | 125.7 113.2 40.4 | 35.0 | 32.3 10.4 61.2 | 0.0 | 9.7 12.3 61.8 | 0.0 |
| Average | 21.3 30.9 | 56.7 29.8 | 13.0 34.9 27.6 | 0.5 | 32.0 34.4 27.3 | 7.1 | 3.5 5.7 34.2 | 0.2 | 2.9 6.5 34.3 | 0.2 |

Table 22: PIO: comparison of average results

6.7 Unpicked orders

Our two-stage approach, using either Strategy 2 or Strategy 3, explicitly allows orders $o \in S$ that are dealt with in the second-stage to be unfulfilled. The logic behind this was that in practice in many RMFS environments new orders arrive frequently, and any unpicked orders can be left to be dealt with together with future (yet to appear) orders.

However, based on the instances examined, it is clear that it may be possible to keep the number of unpicked orders to a very low value. For example, referring to Table 22 over the 50 cases considered from $O = 50$ to $O = 500$, ten cases for each value of $O$, the average number of unpicked orders for Strategy 2 with rack reduction and $\tau = 1$ is only 0.5, whilst for Strategy 3 with/without rack reduction the average number of unpicked orders (over the 50 cases) never exceeds 0.6.

Whilst, clearly, the characteristics of customer orders will be different in different practical situations these results indicate that the work presented here has the potential to allow the user to control (via the two-stage approach) the orders which must be fulfilled, whilst only having a low number of unfilled (non-picked) orders.

7 Conclusions

In this paper we have considered the problem of allocating orders and mobile storage racks to pickers in a robotic mobile fulfilment system.

We presented a two-stage formulation of the problem. In this two-stage approach we, in the first-stage, deal with the orders which must be definitely fulfilled (picked), where the racks chosen to fulfil these first-stage orders were chosen so as to (collectively) contain sufficient product to satisfy all orders. In the second-stage we restricted attention to those racks chosen in the first-stage solution in terms of allocating second-stage orders.

We presented three different strategies for first-stage order selection. Strategy 1 only used the first-stage (so no second-stage) and allocated all orders to pickers. Strategy 2 left the set of orders comprising just a single unit of one product to the second-stage, all other orders were considered in the first-stage. Strategy 3 considered the set of orders which only require one rack as first-stage orders, all other orders were considered in the second-stage.
Strategy 3 also ensured that all orders which only require one rack, were all fully supplied by a single rack, thereby minimising the requirement to make decisions as to the rack sequence (i.e. the sequence in which racks are presented to each picker).

We presented a heuristic procedure to reduce the number of racks that need to be considered which was independent of our two-stage approach and hence could easily be incorporated into any other solution approach (heuristic or optimal) for order and rack allocation.

We also presented a heuristic for order and rack allocation based on partial integer optimisation that made direct use of our two-stage formulation.

Extensive computational results were presented for test problems that are made publicly available.
Acknowledgments

The completion of this work by the second author occurred whilst he was a Distinguished Research Fellow at NYU London.

References

E. Angelelli, R. Mansini, and M. G. Speranza. Kernel search: A general heuristic for the multi-dimensional knapsack problem. *Computers & Operations Research*, 37(11):2017–2026, 2012. doi: 10.1016/j.cor.2010.02.002.

K. Azadeh, R. de Koster, and D. Roy. Robotized and automated warehouse systems: review and recent developments. *Transportation Science*, 53(4):917–945, 2019. doi: 10.1287/trsc.2018.0873.

M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. B. Rohler. *Matheuristics: Optimization, simulation and control*, volume 5818 of *Lecture Notes in Computer Science*, pages 171–177. Springer, Berlin, 2009. doi: 10.1007/978-3-642-04918-7_13.

N. Boysen, D. Briskorn, and S. Emde. Parts-to-picker based order processing in a rack-moving mobile robots environment. *European Journal of Operational Research*, 262(2):550–562, 2017. doi: 10.1016/j.ejor.2017.03.053.

N. Boysen, R. de Koster, and F. Weidinger. Warehousing in the e-commerce era: a survey. *European Journal of Operational Research*, 277(2):396–411, 2019. doi: 10.1016/j.ejor.2018.08.023.

J. Cai, X. Li, Y. Liang, and S. Ouyang. Collaborative optimization of storage location assignment and path planning in robotic mobile fulfillment systems. *Sustainability*, 13(10), 2021. doi: 10.3390/su13105644.

CPLEX Optimizer. IBM. Available from https://www.ibm.com/products/ilog-cplex-optimization-studio last accessed October 2021, 2019.

I. R. da Costa Barros and T. P. Nascimento. Robotic mobile fulfillment systems: a survey on recent developments and research opportunities. *Robotics and Autonomous Systems*, 137, 2021. doi: 10.1016/j.robot.2021.103729.

G. Duan, C. Zhang, P. Gonzalez, and M. Qi. Performance evaluation for robotic mobile fulfillment systems with time-varying arrivals. *Computers & Industrial Engineering*, 158:107365, 2021. doi: 10.1016/j.cie.2021.107365.

D. Fussler and N. Boysen. High-performance order processing in picking workstations. *EURO Journal on Transportation and Logistics*, 8(1):65–90, 2019. doi: 10.1007/s13676-017-0113-8.

Z. Ghelichi and S. Kilaru. Analytical models for collaborative autonomous mobile robot solutions in fulfillment centers. *Applied Mathematical Modelling*, 91:438–457, 2021. doi: 10.1016/j.apm.2020.09.059.

G. Guastaroba, M. Savelsbergh, and M. G. Speranza. Adaptive kernel search: A heuristic for solving mixed integer linear programs. *European Journal of Operational Research*, 263(3):789–804, 2017. doi: 10.1016/j.ejor.2017.06.005.

R. Hanson, L. Medbo, and M. I. Johansson. Performance characteristics of robotic mobile fulfillment systems in order picking applications. *IFAC PapersOnLine*, 51(11):1493–1498, 2018. doi: 10.1016/j.ifacol.2018.08.290.

Y. Jaghbeer, R. Hanson, and M. I. Johansson. Automated order picking systems and the links between design and performance: a systematic literature review. *International Journal of Production Research*, 58(15):4489–4505, 2020. doi: 10.1080/00207543.2020.1788734.

X. Li, X. Yang, C. Zhang, and M. Qi. A simulation study on the robotic mobile fulfillment system in high-density storage warehouses. *Simulation Modelling Practice and Theory*, 112:102366, 2021. doi: 10.1016/j.simpat.2021.102366.

M. Merschformann, T. Lamballais, M. B. M. de Koster, and L. Suhl. Decision rules for robotic mobile fulfillment systems. *Operations Research Perspectives*, 6:100128, 2019. doi: 10.1016/j.orp.2019.100128.

M. Mirzaei, N. Zaerpour, and R. B. M. de Koster. How to benefit from order data: correlated dispersed storage assignment in robotic warehouses. *International Journal of Production Research*, pages 1–20, 2021. doi: 10.1080/00207543.2021.1971787.
A. Ouzidan, M. Sevaux, A.-L. Olteanu, E. G. Pardo, and A. Duarte. On solving the order processing in picking workstations. *Optimization Letters*, pages 1–31, 2020. doi: 10.1007/s11590-020-01640-w.

A. Rimélé, M. Gamache, M. Gendreau, P. Grangier, and L.-M. Rousseau. Robotic mobile fulfillment systems: a mathematical modelling framework for e-commerce applications. *International Journal of Production Research*, pages 1–17, 2021. doi: 10.1080/00207543.2021.1926570.

X. Shi, F. Deng, Y. Fan, L. Ma, Y. Wang, and J. Chen. A two-stage hybrid heuristic algorithm for simultaneous order and rack assignment problems. *IEEE Transactions on Automation Science and Engineering*, pages 1–13, 2021. doi: 10.1109/TASE.2021.3100718.

C. A. Valle and J. E. Beasley. Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment. *Computers & Operations Research*, 125(105090):1–32, 2021. doi: 10.1016/j.cor.2020.105090.

F. Weidinger. *A precious mess: on the scattered storage assignment problem*, pages 31–36. Operations Research Proceedings. Springer International Publishing, 2018. doi: 10.1007/978-3-319-55702-1_5.

F. Weidinger and N. Boysen. Scattered storage: how to distribute stock keeping units all around a mixed-shelves warehouse. *Transportation Science*, 52(6):1412–1427, 2018. doi: 10.1287/trsc.2017.0779.

L. Xie, N. Thieme, R. Krenzler, and H. Li. Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems. *European Journal of Operational Research*, 288(1):80–97, 2021. doi: 10.1016/j.ejor.2020.05.032.

X. Yang, G. Hua, L. Hu, T. C. E. Cheng, and A. Huang. Joint optimization of order sequencing and rack scheduling in the robotic mobile fulfilment system. *Computers & Operations Research*, 135:105467, 2021a. doi: 10.1016/j.cor.2021.105467.

X. Yang, X. Liu, L. Feng, J. Zhang, and M. Qi. Non-traditional layout design for robotic mobile fulfillment system with multiple workstations. *Algorithms*, 14(7), 2021b. doi: 10.3390/a14070203.

Y. Zhuang, Y. Zhou, Y. Yuan, X. Hu, and E. Hassini. Order picking optimization with rack-moving mobile robots and multiple workstations. *European Journal of Operational Research*, to appear, 2021. doi: 10.1016/j.ejor.2021.08.003.