Taylor & Francis
Taylor & Francis Group

International Journal of Logistics Research and Applications

A Leading Journal of Supply Chain Management

ISSN: 1367-5567 (Print) 1469-848X (Online) Journal homepage: https://www.tandfonline.com/loi/cjol20

Optimisation of storage allocation in order picking operations through a genetic algorithm

Eleonora Bottani , Margherita Cecconi , Giuseppe Vignali & Roberto Montanari

To cite this article: Eleonora Bottani , Margherita Cecconi , Giuseppe Vignali & Roberto Montanari (2012) Optimisation of storage allocation in order picking operations through a genetic algorithm, International Journal of Logistics Research and Applications, 15:2, 127-146, DOI: 10.1080/13675567.2012.694860

To link to this article: https://doi.org/10.1080/13675567.2012.694860

Published online: 19 Jun 2012.

Submit your article to this journal

Article views: 819

View related articles

Citing articles: 30 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=cjol20

International Journal of Logistics: Research and Applications
Vol. 15, No. 2, April 2012, 127–146

Taylor & Francis
Taylor & Francis Group

Optimisation of storage allocation in order picking operations through a genetic algorithm

Eleonora Bottani*, Margherita Cecconi, Giuseppe Vignali and Roberto Montanari

Department of Industrial Engineering, University of Parma, viale G.P. Usberti 181/A, Parma 43100, Italy

(*Received 10 June 2011; final version received 15 May 2012*)

This paper explores the use of a genetic algorithm (GA) to optimise item allocation in a warehouse, with the ultimate purpose of reducing the travel time of pickers, thus streamlining order picking operations. The GA is described along with a numerical example, reflecting a fast moving consumer goods warehouse, where items are assumed to be allocated according to a class-based storage system. Starting from that configuration, and taking into account the set of orders to be fulfilled, the GA identifies a new item allocation, which significantly decreases the travel distance (by approximately 20%). This involves a corresponding decrease in the cost of picking operations, and allows the warehouse to respond quicker to the requests of customers. The GA and its numerical implementation are supported by a general purpose software, such as Microsoft Excel™, programmed under visual basic for applications; the resulting tool is thus easy to use in real scenarios.

Keywords: order picking; warehouse optimisation; item allocation; genetic algorithm

1. Introduction

Warehouses are important parts in the supply chain; here, products are temporarily stored and subsequently retrieved from storage locations to fulfil customer orders. The order picking activity, in particular, is the process of selecting and assembling items from a warehouse, for order fulfilment and shipment in response to a customer's request. Orders from customers consist of order lines, each line representing a unique product, or a stock-keeping unit (SKU), which is requested in a defined quantity. Order picking is one of the most time-consuming activities of a warehouse, and is estimated to contribute more than 55% of the total cost of warehouse operations (Coyle *et al.* 1996, Tompkins *et al*. 1996). This is why scientists, as well as logistics managers, consider order picking as one of the most promising areas for productivity improvements (de Koster *et al.* 2007).

The organisation of order picking immediately impacts on the warehouse efficiency, and also affects the performance of the whole supply chain. The faster items are picked from the warehouse, the shorter the time spent in order fulfilment will be; hence, the lead time required for delivering the product to the final customer decreases correspondingly (de Koster *et al.* 2007).

*Corresponding author. Email: eleonora.bottani@unipr.it

ISSN 1367-5567 print/ISSN 1469-848X online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/13675567.2012.694860
http://www.tandfonline.com

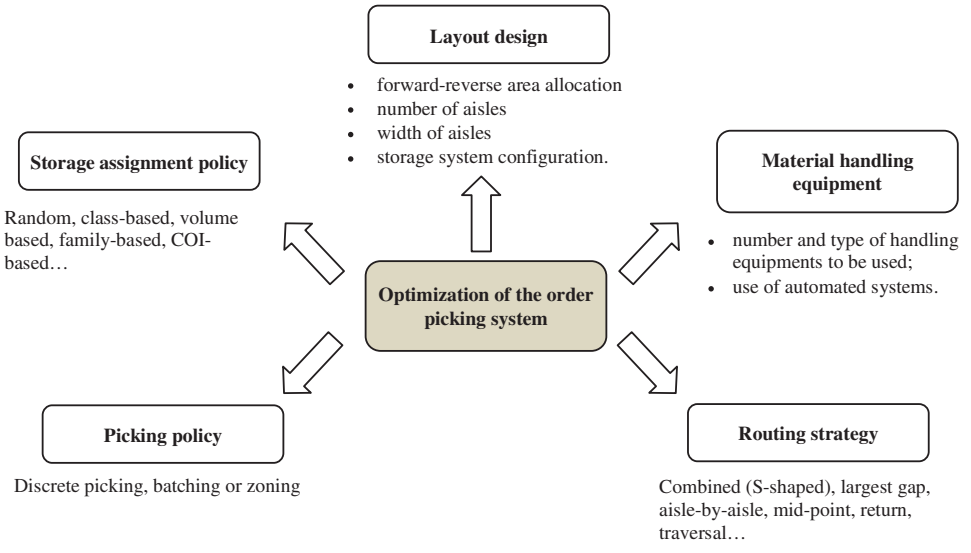

Figure 1. Factors affecting the optimisation of an order picking system.

Order picking involves several processes, which may include the scheduling and/or aggregation of customer orders, assigning available stock to orders, releasing orders to the floor and picking the articles from storage locations (de Koster *et al.* 2007). Accordingly, many authors (Roodbergen and De Koster 2001, de Koster *et al.* 2007, Parikh and Meller 2009) agree that a complex set of tasks contribute to the proper design of an order picking system; such tasks encompass designing the warehouse layout, selecting the storage assignment policy, the picking policy and the routing strategy, and defining the material handling equipment to be used in the warehouse (see Figure 1).

The design and optimisation of an order picking system is usually aimed at maximising the service level delivered to customers, i.e. at reducing the total time required to fulfil customer orders, by taking into account the constraints of available resources, such as workforce, material handling systems or budget (Roodbergen and De Koster 2001, de Koster *et al.* 2007). The total time required to complete picking activities includes several elements, such as the time for reaching the storage location, for picking the item and for further operations (Roodbergen and De Koster 2001). However, in manual warehouses operating with a 'pickers-to-parts' strategy, the time required to reach the picking location (commonly known as the 'travel time') is the most relevant contribution to the total picking time (Tompkins *et al.* 1996). In turn, the travel time is an increasing function of the travel distance, and thus minimising the travel distance is the main lever for optimising the total picking time (Jarvis and McDowell 1991, Hall 1993, Petersen 1999, Roodbergen and De Koster 2001, Petersen and Aase 2004).

Clearly, all factors shown in Figure 1 contribute to the minimisation of the travel distance, both *per se* and due to interdependencies between them (de Koster *et al.* 2007). But it is also evident that such factors cannot be all dealt with and optimised at the same time; this is why researchers often focus on a specific topic to be considered and optimised (de Koster *et al.* 2007). Nonetheless, three main elements emerge as particularly relevant to minimise the travel time, namely warehouse layout, picking policy and storage assignment policy. In this paper, we focus on this latter topic, and in particular on the issue of optimising the allocation of items in a warehouse so as to reduce the travel distance covered by pickers. As we focus on the optimisation of the storage assignment policy, we suppose that the remaining factors proposed in Figure 1 have been defined, and we do not directly deal with them in our analysis. The optimisation of the storage assignment policy is

performed by a heuristic model, based on a genetic algorithm (GA). The model is developed and implemented in Microsoft Excel™, exploiting several macros programmed under visual basic for applications (VBA); the model is then tested in a numerical case study, representative of a fast moving consumer goods (FMCG) warehouse.

The remainder of the paper is organised as follows. Section 2 provides some fundamentals of GA and reviews their application to the context of logistics and warehouse design. The development of the model is proposed in Section 3, along with the details of the numerical case study and an exhaustive analysis of the results. Managerial implications, limitations and future research directions are presented in Section 4.

2. Background: GAs and applications

GAs are stochastic search techniques aimed at reproducing the mechanism of natural selection and natural genetics (Holland 1975). A GA tries to optimise (i.e. maximise or minimise) an objective function, moving from a current collection of entities, called 'chromosomes', to a new one, by means of natural operations, such as selection, crossover, mutation and inversion. The following nomenclature is commonly used to describe GAs (Goldberg 1989, Beasley *et al.* 1993a, 1993b):

- Population: a collection of chromosomes;
- Chromosome: a string of genes, representing a particular solution to the problem under investigation;
- Allele: a specified set of alternatives for each gene. The set of alternatives for genes is problem-specific, although the original GAs coded such alternatives using binary representation (i.e. 0 *vs.* 1);
- Locus: the position of a gene in the chromosome.

Typical operations performed when running a GA include: (1) chromosome selection according to the fitness function; (2) crossover operations to reproduce the new generation of chromosomes; (3) inversion; and (4) random mutation of chromosomes. The starting population of the GA is an initial set of random solutions for the problem under investigation. The chromosomes of the population evolve through successive iterations, called generations, by means of an initial evaluation and different subsequent operations. As a first step, the chromosomes are selected to identify those which will be allowed to reproduce; the choice is made on the basis of a measure of their fit to a defined objective function, and in particular the chromosomes with the best fit to the function are selected for reproduction. The crossover is an operation between two chromosomes, and consists of exchanging some parts of them, following the natural procedure of recombination between two single-chromosomes. The mutation is a random operation, which arbitrarily changes the allele value of some locations in the chromosomes. Finally, by inversion we mean an operation which reverses the order of a contiguous section of the chromosome. By evolving through different generations through the operations mentioned above, the algorithm converges to the best solution for the problem examined.

GAs have gained significant popularity in real-world engineering and optimisation problems; examples of recent applications of GAs in the field of industrial engineering include the optimisation of reorder policy (Pasandideh and Niaki 2008), distribution networks (Jawahar and Balaji 2009), pallet loading operations (Lau *et al.* 2009) and assembly operations (Galantucci *et al.* 2004). The application of a GA to such engineering problems is interesting because of their ease of use and their capability of finding an optimal solution to the problem considered, even in multi-optimisation problems. Moreover, for many engineering problems, there is an increasing

need for intelligent and automated data processing tools (Choudhary *et al.* 2009). In the context of warehousing and logistics, GA-based optimisation models have been proposed by Zhang and Lai (2006), Hsu *et al.* (2005), Yao and Chu (2008) and Silva *et al.* (2008); at the same time, however, there are no applications of GAs for the optimisation of item allocation policies. Zhang and Lai (2006) focus on the solution of the multiple-level warehouse layout problem, i.e. a real scenario where the warehouse has multiple-level storage space, and an elevator transports items between the ground and the other storage levels. To this extent, the authors propose a class of heuristic models, combining GAs and path relinking methods, and demonstrate the achievable performance through extensive numerical simulations. Silva *et al.* (2008) examine the problem of optimising the logistics processes of a warehouse, and, in particular, the order fulfilment activity. They consider the simulation of a simplified scenario, taken from a real case study company, and propose the combined application of GA and ant colony optimisation to improve the order fulfilment process. Yao and Chu (2008) investigate the optimisation of replenishment cycles of multiple products in a warehouse, with the aim of minimising the total storage space required. They apply GA to solve an objective function describing the warehouse space, and carry out extensive experiments to evaluate the improvement of the proposed method against published research. Hsu *et al.* (2005) exploit GA to define a procedure for automatically grouping customer orders into batches; the approach can be easily adapted to different kinds of batch structure and of warehouse layout. The authors conclude that GAs provide interesting results to the order batching problems, for which an exact solution is extremely difficult to obtain, due to its dependence on the configuration of formed batches and the layout of the warehouse.

3. Development of a GA to optimise storage allocation

3.1. *The scenario under investigation*

To develop and apply the GA, we first need to define a reference scenario. In this study, we refer to a FMCG warehouse, whose characteristics and processes were deduced from previous studies in the field of FMCG (i.e. Oke and Long 2007, Bottani and Rizzi 2008, Dallari *et al.* 2009, Pourakbar *et al.* 2009, Bottani *et al.* 2010). The warehouse examined is thus hypothetical; nonetheless, grounding on existing studies, we have tried to define a scenario that can be considered representative of real industrial picking processes, so that the problem under investigation appears meaningful. Limitations related to the scenario hypothesised are discussed in the conclusions section.

The storage capacity of the warehouse is set at 1824 SKUs, and the warehouse is organised into six levels of 304 stock locations each. The first (ground) level contains the stock used for picking operations and is thus the focus of the present study. The warehouse handles 304 different products, so that there is exactly one picking location per product handled. Picking operations allow fulfilment of approximately 50 customer orders per day, which is a typical value for a FMCG warehouse of small size (Bottani and Rizzi 2008). We hypothesise a low-level, picker-to-part system, which represents the very large majority of picking systems in manual FMCG warehouses (Dallari *et al.* 2009). Moreover, pickers operate according to an order picking strategy; hence, they start from the input/output (I/O) with a picking list, including all items required in a customer's order, visit the corresponding picking locations, and return to the I/O when the order is completed.

The warehouse layout includes eight parallel pick aisles of 1.9 m width with picking locations on either side of each aisle; moreover, two cross aisles of 2.0 m width can be used by pickers to change the pick aisle. The I/O is located at the bottom left side of the warehouse. A scheme of the warehouse plan, together with the corresponding measurements, is shown in Figure 2.

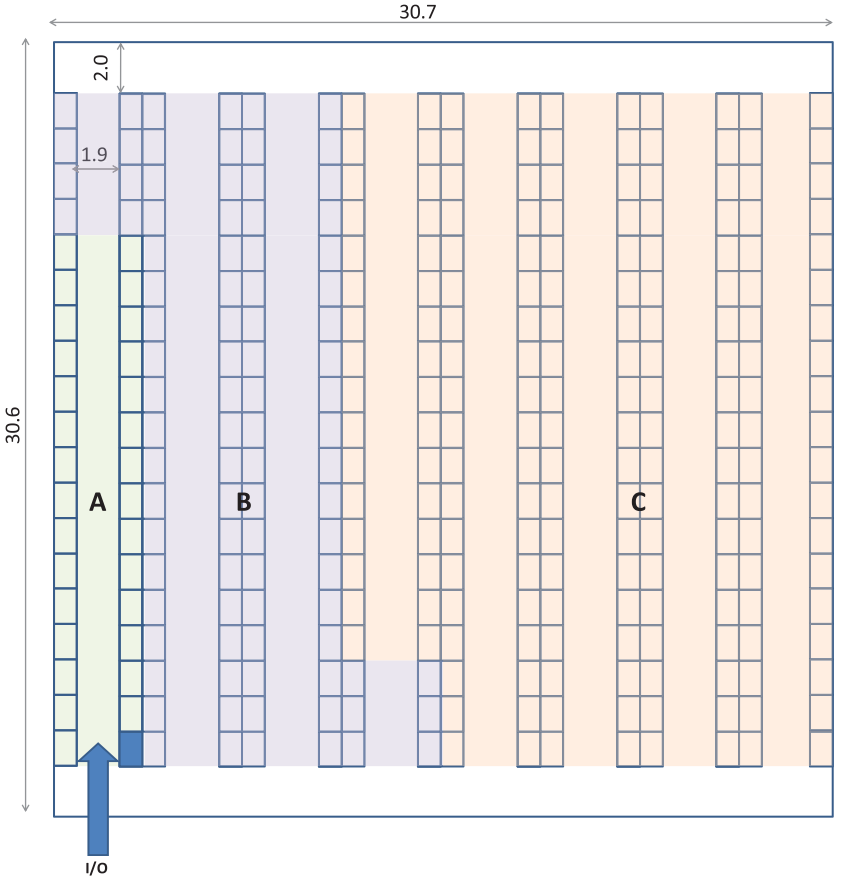

Figure 2. Plan of the warehouse under investigation (measurement in metre).

Table 1. Details of the A–C classes.

Class	Percentage of products included	Number of products products included	Breakable products	Picking probability
A	9.9%	30 (from *1* to *30*)	4 (from *27* to *30*)	70%
B	29.6%	90 (from *31* to *120*)	9 (from *112* to *120*)	20%
C	60.5%	184 (from *121* to *304*)	15 (from *290* to *304*)	10%
Total	100%	304	24	100%

At the beginning of the analysis, products are assumed to be allocated in the warehouse according to a class-based storage policy, which is frequently used in the FMCG context (Bottani and Rizzi 2008). It is thus assumed that the 304 products handled in the warehouse can be grouped into three classes, characterised by different probabilities of being requested in a customer order and including a different percentage of items. Specifically, class A includes few products (9.9%), with high picking probability, whereas class C includes numerous products (60.5%), which feature significantly less frequently in customer orders. We assume the characteristics of the product classes as per Table 1, where the products handled in the warehouse have been indicated as $1, 2, \ldots, 304$. Moreover, each product class also includes breakable products, which must be picked last in the route to avoid breakages.

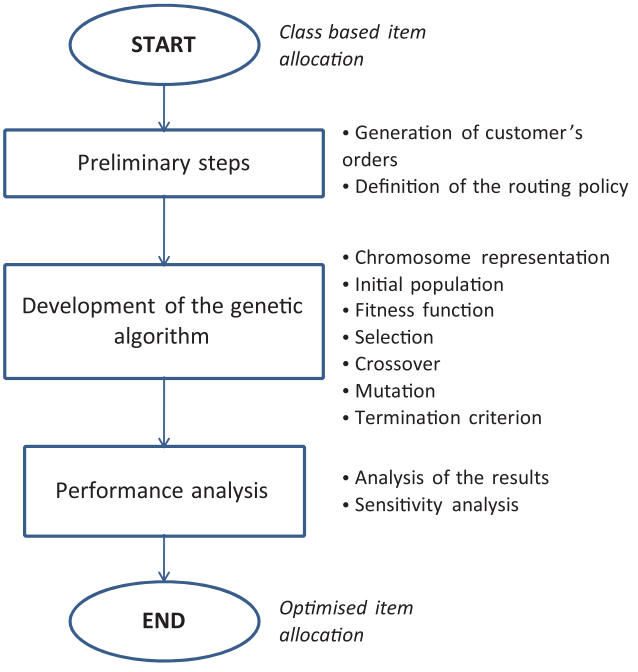

Figure 3. The methodological approach.

The data proposed in Table 1 have been used to generate orders; hence, for instance, products of class A will appear in 70% of the orders from the final customers.

Given the scenario described, the approach followed for the development and implementation of the GA for order picking optimisation consists of three steps and several sub-steps, schematically shown in Figure 3. The first phase of the analysis deals with preliminary steps, namely the generation of a collection of customers' orders to be fulfilled, and the definition of the routing strategy, which is kept unchanged for the remaining steps of the study. Then, we develop the GA for the optimisation of item allocation. As a final phase, we run the GA to obtain a new allocation of items in the warehouse and analyse the results obtained to assess the corresponding performance.

The steps of the study are detailed in the following sub-sections. The entire analysis has been supported by Microsoft Excel™ spreadsheets and by VBA programming.

3.2. *Preliminary steps*

At the beginning of the analysis, some preliminary steps are required. The starting point is the generation of a collection of customer orders to be fulfilled by the warehouse pickers. To generate consistent orders, we embodied the characteristics of the product classes provided in Table 1 in a spreadsheet, which was programmed with an appropriate macro code under VBA (see Figure A.1 for the flowchart of the macro). By running the spreadsheet macro, a collection of 48 customer orders was generated, corresponding to 500 order lines overall and including all 304 products handled in the warehouse. For purposes of illustration, a small extract of the flow of orders generated is shown in Table 2. It can be seen from Table 2 that the number of units of each individual item picked is not considered in the generation of the orders, since its relevance to the optimisation of the travel distance is limited. In this respect, we have implicitly assumed that the

Table 2. Examples of orders generated.

Order number	1	2	...
Number of products included	12	8	...
List of products included	17; 5; 26; 120; 113; 224; 1; 18; 104; 6; 11; 2	2; 7; 10; 1; 12; 15; 24; 16	...

total number of units picked during the preparation of the order is always lower than the capacity of the picker concerned.

On the basis of the orders generated, the optimal routing policy needs to be identified. In particular, the routing policy which minimises the travel distance of pickers, given the orders they have to fulfil, has to be determined. Given our assumption above, during the preparation of the order, the picker is always able to manage the entire load and does not need to invert the route during picking.

In this study, the travel distance was computed as follows. First, picking locations were numbered, and x and y coordinates were defined for each of them, according to the measurements of the warehouse previously shown in Figure 1. To define the coordinates, it is assumed that picking operations are performed when the picker is approximately at the centre of the corridor and arrives at the midpoint of the stock location to be picked from. For instance, to pick from the pallet located in the first picking location, the picker should traverse the cross aisle and move along the y-axis for half of the first location, to reach its midpoint; the resulting distance is $2 + 0.5 * 1.4 = 2.7$ m, which corresponds to the y-coordinate of the first SKU in Figure 4. As regards the x-coordinate, it is clear that the first picking location can be reached by moving along the x-axis for half the aisle width, which corresponds to a distance of $1.9/2 = 0.95$ m. The coordinates of the picking locations for the warehouse examined are illustrated in Figure 4.

We have mentioned that pickers follow an order picking strategy and thus they return to the I/O as soon as they have picked all items required in a customer's order. To minimise the time required to visit all the picking locations, and thus the travel distance, several routing policies have been evaluated through simulation; they are briefly described in the following list.

- *Traversal* (or *S-shaped*): the rationale behind the traversal routing is that any aisle containing at least one picking location to be visited should be traversed through its entire length, unless it is

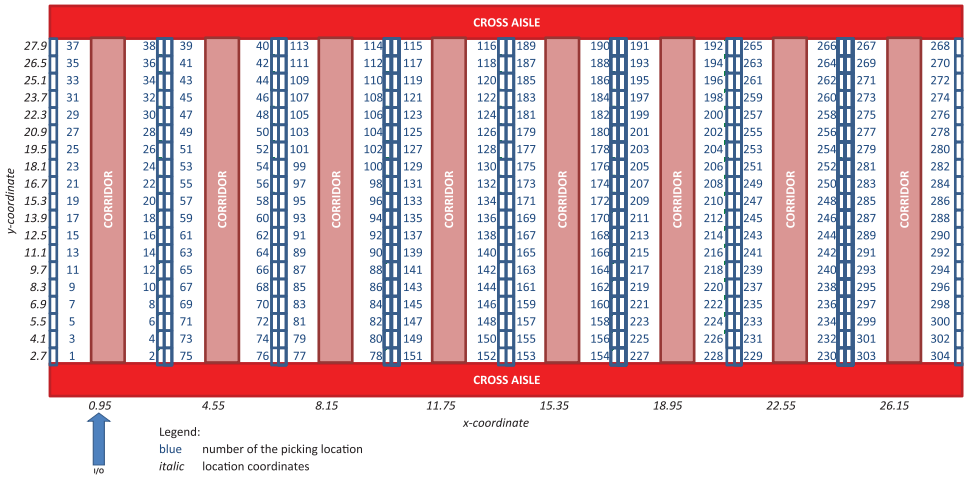

Figure 4. Coordinates of the picking locations (measurements in metre).

the last visited aisle. Conversely, aisles where nothing has to be picked are skipped (Roodbergen and De Koster 2001);
- *return*: according to this policy, a picker enters and leaves each aisle from the same end, by only visiting aisles with picking locations (de Koster *et al.* 2007);
- *combined*: this is a more complex routing method, according to which aisles with picks are either entirely traversed or entered and left at the same end. For each visited aisle, the choice whether to traverse the entire length can be made by using dynamic programming (Roodbergen and De Koster 2001).

For each routing policy, we developed an appropriate heuristic algorithm through VBA macros. The routing algorithms start, as input, from the orders to be fulfilled, determine the routing strategy (i.e. the sequence of items to be picked), and provide, as output, the length of the route created when the picker operates according to the routing policy defined. The flowcharts describing the algorithms are provided in Figures A.2–A.4. To show an example of application of the routing heuristics, Figure 5 provides a picture of the routes generated with the three policies, in the case of fulfilment of an order including items *1, 15, 46, 59, 60, 70, 74, 85, 86, 106, 111, 138, 140, 144, 151*.

By applying the heuristic algorithms to the entire collection of orders generated previously, we found that the combined routing policy usually performs better than the return and S-shaped ones, as it allows generation of routes with shorter distances. This policy was thus selected for implementation and kept unchanged in the remaining steps of the analysis. The total distance covered by pickers to fulfil all customer orders under the combined routing policy was estimated to be 6508.47 m.

3.3. *Development of the GA*

The next step of the analysis reflects the optimisation of the item allocation through the design and implementation of a GA, and it is thus the main part of our study. To apply GAs to the problem of item allocation, we have to solve the issues set out below.

(i) *Chromosome representation*. A chromosome is a viable solution of the problem under investigation, and thus it reflects a possible allocation of items in the warehouse locations. Each gene of the chromosome indicates the location of a particular item, while the allele indicates the item stored in the location. We recall that the items are simply indicated by a sequence of numbers, and thus the allele contains the number which identifies the product stored (from *1* to *304*, according to the description in Section 3.1). Since a product should be assigned only to one location, and all products should be located, genes of a chromosome should differ from one another; hence, chromosomes are possible permutations of integer numbers from *1* to *304*.

(ii) *Initial population*. The initial population is a group of possible solutions to the problem of items allocation for the warehouse investigated. When applying GAs, the size of the initial population should be defined carefully: usually, large populations are time-consuming as they increase the number of evaluations per generation, and the number of generations as well. Conversely, small populations have been proved to lead to premature convergence, without reaching an optimal result, as the diversity of the chromosomes generated decreases quickly (Michalewicz 1999). In this study, an intermediate initial population size of 20 chromosomes was chosen, as a good compromise between computational time and identification of the optimal solution. Frequently, when GAs are used for the optimisation of a function, the initial population is generated randomly. Our situation, however, is slightly different, as we

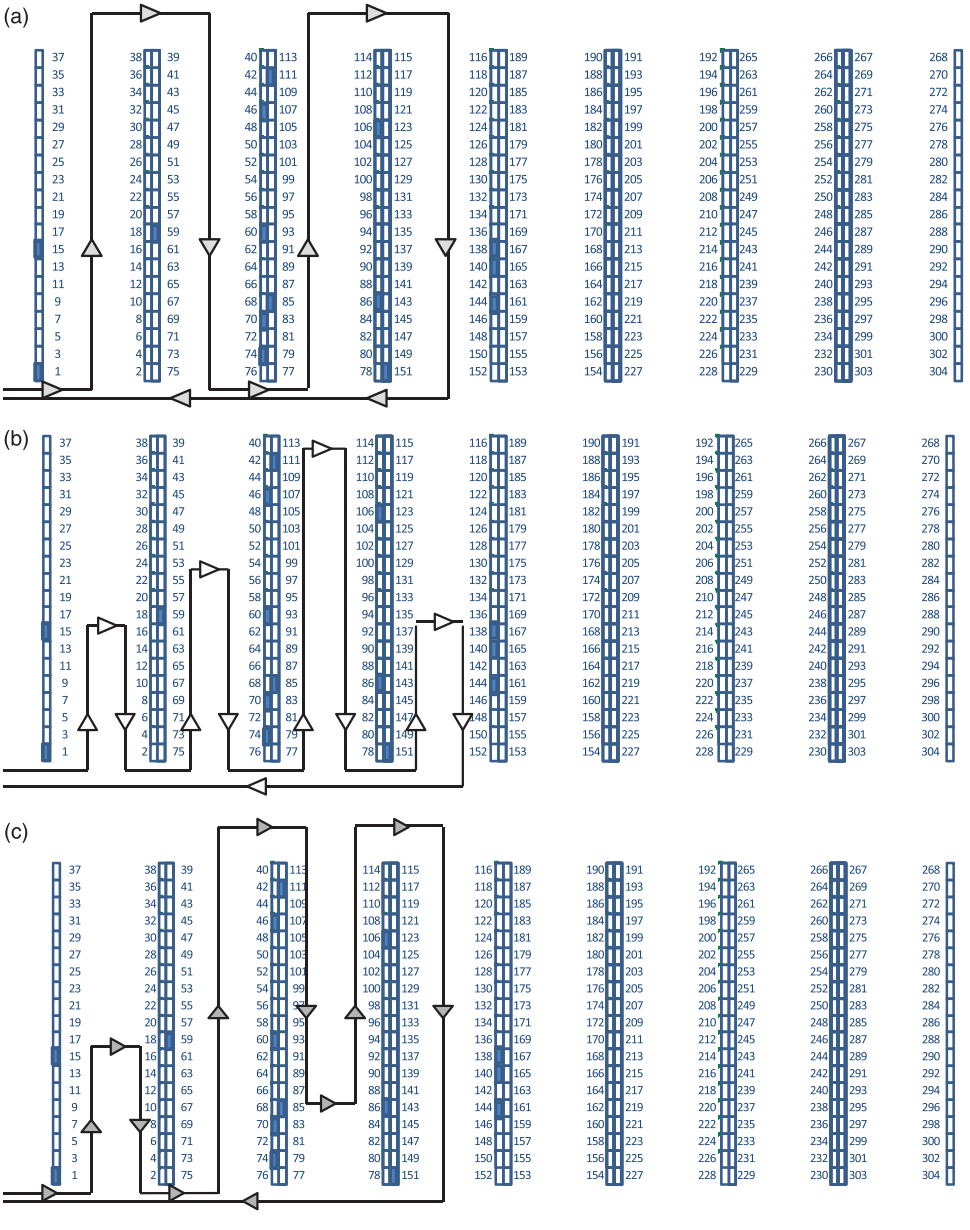

Figure 5. Examples of routes with traversal (a), return (b) and combined (c) policies.

start from an initial solution (i.e. the class-based allocation of items in the warehouse) which we want to optimise. Consequently, the initial population was set as follows:

(I) chromosome 1 was defined manually, and reflects the class-based allocation of items described in Section 3.1 and shown in Figure 2. In particular, in such allocation, we have item 1 stored in location 1, item 2 stored in location 2, etc.;

(II) chromosome 2 was defined manually, and represents a solution where the breakable items of all product classes have been stored close to the I/O, while the remaining items follow the class-based allocation previously described. The rationale behind this

allocation is that breakable products should be picked last, that is, when the picker returns to the I/O;

(III) chromosomes 3–20 have been generated randomly according to a VBA macro. The flowchart describing the functioning of the macro which generates chromosomes 3–20 is provided in Figure A.5.

(iii) *Fitness function*. The fitness function measures the distance covered by pickers to fulfil the overall number of orders. We recall that pickers follow the combined routing policy, and that the breakable items should be picked last in the route. To compute the fitness function of chromosomes, we use the spreadsheet which reproduces the combined routing policy, previously described and shown in Figure A.4. We have already mentioned that the fitness function for chromosome 1 accounts for 6508.47 m, which also reflects the optimal value of the fitness function at the beginning of the analysis.

(iv) *Selection*. Once the fitness function has been evaluated, the selection mechanism simulates the evolutionary process, i.e. the best chromosomes get more copies, the average ones stay the same, and the worst ones die off. We thus gathered the results obtained in (iii) and defined a hybrid procedure to identify the chromosomes to be selected, on the basis of both the probability of selection and the ranking of the fitness functions. We first computed the probability of selection. Let F_k indicate the fitness function of the k-th chromosome ($k = 1, \ldots, 20$) and F_{tot} the overall sum of the fitness functions. According to a minimisation problem, where the best chromosome has the lowest value of the fitness function, the probability p_k that a chromosome is selected can be computed as:

$$p_k = \frac{F_{\text{tot}} - F_k}{\sum_{k=1}^{20} (F_{\text{tot}} - F_k)} \tag{1}$$

with $F_{\text{tot}} = \sum_{k=1}^{20} F_k$. The chromosomes (20 'mothers' and 20 'fathers') to be used for deriving the next generation are identified as follows. Ten chromosomes are chosen on the basis of the selection probability p_k. A further 10 chromosomes are selected randomly according to the roulette wheel mechanism. The selection procedure is supported by an appropriate macro code implemented in MS Excel™, which operates according to the flowchart proposed in Figure A.6.

(v) *Crossover*. The crossover mechanism allows the creation of the chromosomes of the next generation, starting from those selected. It should be recalled that, in the problem under investigation, chromosomes are possible permutations of integer numbers, and offspring should have the same characteristic; hence, the crossover mechanism should be defined to respect this constraint. We chose to generate the first two offspring by preserving the chromosomes with the highest ranking (*elitism*). The remaining 18 offspring are generated on the basis of an adapted uniform crossover, whose procedure has been properly set up to comply with the constraint that the resulting chromosome is a permutation of integer numbers from 1 to 304. More precisely, the crossover procedure is a partially matched crossover (Goldberg 1989), which operates according to the following procedure (cf. Figure A.7). A crossover mask, composed of a sequence of 0 and 1 is generated at the beginning of the crossover. The mask has 304 positions (indicated as $j = 1, \ldots, 304$). In the case, the crossover mask scores 1 in position j, an exchange of genes should be performed. In this case:

(i) the j-th gene of the mother will become the j-th gene of the new chromosome;

(ii) the j-th gene of the father is inserted in the new chromosome in the position indicated by the allele which occupies the j-th position of the mother.

To reduce the computational time, the crossover mask is designed such that the crossover can consider only the first 100 genes, which include the items most frequently handled that

are expected to have the highest impact on the total distance covered by pickers. Hence, the positions of the crossover mask from 101 to 304 are set at 0.

(vi) *Mutation*. In the problem under investigation, mutation is a random inversion of the order of a couple of items among two warehouse locations. As per the crossover, the first chromosome is preserved and does not undergo mutation; conversely, for the remaining chromosomes, mutation can be randomly applied to any couple of genes. Figure A.8 provides a scheme of the VBA macro which performs the mutation of genes in the chromosomes.

(vii) *Termination criterion*. The GA runs until it reaches a defined termination condition. In the case under investigation, we deal with the problem of minimising the travel distance, but the minimum value of the distance is not known *a priori*; this prevents the possibility of applying termination criteria based on the value reached by the fitness function. We thus set the two termination criteria as follows: (1) the GA should run for at least 100 iterations; and (2) the optimal solutions obtained should remain unchanged for at least 10 iterations.

3.4. *Performance analysis*

To complete each iteration, the GA requires approximately 5 min on a 2.4 GHz Pentium IV PC, with 2GB RAM memory, operating under the Windows XP professional platform; hence, to reach the termination criteria set above, the algorithm requires a minimum computational time of 8.3 h. We found that the algorithm reaches convergence after 174 iterations (approximately 14.5 h). Table 3 and Figure 6 show the trend of the optimal solution as a function of the number of iterations of the GA.

As mentioned previously, at the beginning of the analysis (iteration 0), the optimal solution is for 6508.47 m, the distance covered by pickers when items are allocated according to a class-based

Table 3. Optimal solution as a function of the number of iterations.

Iteration	Current optimal solution (m)	Improvement (with respect to iteration 0)
0	6508.47	0.0%
12	6278.38	3.5%
14	6004.94	7.7%
38	5730.86	11.9%
56	5662.71	13.0%
82	5541.7	14.9%
97	5453.08	16.2%
135	5277.9	18.9%
156	5209.66	19.9%
174	5206.88	20.0%

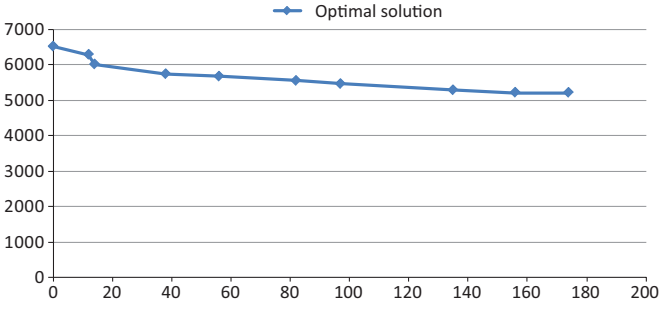

Figure 6. Optimal solution (travel distance in metre) as a function of the number of iterations.

storage system. From Figure 6, it can be appreciated that the proposed GA is effective in identifying a new allocation of items, which decreases the travel distance up to 20%. The GA is particularly effective in the earlier iterations, where the optimal solution improves by approximately 8% (from 6508.47 m to 6004.94 m) through 14 iterations. Conversely, from iteration 135–174, the optimal solution improves only by 1.1% (from 5277.9 m to 5206.88 m), while the computational time increases considerably. The reason for this outcome is that the initial iterations are significantly affected by the crossover operation, which combines the first 100 genes of the chromosome, thus providing good solutions in a relatively short time. As the algorithm approaches convergence, chromosomes become more similar, and thus the crossover is less effective in generating better solutions. At this stage, mutation still affects the optimal solution, but, since it only combines two genes, its effect on the fitness function is limited.

Figure 7 shows the optimal allocation of items obtained at step 174.

From Figure 7, one can see that, in the optimal allocation, most of the products belonging to class A are located close to the I/O, but they no longer follow a rigorous numerical order; rather, the GA recombines them on the basis of their 'real' recurrence in the customer's orders, with the goal of minimising the distance to be covered by pickers. Overall, on the basis of the considerations set out above, the GA modifies the location of about 130 items compared to the original class-based storage system.

A further result is provided in Figure 8, where product classes are examined separately to assess whether, after the implementation of the GA, classes have moved from their original position, previously shown in Figure 2. It can be seen from Figure 8 that, besides combining products inside a class, the GA also partially modifies the position of the product classes: more precisely, in the final configuration, 258 items (29 from class A, 67 from class B and 162 from class C) have been maintained in a location originally assigned to their class, while the remaining 46 are now stored in a position that was originally assigned to a different class.

37		48	199		76	113		114	151		38	189		190	227		163	265		266	303		304
35		44	93		282	224		112	149		139	187		188	225		226	100		264	301		302
96		79	258		80	109		127	147		148	185		87	223		143	141		262	299		300
31		295	84		98	32		108	145		170	182		184	221		165	259		129	297		298
29		30	67		68	105		106	195		222	126		183	219		220	134		273	107		150
27		88	65		103	66		104	111		74	179		180	217		250	255		116	157		294
25		26	34		45	155		253	198		193	177		235	215		245	102		254	291		57
23		9	61		62	99		263	137		296	175		42	213		214	161		252	289		290
21		22	218		52	41		234	135		136	261		174	211		101	249		59	287		228
19		20	292		63	95		240	231		257	171		178	209		210	86		248	285		286
17		18	94		72	73		55	131		50	169		237	207		208	216		246	283		284
11		16	53		54	91		92	181		130	167		168	205		229	243		244	281		142
13		14	51		186	89		256	110		128	200		166	203		204	241		212	279		280
15		12	197		132	60		232	172		260	288		164	201		69	239		144	277		278
24		10	117		152	85		247	123		124	251		162	47		58	146		238	275		276
7		8	64		82	83		36	121		122	159		160	49		274	173		236	71		138
5		6	43		202	81		33	119		120	293		158	242		196	233		70	271		272
3		4	97		28	46		56	153		75	125		156	140		194	230		176	269		270
1		2	39		40	77		78	115		90	118		154	191		192	206		133	267		268

Figure 7. Optimal item allocation obtained through GA (174 iterations).

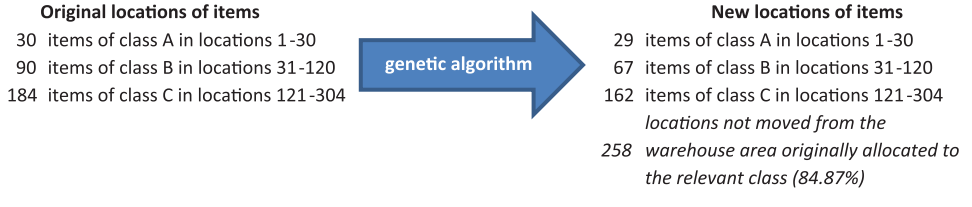

Figure 8. Analysis of product class location changes after GA implementation.

On the basis of the above findings, we can argue that there are two main reasons why product locations can be recombined by the GA, and these are explained below.

- When setting the warehouse parameters, we have defined the probability of being picked for a class of products. For instance, such probability accounts for 70% in the case of class A. Such a probability indicates that, on average, each product in this class is expected to be present in 0.7/30 of the picks. However, this is an average probability value which holds for the class of products as a whole, while (looking at the specific orders generated) each product can have a 'real' probability of being picked which could be slightly different from 0.7/30. In this case, the GA would probably recombine the product locations within the same class, to take into account the real occurrence of the products in the customer orders;
- Moreover, let us consider the case of two products (say product x and product y) which, on the basis of their picking probability, belong, respectively, to class A and class B. Such a classification indicates that product x is more frequently picked than product y; however, it does not take into account possible correlations among the products. In practical cases, it could happen that product y, when picked, is often coupled with product x (i.e. a positive correlation exists between those products). In this case, the GA would recombine the product locations, and in particular, it would move product y close to product x, although the original class is different, because this recombination would significantly reduce the distance covered by pickers.

Note that the locations of products in class A are little changed as a result of GA implementation, suggesting that products of this class should be preserved close to the I/O to minimise the travel distance. Modified locations for products from classes B and C are more numerous (23 out of 90 and 22 out of 184), indicating that higher benefits in the travel distance can be gained by intervening in the allocation of those products.

The above results demonstrate that the GA developed can be considered as a viable tool to solve the problem of an optimising item allocation in a warehouse, to minimise the travel distance covered by pickers. Although effective, however, a limitation of the approach can be found in the termination criteria; these latter require the GA to run for at least 100 iterations and the optimal solution to be stable for at least 10 iterations, but this does not ensure that the GA reaches the optimal solution of the problem under investigation. A sensitivity analysis for the optimal solution was thus performed by modifying the second termination condition, and, in particular, by hypothesising that the stability of the optimal solution should hold for at least 100 iterations. With this constraint, the GA runs for approximately 500 iterations before reaching the termination condition. In this case, we found that the optimal solution previously proposed in Figure 6 can be improved further, reaching 4864.9 m, which provides an additional 8% decrease in the travel distance (see Table 4). However, under this scenario, the main problem associated with the use of the GA is the computational time, which increases up to approximately 42 h. Moreover, the GA highlights an asymptotic trend, as the optimal solution does not significantly change during the last 150 iterations.

The solution obtained after 500 iterations modifies the location of 78 products, while 226 items maintain a location which is inside the area originally allocated to the class concerned (Figure 9); moreover, as per the previous case, most of the products whose location is changed belong to classes B and C.

The results obtained can be expressed in economic terms, as proposed in Table 5, according to the procedure described below. The optimal travel distance resulting when setting the first termination condition is 5206.88 m, which corresponds to approximately 1301.62 m saved with respect to the original configuration. This is a daily saving, as it is computed starting from the number of orders fulfilled each day by the pickers. Hypothesising that the picker moves at 1.1 m/s, which is a typical walking speed, a travel time of 1183.29 s is saved each day.

Table 4. Optimal solution as a function of the number of iterations, with a different termination condition.

Iteration	Current optimal solution (m)	Improvement (with respect to iteration 0)	Time (min)
0	6508.47	0.00%	0
12	6278.38	3.54%	60
14	6004.94	7.74%	70
38	5730.86	11.95%	190
56	5662.71	12.99%	280
82	5541.7	14.85%	410
97	5453.08	16.22%	485
135	5277.9	18.91%	675
156	5209.66	19.96%	780
174	5206.88	20.00%	870
204	5122.78	21.29%	1020
226	5022.59	22.83%	1130
244	4938.45	24.12%	1220
279	4882.59	24.98%	1395
319	4810.09	26.09%	1595
330	4810.09	26.09%	1650
350	4721.39	27.46%	1750
408	4684.9	28.02%	2040
430	4684.9	28.02%	2150
450	4684.9	28.02%	2250
480	4684.9	28.02%	2400
500	4684.9	28.02%	2500

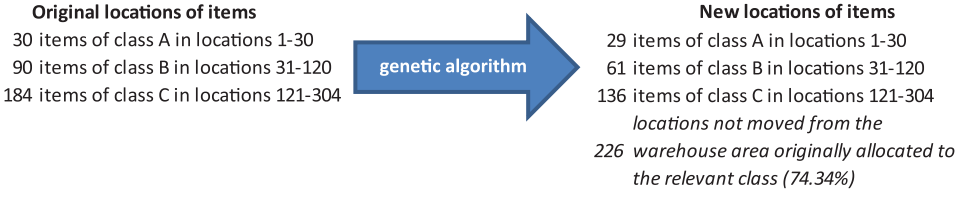

Figure 9. Analysis of product class location changes after GA implementation with a different termination condition.

Table 5. Financial analysis.

	Optimal solution	
	Termination criteria: (1) the GA should run for at least 100 iterations; (2) the optimal solutions obtained should remain unchanged for at least 10 iterations	Termination criteria: (1) the GA should run for at least 100 iterations; (2) the optimal solutions obtained should remain unchanged for at least 100 iterations
Travel distance (m/day)	5206.88	4864.9
Savings in travel distance with respect to the original configuration (m/day)	1301.62	1643.6
Savings in time (s/day)	1183.29	1494.18
Savings in time (h/year)	72.31	91.31
Financial savings (€/year)	2169.37	2739.33

Considering a 1-year operating horizon of the warehouse (e.g. 220 working days), this will lead to an overall decrease in travel time of 72.31 h, corresponding to a financial saving of approximately €2200/year. When setting the second termination condition, we obtain higher savings of some €2700/year.

4. Managerial implications, limitations and conclusions

Travel time accounts for a large proportion of the time required for order picking operations, and its reduction has obvious benefits for the cost of warehouse operations and on the time required to fulfil customer orders. In this research, we have designed a GA which, starting from the customer orders to be fulfilled, identifies a new allocation of items in the warehouse, in order to minimise the travel distance covered by pickers. The algorithm has been designed and tested exploiting a numerical example, whose features and figures reflect a typical FMCG warehouse. From the results obtained, we can conclude that, by setting adequate termination conditions, the GA is able to provide good performance in a reasonable computational time. In fact, the item allocation provided by the GA allows an improvement in the travel distance of pickers of some 20% compared to the traditional class-based allocation method. From a managerial point of view, the reduction of travel time has direct effects on personnel costs, which will decrease proportionally. Other benefits derive from the capability of the company to fulfil orders more quickly, reducing the throughput time and ultimately improving customer service.

The fact that the algorithm was tested in a numerical example could be seen as a current limitation of the study, since the assumptions made for the warehouse (e.g. size, number of picking locations, number of picks per day, etc.) describe a hypothetical and simplified scenario. Obviously, since the aim of our study is to demonstrate the application of a GA in the context of warehouse management, defining a reference scenario is mandatory. At the same time, although we have considered a hypothetical scenario, some of the assumptions made, such as for instance the presence of a picking location at ground level for each product handled by the warehouse or the chosen warehouse structure, are quite frequently observed in real contexts. Furthermore, we have tried to define a sufficiently large problem, so that the application and the results proposed are meaningful.

We also acknowledge that picking operations could be different for a warehouse with a different characteristics (e.g. structure, size or number of orders handled per day), and thus the methodology proposed would need to be modified to be applicable to any new context. Nonetheless, it is worth mentioning that the approach developed is general in nature, and has the potential to be adapted to many different contexts. Indeed, since the GA was programmed through general purpose software such as Microsoft Excel™, it is relatively simple to modify the algorithm to include a different number of items, or a different routing policy, or a different initial allocation of items in the warehouse. In future work, we will consider the implementation of our approach in an existing scenario, to get insights from a real case application.

Sensitivity analysis has shown that the GA has the potential to provide further reduction of the travel distance (up to 28%), although this result is achieved with a significant increase in computation time. An obvious trade-off exists between improvements achievable by means of the algorithm and computation time; hence, in real cases, it will be the company analyst's responsibility to identify and set appropriate termination conditions for the GA, to ensure adequate balance between these aspects. The sensitivity analysis also highlighted that a further shortcoming of the proposed approach is the large increase of the computation time with an increased number of iterations, again in trade-off with the quality of the results obtained. Of course, in real contexts, there is no need to frequently modify the allocation of items in a warehouse, and thus the proposed algorithm would be used only occasionally. Specifically, the warehouse manager could plan to run

the GA at fixed time intervals, partially mitigating the issue related to the high computation time. Nevertheless, further research can be directed towards finding solutions to reduce the computation time of the algorithm.

References

Beasley, D., Bull, D.R., and Martin, R.R., 1993a. An overview of genetic algorithms, part 1: fundamentals. *University Computing*, 15 (2), 58–69.

Beasley, D., Bull, D.R., and Martin, R.R., 1993b. An overview of genetic algorithms, part 2: research topics. *University Computing*, 15 (4), 170–181.

Bottani, E. and Rizzi, A., 2008. Economical assessment of the impact of RFID technology and EPC system on the Fast Moving Consumer Goods supply chain. *International Journal of Production Economics*, 112 (2), 548–569.

Bottani, E., Montanari, R., and Volpi, A., 2010. The impact of RFID and EPC Network on the bullwhip effect in the Italian FMCG supply chain. *International Journal of Production Economics*, 124 (2), 426–432.

Choudhary, A.K., Harding, J.A., and Tiwari, M.K., 2009. Data mining in manufacturing: a review based on the kind of knowledge. *Journal of Intelligent Manufacturing*, 20 (5), 501–521.

Coyle, J.J., Bardi, E.J., and Langley, C.J., 1996. *The management of business logistics*. Minneapolis/St Paul: West Publishing Company.

Dallari, F., Marchet, G., and Melacini, M., 2009. Design of order picking system. *International Journal of Advanced Manufacturing Technology*, 42 (1–2), 1–12.

Galantucci, L.M., Percoco, G., and Spina, R., 2004. Assembly and disassembly by using fuzzy logic and genetic algorithms. *International Journal of Advanced Robotic Systems*, 1 (2), 67–74.

Goldberg, D., 1989. *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Hall, R.W., 1993. Distance approximation for routing manual pickers in a warehouse. *IIE Transactions*, 25 (4), 76–87.

Holland, J., 1975. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Hsu, C.-M., Chen, K.-Y., and Chen, M.-C., 2005. Batching orders in warehouses by minimizing travel distance with genetic algorithms. *Computers in Industry*, 56 (2), 169–178.

Jarvis, J.M. and McDowell, E.D., 1991. Optimal product layout in an order picking warehouse. *IIE Transactions*, 23 (1), 93–102.

Jawahar, N. and Balaji, A.N., 2009. A genetic algorithm for the two-stage supply chain distribution problem associated with a fixed charge. *European Journal of Operational Research*, 194 (2), 496–537.

de Koster, R., Le-Duc, T., and Roodbergen, J.K., 2007. Design and control of warehouse order picking: a literature review. *European Journal of Operational Research*, 182 (2), 481–501.

Lau, H.C.W., *et al.*, 2009. An AI approach for optimizing multi-pallet loading operations. *Expert Systems with Applications*, 36 (3), 4296–4312.

Michalewicz, Z., 1999. *Genetic algorithms + data structures = evolution programs*. 3rd ed. London: Springer.

Oke, A. and Long, M., 2007. An analysis of the downstream logistics operations of a South African FMCG producer. *International Journal of Production Economics*, 108 (1–2), 176–182.

Parikh, P.J. and Meller, R.D., 2009. Estimating picker blocking in wide-aisle order picking systems. *IIE Transactions*, 41 (3), 232–246.

Pasandideh, S.H.R. and Niaki, S.T.A., 2008. A genetic algorithm approach to optimize a multi-products EPQ model with discrete delivery orders and constrained space. *Applied Mathematics and Computation*, 195 (2), 506–514.

Petersen, C.G., 1999. The impact of routing and storage policies on warehouse efficiency. *International Journal of Operations & Production Management*, 19 (10), 1053–1064.

Petersen, C.G. and Aase, G., 2004. A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, 92 (1), 11–19.

Pourakbar, M., Sleptchenko, A., and Dekker, R., 2009. The floating stock policy in fast moving consumer goods supply chains. *Transportation Research Part E: Logistics and Transportation Review*, 45 (1), 39–49.

Roodbergen, K.J. and De Koster, R., 2001. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39 (9), 1865–1883.

Silva, C.A., Sousa, J.M.C., and Runkler, T.A., 2008. Rescheduling and optimization of logistic processes using GA and ACO. *Engineering Applications of Artificial Intelligence*, 21 (3), 343–352.

Tompkins, J.A., *et al.*, 1996. *Facilities planning*. 2nd ed. New York: John Wiley & Sons.

Yao, M.-J. and Chu, W.-M., 2008. A genetic algorithm for determining optimal replenishment cycles to minimize maximum warehouse space requirements. *Omega*, 36 (4), 619–631.

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

Appendix. Flowcharts of the visual basic macros

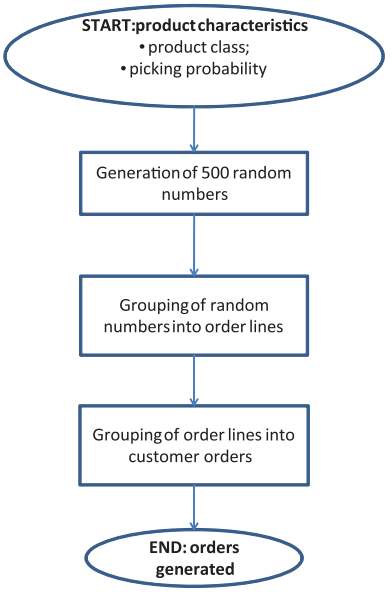

Figure A.1. Flowchart of the VBA macro for the generation of customer orders.

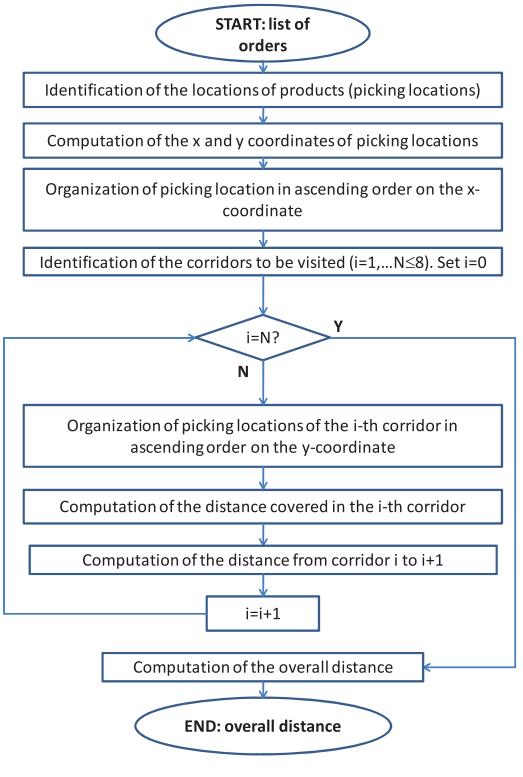

Figure A.2. Flowchart of the heuristic algorithm for traversal routing policy.

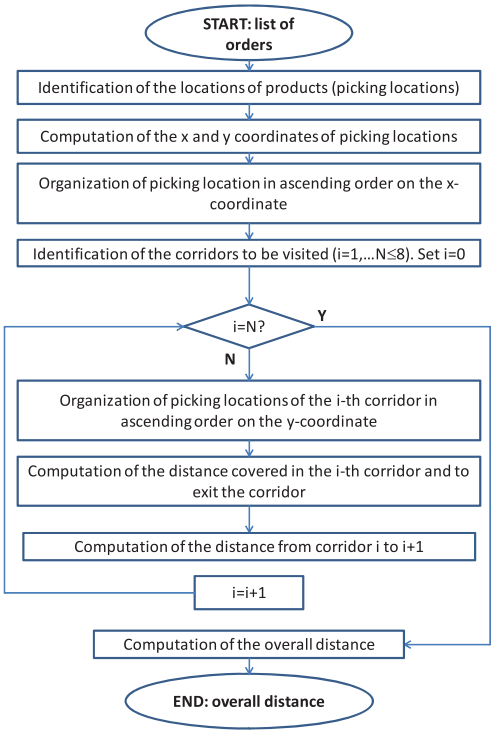

Figure A.3. Flowchart of the heuristic algorithm for return routing policy.

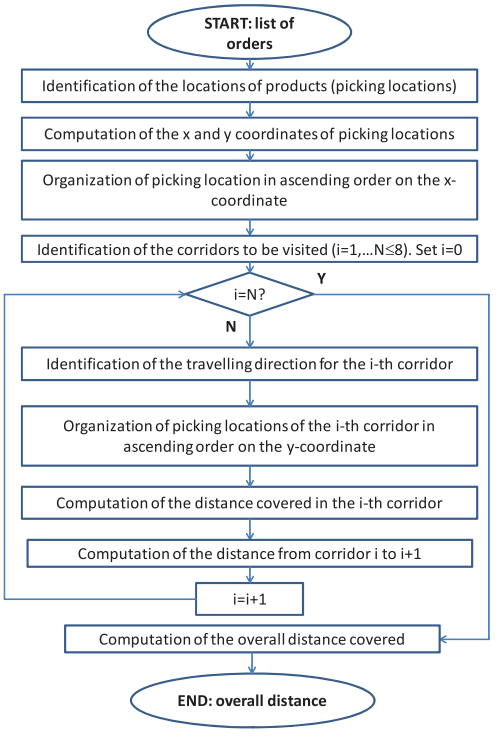

Figure A.4. Flowchart of the heuristic algorithm for combined routing policy.

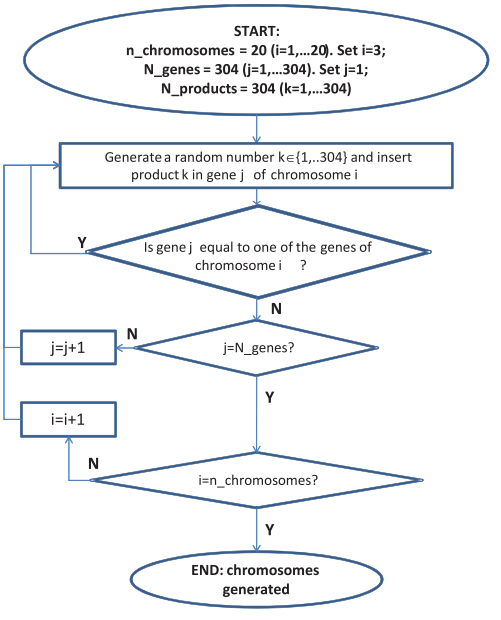

Figure A.5. Flowchart of the macro for the generation of the initial population of chromosomes.

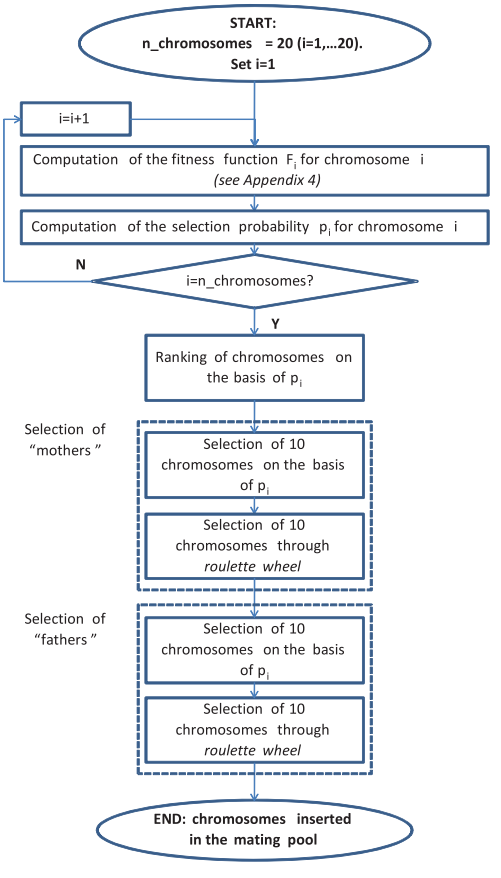

Figure A.6. Flowchart of the macro for the selection mechanism.

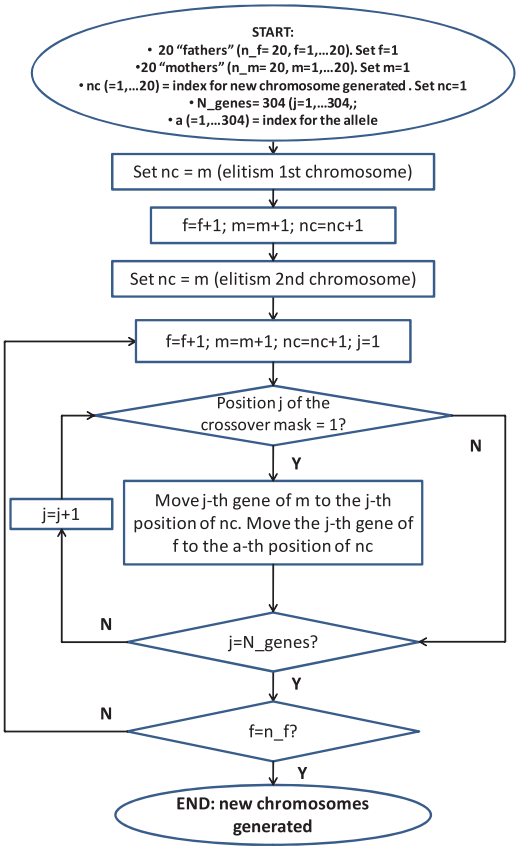

Figure A.7. Flowchart of the macro for the crossover mechanisms.

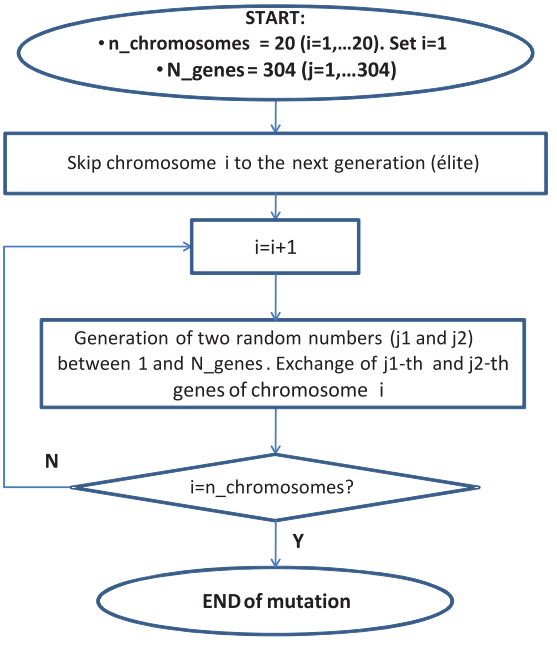

Figure A.8. Flowchart of the macro for the mutation mechanisms.