PART III

Industry 4.0, Logistics 4.0 and Smart Manufacturing

CHAPTER-13

On the Order Picking Policies in Warehouses
Algorithms and their Behavior

Ricardo Arriola, Fernando Ramos, Gilberto Rivera, Rogelio Florencia, Vicente García* and *Patricia Sánchez-Solis*

This chapter explores the relationship among different routing policies for order picking and the features of the problem (describing both warehouse layout and orders), the results obtained by simulation show that some policies are especially sensitive to the presence of certain conditions that are likely to be present in real-world cases.

Moreover, the routing policies are represented—for the first time in the literature as far as our knowledge—on structured algorithms. This contribution can facilitate their implementation because the features of the policies are modeled by formal mathematical structures, laying the foundations to standardize the way they operate.

1. Introduction

A warehouse is a fundamental part of a company, and its performance can impact the entire supply chain [1].

Order picking is a problem that is present in all companies. It has received special focus from research areas related to planning and logistics. This fact is a consequence of several studies that identify order picking as the activity that demands more resources inside the warehouses, reaching up to 55% of the operational cost of the entire warehouse [2].

This activity has a strong impact on production lines, so companies with complex warehouses have areas dedicated to improving their product collection processes.

Universidad Autónoma de Ciudad Juárez, Av. Hermanos Escobar, Omega, 32410 Cd Juárez, Chihuahua, México.
* Corresponding author: gilberto.rivera@uacj.mx

There are optimization models to support the resolution of this problem in the generation of product-picking routes; however, being considered as an NP-complete problem is not feasible to solve the models when working at medium and large scale due to the high cost that this represents. Thus, it is possible to apply some heuristics to get an approximate solution in real cases.

Although several studies in the literature (e.g., [3]) show that these procedures are far from finding solutions close to the optimal one; these heuristics are still applied to real problems due to the simplicity and the way they relax the problem, granting a good balance between the quality of the solution and the ease of implementation.

The picking routes are dependent on the structure of the warehouse and the properties of the orders, so studies have stated [4] that the input elements and the performance of the routes obtained are highly related. For example, a greater number of cross aisles facilitates movements inside the warehouse, so that the distance of the route tends to decrease.

Throughout this chapter, we are going to define the algorithms for five of these heuristics and deepen on the study of which of them are more sensible to the characteristics describing the layout of a warehouse.

2. Background

Warehouses are an important part of supply chain in a factory, and the main activities inside of it, like reception (receive and collect all product data), storage (move products to their locations), pick up (pick products from their storage location), packing (prepare for being transported), and shipping (place product in the transport medium). On this last step, the warehouse operation ends.

2.1 Warehouse Layout

The chief features of the warehouse are: the central depot, where the picker starts its route and finish it, also usually is where the picker gets the order. The picker has to walk by the—the second element— picking aisles. A picking aisle is the space between two racks and facilitates the picker to pick product from the shelf, and the aisles have the following characteristics: length (distance between front aisle and rear aisle), and distance between aisles (distance that exists from the center of one aisle to center of next aisle); based on the distance, we can classify the aisles as short or long, in this case, we are going to use short aisles, which means that we can get the products from the shelves without making lateral displacements over the aisle. Cross aisles are perpendicular to picking aisles and are used to travel from one aisle to another, picking node is the location in the shelf where you can get the product, shelf is the space in the rack where products are stored, block is the area between the cross aisle and picking aisle, and sub-aisle is the area between the picking aisles inside the block if the blocks have picking aisles. Finally, picker is the person, tool or machine that picks up the products.

Figure 1 represents an example of a layout of a warehouse with five cross aisles, four blocks, six picking aisles, and a central depot.

Also, we are going to clarify the key concepts and briefly explain how the order picking routing policies work.

2.2 Steiner STP

In the literature, it is verified that TSP (Travel Salesman Problem) is on the classification of NP-hard problems [5]; likewise, TSP and Order Picking have a close relationship. Unfortunately, the optimal solution may require intolerable run times for high-scale problems. Considering an approximate solution might be more convenient as this provides an favourable relationship between time and cost.

Optimization algorithms based on local searches focus on achieving a good quality solution in a reasonable time to get a minimum or maximum value and avoid being stuck in a local optimum.

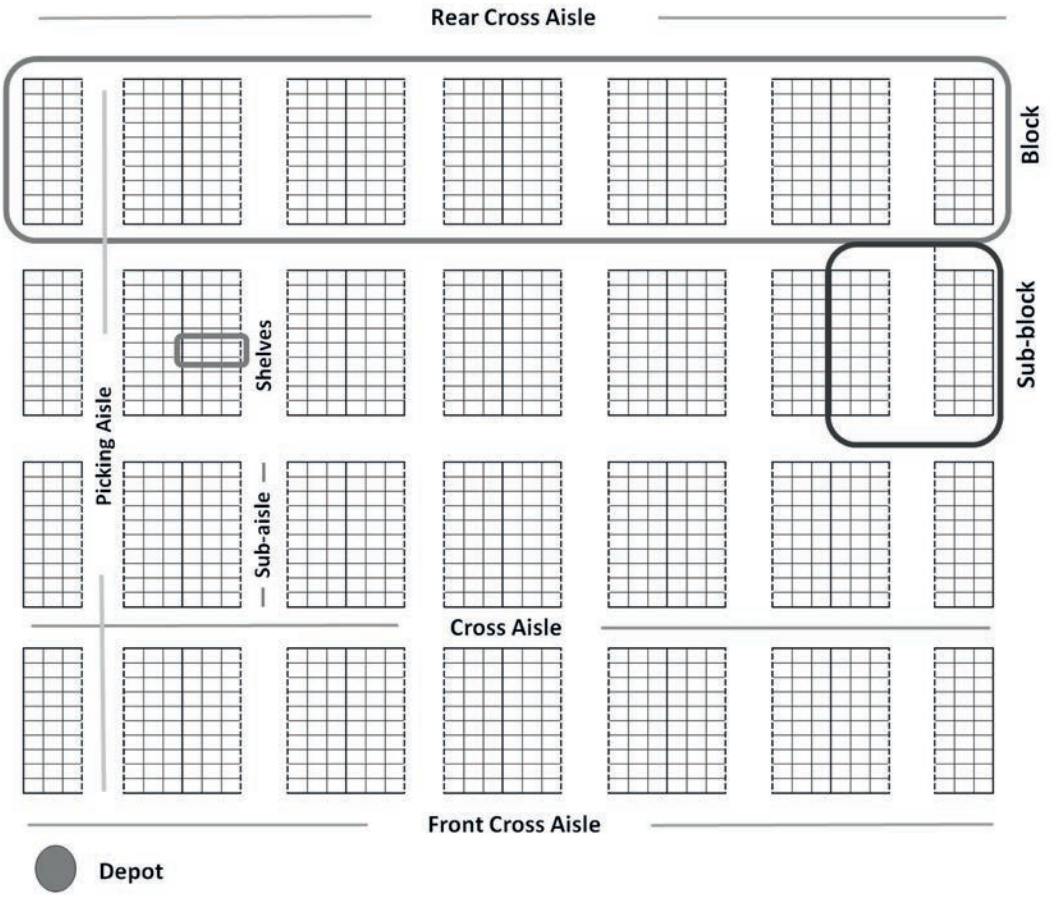


Figure 1: Warehouse layout elements.

It is necessary to start from a solution and, by applying operators, calculate solutions better than the initial solution. Normally, this strategy is applied to NP-hard problems, where heuristic functions are used to eliminate non-promising routes [6].

The solution for this project is represented as SPRP (Single-Picker Routing Problem), which consists of finding the shortest path that includes all the products to pick up [7]. This problem could be represented as a special case for TSP and can be applied as such to solve the initial SPRP problem. The objective is minimizing the distance and the time travel of the picker, either a human or machine, so it becomes a TSP. TSP consists of a salesman and a set of cities. The salesman must visit each of the cities, starting from a specific location (for example, native city), and come back to the same city. The challenge of this problem is that the salesman wishes to minimize the total duration of his/her trip.

SPRP could be modeled as TSP, where the vertices of the correspondent graph are defined by the location of the available products inside of the warehouse and the location of the depot, as presented in Figure 2.

This graph shows all the vertices and not only the picking ones, so the SPRP was modelled as a Steiner TSP (which is a variant of the classical TSP) that is defined as follows:

Let $G = (V, E)$ be a graph with a set of vertices V and a set of edges E. Let P be a subset of V. The elements of $V \backslash P$ are Steiner points. On a Steiner route, each vertex of P is visited only once. Steiner points should not be visited multiple times. However, a Steiner route could travel through

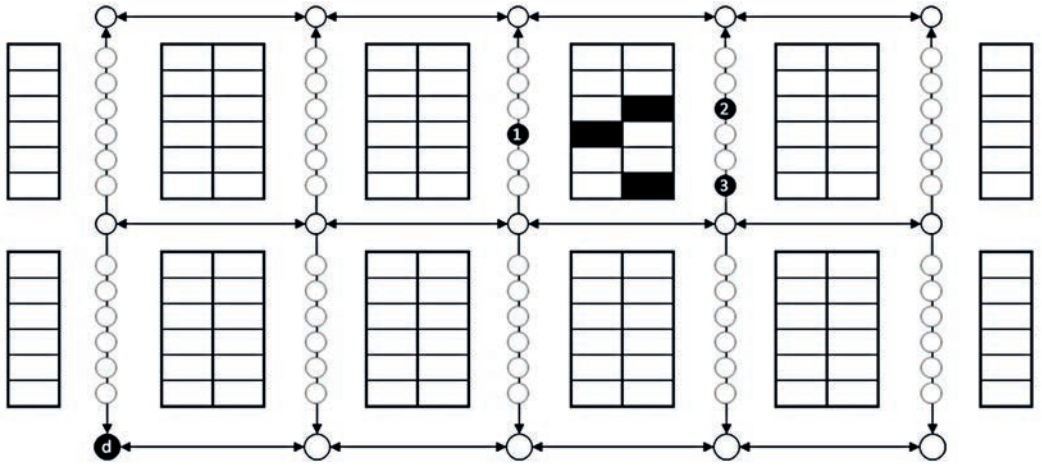


Figure 2: Example of a warehouse structure as a set of V vertices.

some vertices and edges more than one time. In conclusion, the TSP of Steiner consists of finding a route of Steiner with the minimum distance [8].

Figure 3 shows an example of a warehouse layout with different parameters and variables.

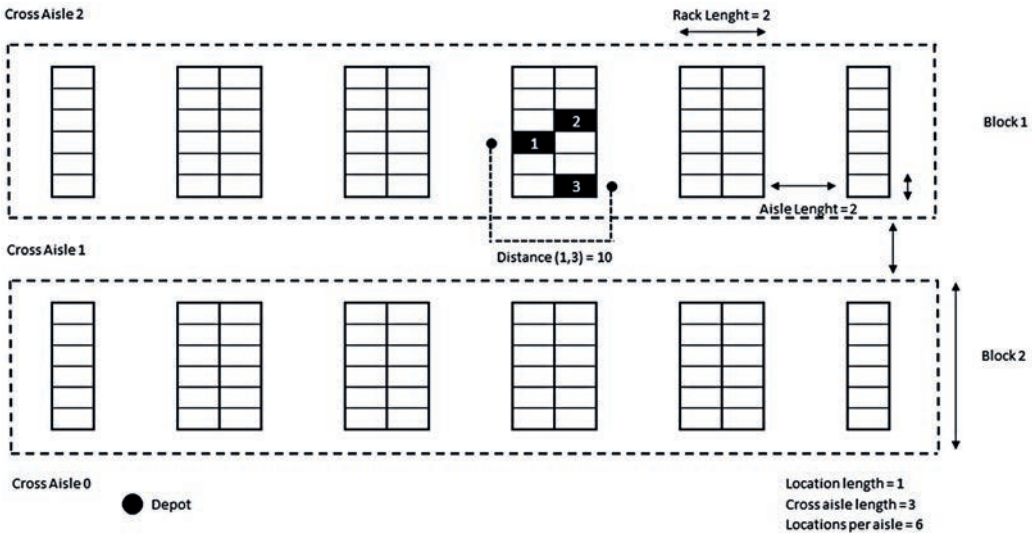


Figure 3: Example of a warehouse structure with different parameters.

In Figure 4, the black-filled vertices are the picking vertices and the initial vertex, also known as the depot. This set of vertices is the set *P*, a subset of all the vertices *V*. This subset will form a Steiner graph, and the vertices formed at the intersections of the cross aisles and the picking aisles we will call Steiner points.

Once the graph is obtained, the objective is to find a Hamiltonian circuit with the minimum cost. The initial and finish point of this circuit will always be the depot.

Also, it is important to know that there are six different ways to travel through a picking aisle [9].

Figure 5 describes each one over one example of a unique block, one front cross aisle, and one rear cross aisle.

Picker enters by the front cross aisle, crosses it completely, picks up all required products, and finishes leaving the aisle by the rear cross aisle.

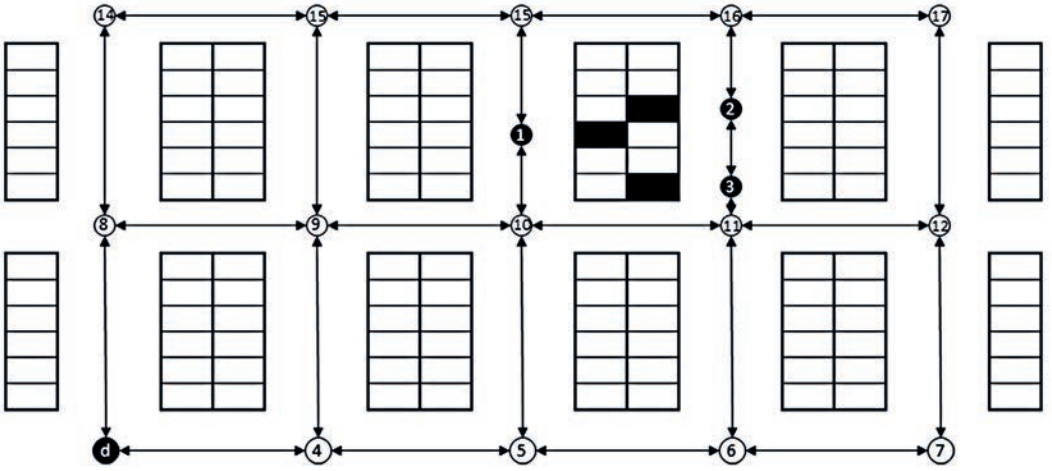


Figure 4: Example of a Steiner graph.

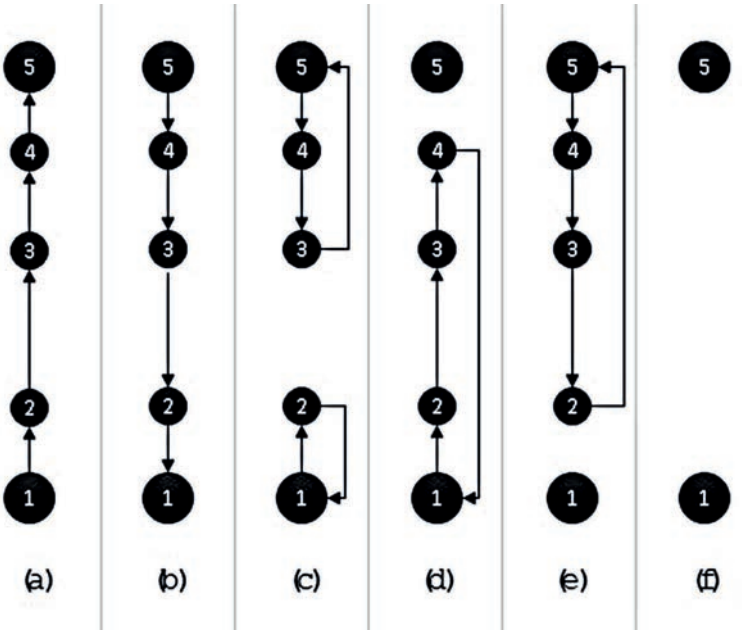


Figure 5: Six ways to travel edges through the picking aisles.

Picker enters by the rear cross aisle, crosses it completely, picks up all required products, and finishes leaving the aisle by the front cross aisle.

Picker enters and leaves twice through the aisle, enters once through the front cross aisle and once more through the rear cross aisle, picker enters and leaves by the same place. The picker will make its return defined by the largest gap, which is the largest distance between two adjacent picking vertices or the picking vertex and cross aisle.

Picker enters through the front cross aisle, and its return point is the picking vertex farthest from the front aisle.

Picker enters through the rear cross aisle, and its return point is the picking vertex farthest from the rear aisle.

The picker doesn't need to travel through the aisle because there are no picking vertices inside.

These ways to travel are combined, generating different routing policies, which are highly popular in practice.

3. Routing Policies

The routing policies determine the collection sequence of the SKUs (Stock-keeping unit) [10]. The objective of the routing policies is minimizing the distance traveled by the picker using simple heuristics [3].

To achieve this, it is necessary to consider the following features of the warehouse layout and the product orders, which can influence the final performance of each policy: quantity of products in the order, picker capacity, aisle length, and the number of aisles.

Five of these heuristics are described below.

3.1 S-Shape

The picker must start by entirely crossing the aisle (with at least one product) that is at the left or right end (depending on which is the closest one to the depot) until reaching the rear cross aisle of the warehouse. Then, the sub-aisles that belong to the farthest block of the depot are visited one by one until they end up at the opposite end of the warehouse. The only case where it is not necessary to cross a sub-aisle completely is when it is the last one in the block. In this case, after picking up the last product, the picker returns to the cross aisle from where it entered the sub-aisle. When changing blocks, the picker visits the closest sub-aisle to the last visited sub-aisle of the previous block. After picking up all the products, the picker must return to the depot [11]. Figure 6 shows an example.

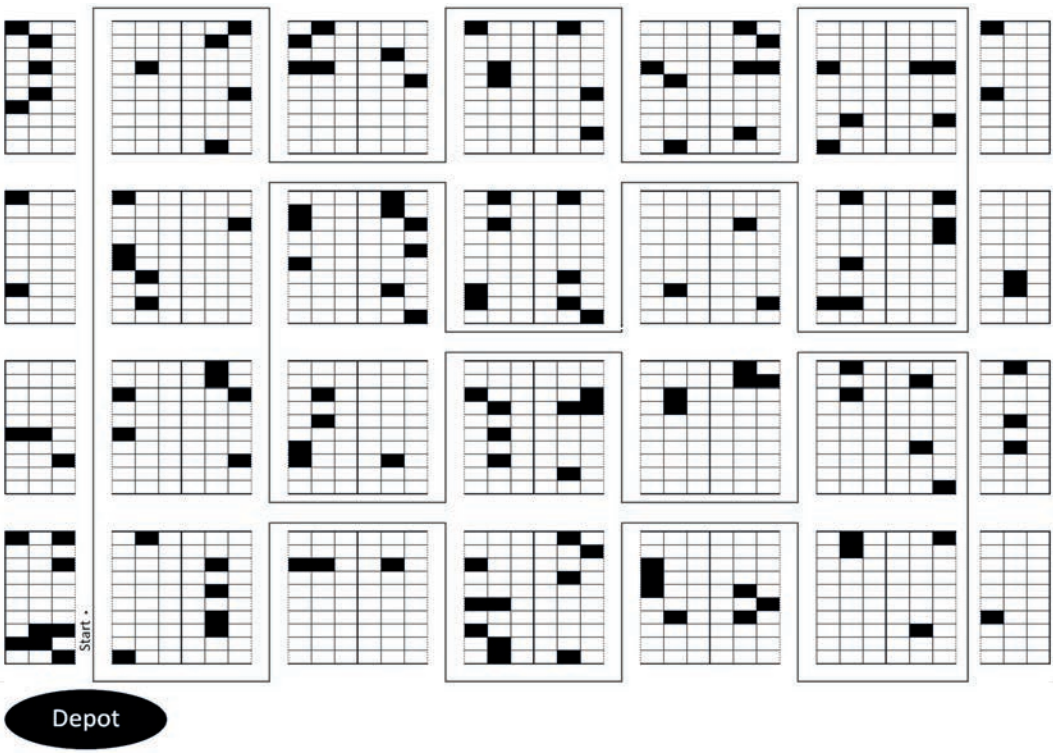


Figure 6: An example of a route applying S Shape.

3.2 Largest gap

This routing policy consists of identifying which is the largest gap in each sub-aisle and then avoiding crossing it [12]. This gap can be either between the rear cross aisle and the first product to be picked, between adjacent products or between the last product of the sub-aisle to the front cross aisle. All products that are before this gap will be picked up from the rear cross aisle and afterwards the picker must return to the cross aisle where it departed and do the same until all the sub-aisles of the block are explored, then, pick up all the remaining products from the front cross aisle.

The sub-aisles that are completely crossed are those belonging to the first visited picking aisle and the last one of each block (this way, it passes from one cross aisle to another).

When the picker picks up all the products, it must return to the depot. Figure 7 shows an example.

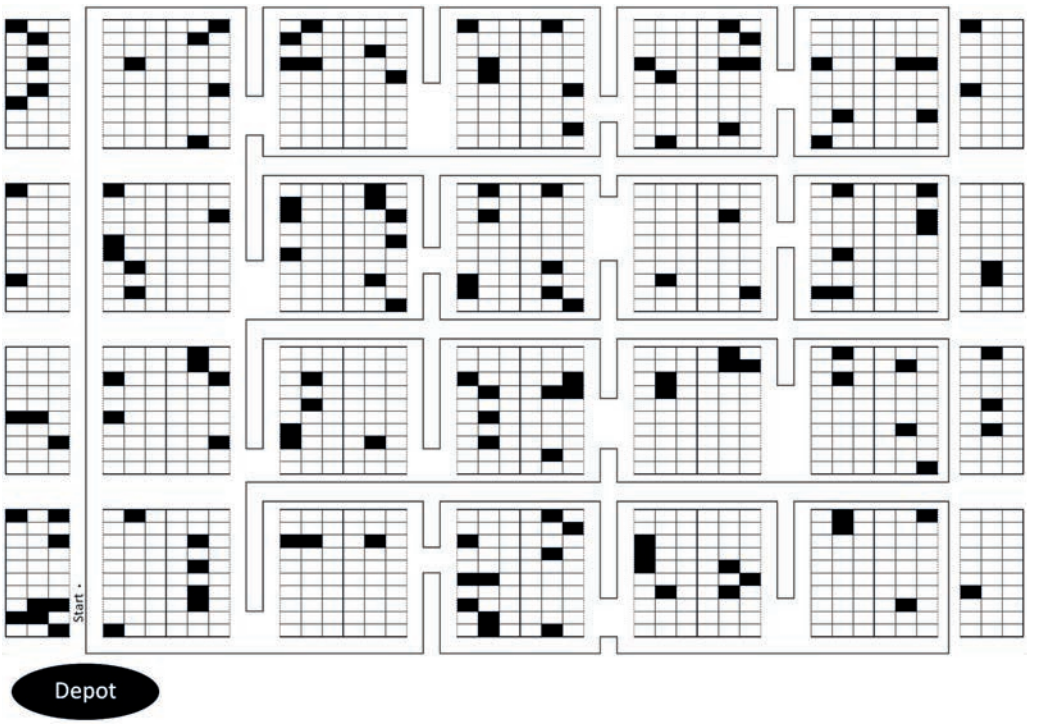


Figure 7: An example of a route applying Largest Gap

3.3 Midpoint

This routing policy is similar to Largest Gap; the main difference is that the picker identifies the product to pick closest to the center of each sub-aisle, which is considered the travel limit [13]; at first, the products that are in the upper half of the sub-aisle are picked up from the rear cross aisle, then, after picking up all the upper half products of the entire block, continues picking up the remaining products from the front cross aisle. If the product is exactly in the center, the picker takes it from either of the two cross aisles. In the end, the picker must return to the depot. An example is represented in Figure 8.

3.4 Return

When applying this routing policy, the picker enters and leaves the sub-aisle from the same cross aisle; this means that, after picking up the last product of the sub-aisle, the picker must return to the cross aisle [14].

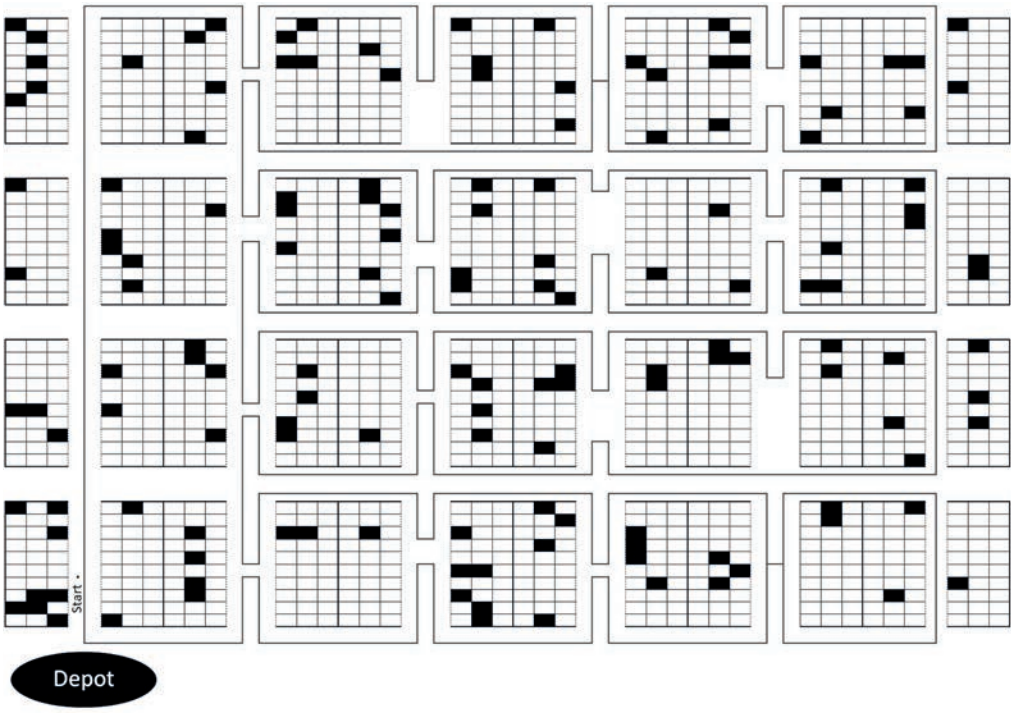


Figure 8: An example of a route applying Midpoint.

In the case that the warehouse configuration contains more than one block, the picker visits all the sub-aisles of the two adjacent blocks to the cross aisle alternately. After that, the picker moves to the next cross aisle that is adjacent to two unexplored blocks. The picker must return to the depot once all the products have been picked. This route is shown in Figure 9.

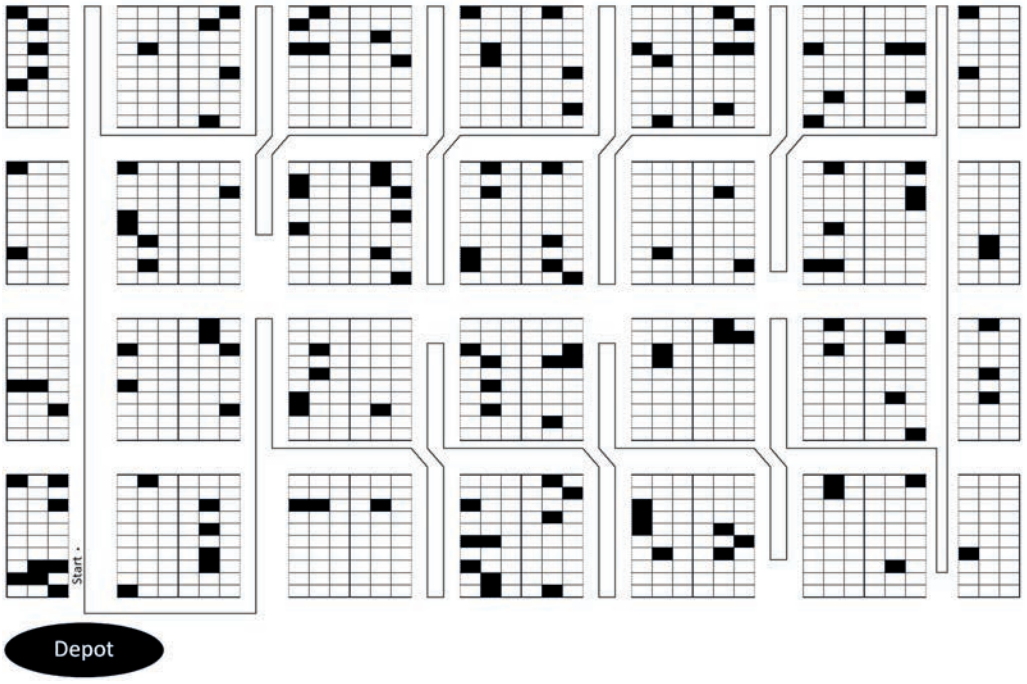


Figure 9: An example of route applying Return.

3.5 Combined

This routing policy is considered a combination of S Shape and Largest Gap policies. When all products are picked up completely, the picker must decide between (1) continuing through the front cross aisle, or (2) returning to the rear cross aisle [14]. This decision is made according to the shortest distance to the next product to pick up. An example of this route is shown in Figure 10.

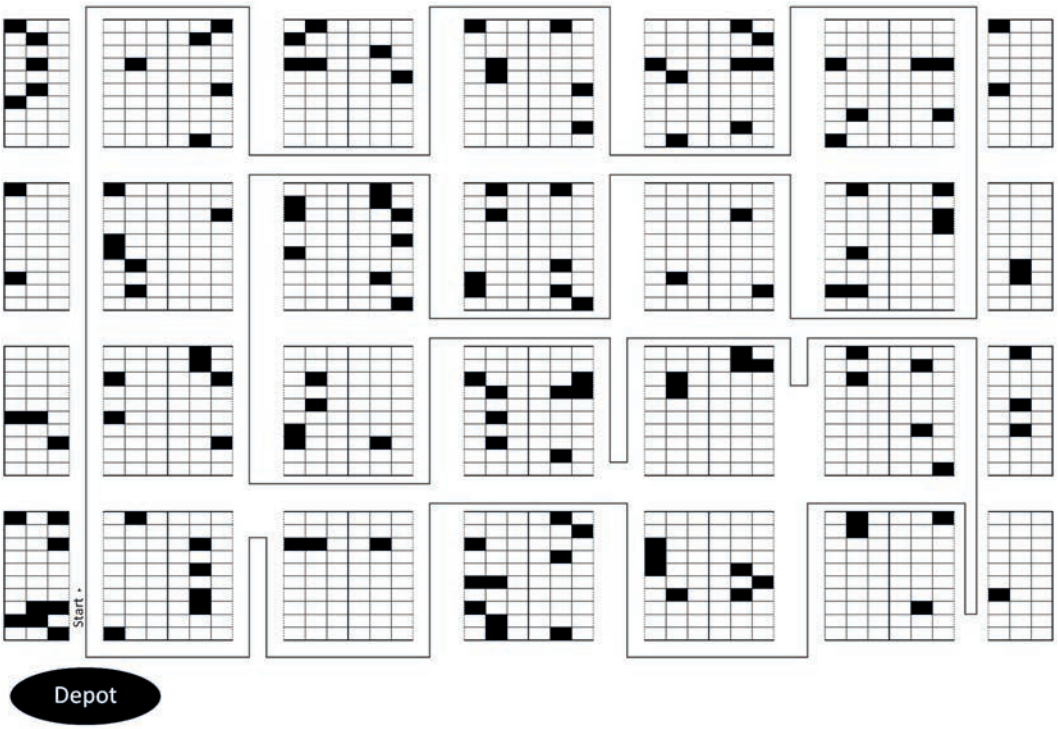


Figure 10: An example of a route applying Combined.

4. Development of Routing Policies

The necessary elements for the development and application of the algorithms for each heuristic will be defined below.

4.1 S-Shape

A fundamental part of the implementation of this heuristic is to define the order in which the picker will visit the sub-aisles, an example of the correct order according to its characteristics is shown in Figure 11.

Also, to help obtain the order, it is necessary to assign "auxiliary coordinates" to each sub-aisle. Figure 12 represents an example.

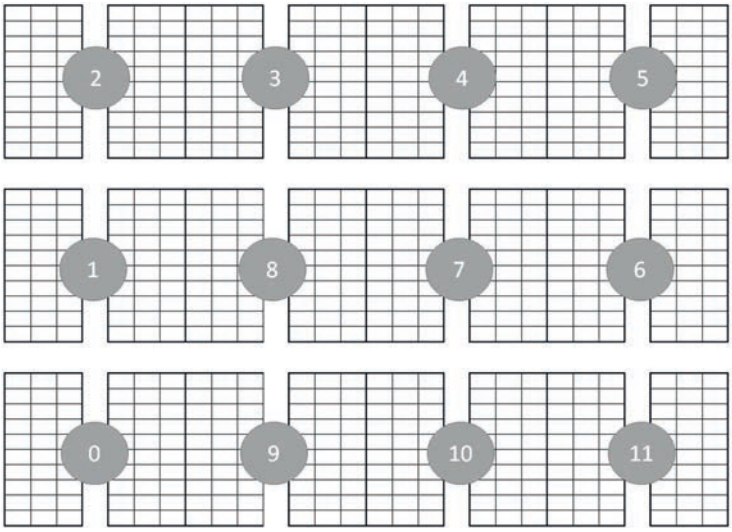
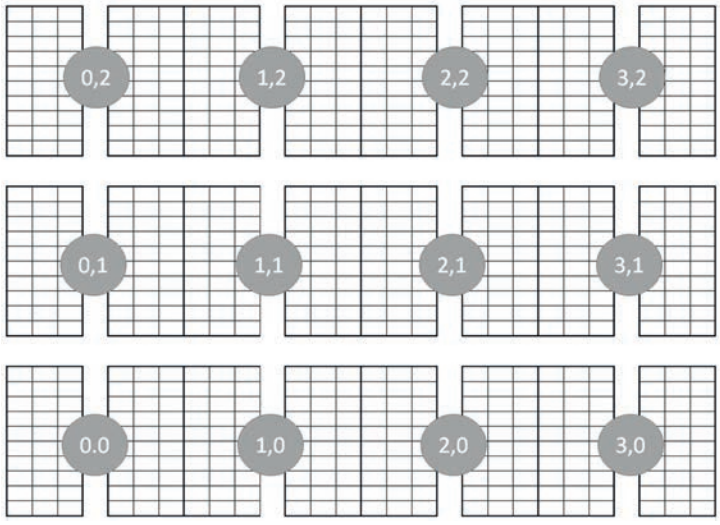


Figure 11: An example of sub-aisles order to visit for S-Shape.



$x_{max} = 4$

$y_{max} = 3$

Figure 12: An example of auxiliary coordinates for each sub-aisle.

The following equation returns the picking order of each sub aisle, given x and y coordinates:

$$f(x,y) = \begin{cases} y & \text{if } x = 0, \\ y_{max} + (x_{max} - 1)(y_{max} - (y+1)) + x - 1 & \text{if } y_{max} - (y+1) \bmod 2 = 0, \\ y_{max} + \big((x_{max} - 1)(y_{max} - (y+1))\big) + (x_{max} - (x+1)) & \text{otherwise}. \end{cases} \quad (1)$$

where:

y_{max} is the quantity of sub-aisles per aisle,

x_{max} is the quantity of sub-aisles per block,

x is the current picking aisle, and

y is the current block.

Once the order is defined, the next step is to obtain the direction in which the picker will pick products from each sub-aisle.

Algorithm 1 describes the procedure used to obtain the final path.

Algorithm 1. S Shape

Input: Sub-aisles *s* with at least one product to pick up, visit order of sub-aisles
Output: Final path *C*
1 **Begin**
2 **While** there is unexplored *s* with product **Do**
3 Select next *s* according to the visit order
4 If *s* is part of the first picking aisle
5 Add products in *s* in ascending order to *C*
6 Else if *s-1* was explored in an ascending direction
7 Add products in *s* in descending order to *C*
8 Else if *s*-1 was explored in a descending direction
9 Add products in *s* in ascending order to *C*
10 If the current block was explored completely
11 Add products in *s* in descending order to *C*
12 **While end**
13 **Return *C***
14 **End**

Where all the sub-aisles that are part of the first picking aisle are explored and added to the final path ascendingly (lines 4–5); then, the direction must alternate between each sub-block (lines 6–9) until the picker has visited the block completely; in that case, the first sub-block the picker visits in the new block is always traversed in a descending direction (lines 10–11).

4.2 Largest gap

The order in which the picker will visit the sub-aisles in this heuristic is different in comparison to S-Shape. In Largest Gap, each explored block starts and ends from the same sub-aisle. Figure 13 shows an example.

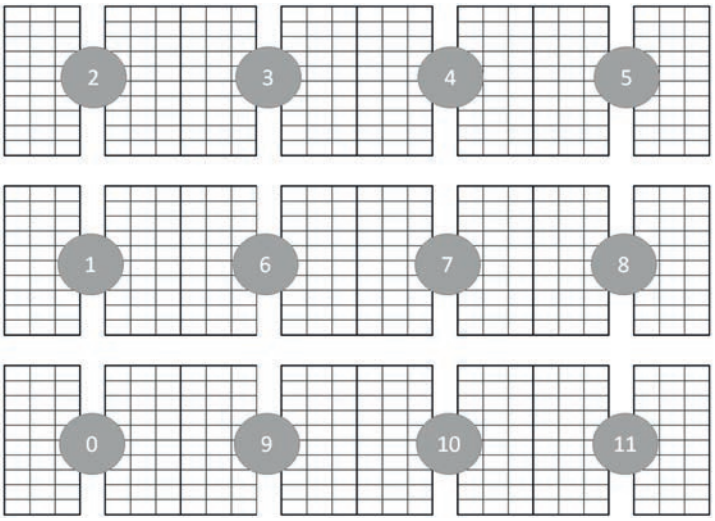


Figure 13: An example of sub-aisles visit order generated by Largest Gap.

The application of this equation only depends on whether the sub-block is on the first picking aisle; in any other case, the blocks are explored from left to right. The following equation represents this:

$$f(x,y)=\begin{cases} y & \text{if } x=0, \\ y_{\max}+(x_{\max}-1)(y_{\max}-(y+1))+x-1 & \text{otherwise}. \end{cases} \quad (2)$$

Once the order in which the picker will visit the sub-blocks is defined, the generation of the final route begins. This method is described in Algorithm 2.

Algorithm 2. Largest Gap

Input: Sub-aisles *s* with at least one product to pick up, visit order of sub-aisles
Output: Final path *C*

```
1  Begin
2      While there is unexplored s with product Do
3          Select next s according to the visit order
4          If s is part of the first picking aisle
5              Add products to C in an ascending direction
6          Else
7              Valuate elements in s
8                  Calculate the distance between the current element and the next one
9                      If the current distance is higher than the current limit
10                         The new limit is the current element.
11             Traverse elements in s
12                 If the current element is higher than the limit
13                     Add the current element to C
14                 Else
15                     Add the current element to pending
16         If the current block was explored completely
17             Add elements in pending to C.
18     While end
19     Return C
20 End
```

Where the direction in which the picker traverses the sub-aisles that are in the first picking aisle is ascending, adding the elements to the final path (lines 4–5). Then, the largest gap of each sub-aisle is calculated by the distance between each of the elements that it contains; the new limit is defined by the element that detects the highest travel distance (lines 8–10), the next step is to add the elements that are above this limit to the final path (lines 11–12) and the elements that are below are stored in a *stack* (14–15); once all the elements of the block—that are above this limit—have already been added to the final path, lines 16–17 insert the pending elements in LIFO order (Last In, First Out).

4.3 Midpoint

It is a heuristic similar to Largest Gap. They share the order in which the picker traverses the sub-aisles (Figure 13).
Hereunder, the algorithm designed for the generation of the final path for Midpoint:

Algorithm 3. Midpoint

Input: Sub-aisles *s* with at least one product to pick up, visit order of sub-aisles, locations per sub-block *LS*, locations per aisle *LA*.
Output: Final path *C*

```
1  Start
2       While there is unexplored s with product Do
3            Select next s according to the visit order
4            If s is part of the first picking aisle
5                 Add products to C in an ascending direction
6            Define the Midpoint value of the current block: LP-(LS/2)
7            Traverse elements on s
8                 If the current element is higher or equal to Midpoint
9                      Add element to C
10                Else
11                     Add element to pending
12           If the current block was explored completely
13                Add elements on pending to C
14                Update Midpoint value of the next block: Midpoint-LS
15      While end
16      Return C
17 End
```

Where the sub-aisles that are in the first picking aisle must be traversed in an ascending direction while adding all the elements to the final path. (Lines 4–5). After, to obtain the midpoint and take it as a limit (line 6), starting from the farthest block to the depot, this value is obtained as follows:

$$Mp = LA - \frac{LS}{2} \tag{3}$$

Where:

Mp is the midpoint,

LA represents the locations per picking aisle, and

LS represents the locations per sub-aisle.

This value is defined as a limit until there is a block change (lines 12 and 14), where it must be updated as follows:

$$Mp = Mp - LS \tag{4}$$

If the element is above the midpoint, it will be added directly to the final path, storing the rest on the pending elements (lines 8–11). The elements of the last sub-block of the block must be added completely in a descending way; so, it passes to the lower cross aisle and then starts adding the pending elements to the final path (line 13).

4.4 Return

The first step in the implementation of this routing policy is to define the order in which the picker will visit the sub-aisles; Figure 14 shows an example of the correct order according to the previously described properties.

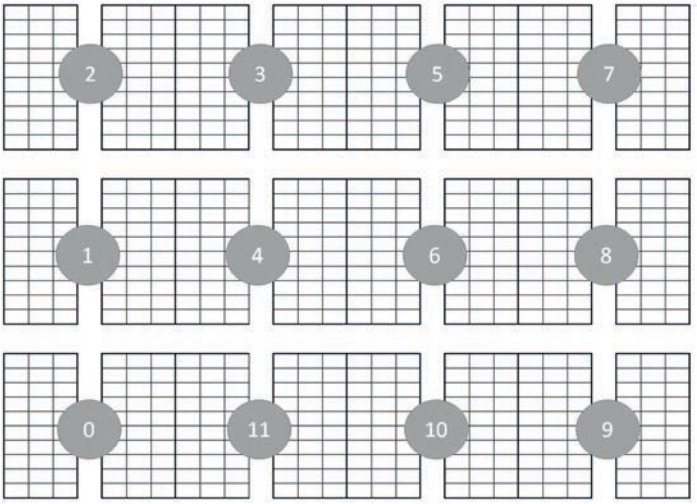


Figure 14: An example of sub-aisles order to visit for Return.

The following equations return the previously mentioned values:

$$g(x,y)=\begin{cases} y & \text{if } x=0, \\ h_1(x,y) & \text{if}(y_{\max}-(y+1)) \bmod 4<2, \\ h_2(x,y) & \text{otherwise}. \end{cases} \tag{5}$$

$$h_1(x,y)=\begin{cases} y_{\max}+(x_{\max}-1)(y_{\max}-(y+1) & \\ h_1(x-1,y)+1 & \text{if}(y_{\max}-(y+1)) \bmod 2=0, \\ & x=1 \\ h_1(x-1,y)+2 & \text{if}(y_{\max}-(y+1)) \bmod 2=0, \\ & y=0 \\ h_1(x,y+1)+1 & \text{if}(y_{\max}-(y+1)) \bmod 2=0, \\ & \text{otherwise} \end{cases} \tag{6}$$

$$h_2(x,y)=\begin{cases} y_{\max}+(x_{\max}-1)(y_{\max}-(y+1)) & \\ h_2(x+1,y)+1 & \text{if}(y_{\max}-(y+1)) \bmod 2=0, \\ & x=x_{\max}-1 \\ h_2(x+1,y)+2 & \text{if}(y_{\max}-(y+1)) \bmod 2=0, \\ & y=0 \\ h_2(x,y+1)+1 & \text{if}(y_{\max}-(y+1)) \bmod 2=0, \\ & \text{otherwise} \end{cases} \tag{7}$$

In this policy, the patterns of the order of travel are more complex than those seen before, so it was necessary to define a series of equations which consist of a function $g(x, y)$ on which you can obtain direct results and functions $h_1(x, y)$ and $h_2(x, y)$ where it is necessary to call them recursively to reach the desired result. For the application of these equations, it is necessary to use the auxiliary coordinates exemplified in Figure 12.

Once having the sub-aisles visit order, the process to generate the final path is shown in Algorithm 4:

Algorithm 4. Return

Input: Sub-aisles *s* with at least products to pick up, visit order of sub-aisles, locations per sub-block *LS*, locations per aisle *LA*.
Output: Final path *C*

```
1  Start
2      While there is unexplored s with product Do
3          Select next s according to the visit order
4          If s is part of the first picking aisle
5              Add products to C in an ascending direction
6          If the quantity of the blocks is an even number
7              If s is part of a block with an even coordinate
8                  Add elements in s to C in an ascending direction
9              Else
10                 Add elements in s to C in a descending direction
11         Else
12             If s is part of a block with a even coordinate
13                 Add elements in s to C in a descending direction
14             Else
15                 Add elements in s to C in an ascending direction
16             If s is part of the block on y=0
17                 Add elements on s to C in an ascending direction
18             If the current block was explored completely
19                 Add elements in s+1 to C in an ascending direction
20     While end
21     Return C
22 End
```

The elements found in the first picking aisle are added to the final path in an ascending direction (lines 4–5). Subsequently, it is important to define whether the number of blocks in the warehouse is even or odd (line 6). This fact influences because the characteristics of the routing policy states that the picker must alternate between the sub-blocks of two blocks, this implies that, in cases where the warehouse configuration has an odd number of blocks, the sub-blocks belonging to the last block to explore (closest to the depot) must be explored continuously (without alternating) (line 16). When the total number of blocks is even, all sub-aisles belonging to a block are crossed ascendingly, and in the odd ones in a descending direction (lines 6–10). In the opposite case (warehouses with an odd number of blocks), the sub-aisles that belong to the odd-number blocks are traversed in an ascending direction, and the sub-aisles that belong to the even-number blocks are traversed in a descending direction (lines 11–15). On the last block to be explored, all sub-aisles are visited ascendingly (line 16). In both cases, at the end of every block, the elements of the first sub-block of the next block (lines 18–19) are added to the final path in a descending direction.

4.5 Combined

The order of how to traverse the sub-aisles is similar to S Shape (represented in Figure 11), but there are cases where it can vary because of the characteristics of this routing policy; when picking up the last element of every block, it is important to define which is the sub-aisle of the next block with product that has the smaller distance, if this sub-aisle is on the left end, the path of the block turns in the direction from left to right; otherwise (if the sub-block with the nearest product is in the extreme right sub-aisle), the opposite direction must be taken. The following equation represents the above behavior:

$$f(x,y)=\begin{cases} y & \text{if } x=0, \\ y_{\max}+(x_{\max}-1)(y_{\max}-(y+1))+x-1 & \text{if } d_1<d_2, \\ y_{\max}+(x_{\max}-1)(y_{\max}-(y+1))+(x_{\max}-(x+1)) & \text{otherwise}. \end{cases} \tag{8}$$

Where d_1 is the distance between the last element of the block and the sub-block with the leftmost product and d_2 is the distance between the last element of the block and the sub-block with the product on the rightest location. Once having the order in which the picker will visit the sub-aisles, the final route is generated as presented in Algorithm 5.

Algorithm 5. Combined

Input: Sub-aisles *s* with at least one product to pick up, visit order of sub-aisles.
Output: Final path *C*
1 **Start**
2 **While** there is unexplored *s* with product **Do**
3 Select next *s* according to the visit order
4 **If** *s* is part of the first picking aisle
6 Add elements on *s* to C in an ascending direction
7 Capture the last element on *s*
8 Calculate d_1: the difference between the last element on *s* and the first one on *s+1* from the rear cross aisle.
9 Calculate d_2: the difference between the last element on *s* and the first one on *s+1* from the front cross aisle.
10 **If** d_2 is greater than d_1
11 Add elements on *s* to *C* in an ascending direction.
12 **Else**
13 Add elements on *s* to *C* in a descending direction.
14 **If** the current block was completely explored
15 Add elements in *s+1* to *C* in an ascending direction
16 **While end**
17 **Return *C***
18 **End**

Where the elements that belong to the first picking aisle are added to the final path in an ascending direction (lines 4–5), from this point, the distance from the last element of each sub-aisle to the first element of the next sub-aisle accessing from the front and rear cross aisle must be evaluated (lines 7–9). In the case that the distance to the first element from rear cross aisle is slower, the elements on the next sub-aisle have to be added in a descending direction to the final path; in the opposite case, the elements on the next sub-aisle are added in an ascending direction (lines 10–13).

4.6 Efficiency measurement

To measure the effectiveness of the result, it is necessary to calculate the distance matrix between all product nodes and artificial nodes by applying Dijkstra. Subsequently, a sum of all the distances between consecutive nodes on *C* is made, represented by the following equation:

$$d(C) = \sum_{i=1}^{p-1} D(x_i, x_{i+1})$$

where:

$C = <x_1, x_2, x_3, \ldots, x_p>$ is the sequence of elements to evaluate,

p is the number of vertices that forms a circuit, and

D is the distance matrix.

5. Experimental Results

In this section, the results obtained by this project are shown and interpreted.

A total of 125 different warehouse layouts were processed and defined according to the combination of the following values:

Number of products capacities per sub-block: 18, 30, 42, 54 and 60

Number of picking aisles: 3, 6, 9, 12 and 15

Number of cross aisles: 1, 3, 5, 7 and 9

In each layout, five routing policies were processed and applied to five different orders with 4, 7, 10, 12, and 15 percent of the full capacity of the products in the warehouse.

The results give a total of 3125 different instances (625 results per routing policy). This information was processed by SAS 9.4 software to generate a correlation analysis.

The rank is one of the different variables to consider, it represents the position obtained by each routing policy in comparison to the other ones. The first position is assigned to the one that gets the lower distance and the fifth the highest. The locations per sub-block with or without product, the quantity of picking aisles, the number of cross aisles, and the percent of total locations of the warehouse that contains products to pick up were other variables used.

5.1 Insights

Let us remember that, as the correlation gets closer to 1 or –1, the correlation is greater. Because this is a minimization problem, a correlation with the performance is better if it is negative. So, the main insights are:

- S Shape tends to be sensitive to the number of locations by sub-aisles (negative correlation of 0.41824) and the number of products in the warehouse (negative correlation of 0.24840) (Figures 15a and 16a).
- For Largest Gap, the correlation coefficient that stands out is the number of picking aisles, obtaining a positive 0.37988 (Figures 15b and 16b). Figure 15b demonstrated the tendency generated by this result, where the level of efficiency compared with the other policies decreases as this value becomes greatest.
- For Midpoint, the two most relevant variables are the number of aisles and the number of products to pick up, both with a positive correlation of 0.33772 and 0.26812, respectively (Figures 15c and 16c).
- The variable with the most effect over the performance of Return is the number of locations by aisle, where it gets a positive correlation of 0.58660. The more locations by aisle, the more competitiveness Return obtains. Figures 15d and 16d show that this policy gets better results as the number of aisles increases.
- Regarding the results of Combined, the number of locations by sub-aisle seems to be an important feature, with a negative coefficient of 0.48762, considerably higher compared to the other variables (Figures 15e and 16e show).
- Combined has better results in warehouses where the number of locations by aisle is the greatest variable, while the most degraded is Return.
- A high number of aisles tends to affect the performance of the five policies, but the policy with the most unfavorable results is Largest Gap. Being S-Shape and Combined the least affected.
- In the case of cross aisles, there was no improvement in the performance of any studied policy. The policies where its effectiveness decreases are Return and Combined. While in S Shape is just a little bit affected.
- S-Shape is the most benefited policy in warehouses where the number of products to be picked up increases.

To be able to develop this project, it was necessary to process orders, get five different results, and compare them over different methods. A benchmark of instances was synthetically created, and the performance in this wide range of different conditions was measured.

a) S shape

Pearson Correlation Coefficients, N = 625 Prob > |r| under H0: Rho=0				
	ProdPerSubBlock	Aisles	CrossAisles	OrderSize
Rank	-0.41824 <.0001	-0.12243 0.0024	0.04785 0.2360	-0.34840 <.0001

b) Largest Gap

Pearson Correlation Coefficients, N = 625 Prob > |r| under H0: Rho=0				
	ProdPerSubBlock	Aisles	CrossAisles	OrderSize
Rank	-0.13199 0.0010	0.37988 <.0001	0.14355 0.0004	0.16394 <.0001

c) Midpoint

Pearson Correlation Coefficients, N = 625 Prob > |r| under H0: Rho=0				
	ProdPerSubBlock	Aisles	CrossAisles	OrderSize
Rank	0.01836 0.6496	0.33772 <.0001	0.11537 0.0042	0.26812 <.0001

d) Return

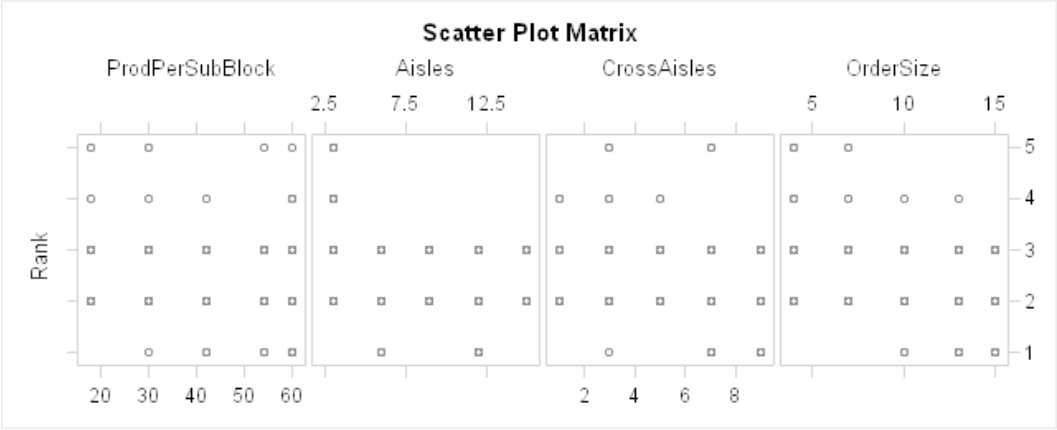
Pearson Correlation Coefficients, N = 625 Prob > |r| under H0: Rho=0				
	ProdPerSubBlock	Aisles	CrossAisles	OrderSize
Rank	0.58660 <.0001	-0.22508 <.0001	-0.18367 <.0001	0.08751 0.0300

e) Combined

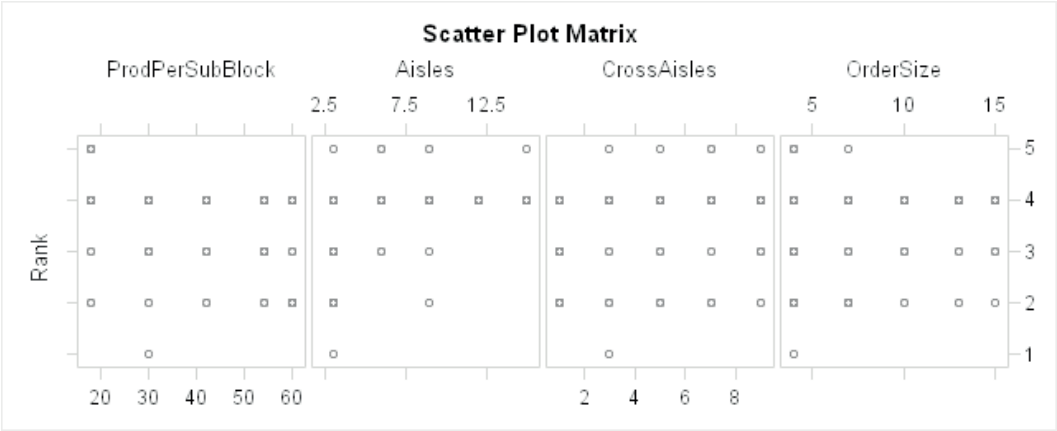
Pearson Correlation Coefficients, N = 625 Prob > |r| under H0: Rho=0				
	ProdPerSubBlock	Aisles	CrossAisles	OrderSize
Rank	-0.48762 <.0001	0.11624 0.0039	0.21212 <.0001	-0.03980 0.3245

Figure 15: Pearson correlation coefficient.

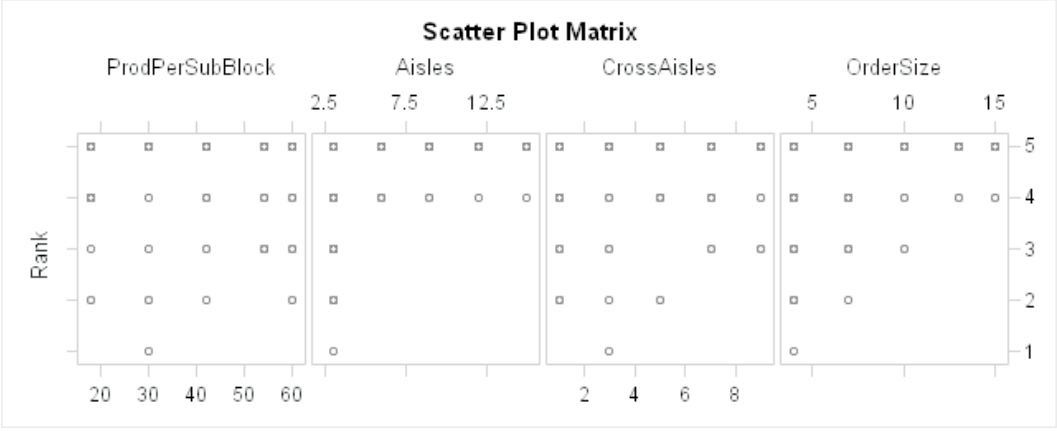
The main purpose of this project is to explain and get knowledge on policies and to reduce traveled distances in order picking processes in warehouses, offering an encouraging panorama for the construction of more complex routing policies.



a) S shape

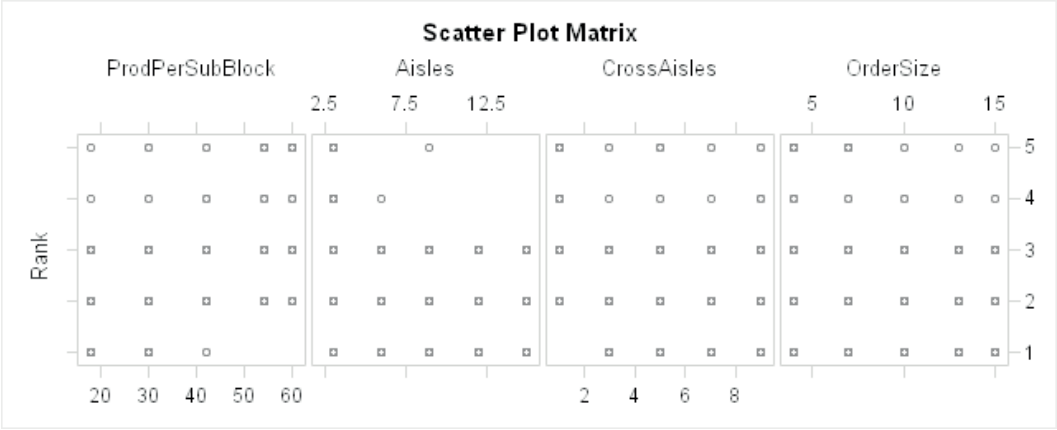


b) Largest Gap

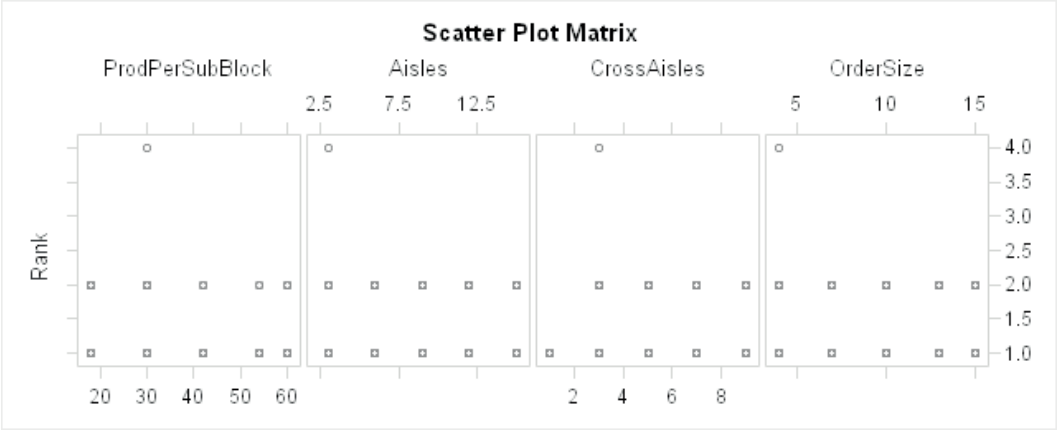


c) Midpoint

Figure 16 contd. ...



d) Return



e) Combined

Figure 16: Pearson correlation coefficient Scatter Plot Matrix.

6. Conclusions and Directions for Future Work

In this chapter, we studied and developed five routing policies to apply them in order to multiple scenarios and situations and generate enough information to find trends and understand the behavior of these heuristics.

These experiments provide evidence that the performance of these policies can be highly sensitive to the different characteristics of the warehouse; for example, in the case of the S Shape routing policy, its performance is mainly affected by (1) the number of locations per sub-block, and (2) the size of orders of products to be picked. In the case of Largest Gap, the most marked trend is defined by the number of aisles, affecting its performance. Similarly, although not as marked as in Largest Gap, in Midpoint, the number of aisles tends to reduce its efficiency. The Return policy seems to be highly sensitive to the number of locations per sub-aisle, with performance deteriorating in a highly marked way. On the contrary, Combined improves as the number of locations increases.

These insights can be used in the future to design a heuristic enriched with these key elements about routing policies. In this way, such a heuristic can foresee aspects of the test instances that can affect its performance and mitigate the consequences.

References

[1] Ochoa Ortiz-Zezzatti, A., Rivera, G., Gómez-Santillán, C., Sánchez–Lara., B. Handbook of Research on Metaheuristics for Order Picking Optimization in Warehouses to Smart Cities. Hershey, PA: IGI Global, 2019. doi.org/10.4018/978-1-5225-8131-4.

[2] Tompkins, J.A., White, J.A., Bozer, Y.A. and Tanchoco, J.M.A. 2010. Facilities planning. New York, John Wiley & Sons.

[3] Petersen, C.G. and Aase, G. 2004. A comparison of picking, storage, and routing policies in manual order picking. Int. J. Production Economics, 92: 11–19.

[4] De Koster, R., Le-Duc, T. and Roodbergen, K. 2007. Design and control of warehouse order picking: A literature review. European Journal of Operational Research, 182: 481–501.

[5] Theys, C., Bräysy, O., Dullaert, W. and Raa, B. 2010. Using a TSP heuristic for routing order pickers in warehouses. European Journal of Operational Research, 200(3): 755–763.

[6] Pansart, L., Nicolas, C. and Cambazard, H. 2018. Exact algorithms for the order picking problem. Computers and Operations Research, 100: 117–127.

[7] Scholz, A. 2016. An exacto solution approach to the single-picker routing problem in warehouses with an arbitrary block layout. Working Paper Series, 6.

[8] Henn, S., Scholz, A., Stuhlmann, M. and Wascher, G. 2015. A New Mathematical Programming Formulation for the Single-Picker Routing Problem in a Single-Block Layout, 5: 1–32.

[9] Ratliff, H.D. and Rosenthal, A. 1983. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Operations Research, 31(3): 207–521.

[10] Gu, J., Goetschalckx, M. and McGinnis, L.F. 2007. Research on warehouse operation: A comprehensive review. European Journal of Operational Research, 177(1): 1–21. doi.org/10.1016/j.ejor.2006.02.025.

[11] Hong, S. and Youngjoo, K. 2017. A route-selecting order batching model with the S-shape routes in a parallel-aisle order picking system. European Journal of Operational Research, 257: 185–196.

[12] Cano, J.A., Correa-Espinal, A.A. and Gomez-Montoya, R.A. 2017. An evaluation of picking routing policies to improve warehouse. International Journal of Industrial Engineering and Management, 8(4): 229–238.