Picking Optimization in U-Shaped Corridors with a Movable Depot

Roberto Montemanni(✉), Agnese Cervino, and Francesco Lolli

University of Modena and Reggio Emilia, 42122 Reggio Emilia, Italy
{roberto.montemanni,francesco.lolli}@unimore.it

Abstract. We consider an order-picking system for a warehouse divided into corridors with two-layer shelves being arranged in the shape of a U in each corridor. Given an order in a corridor, the focus is on the optimization of the picking sequence and on locating the movable depot in the most convenient location. Two iterative algorithms based on constraint programming are proposed. Computational experiments position the new methods in the existing literature, showing that they are operatively effective.

We also show how allowing the depot to be allocated away from the central axis of the corridor can lead to substantial time savings, especially for small orders. This strategic option had not been considered in the previous literature, but can be easily implemented in modern warehouses.

Keywords: Warehouse management · U-shaped corridors · Picking optimization · Depot position · Constraint programming

1 Introduction

Statistics indicate that order picking accounts for 50% to 75% of the total costs to operate a warehouse [1]. Human pickers are still today carrying out picking activities in most of the scenarios, due to their still unmatchable flexibility [2,3]. However, in modern warehouses workers are supported by technology to ease their tasks, both in terms of smart vehicles to help them transporting goods, and devises with smart interfaces to drive them through the picking process. In such a case the benefit is twofold: the picker job becomes lighter both from a physical and intellectual viewpoints, while the company has an economical return associated with the enhanced efficiency. In this work we cover the use of mathematical modelling and optimization to provide pickers with efficient strategies for their picking tasks. The outcome of our approach are an order-based optimized picking sequencing and an optimized location within the corridor for the movable depot in which the picker is operating. All these information can be communicated to the workers through standard devices such as tablets.

The warehouse investigated is characterized by U-shaped corridors, where products are organized in stillages that can be stacked on top of each other for

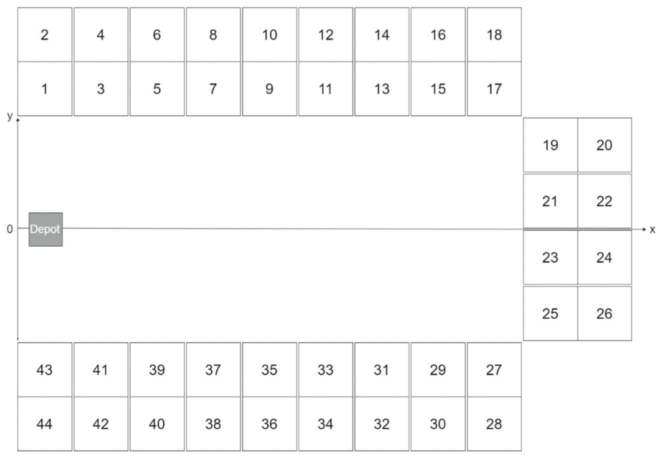

Fig. 1. Example of a U-shaped corridor with 44 stillages

a total of two layers. An example of such a corridor layout is provided in Fig. 1, where the dark grey square is the movable depot. It can be positioned based on the order characteristics before the start of the picking process.

In Sect. 2 we present a literature review on optimization in warehouses, focusing mainly on U-shaped corridors. In Sect. 3 we formally describe the problem we treat, while in Sect. 4 a constraint programming model, based on a for picking optimization is described. The optimization of the location of the depot is instead discussed in Sect. 5. Computational experiments are presented in Sect. 6, where the new methods we propose are compared with state-of-the-art algorithms from the literature. Conclusions are finally summarized in Sect. 7.

2 Literature Review

Design and operations of warehouses in the context of order picking has been vastly studied in the literature. Topics covered include layout design, storage assignment, zoning, batching and routing. General analysis and surveys on zoning, batching and routing problems can be found in [2]. The layout design problem can be aggregated at various levels. The perspective ranges from location planning [4], department arrangement inside the warehouses [5], to determining the number, orientation, and the arrangement of the shelves etc. The latter problems are highly interdependent with order picking problems, particularly in the field of storage assignment, zoning and routing. Manual picking systems are still the most popular ones in several contexts, and in [6] the average travel distance of a picker is analyzed with different routings on different layout arrangements.

A traditional layout for warehouses that consist of multiple zones with shelves organized in U-shape, is considered in [7] and in [8], where a detailed literature survey is also available. 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 [7]. A study on improving the performance of the U-shaped layout through pickers learning is presented in [9], while the use of semi-empty pallets that allow easier object extraction is discussed in [10]. Optimization tools are very popular to enhance efficiency in logistics ([11–13]). In [14] it is shown how to optimize layout design, storage assignment and depot location in a U-shaped corridor, both in terms of economics or ergonomics. An extension of the work, that considers more degrees of freedom for the depot position, is discussed in [15].

3 Problem Description

This paper considers order picking in a U-shaped area as outlined in [7] and [8]. In the warehouse considered items are stored in N stillages, 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, and the geometry of the corridor can change, but is given as an input. The depot, where each order picking tour starts and ends and the collected goods are grouped, is brought to and removed from the U-zone by a forklift truck, and its position within the corridor can be decided in advance. The picker travels on foot along the shelves of the U-zone, possibly pushing or pulling a cart or a similar device.

We consider the operations of a picker while processing a single order in a single U-zone. Each order consists of a set $K \subseteq N$ of stillages that need to be visited to collect items, that have to be grouped to the depot D. For each stillage $k \in K$, a quantity d_k of goods collected is also provided (this can be weight, dimension, or a combination of the two). A capacity Q for the picking device used by the picker is also given. The picker must return to the depot when she/he has finished the order or when the transport capacity Q of the picking device has been reached. In the latter case, she/he returns to the depot to empty the picking device, and then continue the picking process. After an order has been completed, the depot is taken away. Note that although the optimization focuses on a single order, during a shift a picker processes multiple orders. We assume orders are planned beforehand, such that each order is independent from other orders in the optic of our optimization.

The U-zone's coordinate system is two-dimensional, and the depot can be placed anywhere in the U-corridor zone. The coordinates (x_D, y_D) of the depot D have to be decided before the order picking process starts, since a forklift will deposit it in the suggested position and take it away once the picking is completed. Euclidean distances are used to calculate the travel distance d_{ij} between two points of interests i with coordinates (x_i, y_i) and j with coordinates (x_j, y_j) of set $N \cup \{D\}$: $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, as we assume that this is the most intuitive way to travel through the pick zone. Note that only stillages containing items to be picked are considered.

When optimizing the location of the depot, the distance travelled by the forklift to position the depot is calculated from the center of the open end of the U corridor (coordinates (0,0)) to the selected location (x_D, y_D) again as a Euclidean distance, but in this case a factor ν is considered to model that this movement is carried out (twice) but the (faster) vehicle in charge of positioning and collecting the depot. The distance travelled to position and collect the depot is therefore calculated as $P_D = \nu\sqrt{(x_D)^2 + (Y_D)^2}$.

4 Sequencing of Picking Operations

Assuming the (feasible) coordinates of the depot (x_D, y_D) inside the U-corridor are given, the picking process can be optimized by solving a Capacitated Vehicle Routing Problem (CVRP) [16], as observed in [7]. Although we can not use this property in our solving method, in [8] is is proven that this problem can be further simplified, since we can consider only picking tours that visit stillages in a clockwise (or counterclockwise) order.

The picking optimization process can therefore be modelled in terms of constraint programming [17] as follows. We have a variable x_{ij} that takes value 1 if location j is visited right after location i in a picking tour, 0 otherwise. A second variable y_j is also defined for each location, containing the cumulative picking quantity collected along a tour, up to location j. It is used to model capacity constraints. Using the syntax of the CP-SAT solver of OR-tools [18], the following model can be used to describe the problem.

$$z(x_D, y_D) = P_D + \min \sum_{i \in N} \sum_{j \in N} d_{ij}^D x_{ij} \tag{1}$$

$$s.t. \quad \text{MultipleCircuit}(x_{ij} : i, j \in N) \tag{2}$$

$$x_{ij} = 1 \Rightarrow y_j = y_i + q_j \qquad i \in N, j \in K \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in N \tag{4}$$

$$0 \le y_i \le Q \qquad i \in N \tag{5}$$

For notational ease, we define z as the cost of the solution of the model when the depot is in position (x_D, y_D). The objective function (1) minimizes the sum of P_D, the (scaled) distance traveled by a forklift to position the depot, and the total distance traveled by the picker. Note that distances d_{ij}^D are used to indicate that the distances are calculated with respect to the given position of the depot. Constraint (2) uses the *MultipleCircuit* statement of CP-SAT [18] to model a vehicle routing feasible solution. Constraints (3) regulate capacity. They are activated only when $x_{ij} = 1$ through the statement *OnlyEnforceIf* of CP-SAT [18], here indicated as "$\Rightarrow$", and set the cumulative weight collected up to customer j in a tour. Constraints (4) and (5) finally define the domains of the variables.

Note that the model can be solved either optimally, or heuristically by truncating the execution of the solver after a given time of T_{max} seconds. This latter option is more likely to be used for large problems, given the $\mathcal{NP}$-hard nature of the CVRP problem [16].

5 Optimization of the Depot Location

In the context of finding an optimized location for the depot, our approach consists in repeatedly solving the model constraint programming model described in Sect. 4 with different tentative locations for the depot, and choose the best. Differently from [7] and [8], where only the locations in the central axis of the corridor are considered, we decided to consider all possible positions within the corridor. The motivation for restricting the search to che center of the corridor was related to the maneuvering space for the picking devices assisting the workers. In our experience, however, warehouses are nowadays equipped with agile picking devices able to move around easily, so we decided to open for more options for the depot location. Note that in case the optimization has to be restricted to the central axis of the corridor, as in [7] and [8], it is enough to set the feasible coordinates to 0 only for the y axis (see Fig. 1).

We will discuss two algorithms, the first regularly scanning all the area with a given step, and the second focussing the search on the promising locations only, through a zooming mechanism.

5.1 The Basic Algorithm

In this method, given a precision step s, all the locations on a grid defined over the feasible area for the depot, and with a step s, will be examined. A visual example is provided in Fig. 2, where the red points are examined.

The pseudocode of the *Basic* approach is provided in Algorithm 1. The method takes in input the coordinates of the top-left and bottom-right corner of the (feasible) search area for the depot location together with the search-step s. After an initialization phase (lines 1–3), the main double loop is entered and for each point of the grid defined by s, the model described in Sect. 4 is solved with the given coordinates for the depot for a maximum of T_{max} seconds. In case the solution is better than the best one retrieved so far, the data of the best solution are updated (line 7 and 8). The method returns the coordinates of the best location for the depot and the relative cost.

5.2 The ZoomIn Algorithm

A major drawback of the *Basic Algorithm* discussed in Sect. 5.1 is that the exploration for the best depot location is carried out evenly along the whole search area. In the reality, nearby locations will tend to give similar costs, and the costs will gradually smoother into different values as the distance increases. Such a situation is depicted in Fig. 3 for the instance *44-(9x4)-10-01*, where *1D* stands for the locations along the central axis ($y = 0$) only and *2D* for all locations; green areas indicate locations with a lower cost and the red ones locations with a higher cost. We can exploit this situation and zooming only in promising areas, aiming at reducing the overall computation time of the search.

The search area for the depot location is first analyzed with a large step s_M and only the surrounding of the most promising location is selected for a

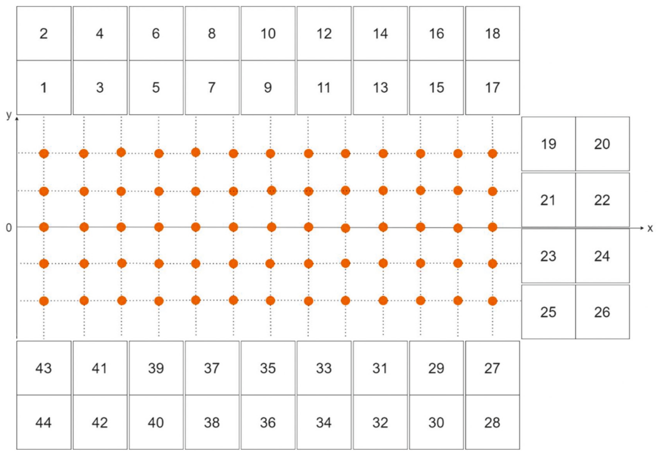

Fig. 2. Depot positions scanned by the *Basic* algorithm with a given step

Algorithm 1 - Basic$((x_m, y_m), (x_M, y_M), s)$

```
1:  $(x^B, y^B) = (x_m, y_m)$
2:  $z^B = +\infty$
3:  for $y_D$ in range$(y_m, y_M, s)$ do
4:      for $x_D$ in range$(x_m, x_M, s)$ do
5:          Solve the model from Section 4 for $T_{max}$ seconds
6:          if $z(x_D, y_D) < z^B$ then
7:              $(x^B, y^B) = (x_D, y_D)$
8:              $z^B = z(x_D, y_D)$
9:          end if
10:     end for
11: end for
12: return $(x^B, y^B)$, $z^B$
```

deeper analysis, carried out with a smaller step. This process is continued until the required precision is reached. It is worth however observing that such a procedure makes the search process more heurristic. Looking again to Fig. 3, for example, a crude initial analysis might lead to search in details the top-right green area, which unfortunately can only lead to a local optimum.

The pseudocode for the algorithm *ZoomIn* is provided in Algorithm 2. the method takes in input the coordinates of the top-left corner and bottom-right corners of the search area for the depot location, together with the initial step value s_M, the final step value s_m and the reduction factor e_s for the step.

The algorithm starts by setting the working boundaries (lines 1 and 2) and the working value of s (line 3). Then the main loop is entered where the *Basic* algorithm is invoked with the working parameters (line 5). The working boundaries are set to the surrounding of the best solution retrieved (lines 6 and 7). Note the use of the min and max operators to constraint the optimization to remain within the feasible region for the depot. The value of s is finally reduced at line 8, to augment the precision in the next iteration. Note that the main

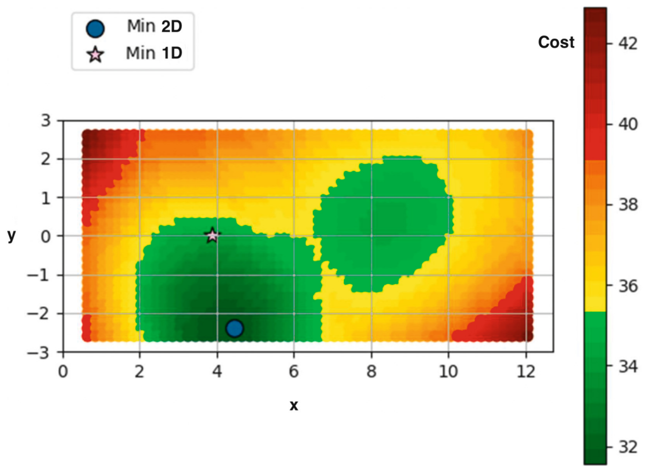

Fig. 3. Heatmap for the distance travelled by the picker for different locations of the depot. Instance *44-(9x4)-10-01*, s = 0.2, t=10 s

Algorithm 2 - ZoomIn$((x_m, y_m), (x_M, y_M), s_M, s_m, e_s)$

```
 1: $(\overline{x}_m, \overline{y}_m) = (x_m, y_m)$
 2: $(\overline{x}_M, \overline{y}_M) = (x_M, y_M)$
 3: $s = s_M$
 4: while $s \geq s_m$ do
 5:     $(x^B, y^B)$, $z^B$=Basic$((\overline{x}_m, \overline{x}_M), (\overline{y}_m, \overline{y}_M), s)$
 6:     $(\overline{x}_m, \overline{y}_m) = (\max\{x_m, x^B - s\}, \max\{y_m, y^B - s\})$
 7:     $(\overline{x}_M, \overline{y}_M) = (\min\{x_M, x^B + s\}, \min\{y_M, y^B + s\})$
 8:     $s = s/e_s$
 9: end while
10: return $(x^B, y^B)$, $z^B$
```

while loop is exited once the desired precision s_m has been reached. The algorithm finally returns the best coordinates retrieved for the location of the depot, and the relative cost.

6 Computational Experiments

The instances used for the experiments are first introduced in Sect. 6.1, The methods we propose are then compared with the existing ones while allowing the depot to be positioned only on the central axis of the corridor ($y = 0$ – *1D* in the remainder of the paper). The analysis is then extended to the case where the depot can be located everywhere inside the corridor (*2D* in the remainder of the paper), aiming at understanding in which measure the extra freedom can lead to more optimized solutions.

6.1 Instances

In order to evaluate our new solution procedures, the instances proposed in [8], that are based on the assumptions presented in [7], are adopted. U-layouts with two different capacities, either 44 stillages or 88 stillages are considered. The layout of an instance is defined by setting n and m, the number of stillages in one horizontal and in one vertical row. We randomly draw K items to be picked for each instance. For the instances with 44 stillages, we set either $|K| \in \{10, 15\}$, for the instances with 88 stillages $|K| \in \{30, 60\}$. Items weights $q_j \in \{1, \ldots, 5\}$ $\forall j \in K$ and the picker capacity is set to $Q = 15$. Finally, the factor for positioning the depot with the forklift is set to $\nu = 1/3$. All instances are labeled as follows: $|K| - (n, m) - |K| - \alpha$ where α is a running index.

The measurements of the stillages to $w = 1.3$ m and the gap between the stillages to $s = 0.05$ m. A minimum distance of 0.65 m has to exist between the depot location and the stillages (or the entrance of the corridor), in order to allow enough working space around the depot itself. The coordinates of a stillage (or the depot) used for the calculation of the distances are given by its center.

The reader interested in a complete description of the derivation of the instances can refer to [8], The instances themselves are available for download at https://doi.org/10.5281/zenodo.4671870.

6.2 1D Case: Comparison with State-of-the-Art Algorithms

In this section we consider the case 1D, where the depot is constrained to be over the central axis of the corridor ($y = 0$), as previously considered in the literature. The results of the algorithms described in Sect. 5 are compared with those of the methods previously appeared in the literature. The following methods are considered:

- *Sweep* from [7]. It was implemented in Mathematica 7.0 [19]. No information about the computer used for the experiments was provided.
- *Benders Decomp.* from [8]. It was coded in C#. The experiments were run on a computer equipped with 8 GB of RAM and an Intel Core i7-3631QM CPU, with CPLEX 12.10 [20] used to solve linear programs. A maximum running time of 3600 s is allowed (a dash in the column *Sec* means that this limit is reached).
- *Dynamic Progr.* from [8]. It was coded in C#. The experiments were run on a computer equipped with 8 GB of RAM and an Intel Core i7-3631QM CPU.
- *Basic* (Sect. 5.1) and *ZoomIn*. It was coded in Python 3. The experiments were run on a computer equipped with 32 GB of RAM and an Intel Core i7 12700F CPU, with OR-Tools CP-SAT 9.6 [18] used as a solver for Constraint Programming model.
- *ZoomIn* (Sect. 5.2). It was coded in Python 3. The experiments were run on a computer equipped with 32 GB of RAM and an Intel Core i7 12700F CPU, with OR-Tools CP-SAT 9.6 [18] used as a solver for Constraint Programming model.

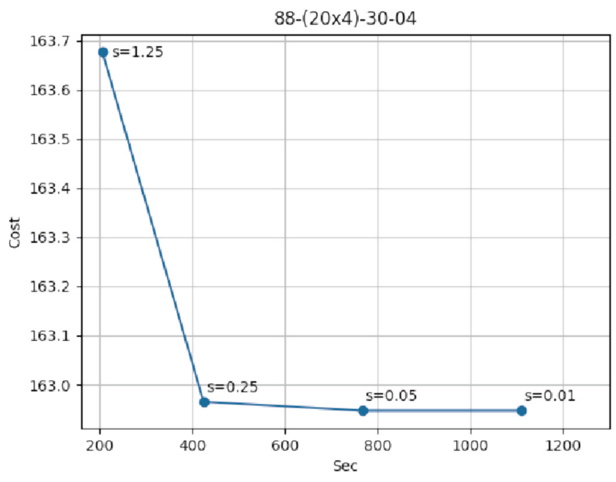

Fig. 4. Algorithm *ZoomIn* with $s_M = 1.25$, $s_m = 0.01$, $e_s = 5$ and $T_{max} = 3$: best cost over different values of s for instance *88-(20x4)-30-04*

The results are summarized in Table 1, where for each instance we report the best cost obtained by the method together with the computation time required. Note that for the algorithms we propose, experiments with different parameter settings are proposed (they are indicated in the table). This suggests that it is possible to tune these methods to me either faster or more precise.

From Table 1 it emerges that the new constraint programming-based methods we propose are competitive when compared with the existing algorithms. Although they are slower than other approaches, they provide high quality results. In particular the *ZoomIn* method match or improves all the previous best-known results, keeping however the computation times below 550 s in the worst case. Such a time is practically sufficient for a Company to plan in advance the pickings for the next day, assuming to have proper hardware, like a suitable multicore architecture.

When comparing the *Basic* and *ZoomIn* methods, the convenience of the latter emerges clearly, since a much smaller (final) step can be achieved in comparable computation times. However, the it can be observed that the gain in the total distance does not improve much when the step becomes very small. This is evident in Fig. 4 where the evolution of the total distance over time is reported. This suggest that settings where the final step is not too small should be preferred to have a better trade off between computation time and quality of the solution. This intuition will be followed for the results reported in Sect. 6.3.

6.3 2D Case: Allowing the Depot Out of the Central Axis of the Corridor

In this section we compare the best results obtained in Sect. 6.2 while considering only locations for the depot along the central axis, with new results obtained allowing the depot to be placed anywhere in the corridor. Our aim is to

Table 1. 1D case: comparison with state-of-the-art algorithms

Instance	Benders Decomp. [8]		Sweep [7]		Dynamic Progr. [8]		Basic Algorithm				ZoomIn Algorithm			
							$s = 1$ $T_{max} = 3$		$s = 1$ $T_{max} = 10$		$s_M = 1.25$ $s_m = 0.01$ $e_s = 5$ $T_{max} = 3$		$s_M = 1.25$ $s_m = 0.01$ $e_s = 5$ $T_{max} = 10$	
	Val	Sec	Val	Sec	Val	Sec	Val	Sec	Val	Sec	Val	Sec	Val	Sec
44-(9×4)-10-01	34.3	0.6	34.3	0.0	34.3	0.8	34.4	1.4	34.4	1.3	34.3	3.9	34.3	3.9
44-(9×4)-10-02	37.4	0.3	37.4	0.0	37.4	0.3	37.5	3.6	37.5	3.7	37.4	5.7	37.4	5.3
44-(9×4)-10-03	44.6	1.4	45.5	0.0	44.6	0.2	44.6	3.3	44.6	3.3	44.6	4.9	44.6	4.5
44-(9×4)-10-04	37.3	0.5	37.3	0.0	37.3	0.2	37.3	3.3	37.3	3.2	37.3	5.8	37.3	5.9
44-(9×4)-10-05	32.7	0.3	32.7	0.0	32.7	0.7	32.7	3.2	32.7	3.5	32.7	6.0	32.7	5.9
44-(9×4)-10-06	39.0	0.7	39.0	0.0	39.0	0.6	39.1	0.9	39.1	1.0	39.0	2.7	39.0	2.4
44-(9×4)-10-07	43.1	1.1	44.0	0.0	43.1	0.4	43.1	2.1	43.1	2.1	43.1	5.4	43.1	5.5
44-(9×4)-10-08	37.5	0.6	37.9	0.0	37.5	0.4	37.5	7.1	37.5	6.9	37.5	10.3	37.5	9.7
44-(9×4)-10-09	34.4	0.4	35.6	0.0	34.4	0.3	34.4	5.3	34.4	5.5	34.4	6.0	34.4	6.5
44-(9×4)-10-10	34.4	0.7	34.4	0.0	34.4	0.2	34.5	9.0	34.5	10.2	34.4	15.8	34.4	19.5
44-(9×4)-15-01	52.4	39.3	52.4	0.0	52.4	0.6	52.4	36.5	52.4	116.8	52.4	124.6	52.4	407.8
44-(9×4)-15-02	57.4	54.8	57.4	0.0	57.4	0.4	57.6	36.4	57.6	120.5	57.4	121.5	57.4	401.7
44-(9×4)-15-03	53.8	99.5	56.4	0.0	56.4	0.6	53.8	36.5	53.8	120.5	53.8	121.5	53.8	401.7
44-(9×4)-15-04	47.4	12.7	47.4	0.0	47.4	0.9	47.4	36.5	47.4	102.8	47.4	121.5	47.4	219.3
44-(9×4)-15-05	42.5	8.1	42.5	0.0	42.5	1.4	42.5	31.1	42.5	63.5	42.5	78.3	42.5	105.7
44-(9×4)-15-06	49.4	19.1	49.4	0.0	49.4	0.4	49.4	36.4	49.4	120.5	49.4	121.6	49.4	401.6
44-(9×4)-15-07	43.7	13.1	43.7	0.0	43.7	1.0	43.7	36.5	43.7	119.8	43.7	121.5	43.7	398.2
44-(9×4)-15-08	47.0	7.5	47.0	0.0	47.0	0.6	47.0	36.5	47.0	94.0	47.0	121.6	47.0	315.6
44-(9×4)-15-09	53.0	79.9	53.0	0.0	53.0	0.4	53.1	36.4	53.1	120.5	53.0	121.6	53.0	401.6
44-(9×4)-15-10	45.8	16.8	46.9	0.0	45.8	1.0	45.8	33.9	45.8	91.0	45.8	73.5	45.8	123.4
88-(20×4)-30-01	148.9	–	145.7	0.2	144.0	4.7	142.0	83.4	141.4	272.7	141.4	163.7	141.3	535.1
88-(20×4)-30-02	148.0	–	140.9	0.2	140.9	2.9	141.7	83.5	139.2	272.8	139.2	164.1	139.1	535.4
88-(20×4)-30-03	146.7	–	139.3	0.2	139.3	3.5	132.6	83.2	131.9	272.6	131.8	163.5	131.8	535.2
88-(20×4)-30-04	177.2	–	167.8	0.2	166.4	2.2	164.1	83.5	163.0	272.7	163.0	163.8	163.0	535.2
88-(20×4)-30-05	145.5	–	142.0	0.2	138.1	2.5	136.8	83.5	136.6	272.8	136.6	163.7	136.5	535.5
88-(20×4)-30-06	172.0	–	159.8	0.2	159.8	2.7	155.1	83.5	155.1	272.9	154.9	164.0	154.9	535.5
88-(20×4)-30-07	143.1	–	134.5	0.2	134.5	6.7	135.4	83.3	132.5	272.6	132.7	163.5	132.5	535.4
88-(20×4)-30-08	151.9	–	147.2	0.2	147.2	4.0	148.1	83.6	147.2	272.8	147.2	163.8	147.2	535.1
88-(20×4)-30-09	154.8	–	148.0	0.2	148.0	2.8	146.9	83.4	146.5	272.6	146.5	163.8	146.4	535.2
88-(20×4)-30-10	144.0	–	137.9	0.2	137.9	3.5	138.0	83.3	138.0	272.8	137.9	163.9	137.9	535.2
88-(20×4)-60-01	297.2	–	255.6	0.7	255.6	6.2	280.0	87.4	255.7	277.2	263.5	171.2	254.3	543.5
88-(20×4)-60-02	305.8	–	237.2	0.7	234.6	8.3	249.5	88.1	239.5	277.6	256.8	173.0	229.3	544.1
88-(20×4)-60-03	288.5	–	235.0	0.7	231.6	10.4	245.1	87.8	230.1	277.9	239.0	172.4	228.5	544.0
88-(20×4)-60-04	376.9	–	285.3	0.8	283.1	6.3	304.8	87.9	275.0	278.0	286.0	172.1	270.3	544.5
88-(20×4)-60-05	309.4	–	239.1	0.7	238.0	6.2	245.8	87.6	236.8	277.5	237.6	171.5	235.7	545.2
88-(20×4)-60-06	341.6	–	266.1	0.8	263.7	5.8	280.7	87.6	261.5	277.0	277.1	172.0	258.0	544.5
88-(20×4)-60-07	339.6	–	270.4	0.8	268.3	5.1	290.1	87.7	267.5	277.4	271.0	171.6	265.8	544.0
88-(20×4)-60-08	334.9	–	275.3	0.8	274.1	5.2	297.4	87.5	263.4	277.7	279.8	171.8	257.4	544.4
88-(20×4)-60-09	310.4	–	240.3	0.8	239.9	6.0	250.4	87.9	241.6	277.5	240.5	172.5	235.9	544.5
88-(20×4)-60-10	293.2	–	252.9	0.7	251.7	9.4	270.7	87.7	252.5	277.8	261.5	172.4	250.0	544.5

Table 2. 2D case: allowing the depot out of the central axis of the corridor

Instance	1D Best	2D ZoomIn $s_M = 2$, $s_m = 0.5$ $e_s = 5$, $T_{max} = 20$			Instance	1D Best	2D ZoomIn $s_M = 2$, $s_m = 0.5$ $e_s = 5$, $T_{max} = 20$		
	Val	Val	Sec	% Imp		Val	Val	Sec	% Imp
44-(9 × 4)-10-01	34.3	31.3	2.7	8.7	88-(20 × 4)-30-01	141.3	141.3	542.2	0.0
44-(9 × 4)-10-02	37.4	37.3	4.8	0.3	88-(20 × 4)-30-02	139.1	136.6	542.4	1.8
44-(9 × 4)-10-03	44.6	42.7	3.0	4.2	88-(20 × 4)-30-03	131.8	131.8	542.4	0.0
44-(9 × 4)-10-04	37.3	36.6	3.7	1.8	88-(20 × 4)-30-04	163.0	163.0	542.5	0.0
44-(9 × 4)-10-05	32.7	29.4	1.4	10.2	88-(20 × 4)-30-05	136.5	135.6	542.5	0.7
44-(9 × 4)-10-06	39.0	37.5	1.4	3.9	88-(20 × 4)-30-06	154.9	152.8	542.4	1.4
44-(9 × 4)-10-07	43.1	42.1	2.6	2.3	88-(20 × 4)-30-07	132.5	128.9	542.4	2.7
44-(9 × 4)-10-08	37.5	35.5	7.3	5.3	88-(20 × 4)-30-08	147.2	146.6	542.4	0.4
44-(9 × 4)-10-09	34.4	34.4	5.1	0.0	88-(20 × 4)-30-09	146.4	145.6	542.6	0.6
44-(9 × 4)-10-10	34.4	31.3	10.1	8.8	88-(20 × 4)-30-10	137.9	137.8	542.6	0.1
44-(9 × 4)-15-01	52.4	51.9	366.6	0.9	88-(20 × 4)-60-01	254.3	249.2	547.6	2.0
44-(9 × 4)-15-02	57.4	57.3	363.5	0.2	88-(20 × 4)-60-02	229.3	229.4	547.7	0.0
44-(9 × 4)-15-03	53.8	53.1	425.0	1.2	88-(20 × 4)-60-03	228.5	225.3	547.7	1.4
44-(9 × 4)-15-04	47.4	45.4	117.4	4.3	88-(20 × 4)-60-04	270.3	265.1	548.0	1.9
44-(9 × 4)-15-05	42.5	41.5	46.5	2.2	88-(20 × 4)-60-05	235.7	232.6	547.8	1.3
44-(9 × 4)-15-06	49.4	47.0	408.0	4.8	88-(20 × 4)-60-06	258.0	258.0	548.0	0.0
44-(9 × 4)-15-07	43.7	43.1	222.1	1.2	88-(20 × 4)-60-07	265.8	264.7	548.0	0.4
44-(9 × 4)-15-08	47.0	46.9	190.5	0.1	88-(20 × 4)-60-08	257.4	253.0	548.0	1.7
44-(9 × 4)-15-09	53.0	50.6	364.2	4.6	88-(20 × 4)-60-09	235.9	235.6	547.6	0.1
44-(9 × 4)-15-10	45.8	45.7	184.2	0.1	88-(20 × 4)-60-10	250.0	247.0	547.5	1.2

understand how much this extra degree of freedom can enhance the quality of the solutions.

The results are summarized in Table 2, where the best known results for the 1D case (from Table 1) are shown together with the results achieved by the 2D-version of the *ZoomIn* algorithm with settings allowing to conclude the computation within 550 s for each instance. A column reporting the improvement of the 2D solution with respect to the 1D one is also incorporated in the table.

The results of Table 2 suggest that considering the possible location of the depot in the whole corridor (2D) and not only in the center (1D) can lead to substantial improvements on the total distance travelled, especially for those orders that require picking from a small proportion of the total stillages. For example, instance *4-(9x4)-10-05* where only 10 stillages are visited over the 44 of the corridor, moving the depot on one side leads to a shortening of the picking process of 10.2%. The advantage is less evident for orders that have to touch most of the stillages (like the last instances, in which 60 stillages over 88 have to be visited).

An average advantage of 2.1% has been obtained over the given instances by moving the depot out of the central axis of the corridor. Given that computation times are in the order of those of the 1D case, we conclude that positioning the depot off-center should be considered by Companies to increase their efficiency.

7 Conclusion

We considered a warehouse with U-shaped corridors and we treated the optimization of the single-order picking process. In particular, the location of the depot is considered as part of the optimization.

From an operational viewpoint, the constraint programming-based algorithms we propose are able to improve the results of the methods available from the literature, although we longer (but still feasible) computation times.

From a strategical perspective, we also show that positioning the depot off-centered with respect to the central axis of the corridor – as it was common in the previous literature – can enhance the overall picking time. This phenomenon is particularly clear for instances where picking involves only a relative small fraction of the stillages. A further study positioning this result from a more strategic point of view is foreseen.

References

1. Coyle, J.J., Bardi, E.J., Langley, C.J.: The Management of Business Logistics. West Publishing Company, St Paul, MN (1996)
2. De Koster, R., Le-Duc, T., Roodbergen, K.J.: Design and control of warehouse order picking: a literature review. Eur. J. Oper. Res. **182**(2), 481–501 (2007)
3. Ognibene Pietri, N., Chou, X., Loske, D., Klumpp, M., Montemanni, R.: The buy-online-pick-up-in-store retailing model: optimization strategies for in-store picking and packing. Algorithms **14**(12), 350 (2021)
4. Kusiak, A., Heragu, S.S.: The facility layout problem. Eur. J. Oper. Res. **29**(3), 229–251 (1987)
5. Heragu, S.S., Du, L., Mantel, R.J., Schuur, P.C.: Mathematical model for warehouse design and product allocation. Int. J. Prod. Res. **43**(2), 327–338 (2005)
6. Roodbergen, K.J., Sharp, G.P., Vis, I.F.: Designing the layout structure of manual order picking areas in warehouses. IIE Trans. **40**(11), 1032–1045 (2008)
7. Glock, C.H., Grosse, E.H.: Storage policies and order picking strategies in U-shaped order-picking systems with a movable base. Int. J. Prod. Res. **50**(16), 4344–4357 (2012)
8. Diefenbach, H., Emde, S., Glock, C.H., Grosse, E.H.: New solution procedures for the order picker routing problem in u-shaped pick areas with a movable depot. OR Spectr. **44**, 535–573 (2022)
9. Grosse, E.H., Glock, C.H.: An experimental investigation of learning effects in order picking systems. J. Manuf. Technol. Manag. **24**(6), 850–872 (2013)

10. Glock, C.H., Grosse, E.H., Abedinnia, H., Emde, S.: An integrated model to improve ergonomic and economic performance in order picking by rotating pallets. Eur. J. Oper. Res. **273**(2), 516–534 (2019)
11. Chou, X., Gambardella, L., Montemanni, R.: A Tabu search algorithm for the probabilistic orienteering problem. Comput. Oper. Res. **126**, 105107 (2021)
12. Dell'Amico, M., Montemanni, R., Novellani, S.: Algorithms based on branch and bound for the flying sidekick traveling salesman problem. Omega **104**, 102493 (2021)
13. Montemanni, R., Smith, D., Gambardella, L.: Ant colony systems for large sequential ordering problems. In: Proceedings of the IEEE Symposium on Swarm Intelligence (SIS), pp. 60–67 (2007)
14. Diefenbach, H., Glock, C.H.: Ergonomic and economic optimization of layout and item assignment of a u-shaped order picking zone. Comput. Ind. Eng. **138**, 106094 (2019)
15. Montemanni, R., Landolfo, A., Chou, X., Loske, D., Klumpp, M.: Ergonomics and storage base position for u-shaped picking zones. In: Proceedings of the 9th International Conference on Industrial Engineering and Applications (ICIEAeu), pp. 199–206. ACM (2023)
16. Toklu, N., Montemanni, R., Gambardella, L.: An ant colony system for the capacitated vehicle routing problem with uncertain travel costs. In: Proceedings of the IEEE Symposium on Swarm Intelligence (SIS), pp. 32–39 (2013)
17. Montemanni, R., Dell'Amico, M.: Solving the parallel drone scheduling traveling salesman problem via constraint programming. Algorithms **16**(1), 40 (2023)
18. Google: OR-Tools. https://developers.google.com/optimization/
19. Wolfram: Mathematica (2023). https://www.wolfram.com/mathematica/
20. ILOG, I.: User's manual for CPLEX (2023). https://www.cplex.com/