OR Spectrum (2022) 44:535–573
https://doi.org/10.1007/s00291-021-00663-8

ORIGINAL ARTICLE

New solution procedures for the order picker routing problem in U-shaped pick areas with a movable depot

Heiko Diefenbach[1] · Simon Emde[2] · Christoph H. Glock[1] · Eric H. Grosse[3]

Received: 30 April 2021 / Accepted: 16 November 2021 / Published online: 7 December 2021

Abstract

This paper develops new solution procedures for the order picker routing problem in U-shaped order picking zones with a movable depot, which has so far only been solved using simple heuristics. The paper presents the first exact solution approach, based on combinatorial Benders decomposition, as well as a heuristic approach based on dynamic programming that extends the idea of the venerable sweep algorithm. In a computational study, we demonstrate that the exact approach can solve small instances well, while the heuristic dynamic programming approach is fast and exhibits an average optimality gap close to zero in all test instances. Moreover, we investigate the influence of various storage assignment policies from the literature and compare them to a newly derived policy that is shown to be advantageous under certain circumstances. Secondly, we investigate the effects of having a movable depot compared to a fixed one and the influence of the effort to move the depot.

Keywords Order picking · Routing · Storage assignment · U-shaped pick area · Benders decomposition · Dynamic programming

✉ Heiko Diefenbach
diefenbach@pscm.tu-darmstadt.de

Simon Emde
siem@econ.au.dk

Christoph H. Glock
glock@pscm.tu-darmstadt.de

Eric H. Grosse
eric.grosse@uni-saarland.de

[1] Institute of Production and Supply Chain Management, Technical University of Darmstadt, Hochschulstraße 1, Darmstadt 64289, Germany

[2] CORAL - Cluster for Operations Research, Analytics, and Logistics, Aarhus University, Fuglesangs Allé 4, Aarhus V 8210, Denmark

[3] Juniorprofessorship of Digital Transformation in Operations Management, Saarland University, Saarbruecken, Germany

1 Introduction

The management of warehouse operations has received ample attention over many years. Especially order picking, which is commonly described as the retrieval of products from storage locations to fulfill customer orders, is on the top of many research agendas (Van Gils et al. 2018). This is because of, firstly, the high amount of manual human labor that is usually associated with picking orders (Grosse et al. 2015), and, secondly, the fact that order picking is a very time-intensive activity with direct impact on customer service (De Koster et al. 2007). For example, in the United States, more than 3.7 million people are employed in warehousing as manual laborers and material movers (Bureau of Labor Statistics 2016). In the European Union's warehousing and transport support sector, 2.6 million persons are employed (Eurostat 2016). These facts render order picking one of the most important cost factors in warehousing (Tompkins et al. 2010; Rushton et al. 2014).

To reduce the cost of order picking, researchers have developed various mathematical models in the past that support warehouse managers in assigning products to shelf locations, in restructuring incoming orders and in routing order pickers through the warehouse (Van Gils et al. 2018). It is usually advisable to adapt planning procedures to the specific layout of the warehouse (Roodbergen et al. 2015). Warehouse layouts that have been studied in research on order picking in the past include layouts of rectangular shape, which are often denoted as conventional warehouses, either with a single block (e.g., Petersen et al. (2005); Grosse et al. (2014)) or with two or more blocks (e.g., Roodbergen and Koster 2001a; Roodbergen et al. 2015). Non-conventional warehouses, such as leaf, chevron or flying-V, are employed less frequently in practice but still play a significant role (e.g., Masae et al. 2020b, 2021).

U-shaped layouts of warehouse zones, which are sometimes also referred to as "picker nests", can be observed quite often in practice, but have not received much attention in the literature so far. Glock and Grosse (2012), for example, studied order picking in a U-shaped zone and developed procedures for assigning products to shelf locations, for finding a location for the depot of the order picker, and for routing the order picker through the U-zone. Inspired by a practical case in automotive part picking, the authors also introduced the concept of a movable depot: the depot from which the picker sets off and to which she returns need not be at a fixed location, but can be moved within certain limits, potentially shortening pick tours. Put differently, the depot location becomes itself a variable to be optimized. Given the special structure of the pick zone, the authors used a simple sweep algorithm to solve the routing problem. Diefenbach and Glock (2019) also studied a U-shaped warehouse and optimized the layout and item assignment for single command picking with regard to two different objectives, namely pick efficiency and ergonomics. They did, however, not study the routing problem since it is not relevant for single command picking.

The paper at hand revisits the setting studied by Glock and Grosse (2012) and extends the existing work by the following contributions:

- We develop the first exact solution procedure for the picker routing problem in U-shaped order picking zones, namely an algorithm based on combinatorial Benders decomposition.
- In a comprehensive numerical study, we compare the solutions of our newly developed exact procedure to the solutions of the sweep algorithm developed by Glock and Grosse (2012) to analyze the latter's solution quality, which had not been done yet.
- We develop a new heuristic approach extending the idea of the sweep algorithm, based on dynamic programming. The newly developed procedure compares favorably in theory and in our numerical experiments.
- In addition, we derive some managerial insights from our numerical experiments. We propose a new radial storage assignment policy that better matches the specific characteristics of a U-shaped order picking zone, compare it to storage assignment policies from the literature, and demonstrate its advantage in certain situations. Furthermore, we investigate the effects of having a movable compared to a fixed depot and the influence of the effort for moving the depot.

The remainder of this paper is structured as follows: Sect. 2 discusses the related literature. Section 3 formally defines the picker routing problem studied in this paper, while Sect. 4 presents exact and heuristic solution methods. Section 5 presents the results of numerical experiments, and Sect. 6 concludes the paper.

2 Literature review

Researchers have developed numerous mathematical models and algorithms to provide managerial decision support in planning manual order picking operations. The aim of these works has mainly been the reduction of order picking time or travel distance and thus the minimization of costs (see, for reviews, Gu et al. 2007; De Koster et al. 2007; Grosse et al. 2017; Masae et al. 2020a)). To reach this goal, several planning problems have to be addressed. These include layout design, routing, storage assignment, and order batching. The reader is referred to the review of Van Gils et al. (2018) for a detailed overview of order picking planning models.

Works on layout design mostly deal with the definition of a suitable warehouse layout, which includes decisions about the number of storage blocks, cross aisles, parallel aisles, as well as height and depth of racks (e.g., Vaughan 1999; Roodbergen and Vis 2006; Roodbergen et al. 2008, 2015). Here, it is important to keep space requirements and other types of restrictions in mind when determining the width of aisles, which can be narrow, wide or mixed (Mowrey and Parikh 2014). The majority of works studies rectangular/conventional layouts, whereas alternative layouts for manual order picking areas are rather rare (Masae et al. 2020a). However, alternative layouts for order picking areas are quite common in practice. These include U-shaped layouts, where shelves or pallets are arranged in the shape of a U within the order picking area (Glock and Grosse 2012; Diefenbach and Glock 2019). We note that Henn et al. (2013) also refer to their considered layout as U-shaped. It is, however, fundamentally different from the one

considered in this paper as it resembles a more conventional warehouse layout, where the cross aisles are arranged in the shape of an H or U. Other alternative layouts such as *fishbone* (Gue and Meller 2009) or *flying-V* designs (Öztürkoğlu et al. 2014) are proposed for unit-load warehouses, where products are picked in pallet quantities.

Routing methods that guide order pickers through the warehouse on preferably shortest routes are mainly developed for conventional warehouses. An exact algorithm that calculates shortest routes exists for one-block warehouses, solving a special case of the traveling salesman problem (Ratliff and Rosenthal 1983; Scholz et al. 2016; Lu et al. 2016; Chabot et al. 2017; Masae et al. 2020a). Although this exact algorithm exists, many authors studied simple routing heuristics (such as the well-known s-shape heuristic) because these easy-to-follow patterns are often applied in practice (Petersen and Aase 2004; Glock et al. 2017). For special cases of rectangular warehouses with more cross aisles, extensions of this exact algorithm (Roodbergen and Koster 2001a; Pansart et al. 2018) as well as heuristic solution approaches exist (Roodbergen and Koster 2001b; Theys et al. 2010; Çelik and Süral 2019; Chen et al. 2019). For alternative layouts, Masae et al. (2020b), for example, propose an exact algorithm to solve the picker routing problem in the chevron layout as well as in the leaf warehouse (Masae et al. 2021), and Çelk and Süral (2014) for the fishbone layout. In U-shaped order picking areas, Glock and Grosse (2012) propose a sweep algorithm to calculate order picking routes. In a similar setting, Glock et al. (2019) assume a sufficient capacity of the order picker to transport all requested items in a single tour, which simplifies the routing problem.

Storage assignment methods assign items to storage positions (Reyes et al. 2019). This can either be random or according to some criteria, such as item demand or volume (Füßler et al. 2019). Common advice for practitioners is to assign frequently requested items to storage locations close to the depot (also denoted as *pick-up/drop-off point*, see Petersen and Aase (2004)). In rectangular warehouses, a common approach is to define item classes (typically A, B, and C items according to demand frequency), which are then assigned to specific aisles or zones (De Koster et al. 2007). Petersen et al. (2004) propose several patterns for class-based storage to classify aisles, e.g., within-aisle, diagonal or rectangular strategies. Several algorithms for class-based storage assignment exist (Muppani and Adil 2008). For U-shaped layouts, Glock and Grosse (2012) propose dedicated storage assignment methods (i.e. horizontal, vertical, and upper/lower assignments). Moreover, further factors can be considered for storage assignment models, such as precedence constraints, item weight (Žulj et al. 2018a), or other objectives than the minimization of travel distance, for example, the minimization of human energy expenditure (Battini et al. 2016; Calzavara et al. 2017, 2019) or workload (Otto et al. 2017; Glock et al. 2019).

Consolidating or splitting up orders, which is commonly denoted as order batching, can save on travel distance (Cergibozan and Tasan 2019). Only small instances of order batching problems can be solved optimally in reasonable time (Gademann and Velde 2005), which is why many researchers propose heuristic or metaheuristic approaches to address this problem (Hong et al. 2012; Henn and Wäscher 2012; Matusiak et al. 2014; Pan et al. 2015; Žulj et al. 2018b). Other authors develop metaheuristic algorithms to solve the combined order batching and picker routing

problems in an integrated fashion (Kulak et al. 2012; Grosse et al. 2014; Van Gils et al. 2019).

This paper addresses two out of the discussed four planning problems, contributing new insights in this research field, namely developing a new heuristic and an exact algorithm based on combinatorial Benders decomposition for the picker-routing problem within the U-zone and proposing a new radial storage assignment method for this special layout.

3 Problem description

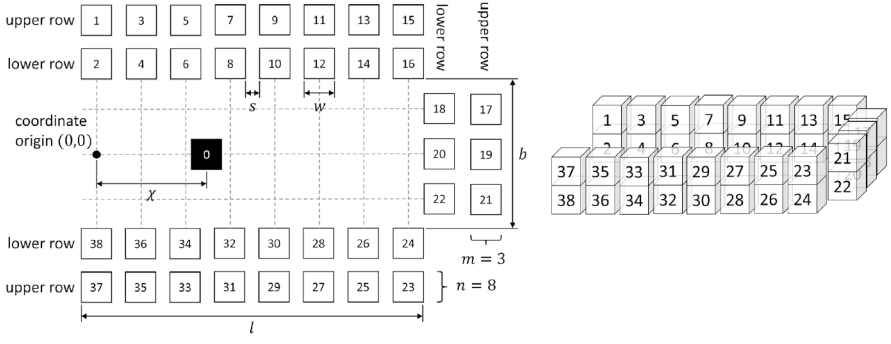
This paper studies order picking in a U-shaped order picking area as outlined in Glock and Grosse (2012). In the considered warehouse setting, items are stored in stillages (i.e., large boxes that can be accessed from the front), with two rows of stillages stacked one atop the other. Each U-zone consists of two horizontal and one vertical shelf as illustrated in Fig. 1. The depot, where each order picking tour starts and ends, is also a stillage that is brought to and removed from the U-zone by a forklift truck, and it is represented by the black box in Fig. 1a. The picker travels on foot along the shelves of the U-zone, possibly pushing or pulling a cart or a related device. U-shaped order picking areas, as the one studied in this paper, can frequently be observed in practice, for example in the automotive or chemical industries (Glock and Grosse 2012; Glock et al. 2019).

In these industries, it is common to prepare so-called kits to supply assembly workplaces with the required materials. For preparing the kits, compact work zones are established in the warehouse that contain the items required at one or more assembly workplaces (e.g., Hanson et al. 2017). Especially in cases where only a small number of items is stored in the kitting zones, U-zones are beneficial because of a clear separation of items and good item accessibility.

3.1 Formal description of the picker routing problem

To model the problem concisely, we make the same assumptions as in Glock and Grosse (2012) and assume a U-zone arranged as in Fig. 1:

- We consider a picker processing a single order in a single U-zone. An order constitutes a set Ω of items that need to be collected and placed together in the depot – for example, a kit destined for a single production station. During a shift, a picker processes multiple orders. We assume orders are planned beforehand and given from the perspective of our problem, such that order picker routing for a single order is independent from other orders.
- The U-zone's coordinate system is two-dimensional, and the depot can be placed anywhere on the center line of the U-zone (i.e., the y-coordinate of the depot is always 0). The location of the depot has to be fixed before the order picker starts processing the order. Moving the depot from the open end of the U deeper into



(a) Two-dimensional representation.

(b) Three-dimensional representation.

Fig. 1 Layout of one U-zone with 38 stillages ($n = 8$, $m = 3$) and one depot

the zone consumes a certain amount of time proportional to the distance that the depot is moved.

- Euclidean distances are used to calculate the travel distance of the order picker, as we assume that this is the most intuitive way to travel through the pick zone. Formally, we define the Euclidean distance between two points $P_1 = \left(x_1^p, y_1^p\right)$ and $P_2 = \left(x_2^p, y_2^p\right)$ as $D^e\left(P_1, P_2\right) = \sqrt{\left(x_1^p - x_2^p\right)^2 + \left(y_1^p - y_2^p\right)^2}$. The distance between stillages is calculated to the center of each stillage or the depot.
- We do not consider service/picking times because they can be considered constant for a given picklist and are therefore not affected by the picker routes. Moreover, we equate travel distances with travel times. Note that, given a fixed average movement speed of the picker, distances can be transformed into durations by simple multiplication with a constant factor.
- Stillages are numbered in a clockwise manner starting at the upper left corner (see Fig. 1).
- The storage assignment policy is selected prior to the start of the order picking process (see Sect. 5.4). Once items have been assigned to storage locations, the assignment is kept constant until all orders have been completed. Each kind of item is stored in a single stillage and each stillage contains only one kind of item. This implies that items have to fit in a single stillage, which is usually the case in practice for U-shaped order picking zones.
- The order picker must return to the depot when he/she has finished his/her order or when the transport capacity of his/her picking device has been reached. In the latter case, after returning to the depot to drop off items there, he/she can continue picking items. After an order has been completed, the depot is removed, and a new depot is brought to the U-zone together with a new order (pick-by-order).
- The demand of any item does not exceed the carrying capacity of the picker.

We consider a pick area with a layout as depicted in Fig. 1, where stillages are arranged in a U-shape around a picking station (depot). The width of a stillage is w, and to facilitate picking/exchanging stillages, the gap between them is s. The entire length of a zone can then be determined as $l = n \cdot w + (n-1) \cdot s$, where n is the number of horizontal stillages in the zone. The width of the zone is $b = m \cdot w + (m+1) \cdot s$, where m is the number of vertical shelves in the zone. The coordinates of the depot are $(\chi, 0)$. For the first 1 to $2 \cdot n$ stillages, the coordinates of the I-th stillage are $\left(\lceil \frac{i-1}{2} \rceil \cdot (w+s), \frac{b}{2}\right)$. For stillages numbered $I = 2 \cdot n + 1, \ldots, 2 \cdot n + m$, the coordinates are $\left(l - \frac{w}{2}, \lceil \frac{2 \cdot n + m - i}{2} \rceil \cdot (w+s)\right)$ for uneven values of m, and $\left(l - \frac{w}{2}, \left(\lceil \frac{2 \cdot n + m - i - 1}{2} \rceil + \frac{1}{2}\right) \cdot (w+s)\right)$ for even values of m. Stillages with index numbers $i = 2 \cdot n + m + 1, \ldots, 4 \cdot n$ are mirror images of the first $2 \cdot n$ stillages along the x-axis. The distance between any two locations i and i' is

$$d_{i,i'} = D^e\left((x_i, y_i), (x_{i'}, y_{i'})\right) = \sqrt{(x_i - x_{i'})^2 + (y_i - y_{i'})^2}. \quad (1)$$

Let $I = \{1, \ldots, |I|\}$ be the set of stillages, and let $\Omega = \{1, \ldots, |\Omega|\}$ be the set of items that need to be picked for a given order with $|\Omega|$ items. Let $\iota(j)$ be the stillage

i where item j is stored, and let the items be indexed in the same clockwise order as their respective stillages, i.e., let $\imath(j) \leq \imath(j')$, $\forall j, j' \in \Omega : j < j'$, be true. Then, item j is located at coordinates $(x_{\imath(j)}, y_{\imath(j)})$. To shorten notation, we define $\tilde{x}_j = x_{\imath(j)}$ and $\tilde{y}_j = y_{\imath(j)}$ as well as $d_{\imath(j),\imath(j')} = \tilde{d}_{jj'}$. Moreover, let Q be the limited carrying capacity of the picker, and let q_j be the weight of item $j \in \Omega$.

We look for a partition of Ω into r subsets $\{\omega_1, \ldots, \omega_r\}$ such that the total weight of the items in each set ω_k does not exceed the carrying capacity of the picker, i.e., $\sum_{j \in \omega_k} q_j \leq Q$, $\forall k = 1, \ldots, r$. Note that the number r of pick tours is not given in advance. Each ω_k stands for one pick tour the picker makes, starting from the depot, visiting all stillages implied by ω_k, and returning to the depot. For brevity of notation, we introduce sets $\tilde{\omega}_k = \{\imath(j) \mid j \in \omega_k\}$, $\forall k = 1, \ldots, r$, to denote the stillages to be visited to pick the items in ω_k.

Furthermore, we look for a position χ of the depot. The depot can be moved along the center line of the U-zone with the default position located at the open end of the U-zone with $\chi = 0$ (see Fig. 1a). If the depot is moved along the aisle, the warehouse worker transporting the depot has to travel an extra distance into the U-zone both when bringing and removing the depot, which leads to a time penalty that has to be considered in the model and which depends on the extra travel distance equaling $2 \cdot \chi$. Assuming that the picker can walk $2 \cdot v$ times faster (or slower, as the case may be) when placing the depot than his/her speed during the order picking process, the additional distance that has to be covered just to move the depot is $\frac{1}{v} \cdot \chi$. All symbols are summarized in Table 1.

A solution to our picker routing problem thus consists of a partition $\{\omega_1, \ldots, \omega_r\}$ of Ω and the depot position $0 \leq \chi \leq l - \frac{w}{2}$. Among all feasible solutions we seek one where the total distance travelled by the picker – including the penalty distance to move the depot and the distance to visit the stillages – is minimal.

Routing the picker for a given set of stillages $\tilde{\omega}_k$ is technically a travelling salesman problem, which is well-known to be strongly NP-hard (Garey and Johnson 1979). However, due to the special structure of the U-shaped pick area, the routing problem is actually tractable.

Proposition 3.1 For a given set $\tilde{\omega}_k$ of stillages to be visited, an optimal route (i.e., sequence of visits) with regard to total travel distance is to move through the stillages in $\tilde{\omega}_k$ in clockwise order in the shape of a polygon without intersecting edges.

Proof Barachet (1957) shows that, in a Euclidean TSP, an optimal TSP tour never crosses itself. Hence, the optimal tour is in the shape of a polygon without intersecting edges, where an edge touching another edge (at a vertex) counts as an intersection as well. Moreover, this polygon is contained within the convex hull around all points to be visited.

Let the points that lie on the convex hull be labeled in clockwise order. Two points i and i', where $i' \geq i + 2$, can never be connected directly in the optimal route. This is because a connection between i and i' would separate the convex hull into two parts, where one contains the points $i'' \in \{i + 1, \ldots, i' - 1\}$ and the other contains the remaining points. Since the optimal route does not cross itself, there exists

Table 1 Notation for the picker routing problem

Sets	
I	Set of stillages
Ω	Set of items to be picked (indices i, j)
ω_k	variable: set of items to be picked on tour $k \in \{1, \ldots, r\}$
$\tilde{\omega}_k$	Auxiliary variable: set of stillages to be visited on tour $k \in \{1, \ldots, r\}$
Parameters	
$d_{ii'}$	Distance from stillage i to stillage i' (meters)
$\tilde{d}_{jj'}$	Distance between the stillage containing item j and the stillage containing item j'
Q	Maximum carrying capacity of the order picker (kilograms)
q_j	Weight of item j (kilograms)
v	Penalty distance factor for moving the depot
b	Width of the aisle (meters)
l	Length of the aisle (meters)
n	Number of stillages in one row of horizontal shelves
m	Number of stillages in the vertical shelf
s	Distance/gap between two adjacent stillages (meters)
w	Width of a stillage (meters)
x_i	x-coordinate of stillage i (meters)
y_i	y-coordinate of stillage i (meters)
$\tilde{x}_j$	x-coordinate of the stillage containing item j (meters)
$\tilde{y}_j$	y-coordinate of the stillage containing item j (meters)

no optimal route that connects the points from the separate parts, because it would intersect the connection between i and i'. It follows that in the optimal route, each point i on the convex hull can only be connected with its neighboring points $i-1$ and $i+1$ on the convex hull or with points that do not lie on the convex hull.

Clearly, all stillages in a U-shaped picking zone lie on a convex polygon, which also constitutes the convex hull. The only point that can possibly not lie on the convex hull is the depot. Hence, in the optimal route, every stillage must be connected to its neighboring stillages (i.e., its predecessor and successor in a clockwise order) or to the depot. Furthermore, the optimal route has only two connections to the depot, one outgoing and one incoming. If the depot lies on the convex hull, it must be connected to its neighboring stillages. If not, the depot must be connected to two consecutive stillages, due to the same argument as before. If the depot would not be connected to two consecutive stillages, the connection would separate the convex hull into two parts that cannot be connected by a route without an intersection. Consequently, the optimal route is to move from the depot to a stillage, move through the stillages in $\tilde{\omega}_k$ in clockwise (or counter-clockwise) order in the shape of a polygon without intersecting edges, and move back to the depot. □

Let the pair (j, j') denote an edge between stillages $\iota(j)$ and $\iota(j')$ and let $\eta_k = \{(j, j') \in \omega_k \times \omega_k : \iota(j) \leq \iota(j') \wedge \{j'' \in \omega_k \mid \iota(j) < \iota(j'') < \iota(j')\} = \emptyset\} \cup \{(\max\{\omega_k\}, \min\{\omega_k\})\}$ be the set of all edges of the convex polygon spanned by the stillages in $\tilde{\omega}_k$. The optimal route's length can then be formalized as

$$\begin{aligned} g(\omega_k, \chi) &= \sum_{(j,j')\in\eta_k} \tilde{d}_{j,j'} + \min_{(j,j')\in\eta_k} \left\{ -\tilde{d}_{j,j'} + \sqrt{(\tilde{x}_j - \chi)^2 + \tilde{y}_j^2} + \sqrt{(\tilde{x}_{j'} - \chi)^2 + \tilde{y}_{j'}^2} \right\} \\ &= \sum_{(j,j')\in\eta_k} \tilde{d}_{j,j'} + \min_{(j,j')\in\eta_k} \{\gamma_{j,j'}(\chi)\}, \end{aligned} \tag{2}$$

where we define $\gamma_{j,j'}(\chi) = -\tilde{d}_{j,j'} + \sqrt{(\tilde{x}_j - \chi)^2 + \tilde{y}_j^2} + \sqrt{(\tilde{x}_{j'} - \chi)^2 + \tilde{y}_{j'}^2}$ for ease of notation. Note that the first term of Eq. (2) stands for the travel distance of the picker along the U, while the second term is the distance from and to the depot, where, by Proposition 3.1, it is optimal to insert the depot visit in-between the two neighboring stillages from $\tilde{\omega}_k$ that minimize the total distance.

The total objective value of a solution consists of the travel distance of the pick tours plus the time to position the depot in the first place, and it is hence

$$G(\omega_1, \ldots, \omega_r, \chi) = \sum_{k\in\{1,\ldots,r\}} g(\omega_k, \chi) + \frac{1}{v} \cdot \chi. \tag{3}$$

Among all feasible solutions consisting of partition $\{\omega_1, \ldots, \omega_r\}$ and depot location χ, we seek one which minimizes $G(\omega_1, \ldots, \omega_r, \chi)$. We refer to this problem as the *picker routing problem in a U-shaped pick area* (PRP-UA).

3.2 Computational complexity

Given that, by Proposition 3.1, routing in a U-shaped pick zone is computationally easier than on general graphs, it may seem that picker routing in U-shaped zones may not be a hard problem at all. However, PRP-UA is intractable as can be seen by the following proposition.

Proposition 3.2 Solving PRP-UA is NP-hard in the strong sense.

Proof We prove Proposition 3.2 by reduction from bin packing, which is well known to be strongly NP-hard (Garey and Johnson 1979).

The decision version of bin packing is concerned with the following question. Given a set S of items i with associated weight w_i and bins with capacity C, does there exists a partition of items into bins such that no subset of items assigned to the same bin exceeds the bin's capacity C and at most k bins are used?

We propose the following transformation from an instance of bin packing to an instance of PRP-UA. Firstly, we set the measurements l and b of the U-zone such that $2 \cdot k \cdot l < b$ holds (i.e., we set n and m such that $2 \cdot k \cdot (n \cdot w + (n-1) \cdot s) < m \cdot w + (m+1) \cdot s$ holds). Secondly, we associate each item $i \in S$ of the bin packing instance with an item $j \in \Omega$ of the PRP-UA instance (with

$w_i = q_j$ for each associated pair of items) and assign the items $j \in \Omega$ to (arbitrary) stillages in the upper row (i.e., in positive y-direction, cf., Fig. 1a) of the U-layout. Finally, we set $Q = C$ and $v \to \infty$ (i.e., $\frac{1}{v} \to 0$).

A solution of an instance of PRP-UA corresponds to a solution of an instance of bin packing if and only if the objective value is less than $(k+1) \cdot b$, as is shown in the following. Clearly, each route cannot contain a subset of items that exceeds the maximum capacity Q, such that we can associate routes with bins. For PRP-UA, by Proposition 3.1, the length of a single optimal route is bounded from above by $b + 2 \cdot l$, i.e., the length of the route if the picker visits every single stillage in the upper row on one tour cannot be longer than this. Likewise, a route's minimum length is b, i.e., visiting only a single stillage directly above the depot cannot be shorter than this. Hence, a solution using $k+1$ tours or more has at least an objective value of $(k+1) \cdot b$ and a solution using at most k tours has at most an objective value of $k \cdot (b + 2 \cdot l) < k \cdot b + b$ (since we set $2 \cdot k \cdot l < b$). Hence, if and only if the objective value of a solution to PRP-UA is less than $(k+1) \cdot b$, it contains no more than k tours, which corresponds to a solution for the corresponding instance of bin packing with at most k bins.

Since the decision version of PRP-UA is strongly NP-complete, the corresponding optimization version is NP-hard in the strong sense, which completes the proof. □

4 Algorithms

In the following, we present new algorithms to solve PRP-UA. Section 4.1 presents the first exact solution approach based on combinatorial Benders decomposition. With PRP-UA being NP-hard, we can expect that larger problem instances cannot be solved exactly, as is also shown in our computational study later on (cf. Sect. 5.2). We therefore present a new heuristic solution approach based on the concept of dynamic programming in Sect. 4.2.

4.1 Logic-based Benders decomposition for the picker routing problem

In addition to being NP-hard, PRP-UA is further complicated by the presence of non-linear (Euclidean) distances, which depend on a variable, namely the position of the depot. There is therefore no obvious way of formulating a compact (mixed-integer) linear programming model without discretization of the depot location χ. For discrete depot locations, the problem would become a capacitated vehicle routing problem. If we disregard the routing aspect and the moveable depot, the remaining problem, i.e., batching items on tours such that the picker capacity is not violated, is a bin packing problem. To avoid making a heuristic choice regarding discretization intervals, in the following, we focus on an exact approach where we consider the depot location χ as a continuous variable.

To make the problem more tractable, we propose a decomposition scheme in the spirit of logic-based and combinatorial Benders decomposition (Codato and Fischetti 2006; Hooker 2007), which has seen success dealing with difficult combinatorial optimization problems (e.g., Kress et al. 2019; Tadumadze et al. 2020; Fang et al. 2021; Huang et al. 2021). The general idea consists of splitting the original problem into a master and a slave component. The master problem is modeled as a mixed-integer linear programming model that is solved by an off-the-shelf default solver. Whenever the solver finds a candidate integer solution for this model, the solution is passed to the slave problem, which calculates the optimal objective value for the given master solution. From the slave solution, combinatorial cuts are generated, which remove suboptimal solutions from the master model. The solver then continues working on the master model with the newly added cuts, passing candidate solutions to the slave problem until no more feasible, undiscarded solutions remain. The best incumbent solution at this point is optimal. For brevity, we refer to this algorithm as *CBD* (combinatorial Benders decomposition).

In Sect. 4.1.1, we describe the master model for our picker routing problem in detail. In Sect. 4.1.2 we present the slave model and describe how it can be efficiently solved. Section 4.1.3 describes how we generate cuts from solutions of the slave problem.

4.1.1 Master problem

The master problem (MP) consists of batching items on tours, i.e., effectively, determining sets $\omega_k, \forall k$. For a given batching, the exact objective value and optimal depot location are then determined by solving the slave problem described in Sect. 4.1.2. We use binary variables $z_{j,j'}$, which have value 1 if and only if item j' is on the same tour as item j and j' is the item with the greatest index on that tour. Formally, the feasible search space of the master model is described by the following constraints.

$$\sum_{\substack{j' \in \Omega : \\ j \le j'}} z_{j,j'} = 1 \quad \forall j \in \Omega \tag{4}$$

$$z_{j,j'} \le z_{j',j'} \quad \forall j, j' \in \Omega : j < j' \tag{5}$$

$$\sum_{\substack{j \in \Omega : \\ j \le j'}} q_j \cdot z_{j,j'} \le Q \quad \forall j' \in \Omega \tag{6}$$

$$z_{j,j'} \in \{0,1\} \quad \forall j, j' \in \Omega : j \le j' \tag{7}$$

Constraints (4) ensure that each item is on exactly one pick tour. If some item j' is on the same tour as item j such that j' has the highest index in that tour, then $z_{j,j'}=$ is forced to 1, as ensured by Inequalities (5). Inequalities (6) ensure that the picker's

carrying capacity is not exceeded, while Constraints (7) define the domain of the decision variables.

While the master model can be solved as a pure feasibility problem, this may not be advisable from a performance viewpoint. Without any objective to guide the search, we can expect many mediocre solutions to be evaluated. We therefore use a lower bound on the objective value as a subproblem relaxation (Hooker 2007) based on the following idea.

Proposition 3.1 states that the optimal route to visit a set $\tilde{\omega}_k$ of stillages and the depot is in the form of a polygon without crossing edges. Clearly, the edge length of the *convex* polygon spanned by the stillages in $\tilde{\omega}_k$ and the depot is a lower bound on the optimal tour length. Given the non-linear Euclidean distances, there is no easy method to calculate the respective convex polygon's edge length in a compact linear model. However, we can calculate lower bounds on the edge length in two ways: first, by using the rectilinear metric (Manhattan metric) and, second, by using the maximum metric (Chebyshev distance). Formally, for two points $P_1 = \left(x_1^p, y_1^p\right)$ and $P_2 = \left(x_2^p, y_2^p\right)$, we define the rectilinear metric distance as $D^r\left(P_1, P_2\right) = |x_1^p - x_2^p| + |y_1^p - y_2^p|$ and the maximum metric distance as $D^m\left(P_1, P_2\right) = \max\left\{|x_1^p - x_2^p|, |y_1^p - y_2^p|\right\}$.

For the first bound, we apply Proposition 4.1.

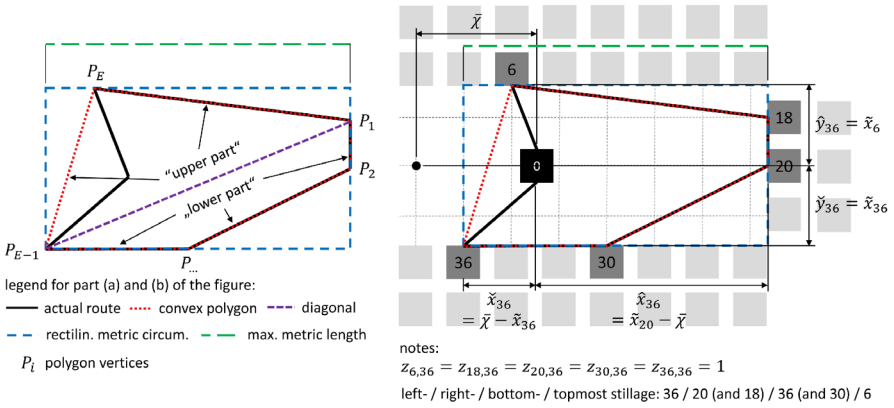
Proposition 4.1 A convex polygon's Euclidean (i.e., actual) circumference is at least its rectilinear metric circumference divided by $\sqrt{2}$.

Proof The proof is based on the labeling in Fig. 2a. The circumference of a polygon with E vertices is given by $C^{\text{poly}} = \sum_{i=1}^{E-1} D^e(P_i, P_{i+1}) + D^e(P_E, P_1)$. In two-dimensional space, the distance between two points measured in the rectilinear metric is at most $\sqrt{2}$ times greater than the distances measured in the Euclidean metric, i.e., $D^e(P_i, P_{i'}) \geq \frac{1}{\sqrt{2}} \cdot D^r(P_i, P_{i'})$ holds (cf., Proposition A.1 in the Appendix). It follows that $C^{\text{poly}} \geq \sum_{i=1}^{E-1} \frac{1}{\sqrt{2}} \cdot D^r(P_i, P_{i+1}) + \frac{1}{\sqrt{2}} \cdot D^r(P_E, P_1) \quad \Rightarrow C^{\text{poly}} \geq \frac{1}{\sqrt{2}} \cdot \left(\sum_{i=1}^{E-1} D^r(P_i, P_{i+1}) + D^r(P_E, P_1)\right)$ $\Rightarrow C^{\text{poly}} \geq \frac{1}{\sqrt{2}} \cdot C^{\text{rect}}$, where C^{rect} is the polygon's circumference in the rectilinear metric. □

Hence, we can calculate the convex polygon's edge length in the rectilinear metric and divide it by $\sqrt{2}$ to attain a lower bound for the actual length. For the second bound, we apply Proposition 4.2.

Proposition 4.2 A convex polygon's Euclidean (i.e., actual) circumference is at least two times its length in the maximum metric.

Proof The proof is based on the labeling in Fig. 2a. Assume any diagonal of the polygon connecting the two shorter sides of its rectilinear circumference. Such a diagonal always exists, since each side of the polygon's rectilinear circumference is connected to at least one of the polygon's vertices. The diagonal splits the polygon into two parts, which we label *upper part* and *lower part* in the following. By the



(a) Concept of using the rectilinear and maximum metric to formulate lower bounds on a convex polygon's circumference.

(b) Illustration of the variables $\check{x}_j$, $\check{y}_j$, $\hat{x}_j$, and $\hat{y}_j$ being used to calculate lower bounds on a route's length.

Fig. 2 Lower bounds on a picking route's length

triangle inequality, the polygon's upper part edge length is longer than the diagonal. The same applies for the lower part's edge length. Therefore, the polygon's Euclidean circumference is at least two times the length of the diagonal. Again by the triangle inequality, the diagonal length is never less than the polygon's maximum length, which is equal to the longer side of the polygon's rectilinear circumference. Hence, two times the polygon's maximum length is never greater than its Euclidean circumference. □

Therefore, two times the convex polygon's length in the maximum metric is also a lower bound for the actual (Euclidean) circumference.

To calculate the bounds, we introduce four sets of auxiliary continuous variables to the master model. Variables $\check{x}_j$ ($\check{y}_j$) denote the distance from the depot to the leftmost (bottommost) stillage on the tour that contains item j. Analogously, variables $\hat{x}_j$ ($\hat{y}_j$) denote the distance from the depot to the rightmost (topmost) stillage on the tour that contains item j. Moreover, we introduce auxiliary variable $\bar{\chi}$ to denote the position of the movable depot. An illustrative example is given in Fig. 2b. We consider the bounds by adding the following objective and valid inequalities to the master model:

$$\text{[MP] Minimize } G = \sum_{j \in \Omega} \hat{d}_j + \frac{1}{v} \cdot \bar{\chi} \tag{8}$$

subject to (4)–(7) and

$$\hat{x}_j \geq \tilde{x}_{j'} \cdot z_{j',j} - \bar{\chi} \quad \forall j, j' \in \Omega : j' \leq j \tag{9}$$

$$\check{x}_j \geq \bar{\chi} - \left(\left(\tilde{x}_{j'} - l\right) \cdot z_{j',j} + l\right) \quad \forall j, j' \in \Omega : j' \leq j \tag{10}$$

$$\hat{y}_j \geq \tilde{y}_{j'} \cdot z_{j',j} \quad \forall j, j' \in \Omega : j' \leq j \tag{11}$$

$$\check{y}_j \geq -\tilde{y}_{j'} \cdot z_{j',j} \quad \forall j, j' \in \Omega : j' \leq j \tag{12}$$

$$\hat{d}_j \geq 2 \cdot \frac{1}{\sqrt{2}} \cdot \left(\hat{x}_j + \check{x}_j + \hat{y}_j + \check{y}_j\right) \quad \forall j \in \Omega \tag{13}$$

$$\hat{d}_j \geq 2 \cdot \left(\hat{x}_j + \check{x}_j\right) \quad \forall j \in \Omega \tag{14}$$

$$\hat{d}_j \geq 2 \cdot \left(\hat{y}_j + \check{y}_j\right) \quad \forall j \in \Omega \tag{15}$$

$$\hat{x}_j, \check{x}_j \in \left[0, \max_{i \in I} \left\{x_i\right\}\right] \quad \forall j \in \Omega \tag{16}$$

$$\hat{y}_j, \check{y}_j \in \left[0, \max_{i \in I} \left\{|y_i|\right\}\right] \quad \forall j \in \Omega \tag{17}$$

$$\hat{d}_j \in [0, 2 \cdot l + 4 \cdot b] \quad \forall j \in \Omega \tag{18}$$

$$\bar{\chi} \in \left[0, l - \frac{w}{2}\right] \tag{19}$$

Objective (8) minimizes the sum of the distance approximations of the routes plus the cost to move the depot. For each route, Inequalities (9) determine the distance between the depot and the polygon's rightmost vertex. If the depot is to the right of the polygon's rightmost vertex, $\hat{x}_j$ assumes 0. Similarly, Inequalities (10) calculate the distance between the depot and the polygon's leftmost vertex. If the depot is to the left of the polygon's leftmost vertex, $\check{x}_j$ assumes 0. Inequalities (11) and (12) calculate the respective distances on the y-axis for every tour. Constraints (13) calculate lower bounds on the tour length based on the rectilinear metric. Constraints (14) and (15) determine respective lower bounds based on the maximum metric. Finally, Constraints (16) through (19) define the domain of the auxiliary variables.

4.1.2 Slave problem

Solving the master model generates feasible item batches, which may, however, not be optimal. Let $\bar{z}$ be a candidate integer solution of the master model, defining $r = \sum_{j \in \Omega} \bar{z}_{j,j}$ pick tours. Each pick tour $k = 1, \ldots, r$ consists of picking the items in set $\bar{\omega}_k = \{j \in \Omega \mid \bar{z}_{j,j'} = 1\}$, where $j' \in \Omega$ is the k-th item for which $\bar{z}_{j'j'} = 1$. We refer

to the set of pick tours derived from the current master solution as $\bar{\Phi} = \{\bar{\omega}_1, \ldots, \bar{\omega}_r\}$, where we use bars to indicate that the sets $\bar{\omega}_k$ and $\bar{\Phi}$ are fixed within the slave problem.

Given an item batching $\bar{z}$, two problems remain to be solved: First, we must decide on the location χ of the depot, and second, we need to determine the optimal picker route for each $\bar{\omega}_k$. As soon as the first problem is solved, the second becomes trivial. Hence, we begin with the former. Note that auxiliary variable $\bar{\chi}$ in the master model does not necessarily correspond to the optimal depot location because the distance values $\hat{d}_j$ are only lower bounds. Instead, the best depot location χ for a given set $\bar{\Phi}$ of tours can be found by minimizing $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ (cf., Eq. (3)) for the given candidate integer solution $\bar{z}$, which can be done as follows.

The term $\frac{\partial G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)}{\partial \chi}$ is not continuous because of the minimum term in each $g(\bar{\omega}_k, \chi)$ (cf., Eq. (2)). However, for a fixed set $\bar{\Phi}$, it is piece-wise continuous in every interval where $\arg\min_{(j,j') \in \eta_k} \left\{\gamma_{j,j'}(\chi)\right\}$ is constant, $\forall k \in \{1, \ldots, r\}$. Hence, to minimize $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$, three steps are necessary. First, we need to determine every interval, in which $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ has a constant derivative. Second, for each interval, we need to determine the minimum of $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ separately. Third, out of all intervals' minima, we need to select the one that is minimal overall. The last step is trivial. In the following, we discuss the first two steps, where we label the minimal value for $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ as $G^*(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ and the respective depot location $\chi^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$.

Starting from the second step, the minimum within an interval can be determined by finding the root of $\frac{\partial G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)}{\partial \chi}$ within that interval. The root within a continuous interval can be determined with arbitrary precision using a gradient descent method (e.g., Newton's method). Note that finding the root of the derivative within each interval is sufficient, since $\frac{\partial^2 \left(-d_{j,j'} + \gamma_{j,j'}(\chi)\right)}{\partial \chi^2} = \tilde{y}_j^2 \cdot \left(\left(\tilde{x}_j - \chi\right)^2 + \tilde{y}_j^2\right)^{-\frac{3}{2}} + \tilde{y}_{j'}^2 \cdot \left(\left(\tilde{x}_{j'} - \chi\right)^2 + \tilde{y}_{j'}^2\right)^{-\frac{3}{2}} > 0$, hence, in a given interval, $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ is convex and only has a single minimum. For the same reason, if the root lies outside of the interval, the interval's minimum is equal to one of its boundaries.

It remains to discuss the first step, i.e., to determine the borders of the intervals in which $\arg\min_{(j,j') \in \eta_k} \left\{\gamma_{j,j'}(\chi)\right\}$ is constant. The edge $\arg\min_{(j,j') \in \eta_k} \left\{\gamma_{j,j'}(\chi)\right\}$, which we call *dominant* edge in the following, can only change at values χ where the functions $\gamma_{j,j'}(\chi)$ and $\gamma_{j'',j'''}(\chi)$ intersect, for two edges $(j,j'), (j'',j''') \in \eta_k : (j,j') \neq (j'',j''')$. This leads to the following iterative procedure. Initially, we determine $(\hat{j},\hat{j}') = \arg\min_{(j,j') \in \eta_k} \left\{\gamma_{j,j'}(0)\right\}$ as the dominant edge for $\chi = 0$ and we initialize $\hat{\chi} := 0$. In each iteration, we determine the consecutive dominant edge. The dominant edge $(\hat{j},\hat{j}')$ changes, when

$$c((\hat{j},\hat{j}'),(j,j'),\chi) = \gamma_{\hat{j},\hat{j}'}(\chi) - \gamma_{j,j'}(\chi)$$

assumes zero, approaching from a negative value with increasing χ, for any $(j,j') \in \eta_k : (j,j') \neq (\hat{j},\hat{j}')$ and $l - \frac{w}{2} \geq \chi > \hat{\chi}$. Let χ' be the smallest value of χ for which $c((\hat{j},\hat{j}'),(j,j'),\chi)$ assumes zero approaching from a negative value, $\forall (j,j') \in \eta_k : (j,j') \neq (\hat{j},\hat{j}')$, and let (j'',j''') be the respective edge. Then we update $\hat{\chi} := \chi'$ and $(\hat{j},\hat{j}') = (j'',j''')$ and start the next iteration. The values of $\hat{\chi}$ found

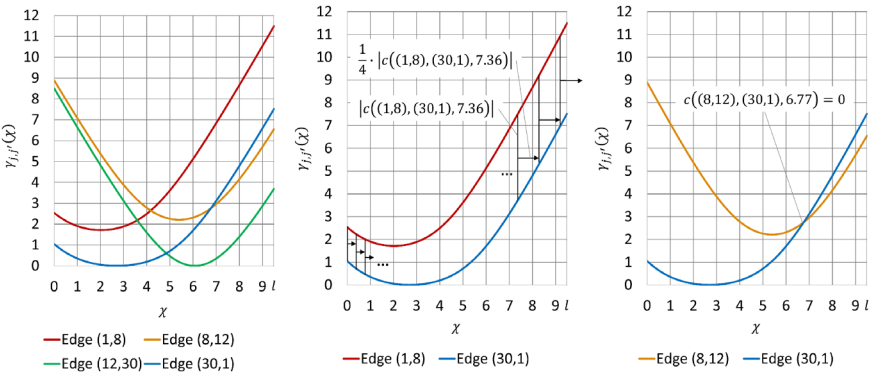
during the iterations mark the intervals' borders. The procedure terminates as soon as no $\chi' \leq l - \frac{w}{2}$ can be found anymore.

During the iterative procedure, we solve $c((\hat{j},\hat{j}'),(j,j'),\chi) = 0$ as follows. At first glance, we cannot rule out that function $c((\hat{j},\hat{j}'),(j,j'),\chi)$ has none, one or multiple roots, such that we cannot use a gradient descent approach, since it may overshoot if there are multiple roots or not terminate if there are none. However, we can conclude that $\left|\frac{\partial c((j,j'),(j'',j'''),\chi)}{\partial \chi}\right| \leq 4$ because $0 \leq \left|\frac{\partial \sqrt{(\tilde{x}_j-\chi)^2+\tilde{y}_j^2}}{\partial \chi}\right| = \left|-\frac{\tilde{x}_j-\chi}{\sqrt{(\tilde{x}_j-\chi)^2+\tilde{y}_j^2}}\right| \leq 1$, $\forall j \in \Omega$ (cf. Eq. (2)). This means that if χ changes by a step size of μ, $c((j,j'),(j'',j'''),\chi)$ changes by no more than $4 \cdot \mu$, which leads to the following iterative sub-routine. Starting from $\tilde{\chi} := \hat{\chi}$, we determine a step size $\mu := \max\left\{\frac{1}{4} \cdot |c((j,j'),(j'',j'''),\tilde{\chi})|, \mu^{min}\right\}$, where μ^{min} is the arbitrarily small numerical precision. We update $\tilde{\chi} := \tilde{\chi} + \mu$ and evaluate $c((j,j'),(j'',j'''),\tilde{\chi})$. We repeat this process until $c((j,j'),(j'',j'''),\tilde{\chi})$ changes from a negative value to a positive value between two iterations or until we reach $\tilde{\chi} > l - \frac{w}{2}$ and return $\tilde{\chi}$ as the solution. The complete procedure to solve the slave problem is summarized in pseudocode in Algorithm 1.

Example For additional clarification, in the following, we exemplary demonstrate how we determine the borders of the intervals in which $g(\bar{\omega}_k, \chi)$ has a continuous derivative. Consider a U-zone as depicted in Fig. 1a, where the items $j \in \omega_k = \{1, 8, 12, 30\}$ should be picked on the same tour. To keep the example simple, we assume every item is stored in stillage i with the same respective index and set $w = 1.3$ m and $s = 0.05$ m (cf., Sect. 5). The set of edges is then given as $\eta_k = \{(1, 8), (8, 12), (12, 30), (30, 1)\}$. Figure 3a depicts the functions $\gamma_{j,j'}(\chi)$, $\forall (j,j') \in \eta_k$.

The procedure starts by determining the dominant edge at $\chi = 0$, which is $(\hat{j},\hat{j}') = \arg\min_{(j,j') \in \eta_k}\{\gamma_{j,j'}(0)\} = (30, 1)$. Starting with edge (1, 8) and with $\chi = 0$, we iteratively increase χ to determine a possible intersection between $\gamma_{30,1}(\chi)$ and $\gamma_{1,8}(\chi)$, which is schematically shown in Fig. 3b. Since there is no intersection, for $\chi \leq l - \frac{w}{2}$, we continue with the next edge, i.e., (8, 12). Here, we find an intersection at $\chi = 6.77$ (cf., Fig. 3). We save $\hat{\chi} := 6.77$ and select the next edge (12, 30) to look for intersections between $\gamma_{30,1}(\chi)$ and $\gamma_{12,30}(\chi)$. We find an intersection at $\chi = 4.85$ and overwrite $\hat{\chi} := 4.85$. At this point, we have checked for intersections between $\gamma_{30,1}(\chi)$ and $\gamma_{j,j'}(\chi)$ for all other edges $(j,j') \in \eta_k : (j,j') \neq (30, 1)$ and found the intersection closest to 0 at $\hat{\chi} := 4.85$. Hence, the fist interval where $g(\bar{\omega}_k, \chi)$ has a constant derivative is [0, 4.85). To determine the next interval, we set the dominant edge to $(\hat{j},\hat{j}') = (12, 30)$ and try to find intersections for $\chi > 4.85$ in the same way as before. Since there are no more intersections, we determine the second (and in this case final) interval as $[4.85, l - \frac{w}{2}]$.

Determining these intervals not only for $\bar{\omega}_k$ but for all $\bar{\omega}_1, \ldots, \bar{\omega}_r$ gives the intervals in which $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ has a continuous derivative. To solve the slave problem, the final step is then to determine the root of $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ within every such interval and to select the one that is minimal overall.



(a) $\gamma_{j,j'}(\chi)$, $\forall (j,j') \in \eta_k$.

(b) Iterative determination of intersections between $\gamma_{30,1}(\chi)$ and $\gamma_{1,8}(\chi)$.

(c) Intersections between $\gamma_{30,1}(\chi)$ and $\gamma_{8,12}(\chi)$.

Fig. 3 Exemplary determination of intervals where $g(\omega_k, \chi)$ has a continuous derivative

Once the optimal depot location $\chi^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$ has been determined, the optimal route for each set $\bar{\omega}_k$, $\forall k = \{1, \ldots, r\}$, follows immediately from the continuous interval in which $\chi^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$ lies. We define $(j_k^*(\bar{\omega}_1, \ldots, \bar{\omega}_r), j_k'^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)) = \arg\min_{(j,j') \in \eta_k} \{\gamma_{j,j'}(\chi^*(\bar{\omega}_1, \ldots, \bar{\omega}_r))\}$. Then the optimal route for the set $\bar{\omega}_k$ visits items $j_k^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$ and $j_k'^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$ directly before returning to and after leaving from the depot, respectively. All other items $j \in \bar{\omega}_k \setminus \{j_k^*(\bar{\omega}_1, \ldots, \bar{\omega}_r), j_k'^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)\}$ are visited in clockwise order in-between (cf., Proposition 3.1).

4.1.3 Combinatorial cuts

Let *UB* be the objective value of the current best known solution (i.e., the best currently known upper bound on the optimal objective value). If no feasible solution is known yet, let $UB = \infty$. If $G^* < UB$, a new best solution has been found, which is stored, and *UB* is updated to G^*. If *UB* is updated, we add the following cut, which we call an *optimality cut*, to the constraint set of the master model:

$$G \leq G^*(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi^*(\bar{z})) - \epsilon \tag{20}$$

where ϵ is a sufficiently small positive number and G is the objective function of the master model. This equation cuts off all solutions whose lower bound is not better than the current upper bound.

Algorithm 1: Algorithm for solving the slave problem.

Input: $\bar{\Phi} = \{\bar{\omega}_1, \ldots, \bar{\omega}_r\}$

$B := \left\{l - \frac{w}{2}\right\}$; // borders of the intervals, in which $\frac{\partial G(\bar{\omega}_1,\ldots,\bar{\omega}_r,\chi)}{\partial \chi}$ is continuous

for $k \in \{1, \ldots, r\}$ **do**

  $b := \{0\}$; // borders of the intervals, in which $\frac{\partial g(\bar{\omega},\chi)}{\partial \chi}$ is continuous

  $\hat{\chi}^{\text{previous}} := 0$; // previous interval border

  $(\hat{j}, \hat{j}') = \arg\min_{(j,j') \in \eta_k} \{\gamma_{j,j'}(0)\}$; // dominant edge

  $\hat{\chi} := 0$; // interval border

  while $\hat{\chi} < l - \frac{w}{2}$ **do**

    $\hat{\chi} := l - \frac{w}{2}$;

    foreach $(j, j') \in \eta_k : \gamma_{\hat{j},\hat{j}'}(\chi) \neq \gamma_{j,j'}(\chi)$ **do**

      $\chi := \max\{b\}$;

      while $c((\hat{j}, \hat{j}'), (j, j'), \chi) < 0 \wedge \chi < l$ **do**

        $\chi := \chi + \max\left\{\frac{1}{4} \cdot |c((j, j'), (j'', j'''), \tilde{\chi})|, \mu^{min}\right\}$;

      if $\chi < \hat{\chi}$ **then**

        $\hat{\chi} := \chi$;

        $(\hat{j}, \hat{j}') = (j, j')$

    $b := b \cup \{\hat{\chi}\}$

  $B := B \cup b$;

$\chi^* := 0$; // optimal depot position for the given $\bar{\Phi}$

$G^* := \infty$; // optimal objective for the given $\bar{\Phi}$

foreach *pair of numerically consecutive elements* β_1 *and* β_2 *(with* $\beta_1 < \beta_2$*) in* B **do**

  $\chi :=$ the solution of $\frac{\partial G(\bar{\omega}_1,\ldots,\bar{\omega}_r,\chi)}{\partial \chi} = 0$ using Newton's method;

  $\chi := \min\{\chi, \beta_2\}$;

  $\chi := \max\{\chi, \beta_1\}$;

  if $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi) < G^*$ **then**

    $G* := G(\chi)$;

    $\chi^* := \chi$;

Output: G^* and χ^*

Regardless of whether a new upper bound has been found, we make use of the following observation to generate further cuts. To find a better solution, the duration of at least one of the r tours must be made shorter, or one of the tours must be dissolved altogether. A tour can only become shorter if the polygon constituting the tour changes its form. The polygon can only change its form if a stillage lying at a vertex changes, but the polygon stays unchanged if a stillage lying on (the middle of) an edge changes. There are two general possibilities where the former is the case.

The first possibility is that either the first or the last stillage to be visited per face of the U-shaped picking area changes. Adding or removing stillages that lie in-between two other stillages on the same face (and that are not visited directly before or after the depot) cannot reduce the duration of the tour because the picker passes by that stillage in any case. The second possibility is that a stillage changes that is visited either directly after leaving or before returning to the depot. For additional clarification, Fig. 4 provides an example.

Let $T = \left\{j \in \Omega \mid \tilde{y}_j = \max_{i \in I} \left\{y_i\right\}\right\}$ be the set of items on the top face of the U-shaped picking area. Analogously, let B be the set of items on the bottom face, and R be the set of items on the perpendicular face. For notational convenience, we define the set $\underline{T}_k = \left\{j \in T \cap \omega_k \mid \tilde{x}_j = \min_{j \in T \cap \omega_k} \{\tilde{x}_j\}\right\}$ and $\overline{T}_k = \left\{j \in T \cap \omega_k \mid \tilde{x}_j = \max_{j \in T \cap \omega_k} \{\tilde{x}_j\}\right\}$ as well as,

analogously, $\underline{B}_k$, $\overline{B}_k$, $\underline{R}_k$ and $\overline{R}_k$ as the extreme items on each face. Then the set of all items that could be changed to alter the route of tour k is given as $\Lambda_k = \underline{T}_k \cup \overline{T}_k \cup \underline{B}_k \cup \overline{B}_k \cup \underline{R}_k \cup \overline{R}_k \cup \left\{j_k^*(\bar{\omega}_1, \ldots, \bar{\omega}_r), j_k'^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)\right\}$.

Using these definitions, we add the following cut, which we call a *progression cut*, to the master model:

$$1 \leq \sum_{k=1}^{r} \sum_{j \in \Lambda_k} \left(1 - z_{j,\max\{\bar{\omega}_k\}}\right) \tag{21}$$

Inequality (21) enforces that at least one stillage at a polygon's vertex of one of the r tours is reassigned. Note that this always makes the current solution infeasible. The solver thus continues solving the master model with the newly added cut(s), intermittently calling the slave problem whenever a new candidate integer solution is found until the search space is empty.

4.2 Heuristic solution approaches

While the proposed CBD is able to solve problem instances of smaller sizes in acceptable runtime, its runtime gets excessively long when working on larger sized instances (cf., Sect. 5.2). Therefore, we also consider heuristic solution approaches in the following.

Glock and Grosse (2012) propose a heuristic sweep algorithm (SA) for the picker routing problem in U-shaped pick areas. In our computational experiments (cf., Sect. 5), the SA produces good results on average. However, for some instances, the optimality gaps can be substantial. Inspired by these findings, we propose a heuristic solution procedure using the concept of dynamic programming (DP) that expands on the idea of the SA. In the following, we briefly review the SA of Glock and Grosse (2012) before discussing our DP approach.

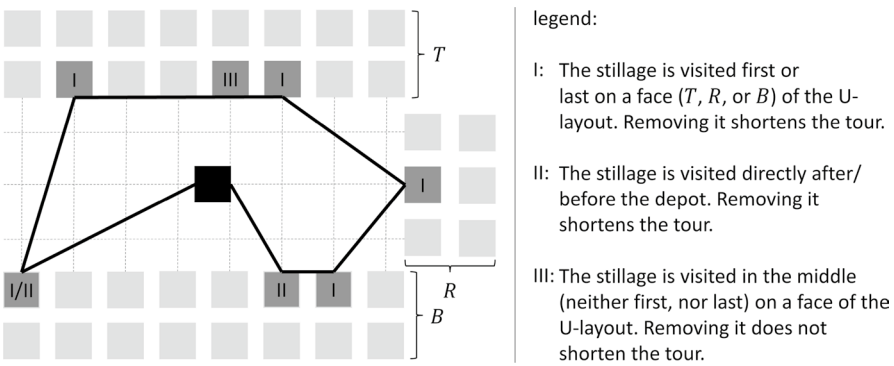


Fig. 4 Example of which stillages are considered in the *progression cut*

4.2.1 Sweep algorithm

The SA starts by assuming a fixed depot position $\bar{\chi}$. Its basic idea is to assign items to sets ω_k in clockwise order. Starting from an initial item, the SA assigns items to the same set ω_k, until the next item would exceed the picker's capacity Q. In that case, the next item is assigned to a new set ω_{k+1} and the previous set is closed. For each closed set of items ω_k, $g(\omega_k, \bar{\chi})$ is derived (cf. Eq. 2). Since $\bar{\chi}$ is given, $g(\omega_k, \bar{\chi})$ can be easily calculated. Once all items have been assigned to sets, the objective value is given by $G(\omega_1, \ldots, \omega_r, \bar{\chi})$ (cf. Eq. 3). The fixed depot position $\bar{\chi}$ and the starting item are varied over multiple iterations of the algorithm. Using $\chi^{\text{step-size}}$ as an arbitrarily small step-size to increment $\bar{\chi}$, the algorithm can be formally described as follows.

1. Set $\bar{\chi} := 0, G^* = \infty$ and $\chi^* = 0$.
2. Set $j^{\text{start}} := 1$.
3. Set $j := j^{\text{start}}$ and $k := 1$.
4. Set $\omega_k = \{j\}$.
5. Increment $j := j + 1$. If $j > |\Omega|$ set $j := 1$, i.e., after the U-zone's final item has been reached, proceed with the first item to continue the clockwise sweeping. If $j = j^{\text{start}}$, i.e., if item j has already been considered in the beginning, go to Step 7.
6. If $\sum_{j' \in \omega_k} q_{j'} + q_j \leq Q$, add $\omega_k := \omega_k \cup \{j\}$ and go to Step 5. Else, increment $k := k + 1$ and go to Step 4.
7. Calculate $G(\omega_1, \ldots, \omega_k, \bar{\chi})$. If $G(\omega_1, \ldots, \omega_r, \bar{\chi}) < G^*$, update $G^* := G(\omega_1, \ldots, \omega_r, \bar{\chi})$ and $\chi^* := \bar{\chi}$. Increment $j^{\text{start}} := j^{\text{start}} + 1$. If $j^{\text{start}} \in \Omega$, go to Step 3.
8. Increment $\bar{\chi} := \bar{\chi} + \chi^{\text{step-size}}$. If $\bar{\chi} \leq l - \frac{w}{2}$, go to Step 2. Else, terminate the procedure.

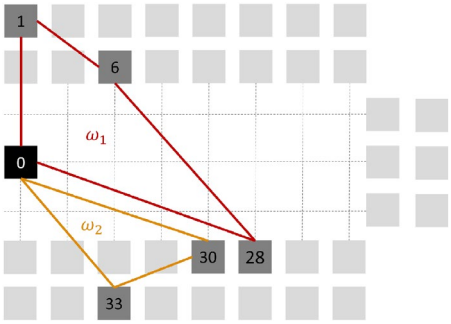
Example Consider a U-zone as depicted in Fig. 1a, where the items $j \in \{1, 6, 28, 30, 33\}$ should be picked. To keep the example simple, we assume every item is stored in stillage i with the same respective index and set $w = 1.3$ m and $s = 0.05$ m (cf., Sect. 5). The weights of the items are given in Fig. 5a and the picker has a capacity of $Q = 5$. For the given initial item $j^{\text{start}} = 1$ and the depot location $\bar{\chi} := 0$, the solution as determined by the SA is given in Fig. 5b with an objective value of 29.43.

4.2.2 Heuristic dynamic programming algorithm

In the SA, items are always added to the currently "open" set ω_k until adding another item would exceed the capacity (cf., Step 6). However, we notice in our computational experiments that this is not always a good choice (see Sect. 5). Instead, sometimes it is better to "close" a set ω_k before the capacity is reached,

j	q_j
1	1
6	2
28	2
30	1
33	2

(a) Example item weights.



(b) Solution found by the SA.

Fig. 5 Example solved by the the SA

and add the next item to the consecutive set ω_{k+1}. The following DP approach takes this observation into consideration.

Similar to the SA, the DP procedure assumes a fixed depot position $\bar{\chi}$ and an initial start item j^{start} at the beginning of each iteration and considers items in a clockwise order. The solution is then constructed piece-wise using a DP scheme, based on the general idea formulated by Bellman (1954).

For given values of $\bar{\chi}$ and j^{start}, the DP consists of $|\Omega| + 1$ stages $p = 0, \ldots, |\Omega|$, each containing one state $\Theta_p = \{j \in J : j^{\text{start}} \leq j < j^{\text{start}} + p \vee 1 \leq j < p + j^{\text{start}} - |\Omega|\}$, denoting the set of items that have already been considered in the partial solution. Starting from initial stage $p = 0$ with state $\Theta_p = \emptyset$, a successor stage $p' > p$ is reached by adding the set $\Theta_{p'} \setminus \Theta_p$ to Θ_p, indicating that items $j \in \Theta_{p'} \setminus \Theta_p$ are picked in the same tour $\omega_{p'}$. A transition is only feasible if $\sum_{j \in \Theta_{p'} \setminus \Theta_p} q_j \leq Q$, i.e., if the capacity is not exceeded.

Let $V(\Theta_p)$ be the set of states from which a feasible transition to state Θ_p exists. The optimal objective value $h^*(\Theta_p)$ of the partial solution in state Θ_p can then be calculated recursively as

$$h^*(\Theta_p) = \min_{\Theta' \in V(\Theta_p)} \left\{ h^*(\Theta') + g\left(\Theta_p \setminus \Theta', \bar{\chi}\right) \right\},$$

with $h^*(\emptyset) = 0$. The objective value of a complete solution in final state $\Theta_{|\Omega|}$ is also the best objective value for the given values of $\bar{\chi}$ and j^{start}. We can obtain the corresponding assignment by backward recovery along the best path. To complete the proposed DP, we increment $\bar{\chi}$ and j^{start} in the same manner as for the SA and save the overall best obtained solution.

The way the DP is set up, it is guaranteed to always find a solution that is at least as good as the sweep algorithm's solution. Concerning the time complexity, for given values of $\bar{\chi}$ and j^{start}, there are $O(n^2)$ transitions. Note that, by careful implementation, it is possible to calculate the objective contribution of all $O(n)$ successors of a state in $O(n)$ time. Let $\sigma = \frac{l - \frac{w}{2}}{\chi^{\text{step-size}}}$ be the number of increments

considered for $\bar{\chi}$. Then the asymptotic runtime of the complete procedure (including varying $\bar{\chi}$ and j^{start}) is bounded by $O(\sigma \cdot n^3)$.

Example Consider the same example as for the SA in Sect. 4.2.1. For the given initial item $j^{\text{start}} = 1$ and the depot location $\bar{\chi} := 0$, Fig. 6 depicts the dynamic programming graph including the optimal path recovered from backtracking. The objective of the DP's solution is 22.63, which is 23.10 % below the one of the SA's solution.

5 Numerical experiments and analysis

5.1 Generating instances for the computational tests

To evaluate our proposed solution procedures and gain some managerial insights, we generate problem instances based on our observations in practice and the assumptions presented by Glock and Grosse (2012). The following section describes how the instances are generated.

We consider U-layouts with two different capacities, either 44 stillages (cf., Glock and Grosse 2012) or 88 stillages. The layout of an instance is defined by setting n and m, the number of stillages in one horizontal and in one vertical row. Since both Glock and Grosse (2012) and Diefenbach and Glock (2019) found that narrow U-shapes are advantageous, we consider the layouts of $(n, m) = (10, 2)$, $(n, m) = (9, 4)$, and $(n, m) = (8, 6)$ if the U contains 44 stillages, and the layouts of $(n, m) = (21, 2)$, $(n, m) = (20, 4)$, and $(n, m) = (19, 6)$ if the U contains 88 stillages. In accordance with Glock and Grosse (2012), we set the measurements of the stillages to $w = 1.3$ m and the gap between the stillages to $s = 0.05$ m.

For the item demands, we assume a 20/60-Pareto distribution (cf., Bender 1981), i.e., 20% of the items are responsible for 60% of demand. Based on the distribution, we randomly draw $|\Omega|$ items to be picked for each instance. For the instances with 44 stillages, we set either $|\Omega| = 10$ or $|\Omega| = 15$; for the instances with 88 stillages,

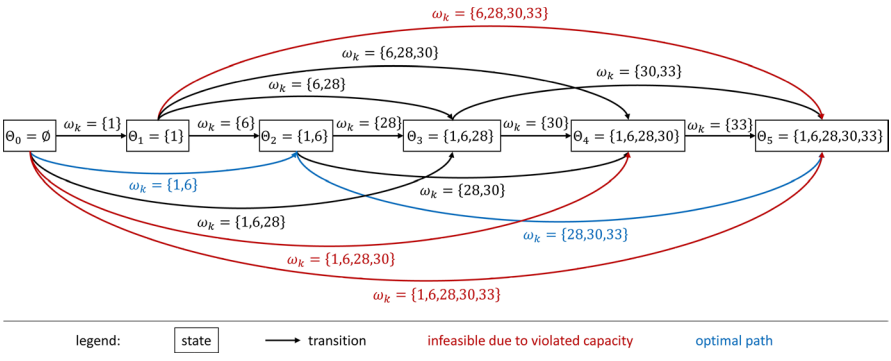


Fig. 6 Dynamic programming graph for the example of Sect. 4.2.1

we set either $|\Omega| = 30$ or $|\Omega| = 60$. Items weights q_j are drawn randomly from the set $\{1, \ldots, 5\}$ and the picker capacity is set to $Q = 15$ (cf., Glock and Grosse 2012). Finally, we assign the items to stillages in a random manner, and set $v = 3$ (cf., Glock and Grosse 2012) as the default value for the penalty to move the depot. Note that we will deviate from the default settings for v and the default item assignment in later experiments.

We generate ten instances for each setting. All instances are labeled as follows: "number of stillages"-(n, m)-|Ω|-*running index*. All instances are available for download at https://doi.org/10.5281/zenodo.4671870.

5.2 Computational performance

This section investigates the performance of the three proposed solution procedures, namely CBD, SA, and DP. All testing is performed on an Intel Core i7-3631QM CPU @ 2.20 GHz and with 8 GB of RAM. All algorithms are implemented in C#, and CPLEX (version 12.10) is used as a default solver for the master model of the proposed CBD at default settings. The maximum runtime is set to 3600 s (1 h). The numerical precision is set to $\mu = \epsilon = \chi^{\text{step-size}} = 0.01$ for all procedures and respective parameters.

In a preliminary test we found that the influence of the exact layout (i.e., the values of n and m for a given number of stillages) on the algorithmic performance is negligible. We therefore restrict the performance tests to the instances with $m = 4$. The results of the performance test are summarized in Table 2.

The computational experiments show that the exact CBD is able to solve all instances with $|\Omega| \leq 15$ in acceptable runtime. However, not surprisingly given the NP-hard, non-linear nature of the problem, for instances with $|\Omega| \geq 30$ the procedure does not terminate within the maximum runtime of 3600 s and the gaps to the lower bounds of the MP (as reported by CPLEX) remain high at the point of termination. Furthermore, both heuristic approaches yield better solutions for $|\Omega| \geq 30$. Especially for larger instances, the time share spent solving the MP (which is above 97% for $|\Omega| \geq 30$) indicates that the MP is the procedure's bottleneck; the SP appears to be sufficiently fast to solve.

The SA runs very fast with a runtime of below 1 s even for $|\Omega| = 60$. For $|\Omega| \geq 15$, where we know the optimum thanks to CBD, the optimality gaps are generally low. For most instances, the optimality gaps are even zero. However, for five instances they are above 2% and can be as high as about 5%. Nonetheless, we can draw the conclusion that the relatively simple sweep algorithm works quite well for U-shaped picking areas. This is certainly good news for practitioners, who may find it easy to implement such a routing scheme.

To further improve the SA, our approach based on dynamic programming yields even better solutions. It completely closes the optimality gap in all small and most medium size instances. Even though the DP is a little slower than the SA, it sill runs fast, with a runtime of at most 10 s for $|\Omega| = 60$, which is clearly sufficient for practical application.

Table 2 Computational performance tests

Instances	Benders decomposition								Sweep algorithm				Dynamic programming			
label	Value	LB CPLEX	Chi	Runtime (in s)	Time MP (in %)	Time SP (in %)	Opt cuts	Prog cuts	Value	Chi	Gap (in %)	Runtime (in s)	Value	Chi	Gap (in %)	runtime (in s)
44-(9x4)-10-01	34.34	34.34	3.87	0.58	75.22	24.78	3	34	34.34	3.87	0.00	0.01	34.34	3.87	0.00	0.76
44-(9x4)-10-02	37.40	37.40	9.13	0.30	80.20	19.80	10	23	37.40	9.13	0.00	0.01	37.40	9.13	0.00	0.29
44-(9x4)-10-03	44.55	44.55	10.83	1.40	60.73	39.27	12	105	45.50	10.10	2.13	0.01	44.55	10.83	0.00	0.19
44-(9x4)-10-04	37.26	37.26	8.75	0.45	90.22	9.78	5	18	37.26	8.75	0.00	0.01	37.26	8.75	0.00	0.22
44-(9x4)-10-05	32.72	32.72	1.74	0.30	79.87	20.13	6	25	32.72	1.74	0.00	0.01	32.72	1.74	0.00	0.66
44-(9x4)-10-06	38.99	38.99	3.07	0.68	57.46	42.54	7	66	38.99	3.07	0.00	0.01	38.99	3.07	0.00	0.58
44-(9x4)-10-07	43.11	43.11	5.47	1.10	58.00	42.00	11	130	43.99	6.54	2.04	0.01	43.11	5.47	0.00	0.35
44-(9x4)-10-08	37.45	37.45	9.80	0.58	75.56	24.44	6	55	37.91	9.56	1.23	0.01	37.45	9.80	0.00	0.42
44-(9x4)-10-09	34.40	34.40	11.69	0.42	81.67	18.33	5	24	35.63	11.55	3.58	0.01	34.40	11.69	0.00	0.28
44-(9x4)-10-10	34.35	34.35	8.19	0.72	61.42	38.58	8	79	34.35	8.19	0.00	0.01	34.35	8.19	0.00	0.19
Mean	37.46	37.46	7.25	0.65	72.04	27.97	7.3	55.9	37.81	7.25	0.90	0.01	37.46	7.25	0.00	0.39
44-(9x4)-15-01	52.42	52.42	9.67	39.27	83.84	16.16	12	1239	52.42	9.67	0.00	0.02	52.42	9.67	0.00	0.64
44-(9x4)-15-02	57.43	57.43	5.12	54.75	85.42	14.58	19	1163	57.43	5.12	0.00	0.03	57.43	5.12	0.00	0.35
44-(9x4)-15-03	53.81	53.81	6.84	99.48	87.21	12.79	12	2096	56.41	6.98	4.83	0.02	56.41	6.98	4.83	0.57
44-(9x4)-15-04	47.42	47.42	5.75	12.66	84.28	15.72	9	381	47.42	5.75	0.00	0.02	47.42	5.75	0.00	0.88
44-(9x4)-15-05	42.47	42.47	3.85	8.07	72.59	27.41	7	472	42.47	3.85	0.00	0.02	42.47	3.85	0.00	1.38
44-(9x4)-15-06	49.35	49.35	6.00	19.14	94.25	5.75	10	220	49.35	6.00	0.00	0.03	49.35	6.00	0.00	0.44
44-(9x4)-15-07	43.67	43.67	3.75	13.14	82.69	17.31	10	412	43.67	3.75	0.00	0.02	43.67	3.75	0.00	1.03
44-(9x4)-15-08	46.98	46.98	6.59	7.54	95.08	4.92	7	78	46.98	6.59	0.00	0.02	46.98	6.59	0.00	0.58
44-(9x4)-15-09	53.01	53.01	7.96	79.91	91.13	8.87	15	1080	53.01	7.96	0.00	0.02	53.01	7.96	0.00	0.40
44-(9x4)-15-10	45.75	45.75	10.06	16.82	69.38	30.62	9	829	46.85	10.03	2.40	0.02	45.75	10.06	0.00	0.99
Mean	49.23	49.23	6.56	35.08	84.59	15.41	11.0	797.0	49.60	6.57	0.72	0.02	49.49	6.57	0.48	0.73
88-(20x4)-30-01	148.87	84.97	16.23	–	96.79	3.21	19	2732	145.65	16.32	−2.16	0.17	144.03	16.31	−3.25	4.73
88-(20x4)-30-02	148.01	84.01	17.24	–	97.67	2.33	21	1750	140.87	17.42	−4.82	0.20	140.87	17.42	−4.82	2.94

Table 2 (continued)

Instances	Benders decomposition								Sweep algorithm				Dynamic programming			
label	Value	LB CPLEX	Chi	Runtime (in s)	Time MP (in %)	Time SP (in %)	Opt cuts	Prog cuts	Value	Chi	Gap (in %)	Runtime (in s)	Value	Chi	Gap (in %)	runtime (in s)
88-(20x4)-30-03	146.74	82.44	14.07	–	98.27	1.73	14	1703	139.27	14.10	− 5.09	0.16	139.27	14.10	− 5.09	3.54
88-(20x4)-30-04	177.22	93.24	16.84	–	97.17	2.83	28	2147	167.76	15.78	− 5.34	0.22	166.42	16.58	− 6.09	2.18
88-(20x4)-30-05	145.45	79.07	11.90	–	98.21	1.79	20	1440	142.00	10.90	− 2.37	0.20	138.11	11.80	− 5.05	2.49
88-(20x4)-30-06	171.98	86.04	12.46	-	98.40	1.60	15	1261	159.75	12.94	− 7.11	0.20	159.75	12.94	− 7.11	2.70
88-(20x4)-30-07	143.08	81.31	12.78	–	98.10	1.90	32	1699	134.48	12.97	− 6.01	0.17	134.48	12.97	− 6.01	6.72
88-(20x4)-30-08	151.92	97.99	13.44	–	99.23	0.77	9	665	147.22	13.53	− 3.09	0.17	147.22	13.53	− 3.09	4.02
88-(20x4)-30-09	154.80	93.21	15.48	–	97.85	2.15	26	1704	148.02	14.35	− 4.38	0.19	148.02	14.35	− 4.38	2.83
88-(20x4)-30-10	144.04	77.70	13.81	–	99.42	0.58	18	457	137.94	13.54	− 4.23	0.19	137.94	13.54	− 4.23	3.52
Mean	153.21	86.00	14.43	–	98.11	1.89	20.2	1555.8	146.30	14.19	− 4.46	0.19	145.61	14.35	− 4.91	3.57
88-(20x4)-60-01	297.23	82.28	14.86	–	98.31	1.69	11	388	255.57	15.84	− 14.02	0.74	255.55	15.83	− 14.02	6.24
88-(20x4)-60-02	305.78	74.83	15.12	–	99.33	0.67	9	132	237.18	15.20	− 22.43	0.72	234.57	15.21	− 23.29	8.32
88-(20x4)-60-03	288.54	69.71	10.56	–	96.20	3.80	14	345	234.99	11.67	− 18.56	0.72	231.58	11.97	− 19.74	10.43
88-(20x4)-60-04	376.85	89.00	12.97	–	96.56	3.44	16	645	285.32	15.20	− 24.29	0.76	283.06	14.81	− 24.89	6.31
88-(20x4)-60-05	309.40	83.28	9.66	–	93.53	6.47	14	1124	239.07	12.25	− 22.73	0.68	237.95	12.65	− 23.09	6.20
88-(20x4)-60-06	341.57	94.20	13.41	–	99.71	0.29	17	67	266.08	15.12	− 22.10	0.76	263.65	15.58	− 22.81	5.77
88-(20x4)-60-07	339.56	98.19	12.69	–	98.29	1.71	10	106	270.38	14.15	− 20.37	0.75	268.30	14.28	− 20.99	5.05
88-(20x4)-60-08	334.89	94.83	13.41	–	97.60	2.40	15	533	275.25	15.14	− 17.81	0.84	274.08	14.86	− 18.16	5.17
88-(20x4)-60-09	310.37	79.73	12.35	–	98.15	1.85	16	414	240.26	13.27	− 22.59	0.78	239.94	13.24	− 22.69	5.98
88-(20x4)-60-10	293.21	77.76	13.19	–	98.73	1.27	25	236	252.90	13.95	− 13.75	0.74	251.66	13.69	− 14.17	9.40
Mean	319.74	84.38	12.82	–	97.64	2.36	14.7	399.0	255.70	14.18	− 19.86	0.75	254.03	14.21	− 20.39	6.89

value = objective value; LB CPLEX = lower bound as reportet by CPLEX at the point of termination; chi = depot position of the best solution; runtime = runtime until termination; time MP =share of the runtime spent solving the master problem; time SP share of the runtime spent solving the slave problem; opt cuts = number of added optimality cuts; prog cuts = number of added progression cuts; gap = relative gap to the Benders Decomposition's best upper bound

– = the procedure was terminated because the runtime limit of 3600 s had been reached

5.3 Effects of having a movable depot

In our default settings, we assume a movable depot that can be positioned at any position $0 \leq \chi \leq l$ and set $v = 3$, which means that moving the depot is six times faster than travel during order picking (cf., Sect. 3.1). However, in practice, companies may find it easier to define a fixed position for the depot. Furthermore, the depot could be set up and moved in a different way. For example, the depot could be moved by a forklift or a manual hand lift truck. Depending on this, the factor v may vary.

This section investigates the effects of having a movable or fixed depot as well as the effects of different values for v. For the experiment, we use the same instances as in the performance evaluation (cf., Table 2). We solve these instances using the DP approach assuming either a movable or a fixed depot. We test four different fixed depot positions at $\chi \in \{0, 0.25 \cdot l, 0.5 \cdot l, 0.75 \cdot l\}$. Furthermore, we test eight different values $v \in \{1, 3, 5, 7, 9, 11, 13, 15\}$. Figure 7 shows the mean results summarized for each instance size.

Obviously, having a stationary depot can never be better than having a movable depot, since the movable depot can always assume the position of the stationary depot. However, the benefits of having a movable depot strongly depend on the size of the U-zone and the length of the picklist, as the results in Fig. 7 show. For larger U-zones and picklists, a stationary depot at $\chi = 0.5 \cdot l$ is almost as efficient as a movable one. For smaller U-zones and picklists, the benefits of the movable depot are more significant. For the 44-(n, m)-10-instances, the gap in the objective value between the movable depot case and the best fixed depot case was on average (over all different values of v) 6.37 %. For the 44-(n, m)-15-instances, the 88-(n, m)-30-instances, and 88-(n, m)-60-instances, the gaps were 3.05 %, 1.60 %, and 0.78 %, respectively.

Given these finding, a practically relevant approach might be to limit possible depot positions to a few discrete locations. As indicated in Sect. 4.1, this would make PRP-UA non-exact and turn it into a type of capacitated vehicle routing problem. While the problem remains NP-hard, the capacitated vehicle routing problem is well researched with various exact and heuristic solution procedures being readily available (cf., Ralphs et al. 2003). Such an approach is beyond the scope of this paper, but may be interesting to investigate for future research.

Overall, $\chi = 0.5 \cdot l$, i.e., placing the depot in the middle of the U, performs best if a stationary position is enforced; only for the 44-(n, m)-10-instances, $\chi = 0.75 \cdot l$ is better. Fixing the depot at the entrance of the U-zone, i.e., at $\chi = 0$, is the worst option according to our experiments and results in objective values that are significantly higher than for a movable depot or any of the other fixed positions.

Except for the case where the depot is fixed at $\chi = 0$, the objective values decrease with increasing values of v. They do, however, not decrease linearly but asymptotically approach a constant value due to the fact that v influences the objective anti-proportionally. Hence, there are diminishing returns for increasing v.

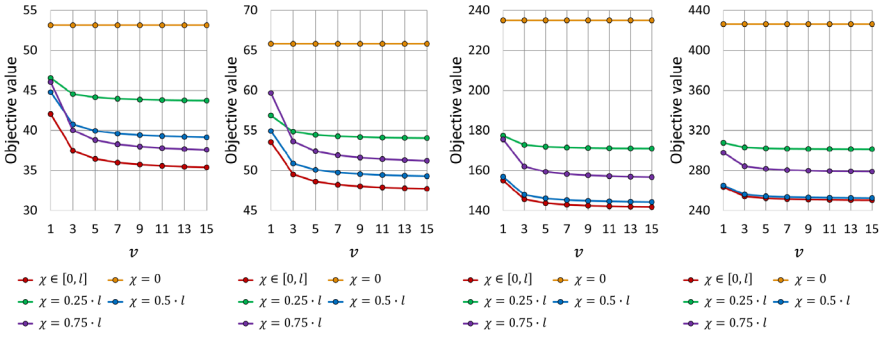
For the movable depot, not only does v affect the objective value, but also the optimized depot position. Figure 8 shows how the mean, maximum and minimum depot positions vary with the value of v. Comparing the sub-figures of the different

instance sizes shows that the ideal depot position varies less for bigger U-zones and larger picklists (i.e., the maximum and minimum positions are closer to each other). The effect of v on the spread is negligible. Furthermore, with increasing values of v, the ideal depot position increases, asymptotically approaching a constant value. Again, this can be explained by the anti-proportional effect of v in the objective function. For increasing values of v, the effort to move the depot approaches zero, such that χ approaches the value that would be ideal to solely minimize the tour length during order picking.

5.4 Effects of storage assignment policies

So far, we have only considered instances with a random assignment of items to stillages. However, the allocation of items to stillages significantly influences the performance of a warehouse with regard to the order picking process. Furthermore, in practice, the (time-)effort associated with exchanging stillages on different levels (upper and lower) needs to be considered in addition to travel distances. Picking items from stillages requires replenishments of empty stillages over time. In the industry case described in Glock and Grosse (2012), stillages are stacked one atop the other, and exchanging stillages in the upper row is less time-consuming than exchanging stillages on the lower level.

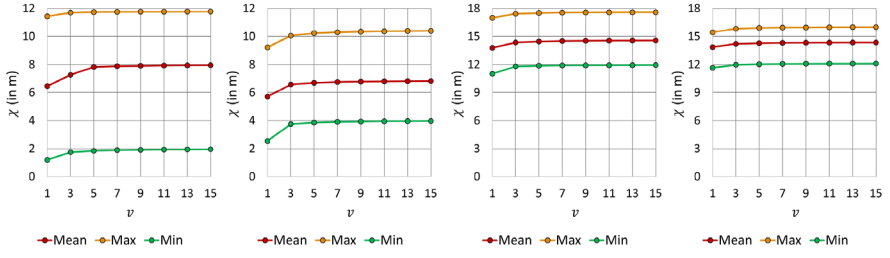
Typically, this problem is addressed with simple rules-of-thumb in both practice and research (cf., Sect. 2). Among various storage assignment methods that have been discussed in the literature, Glock and Grosse (2012) use a dedicated storage assignment policy where each item has a dedicated location in the U-zone, which is kept constant for a set of orders. In this paper, we develop a new assignment policy that we term "radial assignment" and compare this new policy to the four storage assignment policies developed by Glock and Grosse (2012). The storage assignment policies studied in this paper are briefly explained in the following:



(a) 44-(n, m)-10-instances **(b)** 44-(n, m)-15-instances **(c)** 88-(n, m)-30-instances **(d)** 88-(n, m)-60-instances

Fig. 7 Best found objective values for different values of v

1. *Random assignment*: Items are assigned randomly to the stillages. Previous work used this policy as a benchmark for evaluating other assignments, which is also done in this paper. It was found that the random storage assignment usually leads to the highest average travel distance.
2. *Horizontal assignment*: First, items are sorted in descending order of their pick frequency. More frequently requested items are assigned to locations along the stillages on the left side of the zone referred to as zone A in Fig. 9a. Once the stillages in zone A have been filled, items are assigned to zone B, then to zone C etc. Glock and Grosse (2012) showed that a horizontal assignment is especially beneficial for wide aisles, as it helps to avoid unproductive crossings of the aisle.
3. *Vertical assignment*: Items are again sorted in descending order of their pick frequency. Items are then assigned to both parallel aisles from left to right, as can be seen in Fig. 9b. First, stillages in zone A are filled, followed by zones B, C etc. This type of assignment was found to produce better results for longer and narrower zones in earlier research. In such cases, the picker saves more travel distance by crossing the zone to pick items, instead of continuing along the same row of stillages.
4. *Upper/Lower assignment*: This storage assignment policy exploits the difference in the effort associated with exchanging upper and lower level stillages by assigning frequently-required products (that are assumed to result in frequent exchanges of stillages) to upper level stillages (Sect. 5.4). Therefore, items are again sorted in descending order of their pick frequency and then assigned to the upper level first before the lower level stillages are filled. Figure 9c illustrates the preference of item allocation following the order: A,B,..,E,F. Glock and Grosse (2012) reported that this policy is especially beneficial in case the effort of exchanging stillages differs strongly between upper and lower level stillages.
5. *Radial assignment*: This new storage assignment policy is introduced in this paper, where again items are sorted in decreasing order of pick frequency. Frequently required items are allocated radially closer to the depot location. However, the optimal depot location is usually unknown when items are assigned. Therefore, we must assume a given depot location for the radial assignment policy. In this paper, we will investigate four alternative assumed depot locations $\chi \in \{0, 0.25 \cdot l, 0.5 \cdot l, 0.75 \cdot l\}$. We emphasize that the depot location is



(a) 44-(n, m)-10-instances **(b)** 44-(n, m)-15-instances **(c)** 88-(n, m)-30-instances **(d)** 88-(n, m)-60-instances

Fig. 8 Best found depot positions for different values of v

only assumed to determine the assignment; for the optimization, χ is still freely adjustable. Figure 9d depicts the alphabetical sequence of filling the stillages for the assumed depot location $\chi = 0$. It is expected that this will reduce the distance covered at the beginning and end of each tour, and should lead to good results for larger zones.

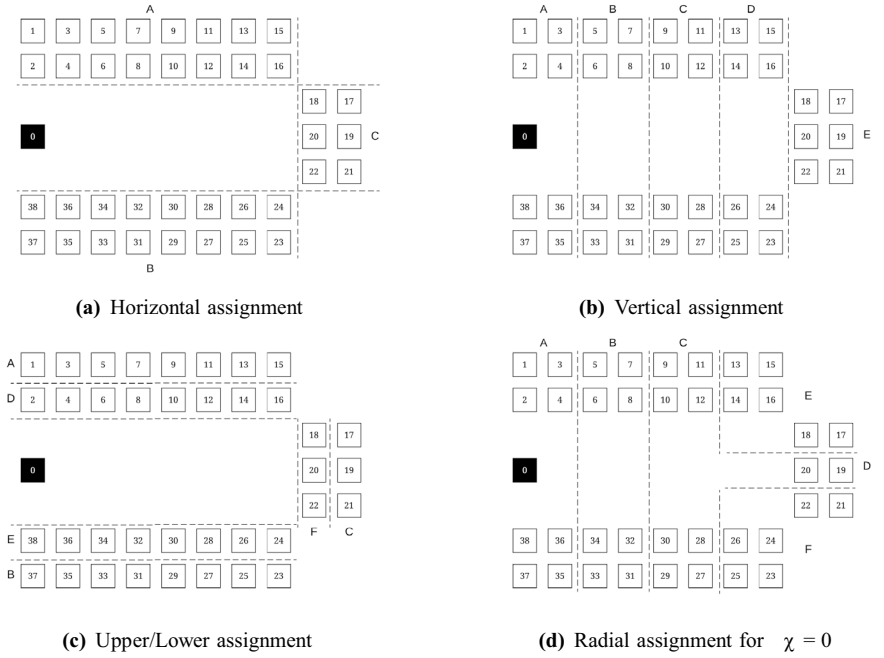
In the following, we investigate the effects of these storage assignment policies. We generate new instances based on the default instances of the performance evaluation (cf., Table 2), by assigning the items according to the presented storage assignment policies instead of randomly. I.e., we neither change the picklists nor the item weights, but only the assignment of items to stillages. Furthermore, we investigate the effect of different layouts, namely $(n, m) \in \{(10, 2), (9, 4), (8, 6)\}$ for the instances with 44 stillages and $(n, m) \in \{(21, 2), (20, 4), (19, 6)\}$ for the instances with 88 stillages.

To evaluate the effort for exchanging stillages, we assume exchanging an upper stillage takes 3 min and exchanging a lower stillage takes 5 min. Assuming a picker has a travel speed of 1.2 $\frac{\text{m}}{\text{s}}$, this means that exchanging an upper (a lower) stillage causes an increase in the objective of $3 \cdot 60 \cdot 1.2 = 216$ ($5 \cdot 60 \cdot 1.2 = 360$), since the objective value is measured in the normalized time the picker requires to travel 1 m. However, stillages only need to be exchanged once they are empty. To account for this, we assume all stillages have an equal capacity of Q^{stillage} and need to be exchanged once the capacity is depleted. We can then calculate the time-share per picklist to exchange a stillage. For example, assume a given picklist where item j is required with q_j and item j is stored in an upper stillage. Based on our assumptions, this would result in an additional objective value of $\frac{q_j}{Q^{\text{stillage}}} \cdot 216$ for the respective item. We evaluate the exchange effort for two different stillage capacities $Q^{\text{stillage}} \in \{50, 100\}$ and solve all instances with the proposed DP approach. The results of the experiment are summarized in Fig. 10.

Figure 10 shows the mean objective values for (a) the 44-(n, m)-10-, (b) the 44-(n, m)-15-, (c) the 88-(n, m)-30-, and (d) the 88-(n, m)-60-instances in separate bar charts. Each bar consists of three stacked bars. The lowest bar shows the objective value if the exchange effort is not considered, the middle bar shows the objective value if $Q^{\text{stillage}} = 100$, and the top bar if $Q^{\text{stillage}} = 50$. Note that all bars are measured form the bottom line (i.e., 0).

Concerning the efficiency of different layouts, our results confirm the findings of Glock and Grosse (2012) and Diefenbach and Glock (2019) in that narrow layouts are beneficial. This is the case for all tested instances. For the most efficient storage assignment policies, the results show that the upper/lower assignment is best if the exchange effort is taken into account. This is already true for $Q^{\text{stillage}} = 100$ but especially evident for $Q^{\text{stillage}} = 50$, the reason being that upper/lower assignment is especially designed to minimize the exchange effort. Interestingly, upper/lower assignment is the worst assignment policy (except for random storage) if solely the pick effort is considered. In that case, the newly proposed radial assignment with $\chi \in \{0.5 \cdot l, 0.75 \cdot l\}$ performs best.

Concerning the efficiency of the upper/lower assignment if the exchange effort is considered, we note that the objective value is strongly dependent on the effort for



(a) Horizontal assignment

(b) Vertical assignment

(c) Upper/Lower assignment

(d) Radial assignment for $\chi = 0$

Fig. 9 Storage assignment policies for U-shaped order picking zones

exchanging stillages and the stillage capacity Q^{stillage}, where higher capacities cause lower exchange efforts. While we only considered $Q^{\text{stillage}} \in \{50, 100\}$ in our experiment, higher capacities are also possible. By extrapolating our results, we can calculate the theoretical break-even stillage capacities, from where on the upper/lower assignment is outperformed by the (former) second best assignment policy, namely radial assignment with $\chi = 0.5 \cdot l$ or $\chi = 0.75 \cdot l$. These break-even stillage capacities are given in Table 3.

Finally, we investigate the influence of the items' demand skewness on the storage assignment policies' efficiency. For the previous experiments, we assumed that item demand follows a 20/60-Pareto distribution (cf., Sect. 5.1). However, more and less skewed demand distributions are also common in practice. Therefore, we also consider a 20/80- and a 20/40-Pareto distribution in the following. For each demand distribution and instance size, we generated ten new instances, which we solved for all layout options and storage assignment policies (except for random storage, where the item demand distribution is insignificant). The results are given in Table 4, which shows the relative change (in %) of the mean objective value (including the exchange effort for stillages, where we set $Q^{\text{stillage}} = 100$) compared to the default 20/60-Pareto distribution. The results are summarized over all layout options.

Table 4 generally indicates that more skewed demand distributions result in lower objective values for all storage assignment policies. This was to be expected due to all considered storage assignment policies being demand-based, meaning they are specifically designed to make use of a skewed item demand distribution. The benefits

of a more skewed demand distribution are primarily relevant if the U-zone's capacity is large compared to the order size $|\Omega|$. Otherwise, the effect is marginal. Moreover, if we compare the best storage assignment policies from the previous tests, namely upper/lower assignment and radial assignment with $\chi \in \{0.5 \cdot l, 0.75 \cdot l\}$, we gain an interesting insight. The demand skewness has a much lower influence for the former than for the latter. This indicates that for highly skewed demand, the radial assignment with $\chi \in \{0.5 \cdot l, 0.75 \cdot l\}$ becomes more beneficial, while for lower demand skewness, the upper/lower assignment is superior.

6 Conclusion

This paper considers the order picker routing problem in U-shaped order picking zones. The assumed order picking zones are built from stillages stacked one atop another and arranged in a U-shape with a movable depot at its center-line, where items are dropped off during order picking. We show that the problem is NP-hard and develop the first exact solution procedure, which is based on combinatorial Benders decomposition. Furthermore, we develop a new heuristic solution procedure based on dynamic programming by expanding the concept of a heuristic sweep algorithm from the literature, such that the new heuristic is guaranteed to find solutions that are at least as good as the ones of the sweep algorithm.

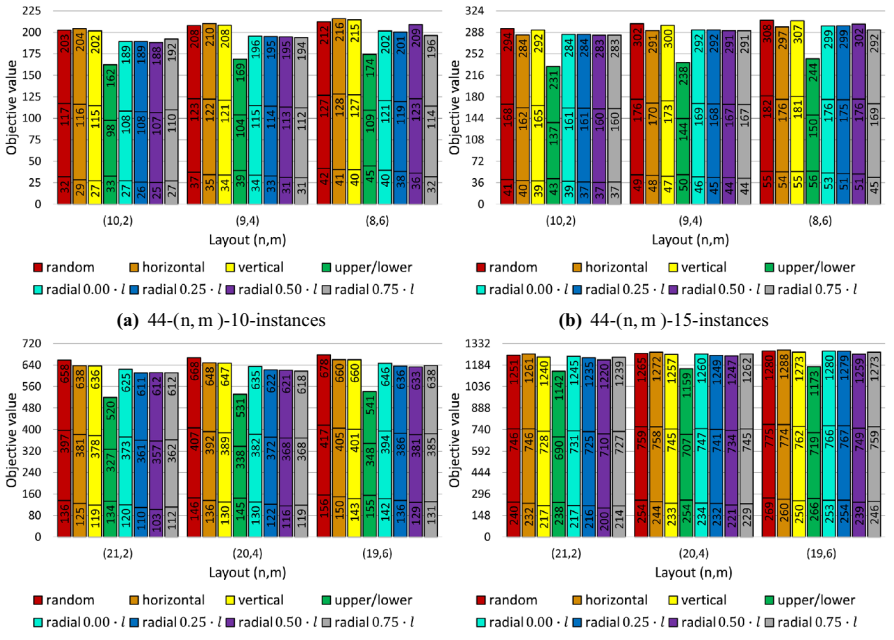


Fig. 10 Effects of different storage and layouts on the efficiency of U-shaped order picking zones

In a computational study, we compare the performance and runtime of our two newly proposed algorithms and the sweep algorithm. We find that the exact procedure is sufficiently fast to solve small problem instances in acceptable runtime but struggles with larger ones. Both the sweep algorithm and the dynamic programming approach run very fast with at most ten seconds runtime even for large instances. Comparing the results of the two heuristics to the exact solutions, we find the optimality gaps are very small at below 1% on average and zero for many instances. Comparing both heuristics, we find that the results of the newly developed dynamic programming approach are on average 0.55% better than the ones of the sweep algorithm.

Beyond that, we investigate the effects of having a movable depot compared to a static depot and the influence of different storage assignment policies, where we suggest a new policy, called radial assignment, and compare it to various policies from the literature. We derive the following managerial insights:

- A movable depot is always favorable compared to a stationary depot. However, for larger U-zones and longer picklists, a stationary depot in the middle of the U-zone is almost as beneficial as having a movable one. In our experiments, objective values were between 0.78% (for large U-zones) and 6.37% (for small U-zones) higher if the depot was fixed compared to the movable depot case.
- Having a stationary depot directly at the entrance of the U-zone is the worst option by a large margin in all of our experiments. It is therefore not advisable.
- In our experiments, the newly proposed radial assignment policy minimizes the effort for order picking, while the upper/lower assignment policy minimizes the combined effort for order picking and exchanging empty stillages. However, the latter is strongly dependent on the assumed stillage capacities. For high stillage capacities, radial assignment remains the best policy even if the effort for exchanging empty stillages is considered. Moreover, radial assignment appears superior for highly skewed item demand distributions, while upper/lower assignment is beneficial for lower demand skewness.
- In our experiments, narrow U-zones are advantageous, which validates the results from the literature.
- The relatively simple sweep algorithm, adapted to U-shaped picking zones with a movable depot, performs quite well. This may be good news for practitioners who do not wish to implement complicated optimization logic.

We base our problem definition on some assumptions. Among the more critical ones are the assumption that the depot is point-like and that pickers travel in Euclidean paths, while actual human walk paths often resemble an elongated S-shape (cf., Diefenbach and Glock 2019). We regard the resulting errors to be comparatively small. Nevertheless, future research may aim to improve or refine our assumptions.

Furthermore, we consider storage assignment only in a rudimentary way by comparing various assignment policies. However, as our computational study shows, storage assignment can greatly influence a U-zone's efficiency. Future research may investigate the possibility to store multiple kinds of items per stillage, which is sometimes found in practice. Our solution procedures are already

Table 3 Stillage capacity from where on the radial assignment is more efficient than the upper/lower assignment

Instances	44-(n,m)-10-instances			44-(n,m)-15-instances			88-(n,m)-30-instances			88-(n,m)-60-instances		
Layout	(10,2)	(9,4)	(8,6)	(10,2)	(9,4)	(8,6)	(21,2)	(20,4)	(19,6)	(21,2)	(20,4)	(19,6)
Break-even at	223.60	197.03	137.96	475.29	487.81	281.00	199.37	217.00	225.37	151.62	183.70	211.81
Break-even with	Radial	Radial	Radial	Radial	Radial	Radial	Radial	Radial	Radial	Radial	Radial	Radial
	$0.50 \cdot l$	$0.75 \cdot l$	$0.75 \cdot l$	$0.50 \cdot l$	$0.75 \cdot l$	$0.75 \cdot l$	$0.50 \cdot l$	$0.75 \cdot l$	$0.75 \cdot l$	$0.50 \cdot l$	$0.50 \cdot l$	$0.50 \cdot l$

Table 4 Influence of the demand skewness on the storage assignment policies' efficiency

Instances		44-(n,m)-10-instances			44-(n,m)-15-instances			88-(n,m)-30-instances			88-(n,m)-60-instances		
Demand skewness		80/20	60/20	40/20	80/20	60/20	40/20	80/20	60/20	40/20	80/20	60/20	40/20
Storage assignment policy	Horizontal	− 4.6	0.0	4.8	1.7	0.0	2.6	− 3.2	0.0	2.5	− 0.1	0.0	0.2
	Vertical	− 3.3	0.0	8.9	− 0.2	0.0	2.0	− 3.1	0.0	1.8	0.1	0.0	1.4
	Upper/lower	− 2.5	0.0	1.7	2.6	0.0	0.2	− 2.2	0.0	0.4	0.7	0.0	0.0
	Radial 0	− 4.9	0.0	10.1	0.4	0.0	1.3	− 4.3	0.0	2.0	− 0.5	0.0	0.9
	Radial 0.25	− 2.4	0.0	9.5	0.4	0.0	1.3	− 3.3	0.0	4.8	− 0.5	0.0	1.1
	Radial 0.5	− 2.5	0.0	8.1	− 0.5	0.0	1.2	− 4.4	0.0	3.6	0.1	0.0	1.6
	Radial 0.75	− 4.0	0.0	8.7	0.6	0.0	2.3	− 4.5	0.0	3.7	− 0.2	0.0	0.3

suited for such scenarios, but we did not consider suitable storage assignment policies. Moreover, it could be advisable to consider a combined storage assignment and routing problem in the future. As the performance of our exact solution approach suggests, this is most likely only possible with heuristic solution approaches, although investigating stronger cuts might also present an interesting and promising opportunity to improve our BD's performance further in the future.

In this paper, we considered a fully manual system, as currently automation plays a subordinate role for U-shaped order picking zones. However, automation becomes increasingly important for order picking in general, where it has achieved significant performance increases in recent years (Jaghbeer et al. 2020). Looking into future developments for U-shaped picking areas, a logical step would be to automate depots, enabling them to (autonomously) relocate while the picker processes pick tours. Future research may investigate the benefits of such (semi-)automated systems and thereby encourage the development of suitable technologies.

With this paper being the most recent addition, U-shaped order picking zones have been increasingly studied in recent years, since they were first introduction by Glock and Grosse (2012). However, there has not yet been a comprehensive comparison between U-shaped layouts and conventional layouts with parallel shelves. Especially from a practical point of view, it might be desirable to have some guidelines about when which layout poses which benefits. The insight gathered in previous works and in this paper may spark future research into this topic.

Appendix

Proposition A.1 Given two points $P_1 = \left(x_1^p, y_1^p\right)$ and $P_2 = \left(x_2^p, y_2^p\right)$ with Cartesian coordinates, the inequality $\sqrt{2} \cdot D^e(P_1, P_2) - D^r(P_1, P_2) \geq 0$ always holds true.

Proof To simplify notation, we define $l_x^p = |x_1^p - x_2^p|$ and $l_y^p = |y_1^p - y_2^p|$ and $\frac{l_y^p}{l_x^p} = \phi \Leftrightarrow l_y^p = \phi \cdot l_x^p$. Using these definitions and the definitions of $D^e(P_1, P_2)$ and $D^r(P_1, P_2)$, it follows that $\sqrt{2} \cdot \sqrt{(1+\phi^2) \cdot \left(l_x^p\right)^2} - (1+\phi) \cdot l_x^p \geq 0$ must hold in order for Property A.1 to be true. Rearranging yields $\sqrt{2} \cdot \sqrt{(1+\phi^2)} - (1+\phi) \geq 0$ $\Rightarrow \sqrt{2} \cdot \sqrt{(1+\phi^2)} \geq (1+\phi) \quad \Rightarrow 2 \cdot (1+\phi^2) \geq (1+\phi)^2 \quad \Rightarrow \phi^2 - 2 \cdot \phi + 1 \geq 0$. Applying the binomial theorem finally yields the inequality $(\phi - 1)^2 \geq 0$, which is always true, since the left side is a quadratic term that cannot be smaller than zero. □

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availibility Statement The test data that support the findings of this study are openly available in Zenodo at https://doi.org/10.5281/zenodo.4671870.

References

Barachet L (1957) Letter to the editor-graphic solution of the traveling-salesman problem. Oper Res 5(6):841–845

Battini D, Glock CH, Grosse EH, Persona A, Sgarbossa F (2016) Human energy expenditure in order picking storage assignment: a bi-objective method. Comput Ind Eng 94:147–157

Bellman R (1954) The theory of dynamic programming. Bull Am Math Soc 60:503–515

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

Bureau of Labor Statistics, U. D. o. L. (2016). Occupational outlook handbook, hand laborers and material movers. http://www.bls.gov/ooh/transportation-and-material-moving/hand-laborers-and-material-movers.htm

Calzavara M, Glock CH, Grosse EH, Persona A, Sgarbossa F (2017) Analysis of economic and ergonomic performance measures of different rack layouts in an order picking warehouse. Comput Ind Eng 111:527–536

Calzavara M, Glock CH, Grosse EH, Sgarbossa F (2019) An integrated storage assignment method for manual order picking warehouses considering cost, workload and posture. Int J Prod Res 57(8):2392–2408

Çelik M, Süral H (2019) Order picking in parallel-aisle warehouses with multiple blocks: complexity and a graph theory-based heuristic. Int J Prod Res 57(3):888–906

Çelk M, Süral H (2014) Order picking under random and turnover-based storage policies in fishbone aisle warehouses. IIE Trans 46(3):283–300

Cergibozan Ç, Tasan AS (2019) Order batching operations: an overview of classification, solution techniques, and future research. J Intell Manuf 30(1):335–349

Chabot T, Lahyani R, Coelho LC, Renaud J (2017) Order picking problems under weight, fragility and category constraints. Int J Prod Res 55(21):6361–6379

Chen F, Xu G, Wei Y (2019) Heuristic routing methods in multiple-block warehouses with ultra-narrow aisles and access restriction. Int J Prod Res 57(1):228–249

Codato G, Fischetti M (2006) Combinatorial Benders' cuts for mixed-integer linear programming. Oper Res 54(4):756–766

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

Diefenbach H, Glock CH (2019) Ergonomic and economic optimization of layout and item assignment of a u-shaped order picking zone. Comput Ind Eng 138:106094

Eurostat (2016). Warehousing and transport support services statistics - nace rev. 2. http://ec.europa.eu/eurostat/statistics-explained/index.php/Business_economy_by_sector_-_NACE_Rev._2

Fang K, Wang S, Pinedo ML, Chen L, Chu F (2021) A combinatorial Benders decomposition algorithm for parallel machine scheduling with working-time restrictions. Eur J Oper Res 291(1):128–146

Füßler D, Boysen N, Stephan K (2019) Trolley line picking: storage assignment and order sequencing to increase picking performance. OR Spectrum 41(4):1087–1121

Gademann N, Velde S (2005) Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Trans 37(1):63–75

Garey M, Johnson D (1979) Computers and intractability: a guide to the theory of NP-hardness. WH Freeman, San Fransisco

Glock CH, Grosse EH (2012) Storage policies and order picking strategies in u-shaped order-picking systems with a movable base. Int J Prod Res 50(16):4344–4357
Glock CH, Grosse EH, Abedinnia H, Emde S (2019) An integrated model to improve ergonomic and economic performance in order picking by rotating pallets. Eur J Oper Res 273(2):516–534
Glock CH, Grosse EH, Elbert RM, Franzke T (2017) Maverick picking: the impact of modifications in work schedules on manual order picking processes. Int J Prod Res 55(21):6344–6360
Grosse EH, Glock CH, Ballester-Ripoll R (2014) A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions. Int J Oper Quant Manag 20(2):65–83
Grosse EH, Glock CH, Jaber M, Neumann W (2015) Incorporating human factors in order picking planning models: framework and research opportunities. Int J Prod Res 53(3):695–717
Grosse EH, Glock CH, Neumann WP (2017) Human factors in order picking: a content analysis of the literature. Int J Prod Res 55(5):1260–1276
Gu J, Goetschalckx M, McGinnis L (2007) Research on warehouse operation: a comprehensive review. Eur J Oper Res 177(1):1–21
Gue K, Meller R (2009) Aisle configurations for unit-load warehouses. IIE Trans 41(3):171–0182
Hanson R, Falkenström W, Miettinen M (2017) Augmented reality as a means of conveying picking information in kit preparation for mixed-model assembly. Comput Ind Eng 113:570–575
Henn S, Koch S, Gerking H, Wäscher G (2013) A u-shaped layout for manual order-picking systems. Logist Res 6(4):245–261
Henn S, Wäscher G (2012) Tabu search heuristics for the order batching problem in manual order picking systems. Eur J Oper Res 222(3):484–494
Hong S, Johnson AL, Peters BA (2012) Large-scale order batching in parallel-aisle picking systems. IIE Trans 44(2):88–106
Hooker JN (2007) Planning and scheduling by logic-based Benders decomposition. Oper Res 55(3):588–602
Huang D, Mao Z, Fang K, Yuan B (2021) Combinatorial Benders decomposition for mixed-model two-sided assembly line balancing problem. Int J Prod Res 1–27
Jaghbeer Y, Hanson R, Johansson MI (2020) Automated order picking systems and the links between design and performance: a systematic literature review. Int J Prod Res 58(15):4489–4505
Kress D, Müller D, Nossack J (2019) A worker constrained flexible job shop scheduling problem with sequence-dependent setup times. OR Spectrum 41(1):179–217
Kulak O, Sahin Y, Taner M (2012) Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. Flex Serv Manuf J 24(1):52–80
Lu W, McFarlane D, Giannikas V, Zhang Q (2016) An algorithm for dynamic order-picking in warehouse operations. Eur J Oper Res 248(1):107–122
Masae M, Glock CH, Grosse EH (2020a) Order picker routing in warehouses: a systematic literature review. Int J Prod Econ 224:107564
Masae M, Glock CH, Vichitkunakorn P (2020b) Optimal order picker routing in the chevron warehouse. IISE Trans 52(6):665–687
Masae M, Glock CH, Vichitkunakorn P (2021) A method for efficiently routing order pickers in the leaf warehouse. Int J Prod Econ 234:108069
Matusiak M, de Koster R, Kroon L, Saarinen J (2014) A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. Eur J Oper Res 236(3):968–977
Mowrey C, Parikh P (2014) Mixed-width aisle configurations for order picking in distribution centers. Eur J Oper Res 232(1):87–97
Muppani V, Adil G (2008) A branch and bound algorithm for class based storage location assignment. Eur J Oper Res 189(2):492–507
Otto A, Boysen N, Scholl A, Walter R (2017) Ergonomic workplace design in the fast pick area. OR Spectrum 39(4):945–975
Öztürkoğlu Ö, Gue K, Meller R (2014) A constructive aisle design model for unit-load warehouses with multiple pickup and deposit points. Eur J Oper Res 236(1):382–394
Pan JC-H, Shih P-H, Wu M-H (2015) Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega 57:238–248
Pansart L, Catusse N, Cambazard H (2018) Exact algorithms for the order picking problem. Comput Oper Res 100:117–127
Petersen C, Aase G (2004) A comparison of picking, storage, and routing policies in manual order picking. Int J Prod Econ 92(1):11–19

Petersen C, Aase G, Heiser DR (2004) Improving order-picking performance through the implementation of class-based storage. Int J Phys Distrib Logist Manag 34(7):534–544
Petersen C, Siu C, Heiser DR (2005) Improving order picking performance utilizing slotting and golden zone storage. Int J Oper Prod Manag 25(10):997–1012
Ralphs TK, Kopman L, Pulleyblank WR, Trotter LE (2003) On the capacitated vehicle routing problem. Math Program 94(2–3):343–359
Ratliff H, Rosenthal A (1983) Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Oper Res 31(3):507–521
Reyes J, Solano-Charris E, Montoya-Torres J (2019) The storage location assignment problem: a literature review. Int J Ind Eng Comput 10(2):199–224
Roodbergen K, Koster R (2001a) Routing methods for warehouses with multiple cross aisles. Int J Prod Res 39(9):1865–1883
Roodbergen K, Koster R (2001b) Routing order pickers in a warehouse with a middle aisle. Eur J Oper Res 133(1):32–43
Roodbergen K, Sharp G, Vis I (2008) Designing the layout structure of manual order picking areas in warehouses. IIE Trans 40(11):1032–1045
Roodbergen K, Vis I (2006) A model for warehouse layout. IIE Trans 38(10):799–811
Roodbergen K, Vis I, Taylor G Jr (2015) Simultaneous determination of warehouse layout and control policies. Int J Prod Res 53(11):3306–3326
Rushton A, Croucher P, Baker P (2014) The handbook of logistics and distribution management: understanding the supply chain. Kogan Page Publishers, New York
Scholz A, Henn S, Stuhlmann M, Wäscher G (2016) A new mathematical programming formulation for the single-picker routing problem. Eur J Oper Res 253(1):68–84
Tadumadze G, Emde S, Diefenbach H (2020) Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines. OR Spectrum 42(2):461–497
Theys C, Bräysy O, Dullaert W, Raa B (2010) Using a tsp heuristic for routing order pickers in warehouses. Eur J Oper Res 200(3):755–763
Tompkins J, White J, Bozer Y, Tanchoco J (2010) Facilities planning. Wiley, New York
Van Gils T, Caris A, Ramaekers K, Braekers K (2019) Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. Eur J Oper Res 277(3):814–830
Van Gils T, Ramaekers K, Caris A, de Koster RB (2018) Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. Eur J Oper Res 267(1):1–15
Vaughan T (1999) The effect of warehouse cross aisles on order picking efficiency. Int J Prod Res 37(4):881–897
Žulj I, Glock CH, Grosse EH, Schneider M (2018a) Picker routing and storage-assignment strategies for precedence-constrained order picking. Comput Ind Eng 123:338–347
Žulj I, Kramer S, Schneider M (2018b) A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. Eur J Oper Res 264(2):653–664