An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses

An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses

Authors:

Roberta De Santis[*], Roberto Montanari[§], Giuseppe Vignali[§], Eleonora Bottani[§]

Affiliations:

[§]Department of Engineering and Architecture, University of Parma, viale G.P.Usberti 181/A – 43124 Parma (Italy)

[*]Logistics division, NUMBER 1 Logistics Group S.p.A., Via Paradigna 110/A, 43122 Parma

Corresponding author:

Eleonora Bottani – Associate professor of Industrial logistics

Department of Engineering and Architecture, University of Parma

viale G.P.Usberti 181/A – 43124 Parma (Italy)

phone: +39 0521 905872; fax +39 0521 905705; email: eleonora.bottani@unipr.it

An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses

Abstract

This paper proposes a new metaheuristic routing algorithm for the minimization of the travel distance of pickers in manual warehouses. The algorithm is based on the ant colony optimization (ACO) metaheuristic, which is combined and integrated with the Floyd-Warshall (FW) algorithm, and is therefore referred to as FW-ACO. To assess the performance of the FW-ACO algorithm, two sets of analyses are carried out. Firstly, the capability of the algorithm to provide effective solutions for the picking problem is analysed as a function of the settings of the main ACO parameters. Secondly, the performance of the FW-ACO algorithm is compared with that of six algorithms typically used to optimize the travel distance of pickers, including exact algorithms for the solution of the travelling salesman problem (where available), two heuristic routing strategies (i.e. S-shape and largest gap) and two metaheuristic algorithms (i.e. the MIN-MAX ant system and Combined+). The comparison is made considering different warehouse layouts and problem complexities. The outcomes obtained suggest that the FW-ACO is a promising algorithm generally able to provide better results than the heuristic and metaheuristic algorithms, and often able to find an exact solution. The FW-ACO algorithm also shows a very efficient computational time, which makes it suitable for defining the route of pickers in real time. The FW-ACO algorithm is finally implemented in a real case study, where constraints exist on the order in which items should be picked, to show its practical usefulness and quantify the resulting savings.

Keywords: logistics; ant-colony optimization (ACO); Floyd-Warshall (FW) algorithm; order picker routing; travel distance.

1 Introduction

Warehouse operation and management is an essential part of manufacturing and service operations (Zhang & Lai, 2006). The efficiency and effectiveness of logistics activities in general and of distribution networks in particular, is in fact largely determined by the way warehouses operate as the nodes of these networks. The logistics cost relating to warehouse processes, including receiving, storage, order picking and shipping, is often high (Rouwenhorst et al., 2000). Among the different processes, order picking is generally recognized as the most expensive activity, because it tends to be

either very labour intensive or capital intensive (Frazelle, 2002). Order picking is the process of selecting a set of items, retrieving them from their storage locations and transporting them to a sorting/consolidation process for order fulfilment and shipment, in response to a customer's request (Rouwenhorst et al., 2000).

The picking process can either be performed manually or (partly) automated. In the case of a manual process, it is estimated that picking operations account for more than 55% of the total cost of warehouse operations (Coyle et al., 1996; Tompkins et al., 1996; Bottani et al., 2015). For this reason, both researchers and logistics managers consider order picking as a promising area for productivity improvement (de Koster et al., 2007). The high cost of picking is mainly due to the fact that pickers spend approximately 50% of the total order picking time (unproductively) travelling. This part of the order picking time is commonly known as the "travel time" of pickers and affects the total order picking time to the largest extent (Tompkins et al., 1996). As the travel time is an increasing function of the travel distance, minimizing this distance has been suggested by many authors as potential leverage for optimising the total picking time of warehouses (Jarvis & McDowell, 1991; Hall, 1993; Petersen, 1999; Roodbergen & de Koster, 2001a; Petersen & Aase, 2004). Reducing the travel distance of pickers has a direct impact on warehouse performance in terms of cost and delivery lead-time, and consequently affects the performance of the whole supply chain. The faster items are picked from the warehouse, the shorter the time spent on order fulfilment; the lead-time required for delivering the product to the final customer therefore decreases correspondingly (de Koster, Le-Duc & Roodbergen, 2007; Bottani et al., 2012).

Researchers (e.g. Gu et al., 2007 or de Koster et al., 2007) agree that several factors affect travel distance in an order picking system, including:

1. the overall structure of the warehouse in terms of size and layout (Roodbergen & Vis, 2006; Parikh & Meller, 2010);
2. the operational strategy, e.g. order picking vs. batch picking (Van Nieuwenhuyse & de Koster, 2009; Le-Duc & de Koster, 2007; Gademann & Van de Velde, 2005; Henn, 2012; Hong et al., 2012);
3. the storage assignment policy (Petersen & Schmenner, 1999; Webster et al., 2012; Bottani et al. 2012; Jane & Laih, 2005);
4. the use of zone picking (de Koster et al. 2012; Petersen, 2002);
5. the picker routing (Petersen & Aase, 2004; Kulak et al., 2012);
6. the use of narrow aisles, which increase the likelihood of congestion whenever more pickers operate simultaneously in the warehouse (Pan & Wu, 2012; Chen et al., 2013, 2016; Mowrey & Parikh, 2014).

The proper design and management of all the factors listed above contribute to minimizing the travel distance of pickers, both *per se* and due to interdependencies between those factors (de Koster et al., 2007). However, these factors cannot all be dealt with and optimised at the same time; researchers typically focus on a specific topic to be analysed and optimised (de Koster et al., 2007; Bottani et al., 2012, 2015; Manzini et al., 2012). In this paper, we focus on the optimization of the routing policy of the picker; this is suggested to have greater potential for dynamic adjustment due to its flexibility (Chen et al., 2013). Moreover, significant savings in the travel time of pickers can be made using a dedicated routing heuristic, because in real cases most warehouses use very simple routing policies and the picking process is carried out manually (Petersen, 1999).

Sequencing and routing pickers in conventional multi-parallel-aisle warehouses starting from the set of items to be picked, is an NP-hard travelling salesman problem (TSP) (Theys et al., 2010; Lu et al., 2016; Scholz et al., 2016). Very few exact algorithms are able to solve this problem, and only apply under specific conditions. Nonetheless, Brezina & Čičková (2011) showed that the ant colony optimization (ACO) metaheuristic algorithm could be used to solve the TSP efficiently. ACO was also found to perform better than other metaheuristic algorithms (i.e. simulated annealing, genetic algorithms and evolutionary programming) when applied to different TSPs (Shtovba, 2005). On the basis of these findings, this paper proposes an adapted ACO metaheuristic algorithm to minimize the travel distance of pickers. The approach developed integrates the traditional ACO and Floyd-Warshall (FW) algorithms (Floyd, 1962; Warshall, 1962), and will be referred to as FW-ACO throughout the paper. The rationale behind their integration is to combine the ability of the FW algorithm to find the shortest path between any pair of nodes even in a complex graph (Cormen et al., 2009) and the effectiveness of the ACO algorithm in solving the TSP. This integration is also expected to be efficient from a computational point of view, as the computational complexity of the FW algorithm is $O(n^3)$, n being the number of vertices of the graph (Cormen et al., 2009).

The remainder of the paper is organized as follows. Section 2 reviews the literature relating to the optimization of the routing of pickers in manual warehouses, provides background on the ACO algorithm and its applications in the picking process and lastly highlights how this study goes beyond existing literature. Section 3 details the FW-ACO algorithm. In section 4, we evaluate the performance of the FW-ACO algorithm by means of two sets of analyses. The first is intended to evaluate the ability of the algorithm to provide effective solutions for the problem investigated, as a function of the settings of the main ACO algorithm parameters. In the second set of analyses, the performance of the FW-ACO algorithm is compared to those of six algorithms typically adopted to route order pickers in manual warehouses. In section 5, we apply the algorithm to a real case, involving a major Italian logistics company. Section 6 concludes by summarising the main findings of this study, discussing the main implications (both practical and theoretical) and outlining future research directions.

2 Literature review

2.1 Routing algorithms for manual warehouses

Routing policies for pickers in manual warehouses range from simple heuristics to optimal procedures. Optimal routing policies typically approach the routing problem as a special case of the TSP and try to find the exact solution to this problem. Such policies obviously result in shorter travel times, but can be negatively affected by some key issues. First, exact routing policies exist only for specific (or simple) warehouse configurations, while for more complex warehouse layouts, no exact procedures are available. To be more precise, an exact algorithm for solving the TSP exists for rectangular warehouses, which only have crossovers at the ends of the aisles (Ratliff & Rosenthal, 1983). The algorithm is fast enough to be applied to 1-block warehouses of any size: as an example, a 50-aisle problem requires about l minute to be solved. De Koster & Van der Poort (1998) also proposed a polynomial algorithm for a 1-block warehouse, extending that created by Ratliff & Rosenthal (1983) to solve the routing problem for a non-central location of the depot. This should reflect a more modern situation, where order picking trucks can pick up and deposit pallets at the head of every aisle without returning to the depot. Scholz et al. (2016) proposed an alternative mathematical formulation of the TSP, to capture some specific features relating to the single picker routing problem. Other authors (Roodbergen & de Koster, 2001b) have extended Ratliff & Rosenthal's algorithm to 2-block warehouses. The authors considered a parallel-aisle rectangular warehouse, where order pickers can change aisle either at the end of the aisle or at the cross-aisle halfway along the aisle. Another extension of the Ratliff & Rosenthal algorithm has been proposed for the dynamic order picking problem, in which the picker might receive instructions for a new picking mission where he/she is located in a random position in the warehouse (Lu et al., 2016). Some authors take a different approach, i.e. they minimize the travel distance of pickers by identifying the optimal layout of the warehouse (in terms of number of aisles and depot location), given the routing policy that will be adopted, both for 1-block (Roodbergen & Vis, 2006) and multiple-block warehouses (Roodbergen, Sharp & Vis, 2008). A similar approach has been adopted by Pan et al. (2014) for high-level picking. An exact algorithm for routing order pickers in a warehouse with fishbone layout has been developed by Çelk & Süral (2014).

Besides the fact that they can only be applied to a limited set of warehouse layouts, exact algorithms, as well as metaheuristic algorithms generating complex routes, involve a second issue in that their logic is not straightforward or easily understood by pickers (Henn et al., 2011). In practice, companies prefer to adopt very simple routing strategies (Petersen, 1999), which are more immediate for order pickers, although they usually do not result in an optimal travel time (Hall, 1993; de Koster et al., 2007). Examples of these simple strategies include S-shape, largest gap or midpoint (see e.g. Petersen,

1997, or Roodbergen & de Koster, 2001a, for a description of these policies). Order pickers become quickly familiar with these strategies, thus minimizing the risk of a missed pick (Petersen, 1999).

For more complex warehouse configurations, i.e. warehouses with more than two blocks, no exact algorithms are available. Moreover, it is no easy matter to extend existing ones to these configurations, as computational complexity increases rapidly with warehouse size and number of items in the picking list (Roodbergen & de Koster, 2001a; Theys et al., 2010; Scholz et al. 2016). Consequently, heuristic algorithms have been proposed for warehouse configurations with more than one cross-aisle. A heuristic algorithm for warehouses with two cross-aisles can be found in Hall (1993). Roodbergen & De Koster (1998) compared three heuristic routing algorithms for warehouses with more than two cross-aisles, covering several different situations, which include a narrow-aisle high-bay warehouse where order picking trucks are used. A heuristic routing algorithm, which makes use of dynamic programming, was also proposed for warehouses with more than two cross-aisles (Vaughan & Petersen, 1999). Roodbergen & de Koster (2001a) extended the heuristics available for warehouses with two cross-aisles to those with more than two cross-aisles. Their proposed algorithm, called Combined+, exploits dynamic programming to determine the order picker route. Theys et al. (2010) have reformulated the routing problem using the Lin–Kernighan–Helsgaun TSP heuristic and evaluated the improvement to performance compared to existing heuristics, including S-shape and largest gap. Heuristic algorithms have also been adopted for batch picking problems (Matusiak et al., 2014; Henn and Schmid,, 2013). Scientific literature typically compares the performance of heuristic routing algorithms to the optimal routing (where available), to assess whether the advantages in terms of simplicity or reduced computational time offset the increase in travel time generated. Largest gap and S-shape routing strategies, although non-optimal, are nonetheless often used as a benchmark (as also done, e.g., by Theys et al., 2010), as both policies are popular in the industry and are frequently adopted in practice because of their simplicity (Petersen, 1999).

Almost all of the abovementioned studies assume random storage of items in the warehouse. On the contrary, very few studies have examined the routing issue coupled with other storage assignment policies, to identify possible interdependencies among these factors. Among them, Caron et al. (1998) evaluated two simple routing heuristics in a manual, single cross-aisle warehouse, with a cube per order index (COI)-based allocation strategy. Petersen (1997) carried out a detailed analysis of variance to examine the interactions of routing policies, warehouse layout and depot location under different warehouse operating conditions. In a subsequent study, the same author addressed the issue of optimizing the routing strategy in a class-based storage system (Petersen, 1999). Dukic & Oluic (2007) evaluated the performances of routing, storage and order batching methods in combination, depending on the given situation (layout, order size and order-picker capacity). The correlation between number of aisles, picking list size and path length in a class-based storage environment has been examined by Rao & Adil (2013) for a 2-block warehouse.

2.2 ACO in the picking context

The ACO algorithm was originally introduced by Colorni et al. (1991) and Dorigo et al. (1996) as a probabilistic technique for solving computational problems. ACO is inspired by real ant colonies: when moving, ants leave pheromone on the ground, thus marking the path. An isolated ant is expected to move essentially at random; conversely, an ant encountering a previously laid trail can detect it and decide, with high probability, to follow it, thus reinforcing the trail with its own pheromone. Overall, the greater the number of ants following a trail, the more attractive that trail becomes for other ants (Dorigo et al., 1996).

ACO is commonly adopted to solve optimization problems where the goal is to identify the shortest path and has been applied to this end to solve the TSP, both in its traditional formulation (Brezina & Čičková, 2011; Colorni et al., 1991) and in the MAX-MIN ant system (MMAS) variant (Stützle & Hoos, 2000; Shtovba, 2005). Moreover, ACO has been adopted in many different engineering fields (see, e.g., Mariano & Morales, 1999; Eggers et al., 2003; De Jong & Wiering, 2001; Lucic, 2002), suggesting that the algorithm has a potential for numerous areas of application. Looking specifically at the order picking context, the application of ACO to this problem is primarily intended to minimize the travel distance by identifying the shortest path for pickers. However, we have only found a limited number of studies proposing the use of the ACO algorithm for the case of picking in manual warehouses. More precisely, Xing et al. (2010) applied the ACO algorithm to optimize the travel path of a storage and retrieval machine used for batch order picking. They found that ACO effectively minimizes the path of the machine considered. An adapted ACO algorithm, called G-A (Genetic-Ant colony) was developed by Fu et al. (2011) with the aim to increase the efficiency of high-level order pickers in a warehouse. The algorithm performed well in terms of search ability and was able to reduce the travel time of pickers significantly, compared to the traditional S-shape heuristic, which reflected the most frequently investigated heuristic in order picking literature. There are also examples of ACO-based routing algorithms for two (Chen et al., 2013) or multiple pickers (Chen et al., 2016), with congestion considerations. The performance of these algorithms was assessed and compared to that of the traditional S-shape routing strategy, finding that ACO-based algorithms perform better than that policy in most of the scenarios examined.

2.3 Gaps and contributions of this study

From the review of the studies proposed above, several considerations can be made. Firstly, exact routing algorithms are only available for a limited set of warehouse layouts. Extending the exact approaches to the case of a more complex warehouse is non-trivial, and in any case, the computation times increase rapidly when more than two blocks need to be evaluated (Roodbergen & de Koster, 2001a; Theys et al., 2010). The complexity of the TSP increases with the number of items in the picking list as well (Scholz et al., 2016). Consequently, authors have proposed (and are still proposing)

heuristic algorithms to route order pickers in complex scenarios. This suggests that there is room for new algorithms that are effective for warehouse configurations with more than two blocks, from the point of view of both solution quality and computational performance. Secondly, existing studies show that the ACO algorithm is effective in solving many engineering problems, including the TSP, of which order picking is a special case. Authors sometimes combine ACO with other algorithms (e.g. genetic algorithms) to enhance its solution potential. The algorithm we propose in this paper is also a combination, as it integrates ACO with an operational research algorithm (i.e. the Floyd Warshall algorithm); this is expected to enhance the solution potential of the algorithm and to make it effective from a computational perspective. To date, applications of ACO within the picking context are limited in number and often focus on very specific issues. Only two examples refer expressly to the case of order pickers in manual warehouses, despite the fact that this is the context where the incidence of travel time of pickers on the overall effectiveness of the picking process is higher (Tompkins et al., 1996).

On the basis of the abovementioned considerations, we have developed an adapted ACO algorithm, integrated with the FW algorithm, with the purpose of enhancing its capability to identify the shortest path of pickers. The FW-ACO algorithm is specifically designed to minimise the travel distance of pickers in a context where its incidence is particularly significant, i.e. the case of manual warehouses operating in accordance with a low-level, picker-to-parts strategy. The algorithm is also designed to be flexible enough to adapt easily to any warehouse layout, and, in particular, is expected to be useful for warehouse configurations that cannot be solved by means of exact approaches. To this end, it will be shown that the algorithm can be effectively adopted in complex warehouse configurations and with high number of items in the picking list, and that it is able to provide highly effective solutions with an efficient computational time.

3 The proposed approach

The overall scheme of the approach developed in this paper, including the FW-ACO algorithm and some preliminary steps, is shown in Figure 1. The nomenclature used during the description of the approach is illustrated in Table 1. In the subsections that follow, we will detail the steps of the approach.

Parameter	Description
Warehouse and picking parameters	
n	number of items in the picking list
A	set of arcs in a graph
V	set of nodes in a graph
N	number of nodes in the warehouse graph
N_p	number of picking nodes in the warehouse graph
N_s	number of service nodes in the warehouse graph
k_x	distance between two subsequent aisles (>0) [m]
k_y	distance between two adjacent storage locations (>0) [m]

Floyd-Warshall parameters	
L_{ij}	length of the arc (>0) connecting nodes i and j $(i, j = 1, \dots N)$ [m]
h	step of the algorithm $(h = 1, \dots N)$
D_{ij}^{h}	length of the shortest path connecting nodes i and j at step h [m]
$D^{(h)}$	distance matrix at step h
P_{ij}^{h}	generic entry of the predecessors' matrix at step h
$P^{(h)}$	predecessors' matrix at step h
FW-ACO parameters	
n'	number of nodes in the warehouse graph that should be visited by the picker
m	number of ants in the system $(k = 1, \dots m)$
$path_{i'j'}$	path connecting points i' and j' $(i', j' = 1, \dots n', i' \neq j')$
t	iteration step
ρ	evaporation rate of the pheromone $(0 < \rho < 1)$
$\tau_{i'j'}(t)$	intensity of the pheromone on $path_{i'j'}$ at time t
$\Delta\tau_{i'j'}^{k}(t)$	quantity of pheromone left by ant k on $path_{i'j'}$ at time t
$d_{i'j'}$	distance between points i' and j'
$\eta_{i'j'}$	visibility between points i' and j'
$d_{i'j'}^{k}(t)$	length of $path_{i'j'}$covered by ant k at iteration t [m]
$p_{i'j'}^{k}(t)$	probability that ant k moves from point i' to point j' at time t
α	relative importance of the pheromone trail
β	relative importance of the distance
Q	pheromone update constant
$J_{i'k}$	list of nodes yet to be visited by ant k, located in node i'
$Loops_{TOT}$	number of iterations allowed for the algorithm $(t = 1, \dots Loops_{TOT})$
n_{sim}	number of runs of the algorithm $(s = 1, \dots n_{sim})$
x_{min}	"best found" solution (shortest path) of the problem returned by the algorithm [m]
x_s	solution returned by the algorithm in run s [m]
G	percentage of runs where the algorithm converged to the "best found" solution [%]
E	average error rate of the algorithm [%]

Table 1: nomenclature.

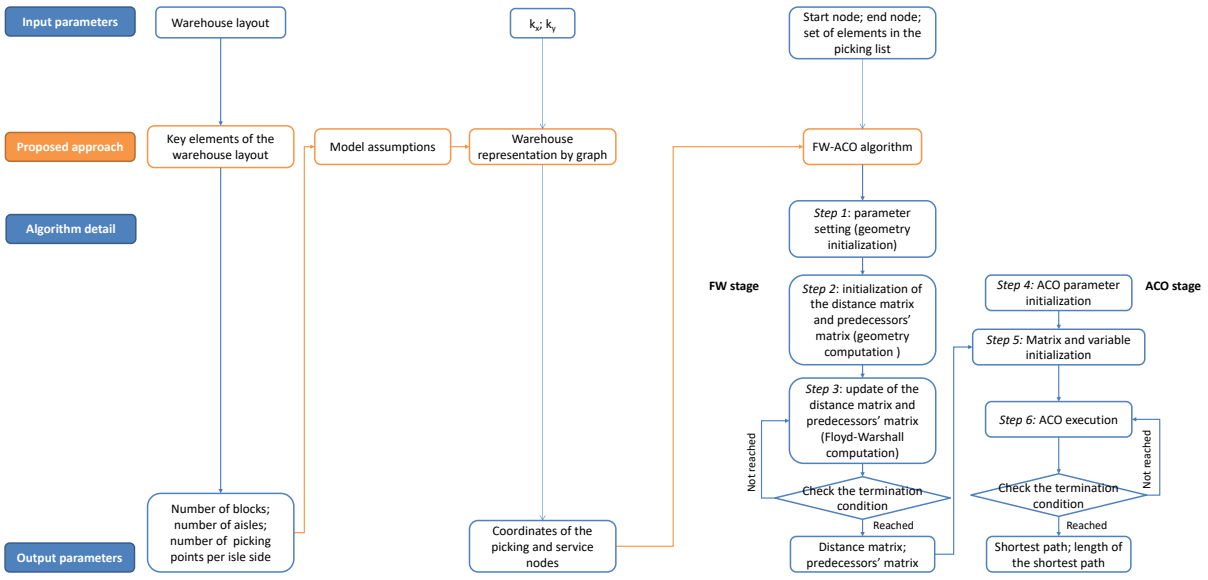

Figure 1: flowchart of the proposed approach.

3.1 Key elements of the warehouse layout

To make the FW-ACO algorithm suitable for adoption in any warehouse layout, it is first necessary to identify a limited number of essential pieces of information, which describe the warehouse layout univocally and can be processed by algorithm in a mathematical form. This is the rationale behind the first step of the approach.

A warehouse of regular shape consists of a number of parallel longitudinal aisles (simply called "aisles") of equal length, with the items being stored in storage locations on both sides of the aisles (De Koster & Van der Poort, 1998). A picking location (highlighted in blue in Figure 2) is a storage location where a picking item is located. Whenever cross-aisles are present, the warehouse is divided into a number of blocks equal to the number of cross-aisles plus one (Roodbergen & de Koster, 2001a). Hence, the essential data required to describe a warehouse layout consist of three elements, namely: (1) the number of blocks; (2) the number of aisles; (3) the number of storage locations per aisle side. Figure 2 shows a representation of the elements listed above in a warehouse layout with 2 blocks, 3 aisles and 12 storage locations per aisle side. Aisles provide access to the storage locations and therefore reflect the points where picking activities take place. Depending on the warehouse layout, starting from a given picking aisle, the picker will have access to one or two picking lanes. Cross-aisles do not provide direct access to the items to be picked; instead, the pickers can change picking lane using them.

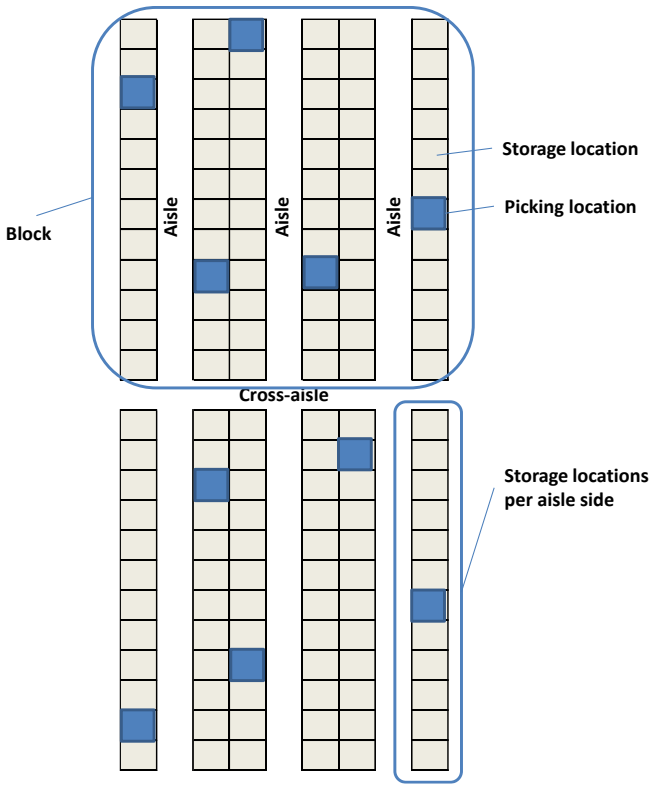

Figure 2: scheme of a generic warehouse indicating the key components.

3.2 Model assumptions

The approach is based on the assumptions described below.

1. Since low-level picking is one of the most frequent operating conditions in a warehouse (Caron et al., 2000), the FW-ACO algorithm will only work on the first level of storage;
2. The minimization of the travel distance of pickers can be approached as a TSP, where the picking locations listed in the picking list reflect the cities to be visited;
3. The longitudinal aisles are narrow enough to allow the operator to pick the item from both sides. We consider the presence of only one picker at a time and thus congestions among pickers (Chen et al., 2013; Mowrey & Parikh, 2014) are not evaluated;
4. The picker can change direction in the aisle;
5. The picker starts at the depot and returns to the depot once he/she has picked the items on one picking list;
6. The aisles can be travelled in both directions;
7. The warehouse has a regular (e.g. rectangular) shape with parallel longitudinal aisles.

3.3 Warehouse representation by graph

The next step is to represent the warehouse layout in the form of a graph able to provide a faithful reproduction of the geometry of the warehouse, taking into account the key elements of the warehouse layout. This step is quite common in literature (e.g. Ratliff & Rosenthal, 1983; Roodbergen & de Koster, 2001b) and, for the purposes of this study, it is required because the FW algorithm is designed to find the shortest path between pairs of nodes in a graph.

Any graph is a set of arcs A and nodes V. In our case, the nodes of the graph (circles) reflect the locations where the picker can move between, while the arcs (arrows) provide the connections between nodes. Figure 3 shows the representation of the warehouse in Figure 2 by graph. We categorise the N nodes of the warehouse graph into picking nodes N_p or service nodes N_s ($N = N_p + N_s$), depending on their function in the warehouse. The picking nodes (e.g. node 1 or 2 in Figure 3) are adjacent to the storage locations, while the service nodes (e.g. node 0 or 13) are used by the picker to reach a different aisle.

Figure 3 also shows that two additional parameters, denoted as k_x andk_y, should be introduced into the warehouse graph. To explain the rationale behind those parameters, we should recall that the structure of a regular warehouse can be represented on the Cartesian plane, as was also done in Figure 3. Because of the regular shape, it is reasonable to assume that nodes located on the x-axis are equidistant from each other and that the same assumption holds true for the nodes located on the y-axis. To describe their position, therefore, a constant value can be used, which is called k_x or k_y depending on the axis considered. More precisely, k_x reflects the distance between two subsequent

aisles, while k_y represents the distance between two adjacent picking positions. The graph nodes can be represented in Cartesian coordinates, exploiting k_x and k_y. For instance, the coordinates of node 0 in Figure 3 are (0;0), while the coordinates of node 55 in the same figure are ($2k_x$;$1k_y$).

It is evident that, once set, k_x and k_y should not be modified, since they reflect the geometrical structure of the warehouse analysed. Nonetheless, by varying k_x and k_y (as well as the number of blocks, the number of aisles and the number of storage locations per aisle side), the layout of any regular warehouse can be represented in mathematical form.

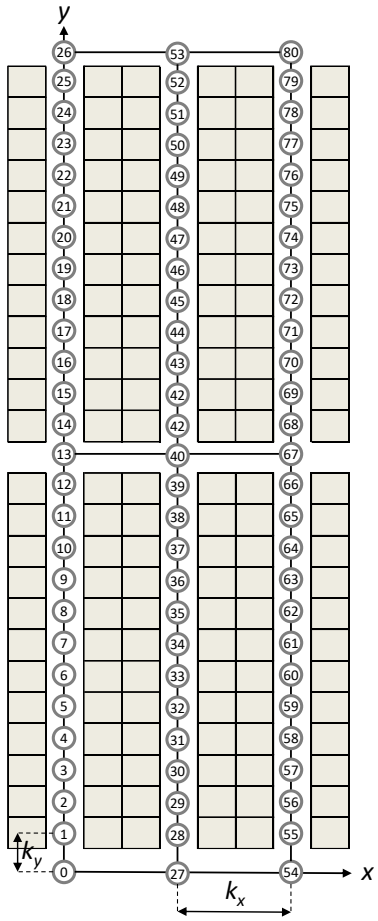

Figure 3: scheme of a warehouse and corresponding representation by graph.

3.4 The FW-ACO algorithm

The FW-ACO procedure consists of 6 steps grouped into 2 stages (see Figure 1). In the first one (*FW stage*), the warehouse layout is schematically represented as a graph and the FW algorithm is used to identify the shortest path connecting each pair of nodes in the graph. ACO is used in the second stage to identify the shortest picker route (*ACO stage*). The corresponding details are provided in the following subsections.

3.4.1 Step 1: geometry initialization

In step 1, the main parameters of the warehouse layout are initialized. According to the description in sections 3.1 and 3.3, these parameters include:

(1) the number of blocks;
(2) the number of aisles;
(3) the number of storage locations per aisle side;
(4) k_x;
(5) k_y.

3.4.2 Step 2: geometry computation

Step 2 involves translating the warehouse layout into a graph, with the corresponding coordinates according to the description in section 3.3.

3.4.3 Step 3: Floyd-Warshall computation

The FW algorithm is introduced in this step to identify the shortest path connecting each pair of nodes (i, j) in the warehouse. The genius of the FW algorithm is in finding a different formulation for the shortest path sub-problem than the path length formulation introduced earlier. This enables the algorithm to find the shortest path between any pair of nodes even in a complex graph (Cormen et al. 2009).

As already mentioned, a graph consists of a set of nodes connected by arcs. In general, for each arc connecting points i and j, a parameter $L_{ij} \in \mathbb{R}$ can be computed. For the purposes of our application, L_{ij} reflects the length of the arc connecting nodes i and j. Accordingly, if $path_{1j}$ describes the path from node 1 to node j, the corresponding length can be computed as $L(path_{1j}) = L_{12} + L_{23} + L_{34} + \cdots + L_{j-1,j}$.

To identify the shortest path, the FW algorithm makes use of two *ad hoc* matrices, known as the "distance matrix" and the "predecessors' matrix", which are generated and updated iteratively applying the following steps:

1. *initialization*. At the beginning of the process ($h = 0$), the structure of the distance matrix is initialized as follows:

$$D^{(0)} = \left(D_{ij}^0\right) \text{ where } D_{ij}^0 = \begin{cases} L_{ij}, if\ (i,j) \in A \\ 0,\ if\ \ i = j \\ \infty, if\ (i,j) \notin \text{A} \end{cases} \quad (1)$$

while the generic entry of the predecessors' matrix P_{ij}^0 is initialized as follows:

$$P^{(0)} = \left(P_{ij}^0\right) \text{ where } p_{ij}^0 = \begin{cases} i,\ if\ i \neq j \\ -,\ if\ i = j \end{cases} \quad (2)$$

2. *matrix update*. A new node is added for the computation of the shortest path between nodes i and j. Therefore, the distance matrix is updated to D_{ij}^{h} applying the following formula:

$$D_{ij}^{h} = min\{D_{ij}^{h-1}, D_{i,h}^{h-1} + D_{h,j}^{h-1}\} \text{ if } i \neq j \quad (3)$$

D_{ij}^{h} describes the updated distance between nodes i and j, computed exploiting h intermediate nodes $\{1, \dots, h\}$. The update of the distance matrix is based on Bellman's (1958) theory of optimality, according to which, if $path_{1j}$ is the shortest path from node 1 to node j and h is an intermediate node between 1 and j along $path_{1j}$, then path $path_{1h}$ connecting node 1 and node h is the shortest path from 1 to h. Overall, at the h-th step of the process, each entry of the distance matrix D_{ij}^{h} indicates the length of the shortest path which connects nodes i and j exploiting h intermediate nodes $\{1, \dots, h\}$. Therefore, the whole set of nodes used for the computation of D_{ij}^{h} is $\{1, \dots, h\} \cup \{i, j\}$.

The predecessor's matrix $P^{(h)} = \left(P_{ij}^{h}\right)$ is updated according to the following formula:

$$P_{ij}^{h} = \begin{cases} P_{h-1j}^{h-1}, \text{ if } D_{ij}^{h} \neq D_{ij}^{h-1} \\ P_{ij}^{h-1}, \quad \text{otherwise} \end{cases} \quad (4)$$

3. *check the termination condition*. If $h = N$, i.e. all the nodes have been added, the algorithm stops. Under this condition, the length of the shortest path connecting nodes i and j can be read in the D_{ij}^{N} element of the distance matrix $D^{(N)}$. The $P^{(N)}$ matrix can instead be used to track back how the shortest path was obtained. Otherwise, if $h < N$, h is updated (i.e. $h = h + 1$) and steps 1-3 are repeated until the termination condition is reached.

In our approach, the input of the FW algorithm is the warehouse graph (N nodes). With this graph, the algorithm generates the (NxN) distance matrix and the (NxN) predecessors' matrix as outputs. The first indicates the shortest distance between any pair of picking positions, on the basis of the physical distance between two adjacent nodes. The predecessors' matrix indicates which nodes are to be visited when moving from node i to node j, with the aim of covering the shortest path between these nodes.

3.4.4 Step 4: ACO parameter initialization

Step 4 consists of the initialization of the main parameters of the ACO algorithm. Specifically, these parameters are α, β, ρ, Q and τ (Colorni et al., 1991). The maximum number of iterations $Loops_{TOT}$ allowed for the algorithm to reach convergence should also be defined. As further input parameters, the ACO algorithm requires the following data:

(1) the starting node;
(2) the set of elements of the picking list (n).

The starting node is typically the warehouse input/output gate: indeed, the picker usually starts from the depot, picks up the items included in the picking list and returns to the depot. Nonetheless, other nodes could be set as starting nodes: for instance, a picker who is already carrying out picking

activities in the warehouse might receive instructions (e.g. *via* a mobile terminal) about an additional picking mission. In this latter case, the starting node is the node where the operator is located at the moment of receiving the instructions and from which he/she starts the new mission. Taking into account these possible scenarios, the starting point was not considered as a fixed element of the warehouse layout; it was instead treated as an ACO input during the initialization phase, to make the algorithm flexible enough to set the start point on the specific problem investigated.

3.4.5 Step 5: matrix and variable initialization

Starting from the distance matrix of the FW algorithm, as well as from the set of n elements of the picking list, a reduced (n'xn') distance matrix is elaborated in this step. This sub-matrix is limited to the $n' \leq n$ nodes of the warehouse that should be visited by the picker, which, in turn, reflect the nodes where the ACO algorithm should work[1]. For the same nodes, the pheromone matrix is initialized at a fixed quantity $\tau_{i',j'} = \tau, (\forall i' \neq j',\ i', j' = 1, \dots n')$. An arbitrary (random) path is set as the "best path" and the corresponding length as the "best length". The iteration counter t is initialized at 0.

3.4.6 Step 6: ACO execution

ACO execution consists of $Loops_{TOT}$ iterations. At each iteration t, the algorithm takes into account the set of n' points $(j' = 1, \dots n')$ to be visited by the m ants $(k = 1, \dots m)$ in the system. The path connecting two points i' and j' $(i', j' = 1, \dots n', i' \neq j')$ is denoted as $path_{i',j'}$ and its length is $d_{i',j'}$, resulting from the distance matrix. The visibility $\eta_{i',j'}$ between two points i' and j' is computed as $\eta_{i',j'} = 1/d_{i',j'}$. The new path connecting the n' nodes is generated by applying the following steps:

1. *computation of the probability to reach each node*. The probability $p^k_{i',j'}(t)$ that ant k moves from point i' to point j' is computed according to the equation below (Shtovba, 2005):

$$p^k_{i',j'}(t) = \begin{cases} \frac{[\tau_{i',j'}(t)]^{\alpha}(\eta_{i',j'})^{\beta}}{\sum_{l \in J_{ik}}[\tau_{i',l}(t)]^{\alpha}(\eta_{i',l})^{\beta}} & if\ j' \in J_{i',k} \\ 0 & otherwise \end{cases} \quad (5)$$

2. *choice of the next node*. Whenever an ant can reach more than one node, the choice of the specific node will be made according to $p^k_{i',j'}(t)$ (the higher the better), reflecting a sort of roulette wheel selection method (Bäck, 1996);
3. *transfer to the chosen node and modification of the path*. The chosen node will be added to the new path; at the same time, the list of nodes $J_{i',k}$ will be updated by removing the node that has been just added to the path.

[1] The number of nodes where ACO should work (n') does not necessarily correspond to the number of elements on the picking list (n). Indeed, the same picking node can give access to two picking locations, i.e. up to two elements on the picking list, depending on the specific case (see e.g. Ratliff & Rosenthal, 1983).

Steps 1-3 are repeated until all the n' nodes have been visited. The total length of the resulting path is then calculated. Whenever this length is lower than the "best length" found so far by the algorithm, the resulting path will become the new "best path" and will represent the benchmark for the evaluation of the paths identified in subsequent iterations. In addition, while the visibility matrix does not vary when running the algorithm, the pheromone matrix will be updated when a new "best path" is found. A hybrid approach between the ant-quantity and the ant-cycle methods (Colorni et al., 1991) has been adopted to update the pheromone trail. The ant-cycle approach is actually an adaptation of the ant-quantity approach, where the pheromone update is made at the end of the tour of an ant, instead of at each movement. Such an approach is effective for the problem in question, as it will prioritise the tours that overall generate a shorter path. Accordingly, if a new "best path" is found at iteration t, the pheromone matrix will be updated as follows:

$$\tau_{i'j'}(t) = \rho * \tau_{i'j'}(t-1) + \sum_{k=1}^{m} \Delta\tau_{i'j'}^{k}(t-1) \quad (6)$$

$\sum_{k=1}^{m} \Delta\tau_{i'j'}^{k}(t-1)$ reflects the overall quantity of pheromone per unit of length left on $path_{i'j'}$ at iteration $t-1$ and is computed as the sum of the quantities of pheromone left on the path by each ant. $\Delta\tau_{i'j'}^{k}(t)$ is computed as follows:

$$\Delta\tau_{i'j'}^{k}(t-1) = \begin{cases} \frac{Q}{d_{i'j'}^{k}(t-1)} & if\ ant\ k\ selected\ path_{i'j'}\ at\ time\ t-1 \\ 0 & otherwise \end{cases} \quad (7)$$

where $d_{i'j'}^{k}(t-1)$ is the distance covered by ant k to move from i to j. With this approach, the increase in the pheromone intensity on $path_{i'j'}$ is inversely proportional to the distance between i' and j', so that shorter arcs are preferred.

Once the algorithm has completed $Loops_{TOT}$ iterations, it provides the shortest path of the picker and the corresponding length as outputs. As a final step, the algorithm makes use of the FW predecessors' matrix to translate the final "best path" into the corresponding tour in the warehouse graph.

The full pseudo-code of the FW-ACO algorithm is proposed in the Appendix.

4 Performance evaluation

The FW-ACO algorithm described above was coded exploiting Python, a widely used general-purpose, high-level programming language (Kuhlman, 2013) and run on a 16 GB RAM, Intel Xeon CPU desktop computer, equipped with the Microsoft Windows 7 operating system.

To evaluate the performance of the FW-ACO algorithm, we carried out two set of analyses. The first one aims to assess the performance of the algorithm as a function of the typical parameters of the ACO metaheuristic. To this end, we consider a fixed warehouse layout and vary the settings of the FW-ACO algorithm. The analysis is expected to help identify the most effective setting for the proposed algorithm, i.e. the setting that allows the best performance to be achieved in terms of the shortest path

for pickers. A formal design of experiment (DOE) analysis is carried out to substantiate the choice of the most effective setting. The second set of analyses aims to compare the performance of the proposed algorithm with other routing policies, including exact and heuristic algorithms, to assess whether, and to what extent, the FW-ACO algorithm outperforms them.

4.1 First set of analyses: identification of the shortest path

4.1.1 Parameter setting

The warehouse layout considered in this set of analyses is fixed and consists of 2 blocks, with 4 aisles per block and 20 storage locations per aisle side; k_x=4 [m] and k_y=2 [m] are set for this warehouse.

The performance of the FW-ACO algorithm is evaluated as a function of:

(1) the numerical values assigned to the main parameters of the ACO metaheuristic, i.e. $\rho, \tau, \alpha, \beta$;
(2) the number of iterations $Loops_{TOT}$ allowed for the algorithm to reach convergence;
(3) the number of items n in the picking list.

With respect to the ACO parameters, the values examined were adapted from Colorni et al. (1991), with the aim of representing sufficiently different operating conditions of the algorithm. More specifically:

- ρ was set at its central value (0.5) and at a value very close to the upper limit (0.9), reflecting two situations where the evaporation of the pheromone trail is moderately slow or extremely slow, respectively;
- α was set at 1 or 2, while β was set at 2 and 5. The combination of those settings allows the performance of the FW-ACO algorithm to be investigated when working with $\alpha = \beta$ or $\alpha < \beta$;
- τ was set at 0.1 or 0.5, reflecting the case of low or high initial quantity of pheromone on the trail.

With respect to the remaining parameters, these were set as follows:

- $Loops_{TOT}$ was set at 1,000, 10,000 and 20,000. Varying the number of iterations is useful to assess whether the performance of the FW-ACO algorithm is affected by the number of iterations allowed to reach convergence;
- n was set at 10, 20 and 50, corresponding to as many picking locations to be visited in the warehouse. The picking locations depend only on the length of the picking list, while they are kept unchanged for a given n. Keeping the picking locations unchanged allows a comparison of the solutions provided by the algorithm to be made with the same predecessor and distance matrices resulting from the FW algorithm, which means that the ACO algorithm operates starting from the same set of distances.

As far as the remaining ACO parameters (i.e. Q and m) are concerned, they were set as follows. After some preliminary tests, Q was set at 2 and was kept unchanged during the analysis, which is in line with several previous studies about the ACO algorithm (e.g. Shtovba, 2005; Zhu & Curry, 2009). With respect to m, a relationship exists between this parameter and $Loops_{TOT}$ (Fidanova & Marinov, 2013). If we fix the number of iterations and double the number of ants in an ACO problem, the execution time is doubled, and, if we fix the number of ants and double the number of iterations, the execution time is also doubled. Therefore, it is advisable to fix one of the parameters; we chose to fix m and vary $Loops_{TOT}$. Finding the exact number of ants required to solve a problem in evolutionary algorithms is typically approached as an empirical problem, as the solution depends on the specific situation (Fidanova et al., 2014). We tried to identify the best number of ants using the following procedure. We set m=1, 2, 5, 7 and 10 ants and solved the smallest instance of the problem (i.e. n=10 items in the picking list) and the largest one (i.e. n=50 items in the picking list), with $Loops_{TOT}$=20,000. We recorded the time required to run the algorithm and the "best found" solution as a function of m. We found that the computational time decreased significantly when increasing m up to 5, while with a higher m the decrease, although present, was less appreciable. No differences were found in the solution returned with m=5, 7 or 10. In line with the fact that limiting the number of ants is useful when solving large problems, as the algorithm will require less memory to be run (Fidanova & Marinov, 2013), m=5 was selected as the most suitable number of ants.

By combining the settings described above, we obtained $2x2x2x2x3x3 = 144$ scenarios where the performance of the FW-ACO algorithm was evaluated. For each scenario, n_{sim}=100 runs of the algorithm were carried out.

4.1.2 Results

Two performance parameters were used to evaluate the results provided by the FW-ACO algorithm in the various scenarios, i.e.:

(1) The percentage of simulation runs G where the algorithm converged to the "best found" solution. The "best found" solution does not necessarily reflect the "global optimal" solution of the problem analysed (i.e. the global shortest path of the picker for a given n). In fact the number of solutions for the problem in question increases very rapidly with n. Verifying all the possible solutions of the TSP exhaustively to identify the global optimal one is not feasible from a practical perspective. Therefore, by "best found" solution we mean the shortest path identified in the resolution of the same problem by using different settings of the ACO parameters. As an example, Table 2 reports G=40% for n=10, obtained setting ρ=0.5, α=1, β=2 and $Loops_{TOT}$=1,000. This result indicates that with this setting, in 40% of the attempts, the FW-ACO algorithm converges to the best solution obtained for the picking problem with n=10, by varying the remaining ACO parameters (i.e. ρ, α, β, τ and $Loops_{TOT}$);

(2) the average error rate E of the algorithm, i.e. the percentage difference between the "best found" solution and the solution returned by the FW-ACO algorithm, in the event it did not converge to the "best found" solution. E is a dimensionless performance parameter computed according to eq.8:

$$E = \frac{\sum_{s=1}^{n_{sim}} \frac{|x_s - x_{min}|}{x_{min}}}{n_{sim}} \quad (8)$$

If a set of runs generates a small E, the results returned by that FW-ACO setting are, on average, very close to "best found" solution of the problem and could be considered as acceptable. Accordingly, the FW-ACO settings that achieved these results can also be considered acceptable.

The detailed outcomes of the performance assessment for the FW-ACO algorithm, in terms of G and E, are shown in Table 2. The key considerations from these outcomes are summarised below.

1. *Ability to generate the "best found" solution*. Overall, the FW-ACO algorithm is able to reach the "best found" solution across the 100 simulation runs (G=100%) in 35 out of the 144 scenarios examined (24.31%). Most of the time, G=100% is obtained with small-scale problems, i.e. n=10 (25 scenarios out of the 48 simulated, 52.08%) and sometimes n=20 (10 scenarios out of the 48 simulated, 20.83%); conversely, G=100% is never achieved with n=50. This is an obvious consequence of the increase in problem complexity as a function of n. Nonetheless, a specific setting of the ACO parameters seems to be particularly effective in identifying satisfactory solutions, regardless of problem complexity. To be more precise, from Table 2 it can be appreciated that when setting $\rho = 0.9$, $\alpha = 1$ and $\beta = 5$ ($\forall\tau$), the solutions identified by the FW-ACO algorithm correspond to the "best found" one (G=100%), with both n=10 and n=20. With respect to the scenarios with n=50, the solutions obtained with the setting detailed above, although not always corresponding to the "best found" solution, exhibit a very low E (from 0.66% to 1.00%), meaning that these solutions are very close to the shortest path ever identified. This suggests that the algorithm is effective in identifying acceptable solutions for the problem investigated, even with a high n.
2. *Effect of the ACO parameters on the algorithm performance*. From Table 2 it is easy to see that the performance of the FW-ACO algorithm tends to get worse when setting $\alpha = \beta = 2$. With this setting, the algorithm converges to the "best found" solution (G=100%) only in one scenario ($\tau = 0.1$, with $Loops_{TOT}$=20,000 and n=10), while it rarely converges to the "best found" solution with n=50, showing, at the same time, a fairly high E. Increasing $Loops_{TOT}$ also tends to improve the results provided by the FW-ACO algorithm: in general, the outcomes obtained are more satisfactory with $Loops_{TOT}$=20,000. This is specifically the case

for scenarios with n=50. This outcome was expected: indeed, by allowing more iterations, the algorithm benefits from more attempts to explore the solution space and identify better solutions.

					n=10		n=20		n=50	
ρ	α	β	τ	$Loops_{TOT}$	G	E	G	E	G	E
0.5	1	2	0.1	1000	60%	1.80%	0%	9.82%	0%	28.59%
				10000	80%	0.47%	0%	9.35%	0%	22.37%
				20000	60%	0.63%	10%	6.59%	0%	25.15%
			0.5	1000	40%	1.72%	10%	5.53%	0%	23.11%
				10000	100%	0.00%	0%	7.47%	0%	20.08%
				20000	90%	0.31%	40%	2.35%	0%	15.39%
		5	0.1	1000	80%	0.16%	70%	0.76%	0%	1.45%
				10000	100%	0.00%	90%	0.35%	0%	1.54%
				20000	100%	0.00%	90%	0.12%	30%	1.49%
			0.5	1000	80%	0.16%	100%	0.00%	0%	1.16%
				10000	100%	0.00%	100%	0.00%	10%	1.04%
				20000	100%	0.00%	80%	0.41%	20%	0.66%
	2	2	0.1	1000	20%	5.70%	0%	19.76%	0%	56.02%
				10000	50%	2.97%	0%	16.29%	0%	57.39%
				20000	100%	3.83%	80%	19.94%	20%	52.49%
			0.5	1000	40%	2.81%	0%	13.24%	0%	38.80%
				10000	70%	1.25%	0%	14.12%	0%	35.77%
				20000	70%	1.09%	0%	11.47%	0%	34.13%
		5	0.1	1000	10%	3.28%	0%	7.71%	0%	6.47%
				10000	70%	0.86%	50%	3.88%	0%	4.22%
				20000	90%	0.08%	10%	4.65%	0%	4.05%
			0.5	1000	20%	1.17%	60%	2.12%	0%	3.15%
				10000	90%	0.08%	50%	3.06%	0%	2.28%
				20000	100%	0.00%	60%	1.24%	10%	1.83%
0.9	1	2	0.1	1000	85%	0.13%	10%	5.64%	0%	8.96%
				10000	100%	0.00%	41%	1.18%	0%	2.73%
				20000	100%	0.00%	41%	0.78%	0%	2.01%
			0.5	1000	50%	0.65%	0%	14.31%	0%	36.10%
				10000	100%	0.00%	35%	4.61%	0%	19.12%
				20000	100%	0.00%	41%	3.09%	10%	9.06%
		5	0.1	1000	100%	0.00%	100%	0.00%	16%	1.00%
				10000	100%	0.00%	100%	0.00%	16%	0.66%
				20000	100%	0.00%	100%	0.00%	17%	0.69%
			0.5	1000	100%	0.00%	100%	0.00%	8%	0.97%
				10000	100%	0.00%	100%	0.00%	8%	0.76%
				20000	100%	0.00%	100%	0.00%	8%	0.69%
	2	2	0.1	1000	60%	1.04%	8%	8.73%	0%	17.25%
				10000	91%	0.07%	25%	3.92%	0%	10.03%
				20000	100%	0.00%	10%	3.19%	0%	11.89%
			0.5	1000	100%	0.00%	25%	6.08%	0%	9.20%
				10000	100%	0.00%	50%	1.96%	0%	2.77%
				20000	100%	0.00%	60%	0.49%	16%	1.56%
		5	0.1	1000	50%	0.39%	60%	0.98%	0%	2.01%
				10000	100%	0.00%	85%	0.34%	10%	1.31%
				20000	100%	0.00%	85%	0.49%	16%	1.00%
			0.5	1000	91%	0.07%	85%	0.59%	0%	1.04%
				10000	100%	0.00%	100%	0.00%	0%	0.83%
				20000	100%	0.00%	100%	0.00%	41%	0.48%

Table 2: experimental results for the first set of analyses - identification of the shortest path.

4.1.3 DOE analysis

A formal DOE analysis (Montgomery & Runger, 2003) was carried out on the outcomes in Table 2 to highlight whether the performance of the FW-ACO algorithm is significantly affected by the ACO settings, as well as to substantiate the choice of the setting to be used in the second set of analyses. Six factors, i.e. $\rho, \alpha, \beta, \tau, n, Loops_{TOT}$, were considered in a 2^6 full factorial design[2]. To this end, we limited the analysis to two values of $Loops_{TOT}$ (i.e. 1,000 and 20,000) and of n (i.e. 10 and 50). It is worth mentioning that n was included in the analysis for completeness, although it is not actually a parameter of the ACO algorithm but rather of the problem examined. Three outcomes were considered, namely: the distance returned by the algorithm; G; and E. Table 3 shows the results of the DOE analysis on these outcomes. Because of the high number of factors, the sparsity of effects principle applies, meaning that main effects and low-order interactions are expected to have the highest effect on the results observed (Box, Hunter, & Hunter, 2005). Accordingly, we limit the presentation of the outcomes (in terms of effect, mean square of the treatment and corresponding significance value) to the single-factor effects. Significant (at p<0.05) values are highlighted in italics in Table 3.

Outcome #1: distance	**factor A: ρ**	**factor B: α**	**factor C: β**	**factor D: τ**	**factor E: $Loops_{TOT}$**	**factor F: n**
effect	-15.64	6.83	-25.55	3.83	-5.79	140.65
mean square of the treatment	1.57E+05	2.98E+04	4.18E+05	9.39E+03	2.15E+04	1.27E+07
sig.	*0.000*	*0.000*	*0.000*	*0.000*	*0.000*	*0.000*
Outcome #2: G	**factor A: ρ**	**factor B: α**	**factor C: β**	**factor D: τ**	**factor E: $Loops_{TOT}$**	**factor F: n**
effect	5.00	-4.00	4.60	0.40	3.60	-17.20
mean square of the treatment	0.39	0.25	0.33	0.00	0.20	4.62
sig.	0.090	0.173	0.118	0.890	0.219	*0.000*
Outcome #3: E	**factor A: ρ**	**factor B: α**	**factor C: β**	**factor D: τ**	**factor E: $Loops_{TOT}$**	**factor F: n**
effect	-0.39	0.03	-1.18	0.01	-0.50	1.13
mean square of the treatment	0.00	0.00	0.02	0.00	0.00	0.02
sig.	0.095	0.903	*0.000*	0.962	*0.031*	*0.000*

Table 3: DOE results for the first set of analyses.

Table 3 shows that all the factors investigated exhibit a significant effect on the total distance returned by the algorithm. More precisely, α and $Loops_{TOT}$ show a positive effect on that outcome, while ρ, β, τ and n have a negative effect. Hence, the most effective setting of those parameters, taking into account the fact that the total distance should be minimized, is as follows: ρ=0.9; α=1; β=5; τ=0.1; $Loops_{TOT}$=20,000. For instance, ρ has a negative effect on the distance, meaning that increasing ρ

[2] To be more precise, we also carried out a 5-factor DOE, excluding $Loops_{TOT}$, as its impact on the algorithm performance would be obvious (the higher the number of iterations allowed, the better the ability of the algorithm to identify the "best found" solution). It is worth mentioning that the outcomes obtained with the 5-factor DOE correspond perfectly with those of the 6-factor DOE.

would reduce the resulting distance returned by the algorithm. Note that two values of this parameter were tested, i.e. $\rho = 0.5$ and $\rho = 0.9$. As the distance should be minimized, the highest value of ρ is to be preferred, i.e. $\rho = 0.9$. Similar considerations can easily be extended to the remaining parameters. It is easy to note that the most effective setting is very similar to that identified as returning the best performance of the FW-ACO algorithm in the previous subsection.

In addition, the results in Table 3 indicate that none of the ACO parameters has a significant effect on G and thus on the ability of the FW-ACO algorithm to converge to the "best found" solution. This is an interesting outcome, as it suggests that the FW-ACO algorithm is robust in this respect. The only significant effect is given by n, which, as already mentioned, is a problem constraint rather than a parameter of the algorithm. It should also be noted that G should be maximized, which justifies the fact that the effects observed against this outcome are opposite to those observed against outcomes #1 and #3. Finally, only two ACO parameters (i.e. β and $Loops_{TOT}$) affect the average error rate E of the algorithm to a significant extent. Both parameters have a positive effect on E, which is to be minimized, thus confirming the choice of the highest values of those parameters.

4.2 Second set of analyses: comparison with other routing strategies

4.2.1 Routing algorithms

In the second set of analyses, we compare the FW-ACO algorithm with six algorithms used to route pickers in manual warehouses. A description of these algorithms is proposed below.

(1) The S-shape routing policy is a simple heuristic strategy, which is based on the assumption that any aisle containing at least one picking location to be visited should be traversed along its entire length, unless it is the last aisle visited. Aisles where nothing has to be picked are skipped (Roodbergen & de Koster, 2001a). As we have already discussed, the S-shape routing strategy is not optimal, meaning that it does not return the shortest path; this is also the case for the largest gap strategy described below. Nonetheless, these algorithms are used as a benchmark because they are both popular in the industry and easy to understood and use by order pickers (Petersen, 1999);

(2) In the largest gap policy, the "gap" represents the distance between any two adjacent picking locations in an aisle, between the first pick and the front aisle, or between the last pick and the back aisle. The rationale of this policy is to avoid covering the "largest gap". Accordingly, if the largest gap is between two adjacent picking locations, the picker performs a return route from both ends of the aisle. Otherwise, a return route from either the front or back aisle is used (Roodbergen & de Koster, 2001a);

(3) The exact algorithm by Ratliff & Rosenthal (1983) is used as a benchmark for the results of the proposed algorithm in 1-block warehouse configurations, for which it is able to provide the exact solution;
(4) The routing algorithm by Roodbergen & de Koster (2001b), which returns the shortest path in warehouses with a middle aisle, is used as a benchmark for 2-block warehouse configurations;
(5) The "Combined+" algorithm (Roodbergen & de Koster, 2001a) is a metaheuristic algorithm that has proven effective in finding good solutions for routing order pickers in warehouses with multiple cross-aisles. It is used as a benchmark for all warehouse configurations examined;
(6) The MMAS (Stützle & Hoos, 2000) is an adaptation of the ACO algorithm, which differs from traditional ACO in three main points, namely the pheromone update rule, the range of values of the pheromone and the initialization of the pheromone on the graph. It is used as a benchmark for all warehouse configurations analysed. The MMAS was chosen because it is derived from the ACO algorithm, which is also the case for the FW-ACO algorithm proposed in this paper. Hence, comparing the related outcomes is useful to judge whether the adaptation introduced by the FW-ACO algorithm is more effective than that of the MMAS. Moreover, the MMAS has been compared with other versions of the ant algorithms and demonstrated to outperform them (Shtovba, 2005).

Algorithms 3-6 were all coded using Python, exploiting the warehouse representation by graph described in section 3.3, while algorithms 1-2 were solved manually for each scenario examined.

4.2.2 Design factors

In this set of analyses, we consider the following design factors:

(1) *Warehouse layout*: three different layouts, denoted as A, B and C, were considered for a rectangular warehouse. These layouts differ in:
    - the position of the depot, which is located in the bottom left corner of the warehouse in configuration A and at the bottom centre of the warehouse in configurations B and C;
    - the orientation of aisles, which are longitudinal in configuration A and B and transversal in configuration C.

    The rationale for the analysis of these layouts is to assess whether changes in warehouse structure could modify or affect the performance of the FW-ACO algorithm;

(2) *number of blocks*: this was varied from 1 to 4 (step 1);
(3) *number of aisles*: this was set at 4 or 10;
(4) *number of items in the picking list*: n was varied from 10 to 30 (step 10).

The warehouse configurations all have k_x=4 [m] and k_y=2 [m] and a number of storage locations per aisle side set at 11. Although these settings do not reflect any specific scenario, they are nonetheless representative of a real warehouse. Moreover, varying the warehouse settings allow to assess whether the algorithm performance depends on warehouse characteristics and size, and thus on problem complexity. For instance, moving from 4 to 10 aisles per block increases the total number of aisles in the warehouse by approx. 120%. Similarly, increasing the number of blocks in the warehouse from 1 to 4 allows the performance of the proposed routing algorithm to be evaluated as a function of the overall number of storage locations, and thus the number of nodes N of the warehouse graph, which increases from less than one hundred to more than five hundred nodes.

Combining these settings with the different warehouse configurations, we obtain $3x4x2x3 = 72$ scenarios. To facilitate the comparison between the outcomes provided by the FW-ACO algorithm and those of the six algorithms used as a benchmark, all the algorithms work on the same set of n items for a given warehouse layout.

4.2.3 Parameter setting

We use the ACO settings that emerged as most effective from the first set of analyses, i.e. ρ=0.9; α=1; β=5; τ=0.1; $Loops_{TOT}$=20,000. Q and m were kept unchanged compared to the first set of analyses. For each warehouse layout, n_{sim}=30 runs of the algorithm were carried out.

With respect to the MMAS algorithm, its main parameters were set as suggested by Stützle & Hoos (2000), i.e.: α=1; β=5; ρ=0.98; p_{best}=0.05; m=5. The computation of the maximum (τ_{max}) and minimum (τ_{min}) value of the pheromone trail was made as follows:

$$\tau_{max} = \frac{1}{1-\rho}\frac{1}{f(s^{opt})} \tag{9}$$

$$\tau_{min} = \frac{\tau_{max}(1-p_{dec})}{(avg-1)p_{dec}} \tag{10}$$

In the above formulae:

- $f(s^{opt})$ denotes the best solution found so far by the MMAS, meaning that any time a new best path is found, τ_{max} is updated accordingly;
- p_{dec} ($= \sqrt[n]{p_{best}}$) denotes the probability of choosing a defined solution component. n is the number of picking locations and $avg = \frac{n}{2}$.

With respect to the initialization ($t = 0$) of the pheromone trail, the following formula was used:

$$\tau_0 = \frac{1}{1-\rho}\frac{1}{f(\bar{s})} \tag{11}$$

where $f(\bar{s})$ was obtained initially as the average length of 10,000 paths generated randomly for each of the 72 scenarios described before. This choice allows τ_0 to be set at a value consistent with the problem complexity.

4.2.4 Results – travel distance

For each scenario, Table 4 shows the travel distance of pickers [m], identified by the seven algorithms tested. Data in *italics* highlight the best results obtained for each scenario, as well as the algorithm(s) that returned the most effective solution for that scenario. The main considerations that can be formulated from the results in Table 4 are summarised below.

1. *FW-ACO vs. S-shape and largest gap*. The FW-ACO algorithm always outperforms the S-shape and largest gap routing strategies in the identification of the shortest path. Indeed, neither of these heuristic policies is able to identify a solution better than that returned by the proposed algorithm for a given warehouse configuration. Only in one scenario (i.e. warehouse type A, n=30, 10 aisles, 3 blocks), the largest gap policy returns the same result as the FW-ACO, which is also the same as the Combined+ algorithm.
2. *FW-ACO vs. exact algorithms* (i.e. Ratliff & Rosenthal, 1983 and Roodbergen & de Koster 2001b). The FW-ACO returns the optimal length of the picking tour in 32 out of the 36 scenarios for which the exact result is available (88.89% of optimal results), showing better performance compared to the Combined+ (24 out of the 36 scenarios, 66.67% of optimal results) and MMAS (28 out of 36 scenarios, 77.78% of optimal results). In particular, the FW-ACO almost always returns the optimal route in warehouse layouts with 4 aisles (94.4% of optimal results), while in warehouse layouts with 10 aisles it is able to identify the optimal solution in 83.3% of the scenarios examined.
3. *Comparison of metaheuristic algorithms for 1-block and 2-block warehouses*. From Table 4 it can be seen that under simple warehouse layouts, i.e. 1-block or 2-block warehouses, the Combined+, MMAS and FW-ACO algorithms all frequently converge to the same path, which is also the shortest, as can be deduced from a comparison with the exact solution. This is, for instance, the case for warehouse types A and B, with n=20, 4 aisles and 1 block. In fact, in 1-block warehouses with 4 aisles, we have N=52 (N_s=8 and N_p=44). If the picking list includes, for instance n=20 items, the picker should visit on average 5 picking locations in each aisle. Since the number of storage locations per aisle side is fixed (and set at 11), the nodes to be visited are likely to be very close to each other, resulting in limited room for optimization. Hence, it is reasonable to expect that the path returned by the different algorithms would be almost the same. Generalising this consideration, we could argue that if the warehouse has a simple structure (i.e. up to 2 blocks), the difference in the path returned by the Combined+, MMAS and FW-ACO is not so appreciable. More precisely, on average the FW-ACO

provides routes that are approximately 0.44% and 2.65% shorter than those generated by the MMAS and Combined+ algorithms in warehouse layouts up to 2 blocks.

4. *Comparison of metaheuristic algorithms for 3-block and 4-block warehouses*. When looking at more complex warehouse configurations (e.g. 10 aisles, 3 blocks or 10 aisles, 4 blocks), an initial consideration is that there are no scenarios where the FW-ACO, MMAS and Combined+ algorithms all return the same travel distance. Conversely, we generally found that the FW-ACO is able to identify paths that are significantly shorter than those returned by both the MMAS and the Combined+ heuristic. Looking at this latter algorithm, only in some limited cases (e.g. warehouse types A, B and C, with n=30, 10 aisles and 3 blocks) the result returned is the same as the FW-ACO. In addition, in 55.56% of the scenarios considered the FW-ACO returns a path shorter than that returned by the MMAS. Even if taking the best solution returned either by the MMAS or the Combined+ algorithm, it is easy to see that the FW-ACO algorithm still returns better results in terms of path length in 16.67% of the scenarios examined, which all refer to either n=20 or n=30. This shows that the performance of the proposed algorithm is particularly appreciable when the number of items in the picking list is high. Overall, on average the routes generated by the FW-ACO are approximately 2.09% and 8.43% shorter than those returned by the MMAS and Combined+ algorithms in warehouse layouts with 3 or 4 blocks.
5. *FW-ACO vs. MMAS*. Because the FW-ACO and the MMAS are both derived from the ACO algorithm, it is useful to provide a detailed comparison of their performance. Table 4 shows that the MMAS has the same performance of the proposed FW-ACO in 62.5% of the warehouse configurations examined overall. Conversely, in the remaining 27 scenarios, the FW-ACO is able to return a shorter path compared to the MMAS. The two algorithms always return the same path when considering picking lists with n=10 items, no matter which warehouse configuration there is. Although the optimal solution is only available for 1-block and 2-block warehouses, the fact that both algorithms always converge to the same result could lead to the conclusion that this is also the optimal routing. Conversely, as n increases (i.e. n=20 and n=30), the FW-ACO generally outperforms the MMAS, which is only able to return the same distance as the FW-ACO in 21 out of the 48 scenarios (43.7%). To be more precise, with n=20 the FW-ACO outperforms the MMAS by approximately 1.07% on average, while with n=30 the percentage difference between the two algorithms reaches 2.72%. Overall, by combining this result with that of the previous points 3 and 4, we can conclude that as problem complexity increases, the FW-ACO algorithm is generally more effective at minimising the travel distance of the picker than the MMAS and Combined+ algorithms.

4.2.5 Results – computational performance

We now compare the computational performance of the Combined+, MMAS and FW-ACO algorithms. With respect to the MMAS and FW-ACO, a preliminary comparison can be made using the outcomes proposed in Table 4, which also reports the computational time required to run these algorithms in all warehouse configurations investigated. From these outcomes, it can be argued that in general the FW-ACO algorithm is more efficient than the MMAS from a computational perspective, as it ensures an average saving of 6.30% in the computational time, with a peak of 8.06% saving where n=30. It is worth noting that these results refer to the ACO stage of the algorithm, whose computational time depends on problem complexity (n) and the number of iterations allowed ($Loops_{TOT}$). Indeed, the ACO stage of the algorithm should be run any time a new picking mission is started, while the FW stage does not depend on n and needs to be carried out only once for a given warehouse layout[3].

As far as the Combined+ algorithm is concerned, its computational performance cannot be directly compared to that of the FW-ACO (and this is why it was not reported in Table 4), because the Combined+ does not make use of iterations but instead of a computational procedure, and thus generates only one route. To carry out a meaningful comparison, for some selected warehouse configurations we have recorded the total time required by the FW-ACO to achieve the "best found" solution. The scenarios examined reflect the most complex and the simplest warehouse configurations (4 blocks, 10 aisles and 1 block, 4 aisles respectively), with n=10, 20 and 30. The related results, averaged on the warehouse layouts (A, B and C) and on the simulation runs (n_{sim}=30), are proposed in Table 5 and compared to the computational time required to run the Combined+ algorithm on the same warehouse configuration, once again averaged on the warehouse layout. The same evaluation was made for the MMAS, with results again reported in Table 5. From these outcomes it is easy to see that both the FW-ACO and the MMAS are quite quick at converging to the "best found" solution (1.41 s vs. 2.14 s on average), but that nonetheless the FW-ACO is generally more effective, ensuring a 34.02% saving in the computational time compared to the MMAS. It can also be seen that for both algorithms the time required to reach the "best found" solution increases with n and warehouse complexity. The computational time of the Combined+ algorithm seems to be less dependent on n, while it slightly increases with warehouse complexity. Overall, when n increases, the Combined+ is usually faster than the FW-ACO algorithm, although the solution returned is as good as that of the FW-ACO algorithm in less than half the configurations tested in Table 5 (see Table 4).

[3] For the sake of completeness, we report some results relating to the FW stage of the algorithm. This stage required on average 35.95 s (maximum 39.35 s, minimum 36.69 s, standard deviation 2.019 s) in the most complex warehouse configuration (i.e. 4 blocks, 10 aisles), and on average 0.080 s (maximum 0.100 s, minimum 0.068 s, standard deviation 0.049 s) in the simplest one (i.e. 1 block, 4 aisles).

Scenario				Benchmark algorithms							Proposed algorithm	
Warehouse layout	n	**Number of aisles**	**Number of blocks**	**S-shape**	**Largest gap**	**Ratliff & Rosenthal (1983)**	**Roodbergen & de Koster (2001b)**	**Combined+**	**MMAS**	**Computational time - MMAS**	**FW-ACO**	**Computational time – FW-ACO**
A	10	4	1	164	152	*148*	-	152	*148*	9.623	*148*	9.487
			2	224	196	-	*176*	202	*176*	9.035	*176*	9.171
			3	304	268	-	-	288	*236*	10.024	*236*	9.244
			4	340	276	-	-	276	*260*	10.762	*260*	8.218
		10	1	312	216	*204*	-	224	*204*	13.108	*204*	13.442
			2	340	316	-	260	260	260	10.945	260	11.002
			3	444	368	-	-	358	*296*	12.607	*296*	9.724
			4	468	464	-	-	440	*316*	9.730	*316*	9.021
	20	4	1	164	172	*152*	-	*152*	*152*	24.253	*152*	18.821
			2	316	284	-	*248*	284	*256*	23.269	*256*	19.201
			3	392	356	-	-	330	320	26.444	*316*	20.460
			4	516	388	-	-	*352*	360	26.998	*352*	25.109
		10	1	332	328	*292*		*292*	*292*	27.183	*292*	26.163
			2	544	436	-	*400*	420	408	23.805	404	21.411
			3	572	432	-	-	424	412	21.087	*404*	19.879
			4	648	644	-	-	550	504	23.257	*492*	24.609
	30	4	1	172	192	*164*	-	*164*	*164*	37.968	*164*	35.886
			2	308	300	-	*264*	290	*264*	31.399	*264*	31.740
			3	472	424	-	-	398	*388*	32.640	*388*	27.660
			4	592	516	-	-	*468*	496	37.021	*468*	28.793
		10	1	356	376	*316*	-	326	*316*	34.769	*316*	28.564
			2	592	524	-	*476*	484	496	33.471	484	29.612
			3	768	540	-	-	*540*	580	34.312	*540*	27.032
			4	888	800	-	-	654	660	33.914	*640*	36.572
B	10	4	1	164	152	*148*	-	164	*148*	10.631	*148*	12.043
			2	220	204	-	*176*	*176*	*176*	14.168	*176*	11.285
			3	316	270	-	-	288	*236*	10.532	*236*	9.819
			4	292	320	-	-	276	*260*	9.326	*260*	11.916
		10	1	312	216	*196*	-	254	*196*	9.128	*196*	9.576
			2	344	320	-	*260*	*260*	264	11.237	260	9.530
			3	456	426	-	-	368	*292*	10.322	*292*	9.320
			4	472	460	-	-	412	*324*	13.132	*324*	12.773
	20	4	1	164	160	*152*	-	*152*	*152*	27.720	*152*	23.612
			2	316	292	-	*256*	*256*	*256*	22.513	*256*	22.339
			3	408	350	-	-	314	*308*	21.213	*308*	21.437

			4	516	452	-	-	*356*	360	23.377	*356*	21.037
		10	1	332	328	*292*	-	296	*292*	22.332	*292*	20.583
			2	544	468	-	*392*	396	400	24.689	396	24.359
			3	552	512	-	-	420	408	24.941	*404*	22.822
			4	660	648	-	-	546	*492*	23.054	*492*	19.612
	30	4	1	172	192	*158*	-	*158*	*158*	37.968	*158*	28.247
			2	308	296	-	*264*	*264*	*264*	31.399	*264*	36.115
			3	476	424	-	-	394	*384*	32.640	*384*	37.944
			4	592	572	-	-	464	492	37.021	*460*	32.547
		10	1	356	376	*316*	-	*316*	*316*	34.769	*316*	29.719
			2	612	556	-	*484*	*484*	500	33.471	*484*	40.685
			3	720	668	-	-	*560*	576	34.312	*560*	31.230
			4	916	876	-	-	654	668	33.914	*640*	32.308
C	10	4	1	164	152	*130*	-	*130*	*130*	10.593	*130*	13.172
			2	224	204	-	*176*	*176*	*176*	10.137	*176*	9.902
			3	320	268	-	-	288	*260*	10.566	*260*	10.277
			4	344	276	-	-	252	*244*	12.024	*244*	10.804
		10	1	312	216	*192*	-	*192*	*192*	10.160	*192*	11.700
			2	324	324	-	*260*	*260*	*260*	9.352	*260*	9.448
			3	444	368	-	-	360	*296*	10.834	*296*	9.190
			4	468	464	-	-	404	*316*	11.027	*316*	11.576
	20	4	1	164	172	*146*	-	*146*	*146*	24.957	*146*	21.882
			2	304	292	-	*256*	*256*	*256*	26.134	*256*	26.318
			3	392	356	-	-	312	316	22.666	*298*	21.984
			4	544	388	-	-	*356*	364	20.251	*356*	24.191
		10	1	332	328	*272*	-	*272*	*272*	22.446	*272*	20.108
			2	504	444	-	*390*	*390*	*390*	21.523	*390*	20.989
			3	528	438	-	-	416	406	25.802	*398*	25.386
			4	648	644	-	-	528	508	22.507	*492*	25.079
	30	4	1	172	192	*158*	-	*158*	162	36.773	*158*	31.571
			2	296	316	-	*260*	*260*	*260*	33.104	*260*	32.610
			3	436	424	-	-	382	386	40.522	*370*	33.442
			4	632	516	-	-	*452*	484	33.648	*452*	34.793
		10	1	356	376	*312*	-	*312*	316	35.435	*312*	27.114
			2	584	540	-	*484*	*484*	504	36.553	*484*	34.894
			3	724	624	-	-	*584*	600	39.606	*584*	37.811
			4	924	800	-	-	642	672	37.099	*630*	30.994

Table 4: experimental results (shortest path in [m] and computational time in [s]) for the second set of analyses - comparison with other routing strategies. *Note*: italics = best found solution.

	Warehouse configuration		FW-ACO	MMAS	Combined+
n	Number of aisles	Number of blocks	*Computational time to reach the "best found" solution [s]*	*Computational time to reach the "best found" solution [s]*	*Computational time to reach the solution [s]*
10	4	1	*0.0052*	*0.0109*	*0.125*
10	10	4	*0.0932*	*0.1246*	*0.445*
20	4	1	*0.0395*	*0.0729*	*0.130*
20	10	4	*2.0774*	*4.8990*	*1.515*
30	4	1	*0.3431*	*0.4581*	*0.140*
30	10	4	*5.9503*	*7.3303*	*1.005*

Table 5: experimental results for the second set of analyses – details of the computational time for FW-ACO, MMAS and Combined+ in some selected warehouse configurations.

5 A case study

We now illustrate the application of the FW-ACO algorithm in the case of *Number1 Logistics Group*, a major Italian company operating as a logistics service provider for the grocery, fast-moving consumer goods, pharmaceutical and Ho.Re.Ca. fields. *Number1* owns a network of 7 distribution centres, 37 transit points and 21 warehouses, all located in Italy, and in 2015 it handled approximately 323 million boxes, with 60 million of them prepared by means of picking activities. Overall, this led to more than 11 million picking locations being visited.

We consider *Number1's* warehouse located in Parma and examine the portion of the warehouse dedicated to baby food. This is a 2-block warehouse, with 12 aisles per block and 66 storage locations per aisle side. The warehouse consists of racks, with 6 levels in height; picking activities are carried out at ground level. A manual, picker-to-part logic is adopted. The average number of pickers in this part of the warehouse is 10-12; however, pickers can increase to 15-18 during peak periods. Overall, this part of the warehouse includes $2x12x66 = 1584$ storage locations. The input/output depot is located at the bottom of the first (left) aisle. Further warehouse parameters are $k_x = 5.4\ m$ and $k_y = 0.93\ m$. The stock consists of approximately 700 different items grouped into 3 categories, labelled as C_1, C_2 and C_3. Product categories reflect the weight of the items, which is higher for C_1 and lower for C_3. Therefore, the picker should pick, in order of priority, the item from C_1, to ensure that they will not damage the remaining items picked, then items from C_2, and lastly items from C_3. The number of storage locations for the item categories is 310, 298 and 976 respectively for C_1, C_2 and C_3. The current routing policy adopted in the warehouse is S-shape. The picker starts from the bottom left corner of the warehouse and picks items from C_1; he then moves to the next aisles to pick items from C_2 and C_3, and lastly returns to the depot.

To show the implementation of the FW-ACO in this context, we consider a picking list with n=30 items, which reflects the average length of the picking lists for baby food handled by the company in 2015. Items are located as shown in Figure 4. Out of the 30 items, 6 are from C_1, 7 from C_2 and remaining ones from C_3. The S-shape routing policy, coupled with a proper allocation of the item

categories, provides an effective route (depicted in Figure 4-a) that respects the order of priority of the items to be picked and is easy to understand by pickers. However, the policy is less effective with respect to the total distance covered, which accounts for approximately 900 m. The FW-ACO algorithm was run on the same picking problem, with the settings described in section 4.1.3. However, compared to the analyses described previously, in this particular case study the picking tour is constrained by the different priorities given to the item categories during picking[4]. This means that the FW-ACO should identify the shortest path and at the same time meet the constraint of picking items from C_1 first and items from C_3 last. This point was taken into account by introducing a slight correction in the ACO stage of the algorithm. In particular, ACO was forced to work on one group of items at a time: items from C_2 can only be selected if all items from C_1 have been picked; otherwise, the probability of visiting the location of items from C_2 is set at zero. The same approach is used for items from C_3, whose probability of being picked becomes higher than zero only if all items from C_1 and C_2 have been picked. The same correction was implemented in the MMAS algorithm. In Table 6 we compare the results returned by the two algorithms, in terms of "best found" path and computational time on a sample of n_{sim}=10 runs[5]. The "best found" paths for MMAS and FW-ACO are depicted in Figure 4-b and 4-c. From Table 6 it is easy to see that, although the computational time required to run 20,000 iterations of the MMAS and FW-ACO algorithms is not significantly different, the FW-ACO needs 58.86% less time to reach the "best found" solution compared to the MMAS. As far as the travel distance is concerned, both the FW-ACO and the MMAS are able to identify routes that respect the order of priority of the items to be picked and that, at the same time, are significantly shorter compared to the S-shape path. The FW-ACO algorithm, in particular, returns a total distance of approximately 608.04 m, with a 32.44% reduction compared to the original one. Also, the "best found" route of the FW-ACO ensures a saving of approximately 3.26% in path length, compared to the MMAS.

	Travel distance [m]	Total computational time [s]	Computational time [s] to reach the "best found" solution
S-shape	900	-	-
FW-ACO	608.04	30.26	5.32
MMAS	628.50	33.48	12.87

Table 6: performance of the MMAS and FW-ACO algorithms in the case study.

[4] Because of the presence of constraints on the order in which items should be picked, neither the Combined+, largest gap nor the optimal algorithm can be applied (in their original form) to this specific case study. Indeed, all these algorithms would return a picking tour that fails to comply with the order of priority of the items picked; therefore, we cannot compare the performance of the FW-ACO with these routing algorithms in the targeted context.

[5] Again, the results in Table 6 refer to the ACO stage of the FW-ACO algorithm. As far as the FW stage is concerned, it required on average 18.38 s (maximum 21.91 s, minimum 17.49 s, standard deviation 0.91 s).

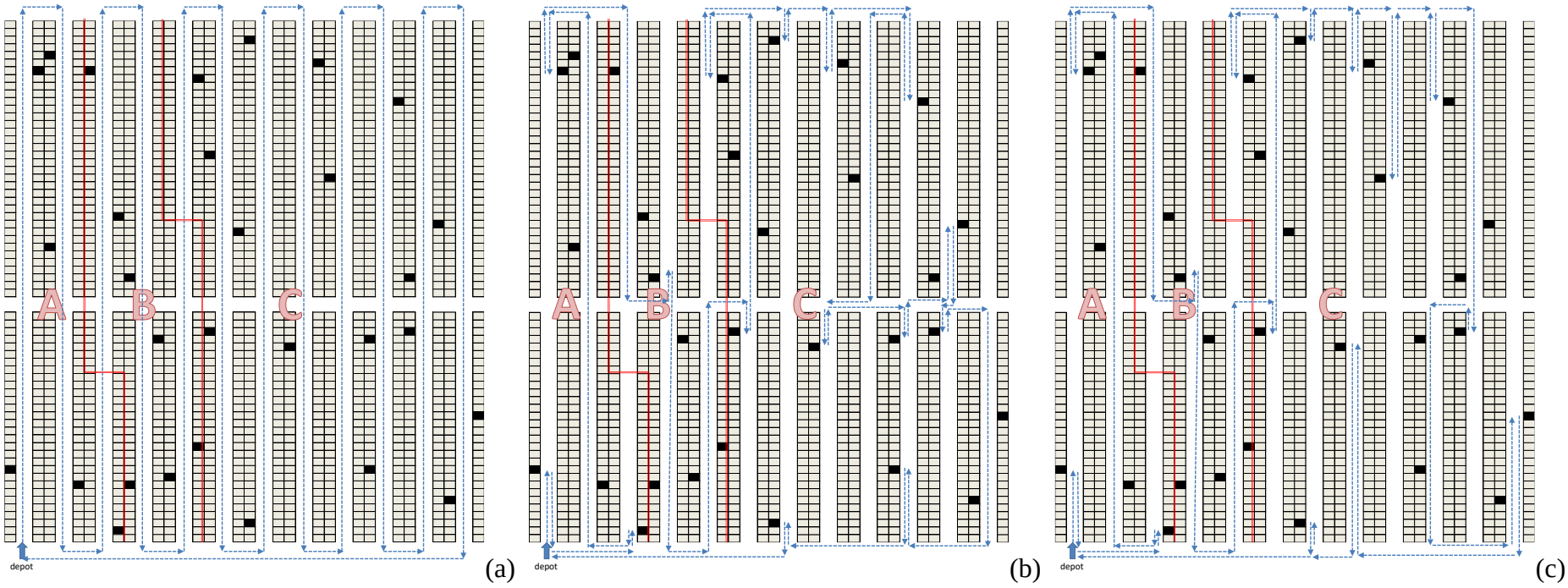

Figure 4: current routing (a) and routes returned by the FW-ACO (b) and MMAS (c) in the case study.

The percentage saving in the travel distance generated by the FW-ACO compared to the S-shape route is expected to decrease slightly with the increase in n. Indeed, as the route is constrained, with more items to be picked the path returned by the FW-ACO is likely to be more similar to that generated by the S-shape policy; conversely, savings in the travel distance could be higher for a lower n. To quantify this point, the analysis was extended to the overall number of picking lists of baby food handled by *Number1's* warehouse in 2015, i.e. approximately 45,000 picking lists, on which we tested both the FW-ACO and the MMAS algorithms. We found that overall, the FW-ACO algorithm is able to reduce the travel distance of pickers by almost 22.05% compared to the current policy used by the company; conversely, the saving generated by the MMAS was less appreciable (14.15%).

Reducing the travel distance has obvious consequences for the time required for picking operations. More specifically, in the portion of warehouse analysed, we estimated that the total saving in the picking distance that can be achieved when adopting the FW-ACO algorithm could reach 19,845 m/day (referring to the picking activities carried out in 2015). Hypothesising an average speed of 1.2 m/s, which is a typical walking speed for a picker (Bottani et al., 2012), and 220 working days per year, such a saving results in approximately 1,010.7 hours/year, which would reduce the cost of picking activities by more than 18,000 €/year.

6 Conclusions

This paper has proposed a new routing algorithm for the minimization of the travel distance of pickers in manual warehouses. The algorithm, called FW-ACO, combines the ACO metaheuristic and the FW algorithm. This latter algorithm was chosen for integration with ACO because of its ability to find the shortest path between any pair of nodes even in a complex graph, as is frequently the case in complex warehouse structures. The FW-ACO is a two-stage algorithm: in the first stage, the warehouse layout is schematically represented as a graph and the FW algorithm is used to identify the shortest path connecting each pair of nodes in the graph. The ACO algorithm is used in the second stage to identify the shortest route of the picker.

The performance of the FW-ACO algorithm was tested by means of two sets of analyses. The first analysis aimed to assess whether the performance of the FW-ACO algorithm can be affected by the settings of the typical ACO parameters ($\rho, \tau, \alpha, \beta, Loops_{TOT}$), as well as by the problem complexity (n). From this set of analyses, it can be concluded that, in general, the FW-ACO algorithm is particularly effective in finding the shortest path of pickers in the case of small-scale problems (n=10 or n=20 items). Nonetheless, with appropriate settings, the algorithm also becomes effective for more complex problems (n=50). A full-factorial DOE analysis confirmed the suitability of the following algorithm settings: ρ=0.9; α=1; β=5; τ=0.1; $Loops_{TOT}$=20,000.

The second set of analyses compared the performance of the FW-ACO algorithm, in the most effective setting, with six routing algorithms, including both exact and heuristic procedures. Different warehouse layouts, in terms of number of blocks, number of aisles and location of the input/output depot, as well as different levels of problem complexity (n), were considered in the analysis. The results enable us to conclude that the FW-ACO always outperforms both the S-shape and largest gap routing strategies in the identification of the shortest path, under all the warehouse configurations examined. The FW-ACO also returns the optimal length of the picking tour in 32 out of the 36 scenarios for which the exact result is available (88.89% of optimal results), resulting in a better performance than the MMAS and Combined+ algorithms. Most importantly, the performance of the FW-ACO algorithm is particularly appreciable for complex warehouse configurations, where the shortest path generated is significantly better than that returned by the remaining routing algorithms. In these configurations, even if taking the best solution returned by either the MMAS or the Combined+ algorithm, the FW-ACO algorithm still returns better results in terms of path length in 16.67% of the scenarios, which all refer to either n=20 or n=30, showing good performance even in case of high n. This suggests that the proposed algorithm goes beyond a simple combination of existing heuristics. In fact, our proposed algorithm is also a combination, but of an existing heuristic plus an operational research algorithm (i.e. the Floyd Warshall one), and this is probably the reason why its performance is better than that of the existing heuristics. Therefore, from the point of view of travel distance, we can conclude that the FW-ACO algorithm adds quality compared to some well-known heuristic routing strategies.

A further relevant point evaluated in this set of analyses is the computational time required by the FW-ACO algorithm. As picker routing is an operational decision, which has to be taken almost in real-time, a routing algorithm should provide good routes in a short time. Because of its two-stage structure, the FW-ACO algorithm is particularly effective in this regard. Indeed, the FW stage of the algorithm, which needs to be carried out only once for a given warehouse layout, required approximately 36 s to be run on the most complex warehouse configuration examined (i.e. 4 blocks, 10 aisles). The ACO stage, which instead needs to be run any time a new picking mission is started, required on average 33 s to be run with n=30 and $Loops_{TOT}$=20,000 in the most complex warehouse configuration. In the same configuration, the FW-ACO required on average 5.95 s to converge to the "best found" solution. Such a computational time is fully compatible with the warehouse manager's need to adopt an algorithm in real time.

As a final step, the FW-ACO algorithm was applied to the warehouse of a main Italian logistics company to show its practical usefulness and quantify the savings resulting from its implementation in a real case. The application context was interesting because of the presence of constraints in picking operations. Specifically, the items handled in the targeted warehouse are grouped into three categories (C_1, C_2 and C_3), with items from C_1 that need to be picked first because of their higher weight, and

items from C_3 that need to be picked last. Some small adaptations were made to the FW-ACO to ensure that the algorithm would find a picking route consistent with these constraints. The application of the proposed algorithm on a picking list with n=30 items showed that the FW-ACO is effective in finding a route that respects the order of priority of the items to be picked and, at the same time, enables the original routing to be reduced by more than 32%. The algorithm also performed better than the MMAS, which was also applied to the case study, both in terms of path lengths (-3.26% route length) and computational performance (-58.88% computational time required to reach the "best found" solution). Overall, the use of the FW-ACO instead of the original routing policy (i.e. S-shape) results in a significant reduction (more than 22%) in the total picking distance covered in 2015 by the company in question.

The above summary leads to the conclusion that the proposed algorithm shows promising results, which are expected to encourage its practical adoption. More specifically, warehouse managers are only expected to run the FW stage of the algorithm on their warehouse configuration once; they could then effectively exploit the ACO stage of the algorithm to identify the shortest path of pickers before they start a new picking mission. Obviously, pickers are generally not expected to understand the logic of metaheuristic algorithms, and therefore when adopting the proposed algorithm, the sequencing of the items should be given to the picker. For instance, the shortest path can be communicated to the picker almost in real time via voice picking or mobile terminals. Furthermore, the algorithm is suitable for application in any warehouse layout, and turned out to be particularly useful in configurations for which exact algorithms are not available; in these configurations, the algorithm provided very good outcomes in a limited computational time. The case study also showed that the FW-ACO algorithm could be adapted to contexts where constraints exist on picking operations (e.g. a warehouse that handles breakable items); these scenarios are rarely considered in scientific literature when dealing with the minimization of picking distances. As a final point, from a scientific perspective the application of the ACO algorithm to the picking problem has been explored in a very limited number of studies. Similarly, the integration of the ACO algorithm with the FW one has not been proposed by researchers to our knowledge. For these reasons, as well as the promising results obtained by the proposed algorithm, we believe that this paper represents an interesting addition to the scientific literature. Future studies could investigate the adaptation of the FW-ACO algorithm to different picking contexts (e.g. batch picking or zone picking), the use of the algorithm in combination with a specific storage allocation policy (e.g. COI-based allocation), or the extension of the algorithm to the case of more pickers, with congestion considerations. More recent nature-inspired algorithms could also be applied to the picking problem and the related performance could be compared to the FW-ACO algorithm to gain further insights. A further interesting future research direction is to embody the FW-ACO algorithm in an *ad hoc* software tool, to automate the algorithm steps and make it directly usable by warehouse managers.

References

1. Bäck, T. (1996). Evolutionary algorithms in theory and practice. USA: Oxford University Press.
2. Bellman, R. (1958). On a routing problem. Quarterly of Applied Mathematics, 16, 87-90.
3. Bottani, E., Cecconi, M., Vignali, G., & Montanari, R. (2012). Optimisation of storage allocation in order picking operations through a genetic algorithm. International Journal of Logistics: Research and Applications, 15(2), 127-146.
4. Bottani, E., Montanari, R., Rinaldi, M., & Vignali, G. (2015). Intelligent Algorithms for Warehouse Management. In C. Kahraman, & S. Çevik Onar, Intelligent Techniques in Engineering Management: Theory and Applications, p.645-667. Switzerland: Springer International Publishing.
5. Box, G., Hunter, J., & Hunter, W. (2005). Statistics for Experimenters: Design, Innovation, and Discovery. Hoboken (USA): Wiley-Interscience.
6. Brezina, I., & Čičková, Z. (2011). Solving the travelling salesman problem using the ant colony optimization. Management Information Systems, 6(4), 10-14.
7. Caron, F., Marchet, G., & Perego, A. (1998). Routing policies and COI-based storage policies in picker-to-part systems. International Journal of Production Research, 36(3), 713-732.
8. Caron, F., Marchet, G., & Perego, A. (2000). Optimal layout in low-level picker-to-part systems. International Journal of Production Research, 38(1), 101-111.
9. Çelk, M., & Süral, H. (2014). Order Picking under Random and Turnover-based Storage Policies in Fishbone Aisle Warehouses. IIE Transactions 46 (3), 283–300.
10. Chen, F., Wang, H., Qi, C., & Xie, Y. (2013). An ant colony optimization routing algorithm for two order pickers with congestion consideration. Computers & Industrial Engineering, 66(1), 77-85.
11. Chen, F., Wang, H., Xie, Y., & Qi, C. (2016). An ACO-based online routing method for multiple order pickers with congestion consideration in warehouse. Journal of Intelligent Manufacturing, 27(2), 389-408. DOI:10.1007/s10845-014-0871-1
12. Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies. Proceedings of ECAL91—European Conference on Artificial Life, (p. 134–142). Paris (France). Retrieved March 3, 2015, from http://faculty.washington.edu/paymana/swarm/colorni92-ecal.pdf
13. Cormen, T., Leiserson, C., & Rivest, R. (2009). Introduction to Algorithms. USA: Massachusetts Institute of Technology.
14. Coyle, J., Bardi, E., & Langley, C. (1996). The management of business logistics. Minneapolis: West Publishing Company.
15. De Jong, J., & Wiering, M. (2001). Multiple ant colony systems for the busstop allocation problem. University of Utrecht, Department of Philology, Utrecht (Holland). Retrieved July 2, 2015 http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.9587
16. de Koster, R., & Van der Poort, E. (1998). Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions. IIE Transactions, 30, 469- 480.
17. de Koster, R., Le-Duc, T., & Roodbergen, J. (2007). Design and control of warehouse order picking: a literature review. European Journal of Operational Research, 182(2), 481-501.
18. de Koster, R., Le-Duc, T., & Zaerpour, N. (2012). Determining the number of zones in a pick-and-sort order picking system. International Journal of Production Research, 50(3), 757-771.

19. Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics - Part B, 26(1), 1-13.
20. Eggers, J., Feillet, D., Kehl, S., Wagner, M., & Yannou, B. (2003). Optimization of the keyboard arrangement problem using an ant colony algorithm. European Journal of Operational Research, 148, 672-686.
21. Fidanova, S., & Marinov, P. (2013). Number of ants versus number of iterations on Ant Colony Optimization algorithm for wireless sensor layout. Institute of Information and Communication Technologies, Bulgarian Academy of Science. Retrieved March 20, 2016, from http://www.iict.bas.bg/acomin/events/8-10-October-2013/Number%20of%20Ants%20Versus%20Number%20of%20Iterations%20on%20Ant%20Colony%20Optimization%20Algorithm%20for%20Wireless%20Sensor%20Layout.pdf
22. Fidanova, S., Marinov, P., Paprzycki, M. (2014). Influence of the number of ants on multi-objective Ant Colony Optimization algorithm for wireless sensor network layout. Large-Scale Scientific Computing, 8353 LNCS, pp.232-239
23. Floyd, R. (1962). Algorithm 97: Shortest Path. Communications of the ACM, 5(6), 345. DOI:10.1145/367766.368168
24. Frazelle, E. (2002). World-class Warehousing and Material Handling. New York: McGraw Hill.
25. Gademann, N., & Van de Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Transactions, 37, 63-75.
26. Gu, J., Goetschalckx, M., & McGinnis, L. (2007). Research on warehouse operation: A comprehensive review. European Journal of Operational Research, 177, 1-21.
27. Hall, R. (1993). Distance approximations for routing manual pickers in a warehouse. IIE Transactions, 25(4), 76-87.
28. Henn, S. (2012). Algorithms for on-line order batching in an order picking warehouse. Computers & Operations Research, 39, 2549-2563.
29. Henn, S., Koch, S., & Wascher, G. (2011). Order batching in order picking warehouses: a survey of solution approaches. Retrieved March 18, 2016, from http://www.fww.ovgu.de/fww_media/femm/femm_2011/2011_01.pdf
30. Henn, S., & Schmid, V. (2013). Metaheuristics for order batching and sequencing in manual order picking systems. Computers & Industrial Engineering, 22, 338-351.
31. Hong, S., Johnson, A.L., Peters, B.A., (2012). Batch picking in narrow-aisle order picking systems with consideration for picker blocking. European Journal of Operational Research, 221, 557-570
32. Jalali, S., & Noroozi, M. (2009). Determination of the optimal escape routes of underground mine networks in emergency cases. Safety Science, 47, 1077-1082.
33. Jane, C.-C., & Laih, Y.-W. (2005). A clustering algorithm for item assignment in a synchronized zone order picking system. European Journal of Operational Research, 166, 489-496.
34. Jarvis, J., & McDowell, E. (1991). Optimal product layout in an order picking warehouse. IIE Transactions, 23(1), 93-102.
35. Kim, B.-I., & Jeon, S. (2009). A comparison of algorithms for origin–destination matrix generation on real road networks and an approximation approach. Computers & Industrial Engineering, 56, 70-76.

36. Kuhlman, D. (2013). A Python Book: Beginning Python, Advanced Python, and Python Exercises. Retrieved July 2, 2016, from http://www.davekuhlman.org/python_book_01.pdf
37. Kulak, O., Sahin, Y., & Taner, M. (2012). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. Flexible Services and Manufacturing Journal, 24(1), 52-80.
38. Le-Duc, T., & de Koster, R. (2007). Travel time estimation and order batching in a 2-block warehouse. European Journal of Operational Research, 176(1), 374-388.
39. Lu, W., McFarlane, D., Giannikas, V., Zhang, Q. (2016). An algorithm for dynamic order-picking in warehouse operations. European Journal of Operational Research, 248 (1), 107-122.
40. Lucic, P. (2002). Modeling transportation problems using concepts of swarm intelligence and soft computing. Polytechnic Institute and State University, Civil Engineering Department, Virginia (USA). Retrieved June 5, 2015, from http://scholar.lib.vt.edu/theses/available/etd-03092002-231724/unrestricted/Panta_Lucic_ED.pdf
41. Manzini, R., Bindi, F., Ferrari, E., & Pareschi, A. (2012). Correlated storage assignment and iso-time mapping adopting tri-later stackers. A case study from tile industry. In R. Manzini, Warehousing in the Global Supply Chain - Advanced Models, Tools and Applications for Storage Systems (p. 373-396). London: Springer-Verlag.
42. Mariano, C., & Morales, E. (1999). MOAQ: An Ant-Q Algorithm for Multiple Objective Optimization Problems. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), 1, p. 894-901. San-Francisco (USA).
43. Matusiak, M., de Koster, R., Kroon, L., & Saarinen, J. (2014). A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. European Journal of Operational Research, 236, 968–977.
44. Montgomery, D., & Runger, G. (2003). Applied statistics and probability for engineers. New York (USA): John Wiley & Sons, Inc.
45. Mowrey, C.H., Parikh, P.J. (2014). Mixed-width aisle configurations for order picking in distribution centers. European Journal of Operational Research, 232(1), 87–97
46. Pan, J.C.-H., & Wu, M.-H. (2012). Throughput analysis for order picking system with multiple pickers and aisle congestion considerations. Computers & Operations Research, 39(7), 1661-1672
47. Parikh, P., & Meller, R. (2010). A travel-time model for a person-onboard order picking system. European Journal of Operational Research, 200, 385-394.
48. Petersen, C. (1997). An evaluation of order picking routeing policies. International Journal of Operations & Production Management, 17(11), 1098-1111.
49. Petersen, C. (1999). The impact of routing and storage policies on warehouse efficiency. International Journal of Operations & Production Management, 19 (10), 1053-1064.
50. Petersen, C. (2002). Considerations in order picking zone configuration. International Journal of Operations & Production Management, 22(7), 793-805.
51. Petersen, C., & Aase, G. (2004). A comparison of picking, storage, and routing policies in manual order picking. International Journal of Production Economics, 92(1), 11-19.

52. Petersen, C., & Schmenner, R. (1999). An evaluation of routing and volume-based storage policies in an order picking operation. Decision Science, 30(2), 481-501.
53. Rao, S.S., & Adil, G.K., (2013). Optimal class boundaries, number of aisles, and pick list size for low-level order picking systems. IIE Transactions, 45, 1309–1321.
54. Ratliff, H., & Rosenthal, A. (1983). Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Operations Research, 31(3), 507-521.
55. Roodbergen, K., & de Koster, R. (1998). Routing orderpickers in a warehouse with multiple cross aisles. In R. Graves, L. McGinnis, D. Medeiros, R. Ward, & M. Wilhelm, Progress in Material Handling Research: 1998 (p. 451-467). Charlotte: Material Handling Institute.
56. Roodbergen, K., & de Koster, R. (2001a). Routing methods for warehouses with multiple cross aisles. International Journal of Production Research, 39(9), 1865-1883.
57. Roodbergen, K., & de Koster, R. (2001b). Routing order pickers in a warehouse with a middle aisle. European Journal of Operational Research, 133(1), 32-43.
58. Roodbergen, K., & Vis, I. (2006). A model for warehouse layout. IIE Transactions, 38(10), 799-811.
59. Roodbergen, K.J., Sharp, G.P. & Vis, I.F.A. (2008). Designing the layout structure of manual order picking areas in warehouses. IIE Transactions, 40, 1032–1045.
60. Rouwenhorst, B., Reuter, B., Stockrahm, V., van Houtum, G., Mantel, R., & Zijm, W. (2000). Warehouse design and control: Framework and literature review. European Journal of Operational Research, 122, 515-533.
61. Scholz, A., Henn, S., Stuhlmann, M., Wäscher, G. (2016). A new mathematical programming formulation for the Single-Picker Routing Problem. European Journal of Operational Research, 253, 68-84
62. Shtovba, S. (2005). Ant Algorithms: Theory and Applications. Programming and Computer Software, 31(4), 167-178.
63. Stützle, T., & Hoos, H. (2000). MAX–MIN ant system. Future Generation Computer Systems, 16, 889–914.
64. Theys, C., Bräysy, O., Dullaert, W., & Raa, B. (2010). Using a TSP heuristic for routing order pickers in warehouses. European Journal of Operational Research, 200(3), 755-763.
65. Tompkins, J., White, J., Bozer, Y., Frazelle, E., Tanchoco, J., & Trevin, J. (1996). Facilities Planning. New York: Wiley.
66. Van Nieuwenhuyse, I., & de Koster, R. (2009). Evaluating order throughput time in 2-block warehouses with time window batching. International Journal of Production Economics, 121(2), 654-664.
67. Vaughan, T., & Petersen, C. (1999). The effect of warehouse cross aisles on order picking efficiency. International Journal of Production Research, 37, 881-897.
68. Warshall, S. (1962). A theorem on Boolean matrices. Journal of the ACM, 9(1), 11-12. DOI:10.1145/321105.321107
69. Webster, S., Ruben, R., & Yang, K.-K. (2012). Impact of storage assignment decisions on a bucket brigade order picking line. Production and Operations Management, 21(2), 276-290.
70. Xing, B., Gao, W.-J., Nelwamondo, F., Battle, K., & Marwala, T. (2010). Ant colony optimization for automated storage and retrieval system. Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 1, p.1-7. Barcelona (Spain).

71. Zhang, G., & Lai, K. (2006). Combining path relinking and genetic algorithms for the multiple-level warehouse layout problem. European Journal of Operational Research, 169(2), 413-425.

72. Zhu, W., & Curry, J. (2009). Parallel ant colony for nonlinear function optimization with graphics hardware acceleration. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, p.1803-1808. San Antonio (USA): IEEE.

Appendix: FW-ACO pseudo-code

```
                                {* Step 1 - GEOMETRY INITIALIZATION *}
1   Set Blocks                          # number of blocks
2   Set aisles                          # number of picking aisles
3   Set Picking/Isle Side Ratio         # number of storage locations per aisle side
4   Set k_x                             # Distance between two adjacent picking positions
                                        along the x-axis
5   Set k_y                             # Distance between two adjacent picking positions
                                        along the y-axis
                                  {* Step 2 - GEOMETRY COMPUTATION *}
6   Calculate Adjacency_Matrix
7   Calculate Adjacency_Graph
                              {* Step 3 - FLOYD - WARSHALL COMPUTATION *}
8   Calculate Floyd-Warshall_Distances_Matrix, based on Adjacency_Graph
9   Calculate Floyd-Warshall_Predecessors_Matrix, based on Adjacency_Graph
                              {* Step 4 - ACO PARAMETER INITIALIZATION *}
10  Set α                               # Trail pheromone importance parameter
11  Set β                               # Visibility importance parameter
12  Set ρ                               # Evaporation coefficient
13  Set Q                               # Pheromone update constant
14  Set Loops_TOT                       # Maximum number of iterations allowed
15  Set Start                           # Start node
16  Set n                               # List of elements in the picking list
                          {* Step 5 - MATRIX AND VARIABLE INITIALIZATION *}
17  Randomize
18  Calculate ACO_Distances_Matrix, based on Floyd-Warshall_Distances_Matrix
19  Calculate Pheromone_matrix  τ_i,j = τ ∀(i,j)
20  Set Best_Path = Random(Path)
21  Set Best_Length = Length(Best_Path)
22  Set t = 0
23  Set n' = Numbers_of_picking_points
                                    {* Step 6 - ACO EXECUTION *}
24  While t < Loops_TOT
25          Set New_Path
26          For i' = 0 To n' Do:
27                  For City Not In New_Path:
28                  Calculate           probability based on Pheromone_matrix and
                                        ACO_Distances_Matrix
29                  Choose city based on probabilities
30                  Add city to New_Path
31          Next i'
32          Calculate New_Path_Length based on ACO_Distances_Matrix
33          If New_Path_Length < Best_Length Then:
34                  Set Best_Path := New_Path
35                  Set Best_Length := New_Path_Length
36                  Update Pheromone_Matrix
37          t = t + 1
38  Translate Path into Tour based on Floyd-Warshall_Predecessors_Matrix
```