Adv. Manuf. (2014) 2:203–211
DOI 10.1007/s40436-014-0074-1

Tackling the storage problem through genetic algorithms

Lapo Chirici · Ke-Sheng Wang

Received: 20 January 2014 / Accepted: 14 April 2014 / Published online: 13 May 2014

Abstract The capability of a company to implement an automated warehouse in an optimized way might be nowadays a crucial leverage in order to gain competitive advantage to satisfy the demand. The order picking is a warehouse function that needs to deal with the retrieval of articles from their storage locations. Merging several single customer orders into one, a picking order can increase efficiency of warehouse operations. The aim of this paper is to define throughout the use of ad-hoc genetic algorithm (GA) how better a warehouse can be set up. The paper deals with order batching, which has a major effect on efficiency of warehouse operations to avoid wastes of resources in terms of processes and to control possibility of unexpected costs in advance.

Keywords Genetic algorithms (GA) · Warehouse management · Order batching · Optimization

1 Introduction

Mastering efficiently warehouse processes is of significant importance for the supply chains management, since underperformance results in an unsatisfactory customer service and/or high costs. Considering the different warehouse functions (receiving, storage, order picking and shipping), this paper will focus on order picking which makes up for the most cost intensive operations. Different studies estimate that between 50 % and 65 % of the total warehousing operating costs can be attributed to order picking [1, 2]. It arises since incoming articles are received and stored in unit loads while customers usually order small volumes of different articles. In manual-order-picking systems where order pickers collect the requested articles while walking or riding through the warehouse, three planning problems can be identified on the operative level: assignment of articles to storage locations in the warehouse (article location assignment), grouping of customer orders into picking orders (order batching) and the determination of routes for the order pickers (picker routing). Improved order batching reduces the total length of the picker tours significantly, which is necessary for picking all requested articles provided by a set of customer orders [3]. These results in a reduction of the total picking time (time needed to collect the requested articles of all customer orders) represent an increase in the efficiency of the picking operations and reduction of processing times and labor costs.

2 Order batching problem

On their tours through the warehouse, order pickers are guided by so-called pick lists. A pick list consists of a set of order lines, where each line denotes an article requested by a customer, the location of the article in the warehouse, and the respective quantity of the article to be picked. The articles of a pick list constitute a so-called picking order. With respect to the relationship between customer orders and picking orders, three cases can be distinguished: (i) a customer order has to be split into several picking orders, since it contains too many articles, which cannot be picked on a single tour; (ii) a customer order is identical to a

L. Chirici (✉)
Department of Computer Science, University of Pisa, Pisa, Italy
e-mail: lapochirici@gmail.com

K.-S. Wang
Department of Production and Quality Engineering, Norwegian University of Science and Technology, Trondheim, Norway

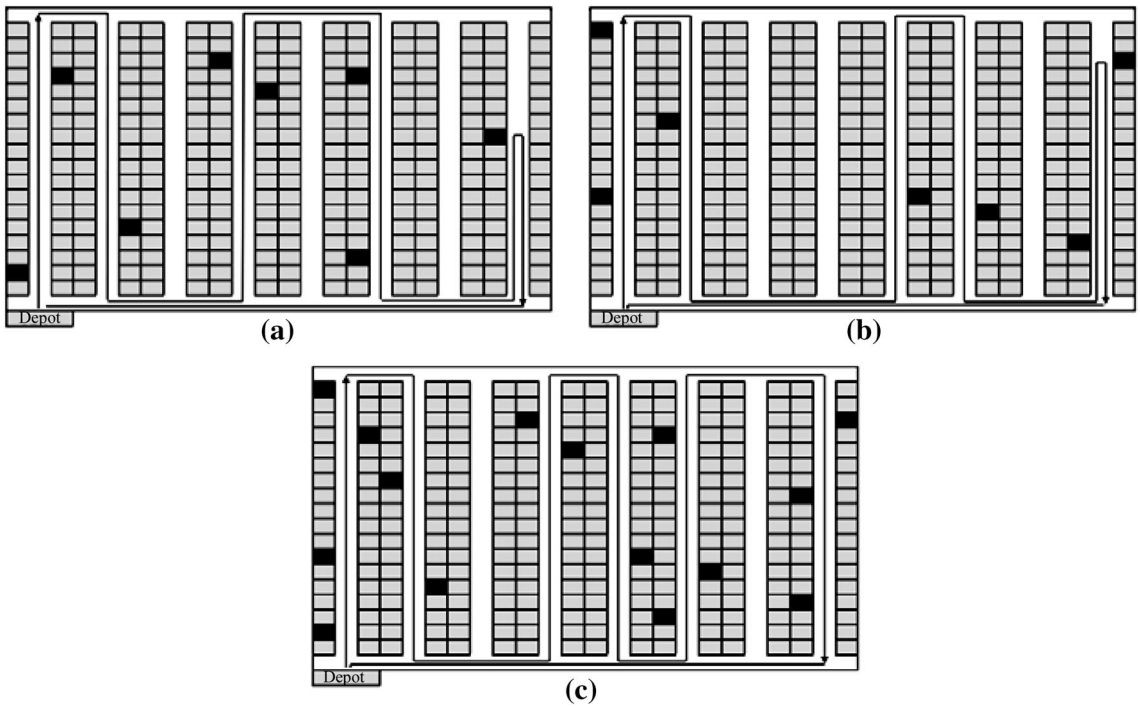

Fig. 1 Different picker tours resulting from pick-by-order and pick-by-batch **a** and **b** the picker routes if each customer orders picked separately (pick-by-order), **c** the corresponding route if the two customer orders joint in a single picking order (pick-by-batch)

picking order; and (iii) a picking order consists of several customer orders. This paper deals with the third case, the grouping (batching) of customer orders into picking orders, which allows for a significant reduction of the total length of the picker tours that is necessary for picking all requested articles provided by a set of customer orders. Figure 1 demonstrates exemplarily the obvious advantages of combining two customer orders into one picking order, in which the black rectangles symbolize the locations of articles to be picked and the black lines symbolize the tours that the order pickers are meant to take.

The warehouse has multiple-level storage spaces, divided in cells with one elevator to transport items from the ground to other levels. It is assumed that the elevator has enough capacity, which means that the vertical transportation operation is always available. Each level is divided into cells of the same dimension.

Different item types need to be stored in the multiple-level warehouse. Each item type has its own monthly demand and inventory volume, vertical unit transportation cost (i.e., the cost to move one unit of the item between the ground and other levels) and horizontal unit transportation cost (i.e., the cost to move one unit of the item 1 m in horizontal distance). Each item must be exactly assigned to one cell. A cell may store more than one item type. The objective is to minimize the total vertical and horizontal transportation cost.

Usually, the order lines on a pick list are already sorted in the sequence in which the articles are to be picked. This sequence determines the route that the order picker takes through the warehouse. According to these routes, each order picker starts at the depot, proceeds to the respective storage locations of the articles, and returns to the depot. Despite the fact that an optimal polynomial-time algorithm exists for the determination of optimal picker routes for some warehouse layouts (among which is the one depicted in Fig. 1), in practice the routes are usually determined by means of so-called routing strategies [5], which can be interpreted as heuristic solution approaches. The advantage of such routing strategies is that they produce schemes that can be memorized fast and can be followed easily, whereas the routing schemes stemming from the optimal algorithm are not always straightforward and sometimes even confusing for the order pickers.

2.1 Minimizing the order picking times

Order picking is considered to be the most labor-cost intensive function in a warehouse, due to the large amount of time-consuming manual operations [4]. Therefore, the minimization of the picking times is of superior interest to a distribution company. In order to minimize the total picking time, it is essential to indentify the different time components of the order picking process, which are setup

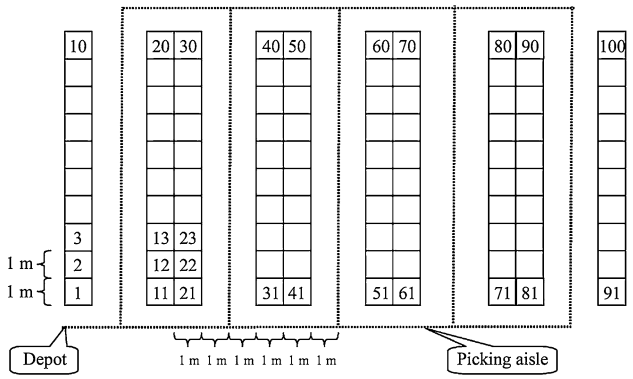

Fig. 2 Schematic layout of warehouse

times, travel times, search times and times for taking the articles from their locations. From these four components, the travel time is the most important one, since it consumes the largest proportion of the total picking time and offers a significant potential for improvement.

Given a fixed assignment of articles to storage locations and a given routing strategy, the order batching problem (OBP) can be defined in the following way as: how a set of customer orders can be grouped (batched) into picking orders (batches) such that the capacity limitation of the picking device is not violated and the total length of all necessary picking tours is minimized [5].

Gademann and van de Velde [6] introduced a straightforward 0–1 optimization model for the OBP whose columns consist of all feasible batches [7] (see Fig. 2).

2.2 Notation to implement the model

In order to present the model, we introduce the following notation:

(i) Index sets

J is the set of customer orders, where $J = \{1,2,\cdots, n\}$;
I is the set of all feasible batches.

(ii) Constants

a_{ij} is the binary entry. $a_{ij} = 1$, if order j ($j \in J$) is included in batch i ($i \in I$); or $a_{ij} = 0$, otherwise;
C is the capacity of the picking device;
C_j is the capacity utilization required by customer order j ($j \in J$);
d_i is the length of a picking tour in which all orders of batch i ($i \in I$) are collected.

(iii) Decision variables

x_i is the binary decision variable. $x_i = 1$, if batch i ($i \in I$) is chosen, or $x_i = 0$, otherwise.

Each element $i \in I$ satisfies the following condition:

$$\sum_{i \in J} C_j a_{ij} \leq C. \tag{1}$$

A batch is feasible, if it contains a set of customer orders that does not violate the available capacity of the picking device. The optimization model can then be formulated as follows:

$$\min \sum_{i \in I} d_i x_i, \tag{2}$$

subject to

$$\sum_{i \in I} a_{ij} x_i = 1, \quad \forall j \in J, \tag{3}$$

$$x_i \in \{0,1\}, \quad \forall j \in J. \tag{4}$$

The sets of constraints (3) and (4) ensure that a set of batches is chosen in such a way that each customer order is included in exactly one of these batches. Numerical experiments based on the above-given model formulation [5] reveal that only small instances can be solved to optimality if all columns (batches) are generated in advance. This can be explained by the fact that the number of possible batches and, consequently, the number of binary variables increases exponentially with the number of customer orders.

2.3 Considerations about the different approaches

Heuristics approaches to the OBP can be differentiated into two main categories: constructive heuristics and metaheuristics. The constructive heuristics can be further distinguished in priority rule-based algorithms, seed algorithms and savings algorithms [8].

In priority rule-based algorithms, customer orders are assigned to batches in the sequence of non-ascending priority values. The priority values can reflect any kind of ordering, e.g., one related to the size of the customer orders or one related to their arrival times. The latter corresponds to the first-come-first-served (FCFS) rule, probably the best known representative of this sub-category. The assignment of customer orders to batches can be done sequentially (next-fit rule) or simultaneously (first-fit and best-fit rule). Using the next-fit rule, a customer order is assigned to a batch, if there is sufficient capacity available in the batch. In case the capacity utilization of a customer order would violate the capacity of the picking device, the current batch will be closed and a new one will be opened for it. Using the first-fit-rule, all batches are numbered in the sequence according to which they have been opened and a customer order will be assigned to the batch with the smallest number, which provides enough capacity. When the best-fit-rule is applied, a customer order will be assigned to the batch with the least remaining capacity which is still sufficient for the inclusion of the customer order.

3 Introduction of genetic algorithm (GA)

3.1 Basics of GA

GAs were first introduced by Holland, who was inspired by the notion of natural and biological evolution. In GAs, the concept that mimics from population genetics and evolution theory is used to construct the optimization algorithms. They attempt to optimize the fitness of a population of elements through recombining and mutating their genes. To apply the genetic evolutionary concept to a real-world optimization problem, two issues must be addressed: encoding the potential solutions, and defining the fitness function (objective function) to be optimized.

A solution, namely a chromosome, is encoded as a string composed of several components (genes). The initial population of chromosomes is generated according to some principles or randomly selected. The algorithm performs an evaluation to measure the quality (fitness) of the potential solutions [9]. Optimization using GAs is achieved by selecting pairs of chromosomes with probabilities proportionate to their fitness, and matching them to create new offspring. In addition to matching (crossover), small mutations are induced in new offspring. The replacement of bad solutions with new solutions is based on some fixed strategies. The chromosomes evolve through successive iterations, called generations. The evaluation, optimization and replacement of solutions are repeated until the stopping criteria are satisfied.

Figure 3 depicts the general structure of a GA. The design of such an algorithm requires several structural decisions, namely concerning the representation of solutions, the initialization of a starting population, and the components of the evolutionary process.

3.2 Exploring GAs structure

The general procedure to develop a GAs can be described as follows, and Fig. 4 is the basic scheme of GA.

Step 1 Define a genetic representation of a feasible solution of the problem.

Step 2 Create an initial population $P(0) = x_1^0, x_2^0, \cdots, x_N^0$, and $t = 0$.

Step 3 Compute the average fitness $\dot{f}(t) = f(x_i)/N$. Assign each individual the normalized fitness value $f(x_i)/\dot{f}(t)$.

Step 4 Assign each x_i a probability $p(x_i, t)$ proportional to its normalized fitness. Using this distribution, select N individuals from $P(t)$. This gives the set of the selected parents.

Step 5 Pair all parents at random forming $N/2$ pairs. Apply crossover with a certain probability to each pair.

Step 6 Apply mutation with a certain probability to each offspring.

Step 7 Form a new population $P(t + 1)$ by using the surviving mechanism.

Step 8 Set $t = t + 1$ and return to Step 3.

There are three major advantages when applying GAs to optimization problems. Firstly, GAs do not have many mathematical requirements for the optimization problems and can handle any kind of objective functions and constraints defined in discrete, continuous, or mixed search spaces. Secondly, the ergodicity of evolution operators makes GAs very effective at performing global searches (in probability) and finding global optima. Lastly, GAs provide a great hybridizing flexibility with domain-dependent heuristics to enable the efficient implementation of a specific solution. GAs have been successfully applied to a wide array of difficult real-world problems.

3.3 Modelling the warehouse with GAs

A chromosome (string) is composed of genes which represent a candidate warehouses in the model (1 if the

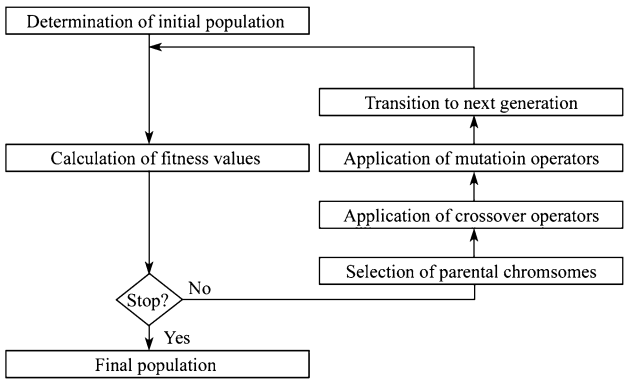

Fig. 3 General structure of a GA [10]

```
Input Data();
Population Init()
while not Finish () do
        for i:=1 to Npop do
            obj[i] := Objective Function(i);
        endfor
        Fitness Function();
        Selection();
        Crossover();
        Mutation();
endwhile
Output Data();
```

Note: *Npop* denotes the number of individuals in a population and obj[*i*] is the objective value of the *i*-th individual.

Fig. 4 Basic scheme of GA

```
function fitness(population,distances,installationCosts);
for i=1:number of individuals '(rows)
        fitness(i)= population (i,:)* installationCost; 'binary chromosomes
        for j=1: number of genes 'warehouses candidates (columns)
                if  population(i,j)==1
                  fitness(i)=fitness(i);
                else
                  fitness(i)=fitness(i)+min(nonzeros(population (i,:)'.*distances(:,j)));
                end
        end
        end
  end
```

Fig. 5 Cost of transporting goods from a warehouse to a customer

warehouse candidate is constructed; 0 otherwise). Each chromosome represents one potential solution. In the initial stage of the optimization, a number of chromosomes are arbitrarily created as initial population. It defines the size of solution pool. More chromosomes may increase the probability of finding optimal solution, but may induce a longer computation time [11].

An objective function is a measuring mechanism that is used to evaluate the status of a chromosome. This is a very important link to relate the GA and the system concerned [12]. Fitness function in the model is considered as the minimization of the total cost including installation cost of warehouses [9, 13]. The cost of transporting goods from a warehouse to a customer is shown in Fig. 5.

4 Design of the experiment

4.1 Parameters

The warehouse layout considered in the numerical experiments is a single-block warehouse with two cross aisles and a depot located in the lower left corner [14]. This layout type is identical to the one shown in Fig. 1 and is frequently used in numerical experiments. The warehouse consists of ten vertically orientated picking aisles. Each picking aisle has a capacity of 90 locations (45 on each side of the aisle), resulting in a total capacity of 900 storage location. Each storage location has a length of one length unit (LU). The necessary movement from the first and the last storage location of a picking aisle into the cross aisle amounts to one LU and the center-to-center distance between two aisles amounts to 5 LUs. The depot is located 1.5 LUs away from the first storage locations of the leftmost aisle. In the warehouse, each article is located at one storage location only and the distribution of the articles within the warehouse follows a class-based storage assignment policy [15]. For this purpose, the warehouse is grouped into 3 classes: articles with the highest demand frequency (A) are located in the leftmost aisle; articles with a medium frequency (B) are located in the 4 subsequent aisles and articles with a low demand frequency (C) are located in the 5 rightmost aisles. Furthermore, it is assumed that 52 % of the demand arises from articles in class A, 36 % from B-articles and the remaining 12 % from C-articles. Similar demand frequencies can be observed in real-world warehouses and have been considered in different values.

4.2 Assignation of classes

For the numerical experiments, the number of customer orders is varied in five steps from 20 to 60 customer orders, where each step has a size of ten customer orders.

The number of articles in a customer order is modeled as a random number which is uniformly distributed in $\{5, 6, \cdots, 25\}$. The capacity of the picking device is defined by the maximum number of articles which can be placed on it. The values considered here are 30, 45, 60 and 75. The underlying routing problem is solved by means of the S-shape Heuristic. Table 1 summarizes the characteristics of the problem classes. Combination of the different specifications results in 20 problem classes. For each problem class, 40 instances have been generated.

Table 1 Characteristics of classes

Characteristic	Specification
Total number of customer orders (n)	20, 30, 40, 50, 60
Capacity of the picking device (C) (No. of articles)	30, 45, 60, 75
Routing strategy	S-shape

Table 2 GA features

Feature	IGA	GGA
Encoding scheme	Real integers	Real integers
Orientation	Item-oriented	Group-oriented
Decoding	Direct representation	Indirect representation
Crossover	Two-point crossover	Two-point crossover
Mutation	Mutation	Migration and mutation

Table 3 Algorithm parameters

Feature	IGA	GGA
Size of the population	4 × number of customer orders	
Number of generations	80	80
Crossover probability	0.50	–
Mutation probability	0.10	0.30
Size of the TOP-part	–	0.10

4.3 Algorithm parameters

GAs can be differentiated according to the realization of some structural aspects of GAs. Here below we have considered two types of GAs: item-oriented algorithm (IGA) and the newly developed group-oriented genetic algorithm (GGA). The distinguishing related features are summarized in Table 2.

The size of the population will be arranged proportionally to the size of the problem classes. In our case the number of individuals in the population will be set to four times the number of customer orders [14].

All other parameters of the algorithms have been fixed across all problem classes. The number of generations has been set to 80 for both algorithms. For the determination of the necessary parameter settings, a series of pre-tests have been carried out in order to determine a suitable combination of the parameters.

For the IGA all combinations of the following parameters have been tested:

Table 4 Solution quality

n	*C*	C&W(*ii*)		IGA			GGA		
	(No. art)	Ø TTL (LU)	Ø No. bat	Ø TTL (LU)	Ø impr./%	Ø No. bat	Ø TTL (LU)	Ø impr./%	Ø No. bat
20	30	4,360	11.3	4,206	3.55	10.7	4,197	3.77	10.7
	45	2,855	7.1	2,745	3.74	6.9	2,700	5.32	6.8
	60	2,283	5.3	2,184	4.31	5.3	2,167	5.02	5.3
	75	1,878	4.3	1,769	5.76	4.2	1,756	6.47	4.2
30	30	6,426	16.5	6,225	3.09	15.8	6,195	3.59	15.7
	45	4,333	10.7	4,152	4.11	10.3	4,119	4.86	10.3
	60	3,288	7.8	3,149	4.17	7.7	3,125	4.90	7.6
	75	2,715	6.3	2,604	3.95	6.1	2,582	4.77	6.1
40	30	8,219	21.2	7,945	3.52	20.2	7,926	3.60	20.2
	45	5,504	13.8	5,308	3.69	13.3	5,273	4.17	13.1
	60	4,282	10.4	4,113	4.11	10.0	4,088	4.44	9.9
	75	3,479	8.2	3,377	3.00	8.1	3,347	3.75	8.0
50	30	10,509	27.0	10,186	3.24	25.9	10,124	3.69	25.7
	45	6,760	16.9	6,608	3.32	16.6	6,580	2.65	16.6
	60	5,254	12.8	5,154	1.92	12.7	5,120	2.50	12.6
	75	4,325	10.3	4,254	1.70	10.1	4,219	2.43	10.1
60	30	12,145	31.6	11,721	3.56	30.1	11,674	3.82	30.0
	45	7,930	20.1	7,803	1.58	19.8	7,755	2.13	19.7
	60	6,147	15.1	6,086	0.98	15.0	6,036	1.75	14.8
	75	5,029	12.1	4,997	0.68	12.0	4,964	1.27	11.9
Minimum			4.30		0.68	4.20		1.27	4.20
Average			13.44		3.15	13.04		3.75	12.97
Maximum			31.60		5.76	30.10		6.47	30.00

Note n number of customer orders; *C* capacity of the picking device in number of articles; *Ø* (TTL) average total tour length of picking tours in length units; *Ø* (impr) improvement obtained by the respective algorithm in comparison to the C&W(*ii*) solution in percent; *Ø* (No. Bat.) average number of batches. For each problem class the best obtained average total tour length is marked in blu

(i) crossover probability ∈ {0.3, 0.4, 0.5};
(ii) mutation probability ∈ {0.05, 0.1, 0.2}.

With respect to the GGA the following combinations are considered:

(i) TOP ∈ {0.10, 0.20, 0.30};
(ii) mutation probability ∈ {0.10, 0.20, 0.30}.

The parameter configuration which provides the best objective function values is depicted in Table 3.

5 Results

5.1 Solution quality

The results of the numerical experiments are summarized in Table 4. In comparison to the total tour length resulting from other application, the IGA reduced the total length of the picker tours by 3.15 % on average. The improvements ranged from 0.68 % (No. of customer orders: $n = 60$, capacity of the picking device: $C = 75$) to 5.76 % ($n = 20$, $C = 75$). By application of the GGA an average improvement of 3.75 % could be obtained. The improvements varied between 1.27 % ($n = 60$, $C = 75$) and 6.47 % ($n = 20$, $C = 75$).

The group-oriented approach outperformed the item-oriented approach not only with respect to the average improvement, but also resulted in a superior average objective function value for every single problem class. The superior results of the GGA can be explained by two aspects. On the one hand, in comparison to the IGA, the GGA generates solutions that incorporate a slightly smaller number of batches on average. A smaller number of batches tend to result in a smaller total tour length. On the other hand, the difference in the number of batches is not large enough to account completely for the differences in the solution quality. Therefore, it can be concluded that the main differences stem from the composition of the batches. The GGA is able to match customer orders better than the IGA does.

The development of the solution quality of the two GAs as a function of the number of generations is given in

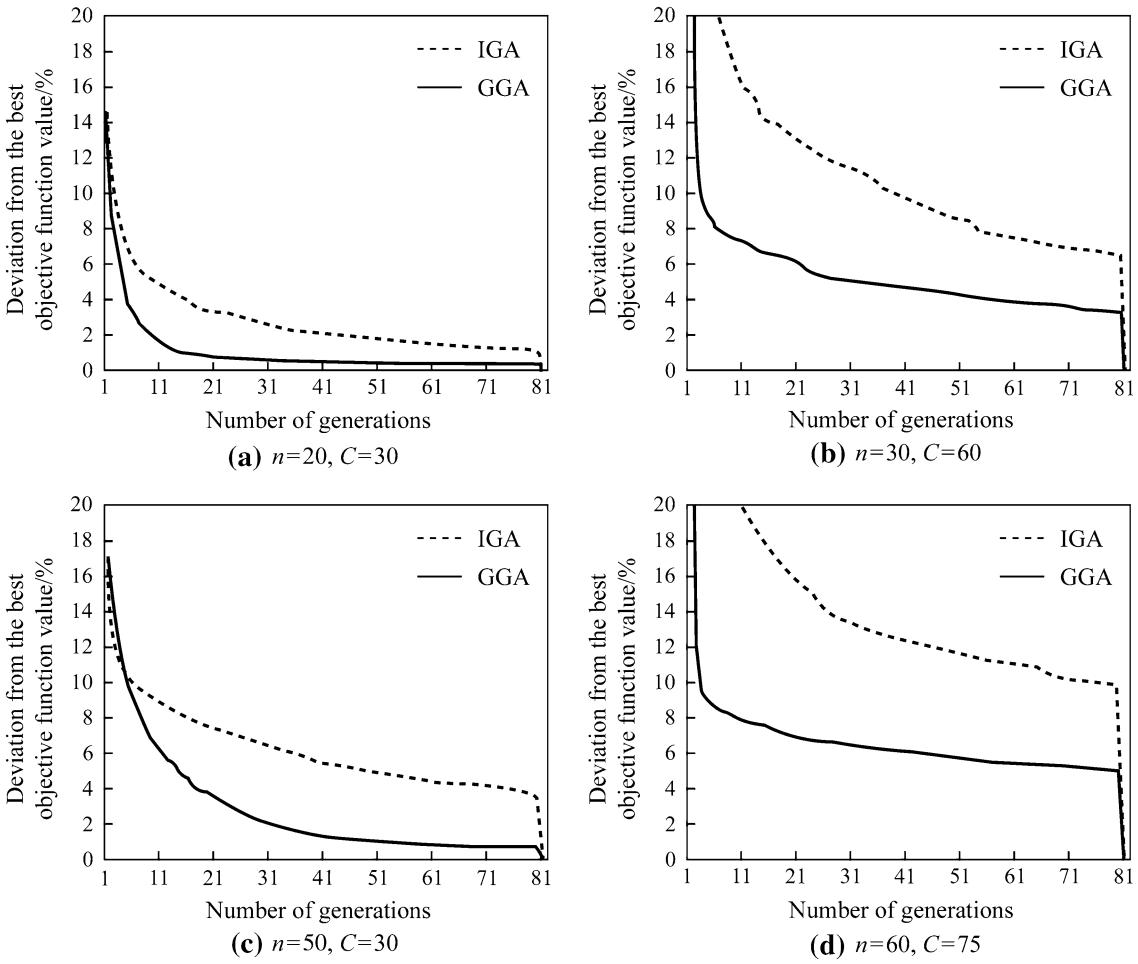

Fig. 6 Solution quality over the generations for different problem classes

Fig. 6. The evolution is shown exemplarily for the four problem classes. The average deviation of the objective function value from the best objective function value obtained at the end of the local search phase is depicted on the *Y*-axis. The number of generations (1–80) is presented on the *X*-axis. Figure 6 demonstrates two important differences between the two algorithms. The GGA needs fewer generations than the IGA in order to identify good solutions. Moreover the quality of the solutions resulting from the application of the "pure" GA (i.e., before the local search is applied) is much better for the group-oriented approach than for the item-oriented approach. It is demonstrated by the different step-sizes after generation 80. Application of the local search phase to the final generation of the GGA results in an additional average improvement between 0.3 % ($n = 20, C = 30$) and 5 % ($n = 60, C = 75$) whereas the application of the local search phase in the IGA resulted in additional improvements between 1.1 % ($n = 20$, $C = 30$) and 9.8 % ($n = 60$, $C = 75$).

5.2 Computing time

Table 5 demonstrates that the computing times are determined by two parameters, namely by the number of customer orders and by the size of the capacity of the picking device. An increase of each of the two parameters leads to an increase of the computing times of the algorithms. Comparison of the computing times of the two algorithms reveals that the item-oriented approach requires less computing time for small and medium-size problems of up to 40 customer orders. For a larger number of customer orders and for larger capacity of the picking device, in particular, the group-oriented approach consumes less time. In general the computing times of both algorithms are reasonable, with the maximum of four minutes for the IGA and three and a half minutes for the GGA, both for the most challenging problem class ($n = 60$, $C = 75$).

Table 5 Comparison of the computing times of the two algorithms

n	*C* (No. art)	Ø(IGA)/s	Ø(GGA)/s
20	30	3.1	7.0
	45	4.2	7.9
	60	4.5	9.0
	75	4.5	9.8
30	30	9.1	17.3
	45	15.0	20.8
	60	17.2	24.5
	75	16.5	27.1
40	30	22.5	34.6
	45	39.5	44.8
	60	48.9	53.7
	75	49.6	59.9
50	30	46.7	60.5
	45	87.4	86.2
	60	117.2	109.4
	75	109.2	116.5
60	30	94.8	99.6
	45	174.8	150.2
	60	230.3	187.2
	75	242.4	207.6

Note n is the number of customer orders; *C* (No. art) is the capacity of the picking device in number of articles; *Ø* is the average computing time in seconds per instance

6 Conclusions

In this paper we have proposed a GA method approach for the large size of warehouse location problem. Delving into the order batching problem, it has been recognized the pivotal importance for maximizing the efficiency of warehouse processes, since improved order batching can result in a significant reduction of the delivery lead times and in a reduction of the operating cost of a warehouse. Two population-based metaheuristics, an IGA and a GGA have been here introduced. The application of both algorithms results in a significant reduction of the total tour length of the order pickers with reasonable computing times. However, the application of the GGA resulted in improvements that are larger than those of the item-oriented one. These results demonstrate that it is very important to choose a GA that is suitable for the problem in order to obtain the best results possible.

References

1. Koch S, Wäscher G (2005) A grouping genetic algorithm for the order batching problem in distribution warehouses. In: Working Paper No. 26/2011. Otto-von-Guericke-Universität Magdeburg
2. Zhang GQ, Lai KK (2010) Tabu search approach for multi-level warehouse layout problem with adjacent constraints. Eng Optim 42(6):775–790
3. Ratliff HD, Rosenthal AS (1983) Orderpicking in a rectangular warehouse: a solvable case of the traveling salesman problem. Oper Res 31:507–521
4. Yang L, Feng Y (2006) Fuzzy multi-level warehouse layout problem: new model and algorithm. J Syst Sci Syst Eng 15(4):493–503
5. Henn S, Koch S, Wascher G (2012) Order batching in order picking warehouses: a survey of solution approaches. In: Manzini R (ed) Warehousing in the global supply chain. Springer, London, p 105
6. Gademann N, Van De Velde S (2005) Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Trans 37(1):63–75
7. Kratica J, Kovacevic-Vujcic V, Cangalovic M (2009) Computing the metric dimension of graph by genetic algorithms. Comput Optim Appl 44(2):343–361

8. Ho YC, Su TS, Shi ZB (2008) Order-batching methods for an order-picking warehouse with two cross aisles. Comput Ind Eng 55(2):321–347
9. Zhang GQ, Xue J, Lai KK (2002) A class of genetic algorithms for multiple-level warehouse layout problems. Int J Prod Res 40(3):731–744
10. Weicker K, Weicker N (2007) Towards qualitative models of interactions in evolutionary algorithms. In: De Jong KA, Poli R, Rowe JE (eds) Foundations of genetic algorithms VII. Morgan Kaufmann, San Francisco, pp 365, 382
11. Äut ÄS, Tuzkaya UR, Doga B (2008) A particle swarm optimization algorithm for the multiple-level warehouse layout design problem. Comput Ind Eng 54(4):783–799
12. Pan CH, Liu SY (1995) A comparative study of order batching algorithms. Omega Int J Manag Sci 23(6):691–700
13. Matic D, Filipovic V, Savic A et al (2011) A genetic algorithm for solving multiple warehouse layout problem. Kragujev J Math 35(1):119–138
14. Tsai CY, Liou JJH, Huang TM (2007) Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. Int J Prod Res 46(22):6533–6555
15. Kratica J, Kovacevic-Vujcic V, Cangalovic M (2008) Computing strong metric dimension of some special classes of graphs by genetic algorithms. Yugosl J Oper Res 18(4):143–151