*International Journal of Combinatorial Optimization Problems and Informatics, 12(2), May-Jun 2021, 32-46. ISSN: 2007-1558.*

www.editada.org

# Genetic algorithms for route optimization in the picking problem

*Heriberto Casarrubias Vargas, Noé Vásquez Godínez, Maribel Chavéz Hernández, Gabriela Gaviño Ortiz*

Centro Universitario UAEM Valle de México, Boulevard Universitario s/n Predio San Javier. Ciudad López Mateos. Atizapán de Zaragoza. Estado de México.
hcasarrubiasv@uaemex.mx

**Abstract.** The picking problem is part of the item picking problem in a warehouse for a given order and is the most expensive in the process; Optimizing the item collection time is essential because it is a bottleneck in the service rate of orders fulfilled and has a significant impact on the efficiency of the supply chain. This work proposes a strategy to reduce the picking time by minimizing the route using a genetic algorithm; a comparison is performed with other heuristic strategies in warehouses with different morphologies and considering orders of different sizes.

**Keywords:** Genetic Algorithms, Picking Problem, Travel Time Optimization, Heuristic algorithms

Article Info
*Received May 16, 2020*
*Accepted May 1, 2021*

## 1 Introduction

Warehouses are an essential component for managing any logistics operation and supply chain management; storage controls the movement, shipment, receipt, and collection of merchandise. They are the component responsible for linking suppliers, production systems, distribution systems, and points of sale; Warehouse performance significantly affects the supply chain [1], so it is essential to establish policies on the operation of the warehouse that improve its efficiency.

In warehouses, the different activities carried can be classified and distinguished into storage, receipt of goods, receipt of orders, preparation of orders, classification, packaging, and shipping. Each of these activities depends on the context and design of the warehouse.

In warehouses, spatially distributed items must be collected; a quite popular system is the so-called picker to parts where the aisles of the warehouse must be travelled to retrieve the products stored on shelves to satisfy a customer's order. Bozer & Kile [2] mentions that each item is distributed in different warehouse locations, which implies a displacement and planning for its collection.

Among the different activities to be carried out in a warehouse, the picking or collection process is the most laborious and repetitive, generating 50% to 70% of the warehouse's operating costs [3]. The picking process time is broken down into travel time 50%, search time 20%, collection time 15%, preparation time 10% and others 5% [4].

The picking problem can be approached analytically using an integer programming model to achieve the optimal solution to the problem. In the work of Valle, Beasley, & da Cunha, [5] a model is proposed that expose two phases, calculating the shortest route between two locations in a warehouse and as a second phase focuses on the optimization problem of finding the best selection sequence to minimize the distances traveled, this is like combinatorial optimization problems, these problems expose high levels of complexity.

The complexity of algorithms is classified into two classes, one of which is of type P whose solution algorithms are polynomial in time; and the other classification is NP problems, whose solution are hard to found using efficient deterministic algorithms; but which can be solved by non-deterministic or heuristic algorithms [6].

The picking problems are NP-Hard type. Heuristics and metaheuristics are used to find the solution; Considering obtaining the optimal travel solution to prepare each order becomes impractical due to the computational cost, the time required, and the time required volume of orders over a period.

## 2 Heuristic strategies for picking

Considering the impact of the cost in travel time on the operation of a warehouse and the cost of finding the optimal solution, heuristic techniques have been developed to planning the route or collection of items of an order that help to decrease the collection time [7]. These intuitive strategies have different travel time costs and generally do not obtain the optimal time. However, they are very usual in practice due to the low cost of implementation and the time required for planning.

Return
For each item to collect, you go to the last item in the aisle, once you take it, go back to the beginning of the aisle and continue with the rest.

S-Shape
The S-shaped strategy, transversal strategy, leads to a route in which the corridors, which must be visited, are completely travelled. Aisles, where there is nothing to collect, are omitted. Therefore, the corridors its visit in an S shape.
The collector enters the aisle from one end and exits the aisle from the other end, starting on the left side of the warehouse. After choosing the last item, the order picker returns to the front of the aisle. This strategy is common because it is straightforward to use and understand.

Largest Gap
In the Largest Gap strategy, the first and last corridors containing items to be collected are completely covered. The picker goes through the other aisles without crossing the largest space between each pair of neighbouring items to collect within the same aisle; this space is the largest gap. The heuristic largest gap is especially useful when the time to change aisles is short, and the number of selections per aisle is low.

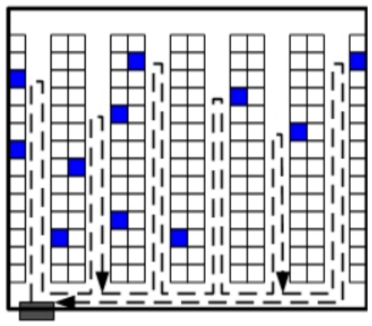
*Figure 3 Return Strategy*

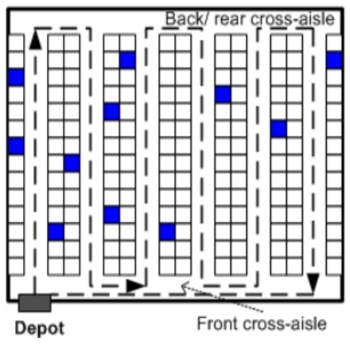

*Figure 2 S-shape strategy*

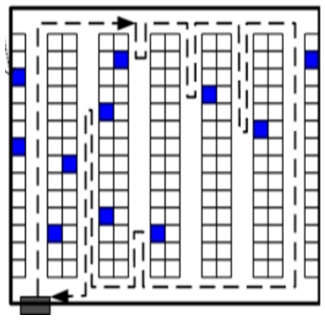
*Figure 1 Largest Gap strategy*

## 3 Warehouse layout

The organization and distribution of space in different operating areas include the design of warehouses; in general, a warehouse has reception, quality control, adaptation of loading units, storage, order preparation, and dispatches. A suitable choice of the distribution of the storage area affects the performance of the operations [8] in general, the choice of the distribution of the warehouse depends on the particular operation.

For the picking problem, the relevant area is the storage area, which is generally considered a rectangular arrangement containing the racks, places where the items are stored, and the corridors where you can walk to collect items.

The part of the warehouse between two adjacent transverse corridors is called a block, and the corresponding part of the corridor through which one can transit is denoted as an aisle or corridor. Storage locations are identical in size and arranged on either side of the aisles, whereby order pickers are enabled to enter and exit to pick items. From this scenario, various solution strategies to the picking problem are analyzed. In the present study, a design of cells with homogeneous storage capacities and a distribution parallel to each other is considered to form a warehouse block; The dispatch point in the upper left is considered, where picking the different items within the warehouse is considered begins and ends. In the example, in Figure 5, the warehouse layout has four vertical and three horizontal aisles.

The parameters that define a layout are:

1. Number of racks, the set of contiguous cells that store the items.
2. Number of cells for each rack.
3. Number of vertical corridors.
4. Front corridor and bottom corridor, first and last horizontal corridor; for a given layout it is considered that at least this pair of corridors exist.
5. Number of horizontal aisles, at least two for every layout.

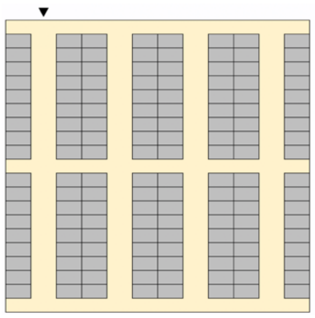

*Figure 4 Warehouse layout, the inverse triangle marks the dispatch point.*

To compare the performance of the picking strategies; the location of the items is distributed with a uniform random distribution, and it is considered a set of layouts with the following parameters:

1. Number of horizontal corridors: 3, 4, 5, ..., 10
2. Number of vertical corridors: 3, 4, 5, ..., 10
3. Number of items in order: 5, 10, …, 50

Each experiment is given by the combination of parameters, horizontal aisles, vertical aisles, and the number of items in one order; four algorithms are evaluated for each experiment: S-shape, Return, Largest Gap, and a solution approach based on a genetic algorithm described below.

## 4 Picking based on genetic algorithms

Genetic algorithms use a strategy based on a heuristic search directed in the space of possible solutions, and the phenomenon of biological evolution inspires them, the first proposal of John Henry Holland [9] has become popular for problems where the analytical solution is impractical.

For genetic algorithms, several elements are required. First, the coding of the problem in a particular representation called phenotype that contains the semantics of the solution to the problem posed. Also, a set of individuals or solutions called population, validation of the individual as a solution to the problem, and operators on the solutions or individuals, mutation operator, and crossover operator working on the solution space to generate new individuals with potentially better performance.

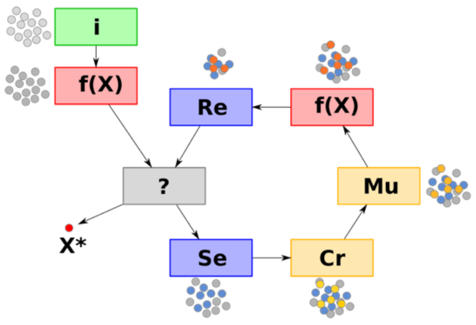

*Figure 5 Genetic Algorithm:i initialization, f(X): evaluation, ?: stop condition, Se: selection, Cr: crossover, Mu: mutation, Re: Replacement, X*: best solution.* [15]

The operation of the genetic algorithm is shown in Figure 6, where the different components are illustrated: set of initial solutions, objective function, operators to generate new solutions based on current solutions, and a stop criterion or when to stop searching for the next best solution.

The genetic algorithm is an iterative optimization procedure that starts from a set of admissible solutions, called the initial population; For the picking problem, an admissible solution is the order of collection of items in the warehouse that is encoded as a vector with numerical entries, which is called the individual or phenotype.

Later, order the solutions according to the efficiency, which is called the aptitude or quality of the individual. Based on the quality of the solutions, it builds a set of new solutions, next generation, taking into account the best characteristics of the individuals in the population. In genetic algorithms, the inheritance of characteristics between generations is a fundamental component and emulated using a binary operator called crossing, and a unary operator called mutation.

In the following subsections, we will detail how to model the warehouse picking problem.

### 4.1 Representation of the travel problem as phenotype

The genetic algorithm requires coding the problem into a vector of characteristics regularly with entries that take integer values, which define a solution to the picking problem for the items of an order.

Due to complexity considerations in the size of the problem and based on the heuristics used in strategies like S-shape, Return, and Largest-Gap; it's proposed one partition of the warehouse area.

Let a set $X$, a partition of $X$ , denoted by $\boldsymbol{P}(X)$ is a family of subsets $Xi$ with the following properties:

- $Xi \qquad = \emptyset$
- $\bigcup i\ Xi = X$

In the warehouse the elements *Xi* in the partition are the same size to get a homogeneous pathlength over each partition element; To the elements in the partition, we denominate them as sites.

The partition of the warehouse area is established as follows, for each rack in the vertical aisle is segmented into three zones of the same length, and for the horizontal aisles of the warehouse, an area covering half of the previous block is associate with the middle of the next block, as shown in Figure 7. Partitioning is applied to generate a level of discretization over the warehouse and allows reducing the search time for the picking route, due it considers a smaller space in factor 1: 3 in the racks.

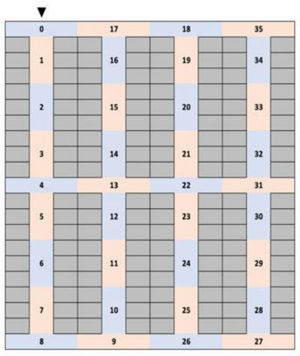

*Figure 6.- Warehouse partition, dispatch point is mark with a triangle inverted*

Note that partitioning in three regions is independent of the length of the rack. In the case of racks of length greater than nine, the partitioning of the rack keeps in three regions, two adjacent to the horizontal aisles and one central. For warehouses with more racks, the partitioning level not change; however, more partitions would be taking on the warehouse layout.

For the search of suitable picking routes; A two-step strategy is used; first, the partitioning zones it is determined, followed by an optimization of picking route over the partition zones.

For an order $O = \{item_1, item_2, \ldots, item_n\}$ with *n* ítems the route to travel, it is encoded using the partitioning and sort by picking preference. Let the el $item_i$ in the $P_{k(i)}$ site; A sequence or order to visit the sites in the warehouse can be represented using the following vector:

$$[\ P_1 \quad P_{k2} \quad P_{k3} \quad P_{k4} \quad P_1\ ]$$

Note that all tours start and end at site 1, the dispatch point. The order of the sites in the vector is interpreted as a priority over which block is more important to visit first, which implicitly establishes the order of collection of items.

For example, consider the items in the order $O = \{\ item_1, item_2, \ldots, item_8\ \}$ spatially distributed in the racks and partitioning zones as shown in Figure 8. The set of sites to consider encoding the travel problem as an individual for the genetic algorithm considers the sites:

$$\{\ P_1, P_5, P_{19}, P_{25}, P_{33}, P_{32}\}$$

The visit to the following sequence of sites gives a codification as a solution to collect the items:

$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5 \rightarrow P_6 \rightarrow P_7 \rightarrow P_8 \rightarrow P_{26} \rightarrow P_{25} \rightarrow P_{24} \rightarrow P_{23} \rightarrow P_{22} \rightarrow P_{21} \rightarrow P_{20} \rightarrow P_{19} \rightarrow P_{18} \rightarrow P_{35} \rightarrow P_{34} \rightarrow P_{33} \rightarrow P_{32} \rightarrow P_{31} \rightarrow P_{22} \rightarrow P_{13} \rightarrow P_4 \rightarrow P_3 \rightarrow P_2 \rightarrow P_1$

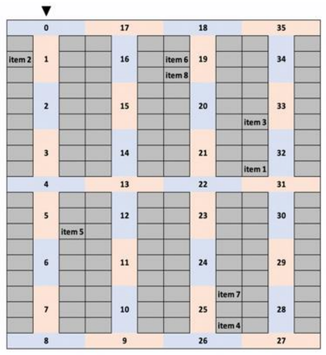

*Figure 7 Example of an order of items distributed in the warehouse layout*

For the type of layout, the permissible displacements given a position in a site are horizontal or vertical movements, a Manhattan type-world. The previous representation can be simplified, considering only the points of change direction in the path. The previous sequence can be equivalently denoted by

$P_1 \rightarrow P_8 \rightarrow P_{26} \rightarrow P_{18} \rightarrow P_{35} \rightarrow P_{31} \rightarrow P_4 \rightarrow P_1$

The previous path has a length of 76 and visits the sites that contain the items in the following order:
$[\ P_1, P_5, P_{25}, P_{19}, P_{33}, P_{32}\ ]$

Another possible route in the warehouse can be represented as:

$P_1 \rightarrow P_8 \rightarrow P_{26} \rightarrow P_{25} \rightarrow P_{26} \rightarrow P_{27} \rightarrow P_{35} \rightarrow P_{18} \rightarrow P_{19} \rightarrow P_{18} \rightarrow P_1$

It has a length of 66, and the order of visit of the sites containing the items is:

$[\ P_1, P_5, P_{25}, P_{32}, P_{33}, P_{19}\ ]$

Note from the above trajectories:

1. The trajectories have variable lengths for the same list of sites to visit or equivalently for collecting items.
2. The length of the route is calculable from the representation in phenotypes.
3. The length of the path is calculable from the partitioned representation.

The phenotype of the length variable in the genetic algorithm is expensive; to reduce the length in the representation, only includes the order of visits for the sites. The individuals are permutations on the priority of visit in the sites that contain the items to be collected; This approach does not specify the path to follow from one partition element to another, it does not describe the list of intermediate sites to traverse. However, considering that the aim is to minimize the length of the path, the shortest path between pairs of sites can be used.

The minimum distance between sites pairs can be pre-calculated using matrix algebra, based on the incidence matrix associated with the warehouse layout, which is defined as follows:

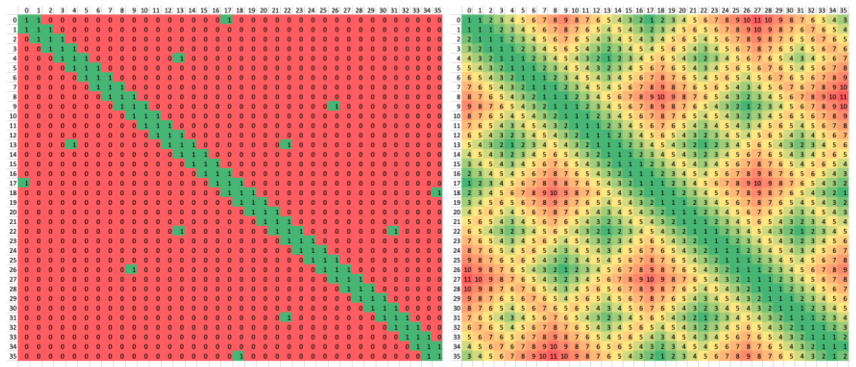

*Figure 8 Left, visualization the adjacency matrix for the layout in the Figure 6. Rigth, visualization of minimum distances between elements of the partition. The partition number is displayed at the edges.*

The adjacency matrix associated with the layout in the warehouse is a square matrix, $A(aij)$, of dimension $n$, where $n$ is the number of sites in the layout partition, where the entry $aij$ of matrix $A$ is defined as one if the site $i$ of the partition is contiguous to site $j$ of the partition. For example, in the layout of Figure 8, the site $P_0$ of the partition is contiguous to the site $P_{17}$ and $P_1$, but it is not contiguous to the site $P_2$.

The powers of the adjacency matrix of the layout corresponding to the warehouse contain information about the paths between the partition sites. For a power $k$ of the adjacency matrix $A$, denoted by:

$$P(p_{ij}) := A^k = \underbrace{A \times A \times \ldots A}_{k-times}$$

The value of the entry $p_{ij}$ in the matrix $P$ takes the value of the number of paths that exist of length $k$ between the site Pi and Pj elements. Based on this property [11] the minimum path length between sites can be determined.

The power matrix corresponding to the adjacency matrix contains the minimum path length between two different sites [10]. The maximum power to calculate that determines the minimum path length corresponds to the cardinality of the set of sites in the layout partition.

```
Input: W,    Warehouse layout
Output: L,   matrix of minimum distances between the sites of the layout
---------------------------------
A ← partition adjacency matrix in layout W
N ← number of blocks in the layout
L ← zero matrix of size NxN

temp ← A
for p in  1 to N
    temp ← temp * A
    for ii in 1 to N
        for jj in in 1 to N
            if temp(i,j) == 0
                continue
            if L(i,j) ==  0
                L(i,j) ← p
```

*Algorithm 1.- Calculation of the minimum distance matrix*

For the example in Figure 7, the maximum power is 35, because there are 35 blocks in the partition. This minimum distance matrix can be pre-calculated using Algorithm 1 and serves for any order of items that need to be optimized.

Figure 8 shows the adjacency matrix and the minimum path length matrix between partition elements, the distance values are displayed as a heat map; Notice the patterns that form due to the spatial distribution of the layout.

The minimum distance matrix provides the minimum path length; however, it does not provide the sequence of sites that make up the minimum length path. To find the minimum length path, see Figure 9, between pairs of elements $Pi$ and $Pj$ different scenarios are considered for the position of the partition elements $Pi$ and $Pj$ in the warehouse layout, which are list below:

1. The sites are in the same vertical corridor
2. The sites are between the same horizontal aisles, and they are on the same line of horizontal racks
3. The sites are in different rows of horizontal racks

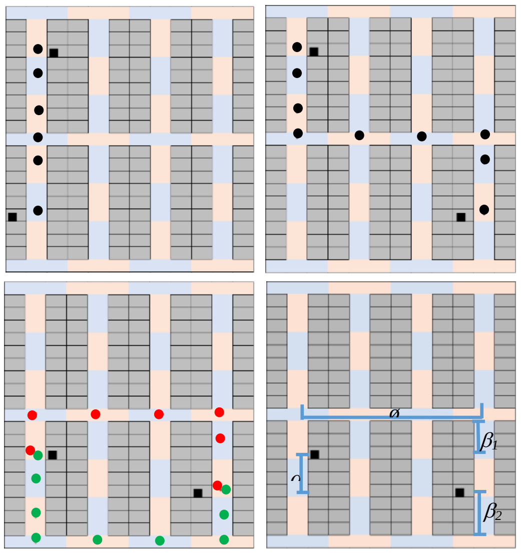

*Figure 9.- Spatial distribution cases for the calculation of the minimum trajectory between pairs of elements of the partition. Upper left corner case 1, upper right corner case 2, lower-left corner case 3, lower right corner detail of distances for case 3*

In the first case, the minimum trajectory is trivial; the second case, the minimum path is obtained by traveling vertically towards the first horizontal corridor in the direction of the destination site, after moving horizontally towards the vertical destination corridor and finally moving vertically towards the destination site; The route using this policy is minimal since a Manhattan layout is assumed. There are two routes to consider in the third case: take the route through the horizontal corridor above or through the horizontal corridor below. The distance is different depending on the distances to the ends, see Figure 9 images of the bottom line. The path above has a length of:

$$\beta 1+\phi+\alpha$$

and the path through the corridor below has a length of:

$$\alpha+\phi+\beta 2$$

analyzing both trajectories in general, the trajectory above its shorter if:

$$\beta 1+\phi+\alpha \leq \alpha+\phi+\beta 2$$

by canceling common terms; the previous inequality is accomplished if and only if:

$$\beta 1 \leq \beta 2$$

Therefore, if $\beta 1 \leq \beta 2$ the minimum path to reach $Pj$ is through the upper corridor, otherwise the route with minimum distance is through the below corridor.

With the above, given two sites of the partition $Pi$ and $Pj$ in the layout of a warehouse, we can always find the minimum route that connects them. For the encoding of the phenotype in the genetic algorithm, it is sufficient to represent the visit order of the sites and not is necessary the complete path.

The approach used to minimize the total path is based on the minimization of the path between pairs of sites, a dynamic programming method supported by Bellman's optimality principle [12] that establishes:

*"An optimal decision sequence that solves a problem must fulfill the property that any subsequent subsequence of decisions, which has the same final state, must also be optimal concerning the corresponding subproblem."*

For the coding of the phenotype of a given order $O = \{item_1, item_2, \ldots, item_n\}$ first the sites to visit are obtained

$$\{P1, P2, \ldots, Pk\}$$

and later an individual is represented as a permutation of the sites to visit:

$$(P_{\pi(1)}, P_{\pi(2)}, \ldots, P_{\pi(k)})$$

where the order of appearance in the tuple is interpreted as the order of visit of the site in the collection of items in the warehouse; the space of possible permutations we denote by $\Omega_k$ with $k$ the number of sites to visit.

The search space for solutions is combinatorial size as a function of the number of sites to visit given an order $O$. If the number of sites to visit is $N$, the solution space has size $N!$, the size of the solution space $\Omega_k$ grows combinatorically in relation to the number of sites to visit associated with the items in an order $O$.

## 4.2 Crossover operator

The genetic algorithm requires a binary operation on the solution space, which, given two solutions encoded as phenotypes, generates a new solution inheriting properties from the parents. The crossover operator, $\bowtie i$, is a binary operation defined on the individuals of the population, $\Omega k$, dependent on the cut-off point $i$:

$$\bowtie_i : \Omega k \times \Omega k \rightarrow \Omega k$$

The definition is based on the operator of [13] with the variant of considering the crossover operator in terms of the parent's travel priority. Let an order $O$, and $P1, P2, \ldots, Pm$ sites to visit and let $X$, $Y$ solutions:

$$X{:}(P\pi x1, P\pi x2, \ldots., P\pi xm)$$
$$Y{:}(P\pi y1, P\pi y2, \ldots., P\pi ym)$$

with $\pi x$, $\pi y \in \Omega m$, $\pi xn$ el $n$-th element of the order induced by the permutation $\pi x$ and $\pi yn$ the nth element of the order induced by the permutation $\pi y$. Let $i$ the cut-off point; then $\bowtie i$ is defined as follows:

$$\bowtie i(X,Y) \longmapsto Z(\ (P\pi x1, P\pi x2, \ldots., P\pi xm), (P\pi y1, P\pi y2, \ldots., P\pi ym)\ )$$
$$\longmapsto (\ P\pi x1, \ldots, P\pi xi, P\pi y(1), \ldots, P\pi y(m-i)\ )$$

where:

- $P\pi y(1)$ is the first entry of $Y$ such as $P\pi y(1)$ is not in $Xi := \{P\pi x1, \ldots P\pi xi\}$
- $P\pi y(2)$ is the first entry of $Y$ such as $P\pi y(2)$ is not in $Xi \cup \{P\pi y(1)\}$
- …
- $P\pi y(m-i)$ is the first entry of $Y$ such as $P\pi y(m-i)$ is not in $Xi \cup \{P\pi y(1), \ldots, P\pi y(m-i-1)\}$

With the crossover operator defined above, the priority of $X$ is considered first up to the entry $i$ at the phenotype, and then the priority of individual $Y$ is considered for the remaining sites. For example, if the sites to visit are $\{P1, P5, P19, P25, P33, P32\}$, let the solutions X, Y be given by:

$$X = [\ P1, P5, P25, P19, P33, P32\ ]$$
$$Y = [\ P33, P19, P1, P25, P32, P5\ ]$$

And let the cut-off $i$=3 then:

$$Z = X \bowtie_3 Y = [\ P1, P5, P25, P33, P19, P32]$$

for the cut-off point $i$=2 we obtain:

$$Z = X \bowtie_2 Y = [\ P1, P5, P33, P19, P25, P32]$$

## 4.3 Mutation operator

The mutation operator is a unary operation on the solution space, which makes a change in the phenotype of the individuals, that is, it makes a change in the order of travel by swapping the position of a pair of sites for a given path.

Let an order $O$, and $P1$, $P2$, …, $Pm$ sites to visit and let $X$ a solution given by:

$$X{:}(P\pi x1, P\pi x2, \ldots, P\pi xm)$$

then it is defined $\propto_{(i,j)}$,the mutation operator over $\Omega_m$, as follows:

$$\propto(i,j)\ (X) \longmapsto Z(\ P\pi x1, \ldots, P\pi xi\ , \ldots, P\pi xj, \ldots, P\pi xm)$$

$$\longmapsto (P\pi x1, \ldots, P\pi xj, \ldots, P\pi xi, \ldots, P\pi xm \qquad )$$

The pair of indices $(i,j)$ are taken arbitrarily for each individual and iteration of the genetic algorithm.
For example, if the sites to visit are $\{P_1, P_5, P_{19}, P_{25}, P_{33}, P_{32}\}$, and let $X$ a solution given by:

$$X = [\ B_1, B_5, B_{25}, B_{19}, B_{33}, B_{32}]$$

and the mutation point is $(i=5, j=2)$, then:

$$\propto_{(2,5)}(X) = [\ B_1, B_{33}, B_{25}, B_{19}, B_5, B_{32}\ ]$$

for the mutation point $(i=1, j=6)$ we obtain:

$$\propto_{(1,6)}(X) = [\ B_{32}, B_5, B_{25}, B_{19}, B_{33}, B_1]$$

Note that $\propto_{(i,j)}(X) = \propto_{(j,i)}(X)$ for all $i, j$.

## 4.3 Fitness operator

The fitness operator measures the quality of solutions to take the phenotype that encoded the sites to visit and compute the distance minimum associated with the trajectory. For compute minimum distance, the matrix L is used, calculated according to Algorithm 1.

Let an order $O$, and $P1$, $P2$, …, $Pm$ sites to visit and let $X$ a solution given by:

$$X{:}(P\pi x1, P\pi x2, \ldots, P\pi xm)$$

then it is defined $\sqcup(i,j)$, the mutation operator over $\Omega_m$, as follows:

$$\sqcup(X) \longmapsto v \in R$$

$$(P\pi x1, \ldots, P\pi xi, \ldots, P\pi xm) \longmapsto \textstyle\sum_{i<m} \quad L(P\pi xi, P\pi xi{+}1) + L(P0, P\pi x1) + L(P0, P\pi xm)$$

where $L(Px, Py)$ denotes the entry in the minimum distance matrix associated with the sites elements. The aptitude operator is a real-valued function and allow sort the solutions by the inverse travel distance.

## 4.4 Selection and reinsertion

Initially, the genetic algorithm creates a population of $K$ randomly individuals of $\Omega k$, the associated permutations space of an order $O$, and between each iteration $r$ individuals are selected to inherit their characteristics to the next generation.

The individuals are selected by the roulette method; which assigns a probability to each individual in base on their aptitude [14]. The roulette method takes r individuals randomly; with this selection method, the best individuals have the best opportunity for selection while still considering the worst individuals to give variability to the population.

The selected individuals inherit their characteristics by the crossing operator; each of them randomly crosses with some individuals from the total population. Also, the selection operation establishes which part of the population to be applied mutation operator. In this way, a part of the individuals that form the next generation is obtained.

In this work, the percent of individuals generated by the cross operator is 40% of the total population; and the mutation operator generates 20 % of the total population. The best individual from the previous generation is too selected for the next generation; this is called elitism.

Since the population size between iterations is always constant, it is necessary to generate 40% of the population of new individuals and randomly take it from Ω*k*. With this, variability in the population is allowed, and local minima are avoided in the optimization problem.

## 5 Results

The genetic algorithm used to perform out the experiments proposed in the Layout section of the warehouse uses the following fixed parameters for the genetic algorithm:

- Population size: 100 individuals.
- Probability of crossing: 20%.
- Probability of mutation: 15%.
- Reinsertion: elitism, the best individual from the previous generation, survives in the new generation.
- Number of iterations: 1000 iterations maximum or stop the iterations if in the last 100 generations there is no improvement in the solution.

Each experiment consists of a tuple of values (h, v, i) that take values in the following ranges:

- *h* the number of horizontal aisles, takes values *3, 4, 5, ..., 10*
- *v* the number of vertical corridors, takes values *3, 4, 5,…, 10*
- *i* denotes the number of items to consider in the order, takes values *5, 10, 15, ... 50*

The possible combinations above are *8 x 8 x 10 = 640*; each possible combination of horizontal and vertical aisles and the number of items to collect tests with *20* scenarios or spatial distributions of items in the layout generated randomly with a uniform probability distribution.

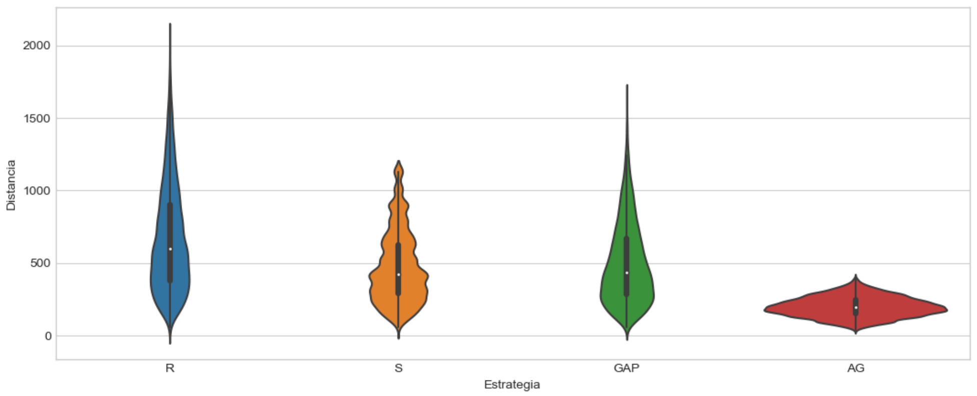

*Figure 10.- General results of the experiments carried out for all the different layouts and items. R Return strategy, S S-shape strategy, GAP Largest Gap strategy and AG genetic algorithm strategy*

In each of the generated scenarios, the performance of the Return, S-shape, Largest Gap algorithms, and the genetic algorithm was compared, obtaining the results shown in Figure 10, note the density distribution shows in the violin charts. On average, the genetic algorithm obtains better results than the other strategies and obtains less variability in the range of values for the calculated routes, which indicates that the genetic algorithm is more consistent in the length of the routes.

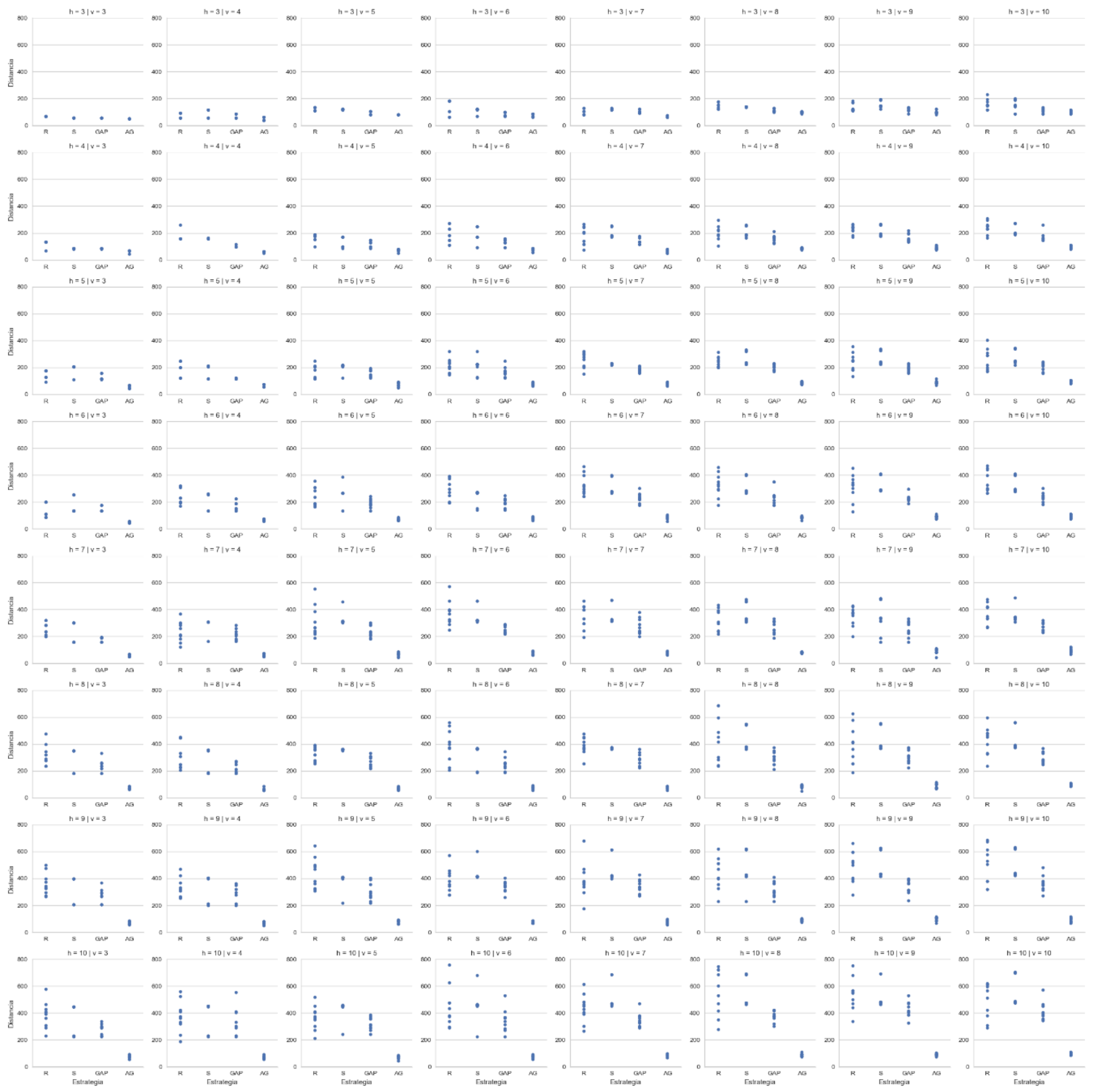

*Figure 11.- Experiment with 5 items in test layouts for strategies Return (R), S-Shape (S), Largest Gap (GAP) and Genetic Algorithm (AG)*

Figure 11 shows the set of tests applied for five items in the layouts considered for comparison. For the 3x3 layout, the algorithms have similar performance, increasing the vertical aisles begins to notice differences in the algorithms. Comparing the first and first columns, which correspond to warehouse layouts type 3 x V and H x 3, shows the behaviour is not symmetrical.

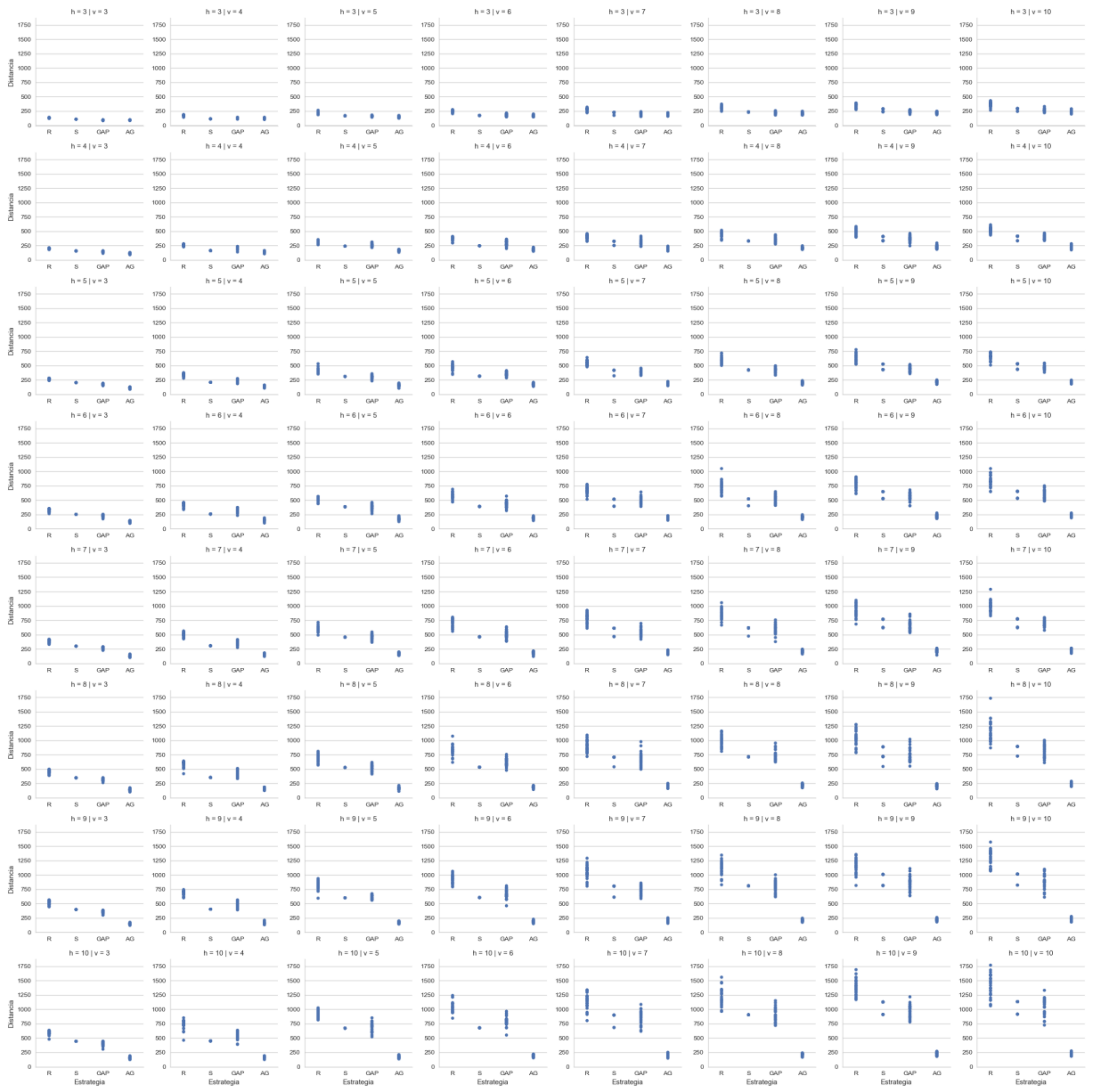

*Figure 12.- Experiment with 25 items in test layouts for Return (R), S-Shape (S), Largest Gap (GAP) and Genetic Algorithm (AG) strategies*

Figure 12 illustrates the distance dispersion diagram for 25 items and the different warehouse layouts considered in the experiment, showing similar behaviour in the first row versus the first column, although the range of values taken in the first column is slightly wider. In the case of 25 items per order, it is observed that square warehouse layout as growth in increasing route times while preserving the relative performance between the different algorithms.

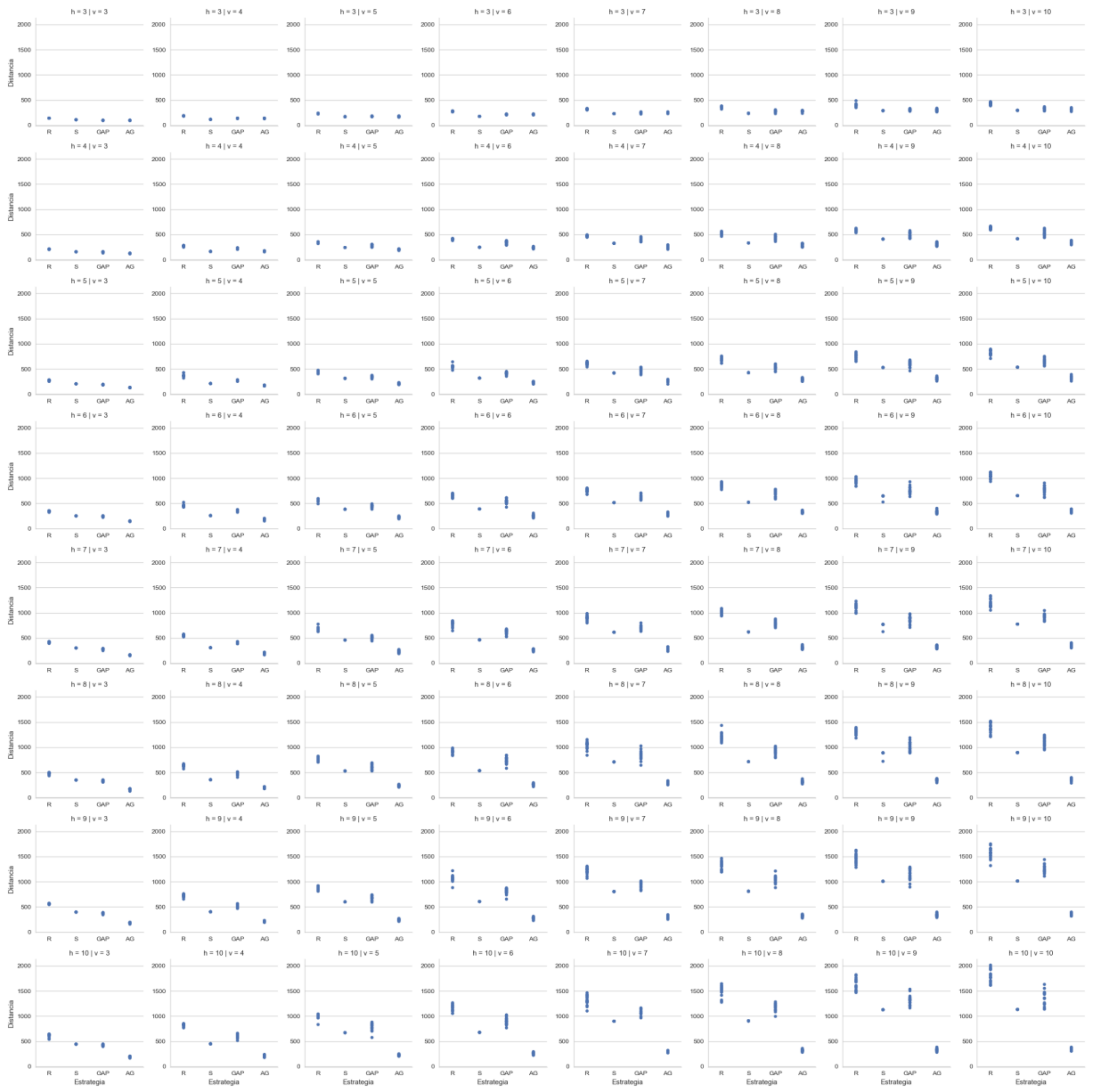

*Figure 13.- Experiment with 50 items in test layouts for Return (R), S-Shape (S), Largest Gap (GAP) and Genetic Algorithm (AG) strategies*

Figure 13 illustrates the dispersion diagram of distances travelled for the simulation performed with 50 items. Comparatively, the first row versus the first column in the results array is similar, the order in the performance of the evaluated algorithms is preserved; however, the first column has a slightly greater distance consistent with the distribution of the morphology of the warehouse layout, structures that are longer than they are wide.

## 6 Conclusions

One of the factors in the picking problem, collecting items for an order in a minimum of time, is the route taken. The choice of route depends on the strategy used, some basic but intuitive, and others efficient but impractical due to the computation time required that makes its use difficult; therefore, it is essential to propose methods that help to find better routes.

The use of a partition in warehouse layout reduces the complexity of the combinatorial problem; and transforms the number of items to collect for sites to visit; This transformation allows the reduction of the dimension of the search space and, therefore, the exploration time. The partitioning morphology is inspired by the largest gap policy determining when it is convenient to make a complete tour of the corridor or only a part and leave the rest for a second visit.

Representing the route using only the elements of the layout partition reduces the dimension of the workspace and simplifies the representation of the path, allowing the application of the Bellman optimality principle to search for the path of minimum length. To obtain the complete succession of sites to visit, an algorithm is presented that allows deciding the minimum path between elements of the partition.

The application of genetic algorithms using partitioning-induced algorithms as an individual model reduces the problem's combinatorial growth and makes it treatable. The defined cross and mutation operators allow the creation of new individuals who inherit characteristics from the parents, who are the solution to the problem and tend to improve their quality.

The experimental results show that, in large warehouses, the classic heuristics do not achieve a good performance compared to the method proposed using genetic algorithms and present greater dispersion in the range, on the other hand, for orders with few items in general, warehouses with more vertical corridors than horizontal is better. For orders in which there are many items, 50 in the experiment, the performance between long and wide warehouses is equivalent, where a long warehouse is one that has more horizontal aisles and few vertical aisles, in contrast, a wide warehouse is one that has more vertical than horizontal aisles.

## References

[1] Roodbergen, K. J., Iris, V., & Taylor, G. (2015). Simultaneous determination of warehouse layout and control policies. *International Journal of Production Research*, 3306-3326.
[2] Bozer, Y., & Kile, J. (2008). Order batching in walk-and-pick order picking systems. *International Journal of Production Research*, 1887-1909.
[3] Bartholdi, J., & Hackman, S. (2014). *WAREHOUSE & DISTRIBUTION SCIENCE Release 0.94.* Atlanta, GA USA: The Supply Chain and Logistics InstituteSchool of Industrial and Systems EngineeringGeorgia Institute of Technology.
[4] Henn, S., Koch, S., & Wäscher, G. (2012). Order Batching in Order Picking Warehouses: A Survey of Solution Approaches. *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems*.
[5] Valle, C., Beasley, J., & da Cunha, A. (2016). Modelling and Solving the Joint Order Batching and Picker Routing Problem in Inventories. *Combinatorial Optimization. ISCO 2016. Lecture Notes in Computer Science*, 81-97.
[6] Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York, NY United States: W. H. Freeman.
[7] Scholz, A., Schubert, D., & Wäsher, G. (2017). Order picking with multiple pickers and due dates - Simultaneous solution of Order Batching, Batch Assignment and Sequencing, and Picker Routing Problems. *European Journal of Operational Research*.
[8] Jinxiang, G., Goetschalckx, M., & McGinnis, L. (2009). Research on warehouse design and performance evaluation: Acomprehensive review. *European Journal of Operational Research*, 539-549.
[9] Holland, J. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* Cambridge, MA, United States: MIT Press.
[10] Aho, A., Ullman, J., & Hopcroft, J. (1983). *Data Structures and Algorithms.* Boston, MAUnited States: Addison-Wesley Longman Publishing Co., Inc.
[11] Korte, B., & Vygen, J. (2010). *Combinatorial Optimization: Theory and Algorithms.* Germany: Springer.
[12] Bellman, R. (1957). *Dynamic Programming.* Princeton, NJ, United States: Princeton University Press.
[13] Davis, L. (1985). Applying adaptive algorithms to epistatic domains. *Proceedings of the 9th international joint conference on Artificial intelligence, IJCAI'85* (págs. 162-164). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
[14] Wolfgang, B., Nordin, P., Keller, R., & Francone, F. (1997). *Genetic Programming - An Introduction.* San Francisco, CA: Morgan Kaufmann.
[15] Nojhan. (3 de April de 2007). *Evolutionary algorithm*. Recuperado el April de 2020, de Wikimedia Commons: https://commons.wikimedia.org/wiki/File:Evolutionary_algorithm.svg