Omega ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Order batching in a pick-and-pass warehousing system with group genetic algorithm ☆

Jason Chao-Hsien Pan [a], Po-Hsun Shih [b,*], Ming-Hung Wu [a]

[a] Department of Logistics Management, Takming University of Science and Technology, 56 Huanshan Road, Section 1, Taipei 11451, Taiwan, ROC
[b] Department of Information Management, Vanung University, 1 Van-Nung Road, Chung-Li, Tao-Yuan 32061, Taiwan, ROC

ARTICLE INFO

Article history:
Received 31 July 2013
Accepted 9 May 2015

Keywords:
Order batching policy
Order picking
Warehouse management
Pick-and-pass system

ABSTRACT

An order batching policy determines how orders are combined to form batches. Previous studies on order batching policy focused primarily on classic manual warehouses, and its effect on pick-and-pass systems has rarely been discussed. Pick-and-pass systems, a commonly used warehousing installation for small to medium-sized items, play a key role in managing a supply chain efficiently because the fast delivery of small and frequent inventory orders has become a crucial trading practice because of the rise of e-commerce and e-business. This paper proposes an order batching approach based on a group genetic algorithm to balance the workload of each picking zone and minimize the number of batches in a pick-and-pass system in an effort to improve system performance. A simulation model based on FlexSim is used to implement the proposed heuristic algorithm, and compare the throughput for different order batching policies. The results reveal that the proposed heuristic policy outperforms existing order batching policies in a pick-and-pass system.

1. Introduction

Warehousing management is a vital and logistical activity that can affect supply chain costs [29]. A pick-and-pass system, also called a progressive zoning system, divides a picking line into picking zones. Each zone is then typically assigned to a picker, and they are often connected by a conveyor in a warehouse [28]. This approach is commonly used for small to medium-sized items, such as household, health and beauty, office, and food products, which can be stored in relatively small and accessible pick locations along the picking line [27]. Since the emergence of e-commerce and e-business, global supply chain management has focused on the fast delivery of small and frequent inventory orders at a low total cost [3,33]. Thus, the operations of the pick-and-pass system play a key role in managing a supply chain efficiently.

The literature survey begins with a review on the warehouse management for pick-and-pass systems, followed by the developments of order batching policy. De Koster [6] approximated picking operations by applying the Jackson network model, and assumed that the service time at each pick station was exponentially distributed, and that customer orders arrived according to a Poisson process. Yu and De Koster [38] proposed an approximation method based on a *GI*/*G*/*m* queuing network modeling technique by using Whitt's queuing network analyzer ([36]) to investigate the effects of order batching and picking area zoning on the mean order throughput time in a pick-and-pass system. Jewkes et al. [24] developed an efficient dynamic programming algorithm for determining the optimal item allocation and picker locations for an order picking line comprising multiple pickers. Jane and Laih [23] proposed several heuristic algorithms for balancing the workload among pickers in a picking line. Gagliardi et al. [11] proposed and analyzed different product location and replenishment strategies for a distribution center that uses a pick-and-pass system for fulfilling orders. Pan and Wu [31] developed an analytical model for a pick-and-pass system by describing the operation of a picker as a Markov chain to determine the expected travel distance of pickers in a picking line, and proposed three algorithms that optimally allocate items to storages. Parikh and Meller [32] proposed a cost model for estimating the cost of each type of picking strategy to mitigate the problem of selecting between batch picking and zone picking strategies. Melacini et al. [28] defined a framework for the pick-and-pass system design to minimize the overall picking costs and meet the required service level.

Order batching problem (OBP), a major decision problem in the design and control of warehousing systems [5] and a key factor for the success of an order picking system [19], determines how

☆This manuscript was processed by Associate Editor Sterna.

* Corresponding author. Tel.: +886 3 4515811 ext 280; fax: +886 3 4621348.
E-mail address: shih@vnu.edu.tw (P.-H. Shih).

http://dx.doi.org/10.1016/j.omega.2015.05.004
0305-0483/

orders are combined into batches to be processed in a picking trip to reduce the travel distance of orders to be fulfilled [30]. Because obtaining precise solutions to the large-scale OBP by exerting reasonable computational efforts is impractical [9], researchers have developed heuristic methods to determine near-optimal solutions. De Koster et al. [4] combined well-known heuristics with new heuristics in an attempt to generate effective, fast, and robust order batching algorithms that are sufficiently simple to use in real-world situations. Gademann et al. [10] addressed batching in a wave picking operation, and presented an exact algorithm that assigns orders to batches to minimize the maximal lead time for each batch. Hwang and Chang [22] investigated order batch processing in which either a part of, or an entire single order or specific pair of orders may be grouped into a batch with a fixed capacity. Hsu et al. [20] developed a genetic-based algorithm for managing OBP with various types of batch structures and warehouse layouts to minimize the total travel distance. Won and Olafsson [37] addressed the typical warehousing problem of batching and order picking to improve the efficiency measured by the picking time and the effective use of vehicles. Hwang and Kim [21] presented an efficient order batching algorithm based on cluster analysis, and validated the performance of the proposed algorithm by comparing it with an existing algorithm regarding total travel time and the number of batches grouped. Tho and De Koster Rene [34] considered an OBP for a two-block rectangular warehouse by assuming that orders arrive according to a Poisson process, and adopted the well-known S-shape heuristic method for routing order pickers. Bozer and Kile [1] developed a new mixed-integer programming model to obtain near-exact solutions to the OBP. Ho et al. [16] developed several order batching methods for an order-picking warehouse with two cross aisles and an input/output point in one corner. Hsieh and Huang [19] developed K-means Batching and Self-organization Map batching polices, and investigated the overall performance of order picking systems by considering the storage assignment, order batching, and picker routing to determine the optimal policy combinations involving different order types. Henn [14] addressed an online order batching problem in which the maximal completion time of the customer orders arriving within a certain time to be minimized. Henn and Wäscher [13] presented two approaches based on the Tabu Search principle for deriving a solution to the OBP in a manual order picking system. Hong et al. [18] discussed an order batching formulation and a heuristic solution procedure suitable for parallel-aisle picking systems. Henn [15] applied variable neighborhood descent and variable neighborhood searches to minimize the total tardiness of a given set of customer orders. Henn and Schmid [12] discussed how metaheuristics can be used to minimize the total tardiness for a given set of customer orders. Lam et al. [25] proposed an order-picking system to manage an order-picking process as batches with common pick locations to minimize the travel distance, and determine the batch-picking sequence.

Previous studies on order batching policy have focused primarily on classic manual warehouses, also known as picker-to-part systems; whereas its effect on pick-and-pass systems has rarely been discussed. In a classic manual warehouse, the travel time of vehicles or pickers is often the dominant component of the operation cost; thus, a decrease in travel time or travel distance is used as the criteria of order batching. In contrast to classic manual warehouses, a pick-and-pass system contains a picking line or a conveyor of a fixed length, and the minimization of the travel distance may not be suitable under such conditions; instead, a reduction in the number of batches formed may serve as a more appropriate goal.

In practice, congestion may occur between two adjacent picking zones in a line since a picker must wait for the preceding picker to finish picking and transfer the container containing the picked items. Therefore, an optimal batching policy must consider both the number of batches and the balance of the picking line concurrently. This paper develops an order batching heuristic algorithm based on the group genetic algorithm (GGA) in a pick-and-pass system by considering the line balancing and the least number of batches formed to minimize the total operation time. The performance of the algorithm is compared and validated with the results generated by simulation models.

The remainder of the paper is organized as follows. Section 2 describes the framework and the order batching problem for a pick-and-pass system under consideration. Section 3 presents the proposed order batching heuristic, followed by an experimental design and a simulation model implemented to investigate the performances of the order batching heuristic in Section 4. Conclusions are highlighted in Section 5.

2. Description of picking operations and order batching problem

2.1. Picking operation for a pick-and-pass system

The pick-and-pass system considered in this study consists of a roller conveyor connecting all pick stations (zones) located along the conveyor line, as shown in Fig. 1. The picking line in the system has n pickers who pick the order items located in the vertical shelves in the respective zones. These vertical shelves are assumed to have precisely the same number of racks as the item types. Dummy items with no demand may be created to ensure that this relationship holds if less item types than racks are present.

A computer-aided picking system (CAPS) can be implemented in practice in a pick-and-pass system. In a CAPS, light indicator modules are mounted to all racks, and these modules automatically guide order pickers toward the pick locations, and show the amount of each item to be retrieved. The advantages of a CAPS include effectively improving the picking productivity by 50% or more and reducing the picking task error [23]. In addition, a CAPS simplifies training for pickers, thus lowering operational costs.

In such an information support system, the light indicator modules can show only the data of items to be picked in one order in a picker's assigned zone at a time in order to avoid pickers picking the wrong items. The picker picks the item quantity

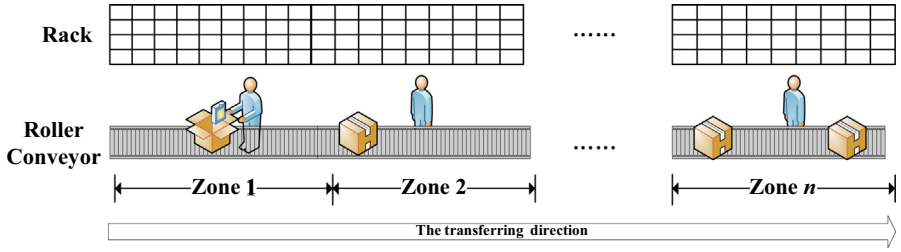

Fig. 1. A pick-and-pass system.

Please cite this article as: Pan JC-H, et al. Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.05.004

displayed on a rack, and then confirms the pick by pressing the lit button [31]. When the picker finishes the pickings of one order in the assigned zone, the CAPS sends the data of the next order to all of the light indicator modules in that zone. For the picking operation, it is reasonable to assume that the seek time of a picker can be omitted. The following assumptions on the pick-and-pass system and the picking operations in this study are considered in this research:

(1) The speeds of the roller conveyor and a picker are constant.
(2) The time to pick an item from a rack is constant.
(3) No orders can be spread, i.e., an order is assigned to a batch only (because of an additional sorting effort).
(4) Each item is independent of the others in an order.
(5) The storage racks are long and undersized; that is, the vertical picking time is negligible.

2.2. Order batching problem for a pick-and-pass system

Let $j=\{1,\ldots, N\}$ be the set of indices of customer orders, C be the capacity of a picking device, c_j be the size of order j, and vector $\boldsymbol{x_i}=(x_{i1}, x_{i2}, \ldots, x_{iN})$ denotes the set of orders to be picked in batch i where $c_j \le C$ for all $j \in J$ and $\sum_{j=1}^{N} c_j \le C$. A classic OBP can be defined with an integer programming model as follows [39,20,40]:

$$\text{Minimize} \sum_{i=1}^{K}\sum_{j=1}^{N} \text{Distance}(x_{ij}) \tag{1}$$

Subject to

$$\sum_{i=1}^{K} x_{ij} = 1, \quad j = 1, \ldots, N. \tag{2}$$

$$\sum_{j=1}^{N} c_j x_{ij} \le C, \quad i = 1, \ldots, K. \tag{3}$$

$$x_{ij} = 0 \text{ or } 1, \quad i = 1, \ldots, K \text{ and } j = 1, \ldots, N. \tag{4}$$

where $x_{ij}=1$ if order j is assigned to batch i; and 0 otherwise, and K is the number of active batches. In the formulation, objective function (1) minimizes the total travel distance of the order batches formed. Constraint set (2) ensures that order j is assigned to batch i only, and constraint set (3) is used to restrict the total size of the orders assigned to a batch not exceeding the capacity of that batch.

However, in pick-and-pass systems, minimizing the total travel distance may be meaningless because the travel distance of each container is constant in a tour, as shown in Fig. 1. Hence, it is more appropriate to consider a bi-criterion OBP in pick-and-pass systems to minimize the number of batches and balance the line concurrently.

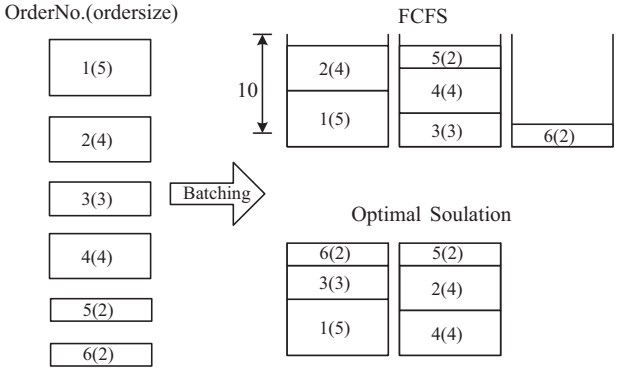

Fig. 2. The data and batching results of Example 1.

The OBP, the purpose of which is to form a minimum number of batches, can be considered an extension of the well-known one-dimensional bin packing problem defined by Falkenauer [7] in determining how to pack a set of boxes of different sizes into containers of a given size to use as few containers as possible. For an OBP in pick-and-pass systems, a container and a box can be regarded as a batch and an order, respectively. Minimizing the number of batches can reduce the efforts to split the batches after the picking is completed. The unused capacity of the batches inevitably produces additional batches; in other words, each batch should be as full as possible, and thus, objective function (1) is revised as

$$\text{Minimize} \sum_{i=1}^{K}\left(C - \sum_{j=1}^{N} c_j x_{ij}\right) \tag{5}$$

Example 1. Fig. 2 shows an OBP for a pick-and-pass system. Six orders are to be picked. Assume that the capacity of a batch is 10 units, and two batching policies are implemented. According to the first-come-first-served (FCFS) discipline, three batches ($K=3$) are required to fulfill orders, and 10 units of the capacity in the batches are left unused. On the other hand, an optimal policy requires only two batches ($K=2$) to fulfill these orders, and no unused capacity remains in the batches.

In addition, the balance of the line must be considered in a pick-and-pass system with multiple pickers. Similar to a manufacturing flow line, an imbalance can cause a severe deterioration of the order throughput time [2,31], and may result in orders not being fulfilled within the scheduled hours of operation during the day [32].

Example 2. Four orders are to be picked in a picking line consisting of four zones. The capacity of a batch is assumed to be 20 units. Two solutions are generated in this example, as shown in Fig. 3. In Solution 1, Batch 1 contains Orders 1 and 2, and Batch 2 contains Orders 3 and 4. Solution 2 also yields two batches where Batch 1 contains Orders 1 and 3, and Batch 2 contains Orders 2 and 4. Although both solutions have an identical total travel time on the picking line as they have the same number of batches, the workload of all zones in Solution 1 is obviously more balanced than that in Solution 2. Consequently, if all of the batches are of the same size in a line, all of the pickers can pick concurrently, with no idle time in the line.

Based on Example 2, the idle time of the entire picking line should be reduced as much as possible because minimizing the total idle time is equivalent to maximizing the line efficiency. Thus, when the line balancing is considered, objective function (1) can be modified as

$$\text{Minimize} \sum_{i=1}^{No\_zones} \text{idle time(zone } i\text{)} \tag{6}$$

The line balancing and OBP are both non-deterministic polynomial-time hard problems; therefore, heuristic methods are required to find the solutions of the two problems. Among them, genetic algorithm (GA) is one of the most widely used methods because it can solve any optimization problem that can be described using chromosome encoding. However, the encoding scheme of a GA is unsuitable because a classic GA regards the orders, whereas the cost function of an OBP depends on the batches. Thus, the structure of the chromosome for an OBP should be batch oriented, instead of order oriented. This implies that solutions may have different lengths of chromosomes since a batch is encoded to one gene. In Example 1, the solution applying

Please cite this article as: Pan JC-H, et al. Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.05.004

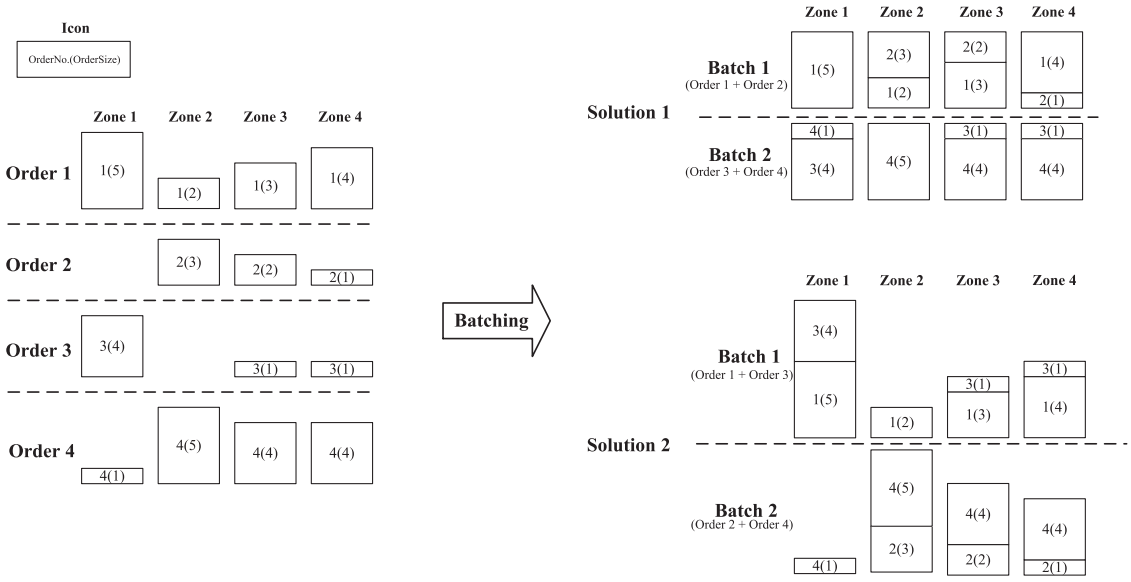

Fig. 3. The data and batching results of Example 2.

FCFS and the optimal solution have three and two genes (batches) in the chromosome, respectively.

3. An order batching heuristic based on GGA

Falkenauer [7] proposed the GGA for the special structure of grouping problems. Compared with the classic GA, the GGA has superior performance in solving grouping problems (e.g., OBP), because the classic GA by Holland [17] reduces the ability of crossovers to reliably estimate the quality of the schemata of a sample [35]. Falkenauer [7] demonstrated that directly applying the classic GA to an arbitrary problem without considering its structure could lead to unfavorable results, and indicated the differences between the two as follows:

- The GGA encodes groups on one gene for a one group basis.
- The GGA uses exponentially large alphabets and features variable lengths of chromosomes.
- Because of the GGA encoding, the operators must manipulate groups of objects, instead of individual objects.

A classic GA can indirectly compute the contents of the batches through the encoding distribution of the orders, but is relatively ineffective in determining a favorable solution to grouping problems. Because the GGA applies groups of orders for genes, the value of a batch is allocated in a straightforward manner according to the set of orders that are the members of the batch it represents. To solve an OBP for a pick-and-pass system by applying the GGA, GGA operators are defined, and the notations used are listed in the subsequent section.

Notation

n	the number of zones in a picking line
N	the number of orders
K	the number of active batches
C	the capacity of a batch
c_{ip}	the number of items to be picked in zone i in order p
$a_i(g)$	the workload of zone i in a batch for gene g in a chromosome
f_j	cost function of chromosome j
$B(g)$	the sum of the absolute difference between the mean workload and the workload of each zone for gene g in a chromosome
$F(g)$	the utilization of the capacity used in a batch for gene g in a chromosome

3.1. Encoding

The classic GA applies a straightforward encoding approach with a string of integers for an OBP. Referring to Example 1, Orders 1 and 2 are grouped into Batch 1; Orders 3, 4, and 5 are grouped into Batch 2; and Orders 6 is grouped into Batch 3 according to FCFS. The codes for this solution can be expressed as (1, 1, 2, 2, 2, 3). This encoding approach is classified as one-to-many [7]; that is, one solution of the group problem is encoded into several genes in a chromosome. Because of the number of possible batches combinations, the degree of redundancy of this approach increases exponentially [7]; consequently, the scope of the searching space of the classic GA is considerably larger than the original scope of solutions. In the encoding method of GGA, each gene on a chromosome represents a set of orders and is named in alphabetical order (A, B, etc.). For example, the two chromosomes of Example 1 can be encoded as

Solution 1 : $A_1B_1C_1$

Solution 2 : A_2B_2

An order group method is developed to generate and encode an initial chromosome as follows:

Step 1: place all orders in the order pool.
Step 2: if the order pool is empty, proceed to Step 5; otherwise, randomly select an order from the order pool as the seed of a new gene.
Step 3: determine the candidate orders: an order with a capacity demand less than or equal to the remaining capacity of the batch is a candidate order.
Step 4: if no candidate order exists, return to Step 2; otherwise, randomly select a candidate order, and add it to the seed and return to Step 3.
Step 5: stop and label all of the genes of the new chromosomes.

Please cite this article as: Pan JC-H, et al. Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.05.004

Table 1
The sizes of the orders of .

Order no.	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5	Total
1	1	–	–	–	1	2
2	2	–	1	–	–	3
3	–	2	–	1	3	6
4	1	–	2	–	–	3
5	–	2	–	2	2	6
6	1	1	1	3	1	7
7	–	2	1	–	–	3
8	1	–	–	2	–	3
9	–	1	3	–	5	9
10	2	3	1	1	2	9

Example 3. Consider a picking line consisting of five zones. The number of orders to be picked is 10. The capacity of a batch is 15, and the sizes of the orders are listed in Table 1. Two chromosomes are generated using the proposed order group procedure, and all of the genes in both chromosomes are named in alphabetical order, as listed in Table 2. To understand the details of the method comprehensively, the process of generating Chromosome 1 is shown in Fig. 4, and is detailed as follows:

(1) From the order pool, randomly select Order 8 as the seed.
(2) The rest are all candidate orders because their sizes do not exceed the remaining capacity of 12.
(3) From the candidate orders, randomly add Order 9 to the seed.
(4) The remaining capacity of the batch is three units, and thus, Orders 1, 2, 4, and 7 are candidate orders.
(5) Randomly add Order 1 to the seed.
(6) No order can be selected as a candidate order; thus, Orders 8, 9, and 1 are clustered as a gene code.

Repeat this process until the order pool is empty to form Chromosome 1.

3.2. The fitness function

As stated, to determine the minimal number of batches formed is one of the two objectives of the OBP in this study. Because the unused capacity of the batches inevitably results in additional batches, maximizing the loading rates of batches is a common objective for an OBP. Falkenauer [7] defined a fitness function as

$$f_{BPP} = \sum_{i=1}^{K} (F_i/C)^{\lambda}/K \tag{7}$$

where F_i is the sum of sizes of orders in batch i and λ is a constant, $\lambda \geq 1$.

Regarding line balancing, calculating the idle time by using mathematical formulas is difficult as the processing time of each station may vary because of different sizes of orders processed. Thus, the idle time has often been measured as a workload in most studies on the assembly line balancing problem. Rachamadugu and Talbot [26] presented the mean absolute deviation (MAD) of the workload for evaluating workload smoothness as

$$\text{MAD} = \frac{1}{No\_zones} \sum_{i=1}^{No\_zones} |T_i - \overline{T}| \tag{8}$$

where T_i is the workload of zone i and $\overline{T}$ is the mean workload.

For an OBP in pick-and-pass systems, a novel MAD is derived from Eq. (8) to estimate the idle time of a picking line as

$$\text{MAD}^{(OBP)} = \left(\sum_{g=1}^{K} \sum_{i=1}^{n} \left(\left| a_i(g) - \frac{C}{n} \right| \right) \right) \tag{9}$$

where $a_i(g) = \sum_{p \in batch_g} c_{ip}$.

Referring to Example 2, $C=20$ and $n=4$. A zero $\text{MAD}^{(OBP)}$ value is yielded by Solution 1 because the picking line is completely balanced as the workloads of all zones are the same, whereas for Solution 2 it is

$$\text{MAD}^{(OBP)} = |9-5| + |2-5| + |4-5| + |5-5| + |1-5| + |8-5| + |6-5| + |6-5| = 16$$

Finally, a fitness function is derived from Eq. (9) for the proposed algorithm as follows:

$$f = \left(1 + \sum_{g=1}^{K} \sum_{i=1}^{n} \left(\left| a_i(g) - \frac{C}{n} \right| \right) \right)^{\frac{-1}{\lambda}} \tag{10}$$

Moreover, the term $\sum_{i=1}^{n} (|a_i(g) - C/n|)$ in Eq. (10) denotes the MAD value of a picking line for batch g. A batch containing a full load and an equal workload for each zone is an optimal solution. Thus, the mean workload C/n can be used to calculate the MAD value. If the number of orders in a batch increases, then the associated MAD value would decrease. In other words, using the proposed fitness function can yield a solution with a minimal number of batches and an effectively balanced picking line, as shown in the following properties:

Lemma 1. *Suppose that order a has k items to be picked in a certain zone, and $|C/n-x|$ is the MAD of the workload of the pickers in that zone. If $k \leq C/(2n)$, then the MAD value after order a is grouped with order b is less than the sum of the MAD values of order a and order b.*

Proof. Let s be the number of items to be picked in order b in the zone under consideration, and define $\varepsilon_a = |C/n-k|$ and $\varepsilon_b = |C/n-s|$.

If $s \geq C/n$, then

$$|C/n-(k+s)| = \varepsilon_b + k \leq \varepsilon_a + \varepsilon_b = \varepsilon_b + C/n - k.$$

If $s \leq C/2n$, then

$$|C/n-(k+s)| = \varepsilon_a - s \leq \varepsilon_a + \varepsilon_b.$$

If $C/2n \leq s \leq C/n$ and $(s+k) \leq C/n$, then

$$|C/n-(k+s)| = \varepsilon_a - s \leq \varepsilon_a + \varepsilon_b.$$

If $C/2n \leq s \leq C/n$ and $(s+k) > C/n$, then

$$|C/n-(k+s)| = k - \varepsilon_b \leq \varepsilon_a + \varepsilon_b. \square$$

Lemma 2. *Suppose the number of items to be picked in order a is greater than or equal to C/n. If $s < C/(2n)$, then the MAD value after order a is grouped with order b will be less than the sum of the MAD values of order a and order b; otherwise, the grouping will generate greater MAD value.*

Proof. If $s < C/(2n)$, then $|C/n-(k+s)| = \varepsilon_a + s \leq \varepsilon_a + \varepsilon_b$; otherwise, for $s \geq C/(2n)$, $|C/n-(k+s)| = \varepsilon_a + s \geq \varepsilon_a + \varepsilon_b$. $\square$

Table 2
The batching results of Example 3.

Chromosome 1				Chromosome 2			
Gene	Order no.			Gene	Order no.		
A_1	8	9	1	A_2	1	10	7
B_1	3	7	5	B_2	4	8	9
C_1	10	2	4	C_2	2	6	
D_1	6			D_2	3	5	

Please cite this article as: Pan JC-H, et al. Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.05.004

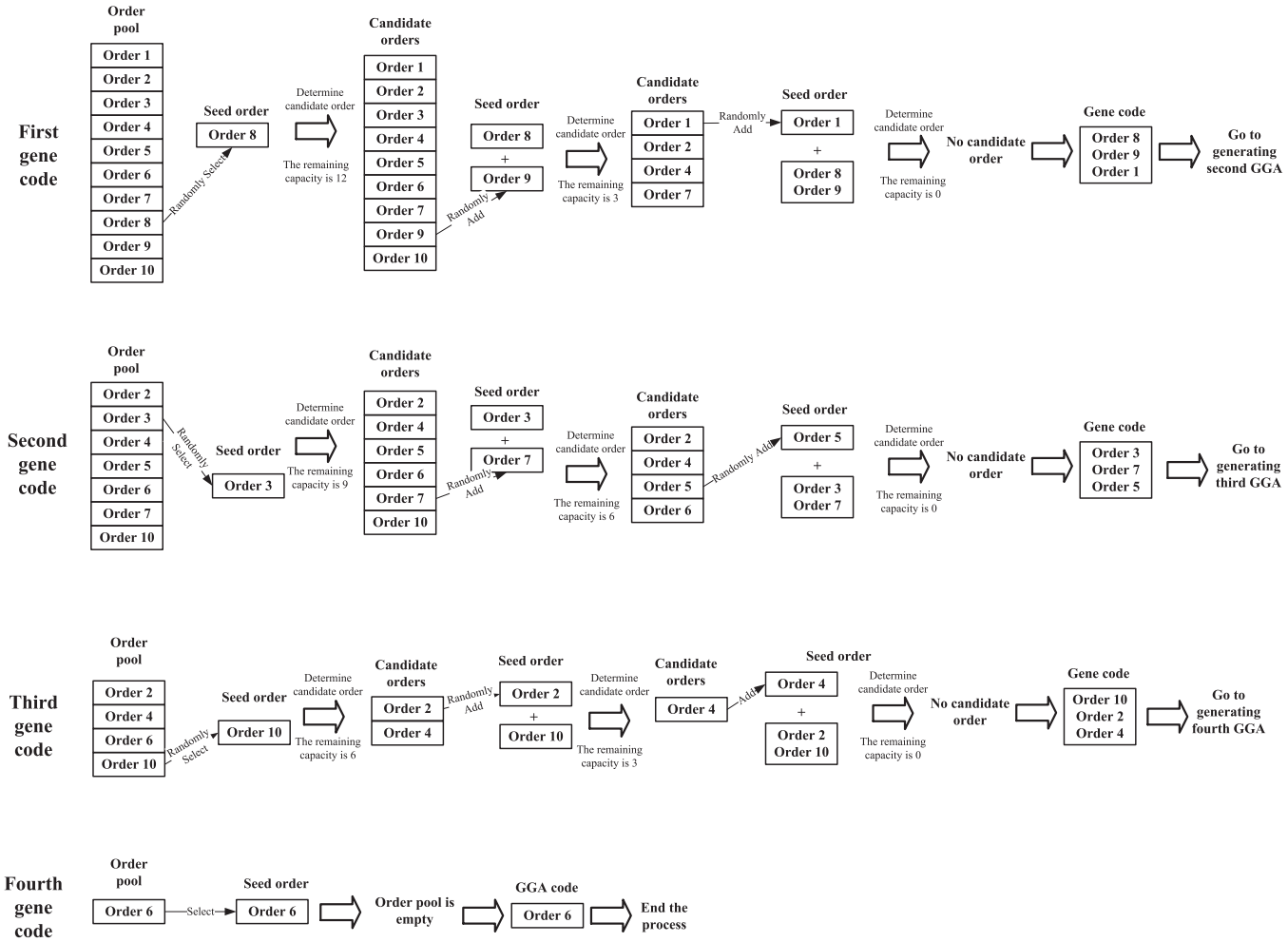

Fig. 4. The illustration of the process of generating chromosome 1 in .

Lemma 3. *Suppose the numbers of items to be picked in order a is between C/(2n) and C/n. The grouping of order a and order b is worse in term of MAD value under two situations: (1) $s \geq C/n$ or (2) $C/(2n) \leq s \leq C/n$ and $C/n+C/(2n) \leq s+k \leq 2C/n$. Otherwise, the MAD value will be reduced.*

Proof.

(1) If $s \geq C/n$, then

$$|C/n-(k+s)| = \varepsilon_b + k \geq \varepsilon_a + \varepsilon_b.$$

(2) If $C/(2n) \leq s \leq C/n$ and $C/n+C/(2n) \leq s+k \leq 2C/n$, then

$$|C/n-(k+s)| = (s+k)-C/n/C/(2n) \geq \varepsilon_a + \varepsilon_b.$$

otherwise,

If $s \leq C/(2n)$ and $(s+k) \leq C/n$, then

$$|C/n-(k+s)| = \varepsilon_a - s \leq \varepsilon_a + \varepsilon_b.$$

If $s \leq C/(2n)$ and $(s+k) \leq C/n$, then

$$|C/n-(k+s)| = s-\varepsilon_b \leq \varepsilon_a + \varepsilon_b.$$

If $C/(2n) \leq s \leq C/n$ and $C/n \leq s+k \leq C/n+C/(2n)$

$$|C/n-(k+s)| = (s+k)-C/n \leq C/(2n) \leq \varepsilon_a + \varepsilon_b. \square$$

Lemmas 1–3 indicate that only a few conditions can generate an adverse case regarding a high MAD value when two orders are grouped in a zone. The results revealed that the sum of the MAD of all zones after orders are grouped is less than before, even if adverse cases existed in a picking line.

Lemma 4. *Let ε_a^z denote the MAD value of order a in zone z and $\varepsilon_{(a+b)}^z$ denote the MAD value of order a grouped with order b in zone z, and k_z and s_z denote the order sizes of order a and order b in zone z, respectively. If $\sum_{z=1}^{n}(k_z+s_z) \leq C$, then $\sum_{z=1}^{n}\varepsilon_a^z + \sum_{z=1}^{n}\varepsilon_b^z \geq \sum_{z=1}^{n}\varepsilon_{(a+b)}^z$.*

Proof. Assume order *a* and order *b* have *m* zones in the worst case, $k_z = s_z = \lceil C/n \rceil$ for $z=1, 2,\ldots, m$. Since $\sum_{z=1}^{n}(k_z+s_z) \leq C$, it follows that $\sum_{z=1}^{m}(k_z+s_z) = 2mC/n$ and $\sum_{z=m+1}^{n}(k_z+s_z) = C-2mC/n = (n-2m)C/n$, where $m=(n-1)/2$ for n is odd and $m=n/2$ for n is even.

Let

$$f \equiv \sum_{z=1}^{n}\varepsilon_{(a+b)}^z = \sum_{z=1}^{n}\left|\frac{C}{n}-(k_z+s_z)\right| = \sum_{z=1}^{m}\left(k_z+s_z-\frac{C}{n}\right) + \sum_{z=m+1}^{n}\left(\frac{C}{n}-(k_z+s_z)\right)$$
$$= \frac{2mC}{n}-\frac{mC}{n}+\frac{(n-m)C}{n}-\frac{(n-2m)C}{n} = \frac{2mC}{n}$$

and,

$$f' \equiv \sum_{z=1}^{n}\varepsilon_a^z + \sum_{z=1}^{n}\varepsilon_b^z = \sum_{z=1}^{n}\left(\left|\frac{C}{n}-k_z\right| + \left|\frac{C}{n}-s_z\right|\right)$$
$$= \sum_{z=1}^{m}\left(k_z+s_z-\frac{2C}{n}\right) + \sum_{z=m+1}^{n}\left(\frac{2C}{n}-(k_z+s_z)\right)$$
$$= \frac{2mC}{n}-\frac{2mC}{n}+\frac{2(n-m)C}{n}-\frac{(n-2m)C}{n} = C$$

Then, it follows that $f'-f=C-(2mC/n)=(C(n-2m)/n)\geq 0$, since $C>0$ and $n\geq 2m$.□

The above lemmas lead to the following theorem.

Theorem. *Suppose that m orders are to be picked in a line of n zones, and the size of each order does not exceed the capacity of the batch. Therefore, using the sum of the absolute difference between the workloads of all zones and the mean workload,* $\sum_{i=1}^{n}(|C/n-a_i(g)|)$, *to evaluate the MAD of the picking line can yield an optimal solution for OBP in a pick-and-pass system with the minimal number of batches and the picking line balance considerations.*

3.3. Selection

A selection operator is part of a GA process, which selects chromosomes from the current generation to be copied into the subsequent generation. Several types of selection methods are used in a GA, and this study adopts linear ranking selection [7] to determine the probability of the chromosomes to be selected. For the ranking selection, chromosomes are sorted according to their fitness values, and rank N is assigned to the ideal chromosome, and rank 1 is assigned to the worst chromosome. The fitness of all of the chromosomes is normalized by using the following formula:

$$f'_j=\frac{f_{max}-f_{min}}{N}(\text{Rank } j) \tag{11}$$

Furthermore, the probability of the chromosomes to be selected is

$$r_j=\frac{f'_j}{\sum_{i=1}^{K} f'_i} \tag{12}$$

3.4. Crossover

According to the GGA, a crossover for an OBP must function with variable-length chromosomes in which genes are grouped independently from each other during a crossover. Because the GGA incorporates the groups of orders for genes, the crossover must manipulate groups of orders, instead of individual orders. Two selection methods are used for crossing genes: one is random selection, and the other incorporates the following two indicators for crossing genes:

$$(1)\ B(g)=\frac{n}{2C(n-1)}\sum_{i=1}^{n}\left(\left|\frac{C}{n}-a_i(g)\right|\right),\quad 0\leq B(g)\leq 1 \tag{13}$$

where a $B(g)$ value of 1, the most unbalanced batch, indicates that all of the items in batch g are allocated in the same zone; and a $B(g)$ value of 0 signifies that all of the items in batch g are allocated equally in each picking zone.

$$(2)\ F(g)=\sum_{i=1}^{n} a_i(g)/C,\quad 0<F(g)\leq 1 \tag{14}$$

where a high value of $F(g)$ indicates a high percentage of the capacity is used in batch g; in particular a value of 1 indicates that the batch is a full load.

A gene that has specific values of these two indicators can be considered to cross over. The proposed crossover procedure is stated as follows:

(1) Step 1: select one of the following selection methods:
Random: randomly select several genes from the first parent.
(2) Eugenics: select the genes with $B(g)\leq\alpha$ and $F(g)\geq\beta$ to form the first parent.
Step 2: use the selection method to select the crossing genes of chromosomes in each of the two parents.
Step 3: randomly determine the crossing site at the second parent.
Step 4: inject genes selected from the first parent at the crossing site of the second parent.
Step 5: eliminate all objects occurring twice from the group when they were members of the second parent.
Step 6: use the order group procedure for all ungrouped orders in the order pool.
Step 7: reinsert new batches into the new chromosomes.
Step 8: re-label all of the genes of the new chromosomes.
Step 9: apply Steps 2–8 to the two parents, with their roles reversed.

An example of the crossover procedure is shown in Fig. 5.

3.5. Mutation

Regarding the crossover of the GGA, the mutation operator for the OBP must function with batches, instead of orders. The mutation operator of the GGA can usually either create a new gene or eliminate an existing gene from the chromosomes. This study chooses to create a new gene for the mutation operator. The proposed procedure can be defined as follows:

Step 1: use the order group procedure to generate a new gene.
Step 2: randomly select a gene from the chromosome, and replace it with a new gene.
Step 3: eliminate all objects occurring twice from the group if they are members of the original parent.
Step 4: use the order group procedure for all ungrouped orders in the order pool.
Step 5: reinsert new batches into the new chromosomes.
Step 6: re-label all genes of the new chromosomes.

The overall procedure of the proposed order batching heuristic based on the GGA for the pick-and-pass system is described as follows:

Step 1: set population size (M), the number of generations (T), crossover rate (Rc), and mutation rate (Rm), $m=1$ and $t=0$.
Step 2: use the order group procedure to generate a chromosome m in generation t.
Step 3: if $m=M$, proceed to Step 4; otherwise, set $m=m+1$, and return to Step 2.
Step 4: compute fj in generation t by using Eqs. (10), $j=1, 2,\ldots, M$.
Step 5: select M chromosomes from generation t by using Eqs. (11) and (12) for the next generation $(t+1)$.
Step 6: randomly select $M\times Rc$ chromosomes from generation $(t+1)$ to be $M\times Rc/2$ pairs of parents.
Step 7: generate $M\times Rc$ offspring by using the crossover procedure from these parents.
Step 8: replace the parents with the offspring.
Step 9: randomly select $M\times Rm$ chromosome from generation $(t+1)$, and apply the mutation procedure.
Step 10: if $t=T$, then stop; otherwise, set $t=t+1$, and return to Step 4.

4. Experimental design and simulation model

To examine the performance of the proposed algorithm, 18 scenarios for the pick-and-pass system, including one, two, and three picking lines, are designed for implementing order batching policies, as listed in Table 3. The GGA parameters are as follows:

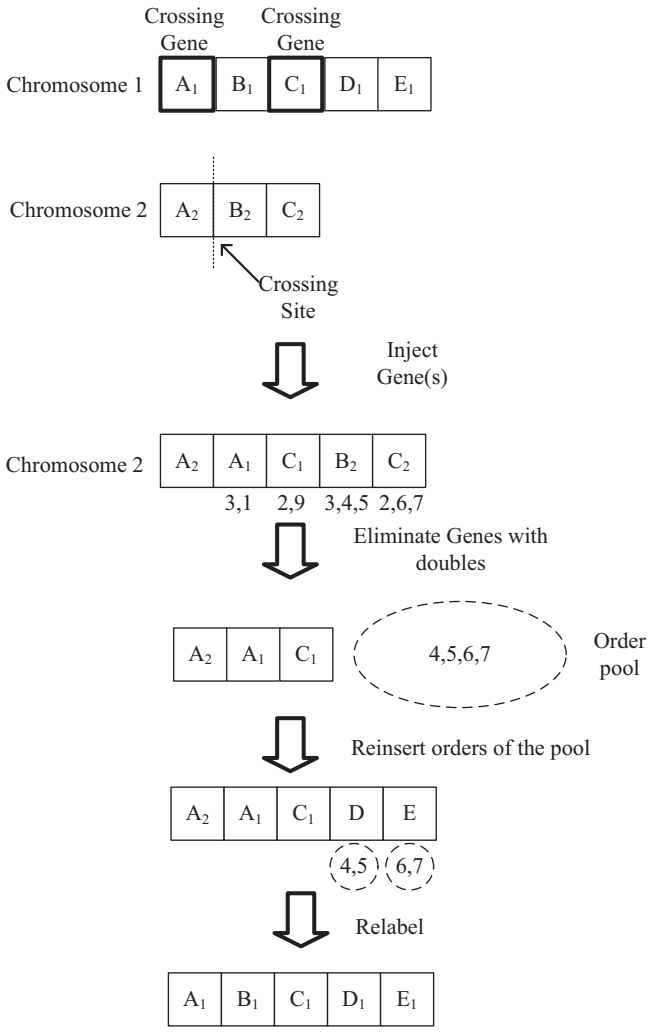

Fig. 5. An example of the GGA crossover procedure.

population size=20, no. of generations=50, crossover rate=0.6, mutation rate=0.05, $\lambda=1$, $\beta=0.1$ and $\alpha=0.95$. In addition, the classic GA and FCFS policies are used to execute these scenarios and compare GGA_E and GGA_R by exploiting Eq. (3) as the fitness function with eugenics and random selection, respectively. The performance procedure of the classic GA for the OBP is derived from Hsu et al. [20]. Ten runs of each scenario are performed for these polices. All of the policies are coded in the Visual Basic application for Excel 2010, and run on a PC equipped with a 3.0-GHz Pentium processor. The batching results are listed in Table 4 by presenting the highest fitness value and the fewest number of batches generated, along with the CPU time consumed by the heuristic policy for each scenario.

As shown in Table 4, the classic GA, GGA_R, and GGA_E yield a significant improvement compared to the FCFS policy. Regarding the optimal fitness value, Scenario 4 shows that the maximal difference rate, defined by (the policy/FCFS), of the classic GA, GGA_R, and GGA_E are 5.177, 5.681, and 5.767, respectively. In the systems consisting of three picking lines, although the difference rates between the heuristic policies and FCFS policy are relatively small, the performance of all of the heuristic policies are still superior to those of the FCFS policy. Because achieving a line balance for multiple picking lines is difficult, both the GGA and classic GA may require additional generations to obtain favorable solutions. In addition, the results reveal that these heuristic methods can reduce the number of batches formed by 90–97% in all of the scenarios which imply that the proposed function can generate favorable solutions when both the picking line balance and the reduction in the number of batches formed are considered.

Compared to the classic GA, the proposed GGA_R and GGA_E are both evidently superior, as listed in Table 4, which shows that favorable solutions are frequently generated by either GGA_R or GGA_E. Fig. 6 shows that GGA_E and GGA_R significantly outperform the classic GA where the improvement rate is obtained by using (GGA − Classic GA)/(Classic GA) × 100%. All of the improvement rates of GGA_R and GGA_E are more than 5%, except for in Scenario 12. In addition, although the CPU times of the classic GA are shorter than those of GGA_R and GGA_E, the classic GA produces the worst fitness values in all of the scenarios after 50 generations of evolution. Among the 18 scenarios, Scenario 2 is randomly selected for probing the relationship between the solution quality and computational efforts regarding the fitness function value and the number of generations that the three algorithms produced, as shown in Fig. 7. The trend in the evolution of GGA_R and GGA_E is obviously more satisfactory than that of the classic GA. Specifically, GGA_E defines a limit for all batches to filter specific genes so as to quickly determine an effective solution because a gene represents a batch, instead of an individual order when applying the GGA. In other words, if the classic GA generates an identical solution to that of GAA_E or GGA_R, additional CPU time is required for more generations of its evolution, as shown in Fig. 8.

Moreover, the actual idle situation of the pickers in a pick-and-pass system in the real world after implementing these polices can further facilitate the understanding of the performance of the GGA. However, as mentioned, calculating the idle time by using mathematical formulas is difficult, so simulation software package Flexsim [8] is applied to construct a virtual system for Scenario 2 to analyze the effect of an imbalanced workload in a picking line. The picking time per item of one second for a picker is assumed to be constant. The picking time includes the time for item handling and administration activities. The simulation model is run 10 times for each order batch method, and the idle rate, standard deviation, and confidence interval for each order batch method are listed in Table 5. A picker could be idle in the following two situations:

(1) Blocking: a picker cannot transport the container to the next zone after completing the operation. The blocking rate is expressed as 1−(a pick's blocking time in the simulation model/total simulation time).

Table 3
Summary of the scenarios in the experimental design.

Scenario no.	Parameters				
	No. of orders (N)	Distribution of an order size	Capacity (C)	No. of zones (n)	No. of lines
1	100	U[1,5]	15	5	One
2	200	U[1,5]	15	5	One
3	300	U[1,10]	20	5	One
4	300	U[5,15]	50	5	One
5	300	U[5,15]	50	10	One
6	500	U[5,15]	50	10	One
7	100	U[1,5]	15	5	Two
8	200	U[1,10]	30	5	Two
9	300	U[5,10]	30	5	Two
10	300	U[5,15]	50	5	Two
11	300	U[5,15]	50	10	Two
12	500	U[5,15]	50	10	Two
13	100	U[1,5]	15	5	Three
14	200	U[1,10]	30	5	Three
15	300	U[5,10]	30	5	Three
16	300	U[5,15]	50	5	Three
17	300	U[5,15]	50	10	Three
18	500	U[5,15]	50	10	Three

Please cite this article as: Pan JC-H, et al. Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.05.004

Table 4
The batching results of the four polices implemented.

Scenario no.	FIFS			Classic GA			GGA_R			GGA_E		
	Highest fitness value	Fewest no. of batches		Highest fitness value	Fewest no. of batches	CPU Time (s)	Highest fitness value	Fewest no. of batches	CPU Time (s)	Highest fitness value	Fewest no. of batches	CPU time (s)
1	0.0762	21	Mean	0.283 (3.720)	**19.0 (90%)**	9.80	0.332 (4.361)	**19.0 (90%)**	13.30	**0.338 (4.441)**	**19.0 (90%)**	13.20
			S.D.	0.014	0.00	0.38	0.022	0.00	0.44	0.032	0.00	0.57
2	0.0364	44	Mean	0.123 (3.396)	41.0 (93%)	17.90	0.141 (3.879)	**40.0 (91%)**	26.10	**0.143 (3.923)**	**40.0 (91%)**	26.00
			S.D.	0.006	0.00	0.54	0.006	0.00	1.04	0.007	0.00	1.79
3	0.0172	93	Mean	0.075 (4.334)	86.2 (93%)	32.50	0.082 (4.766)	**81.0 (87%)**	44.20	**0.083 (4.852)**	**80.7 (87%)**	44.90
			S.D.	0.001	0.43	0.64	0.002	0.00	2.17	0.003	0.44	2.23
4	0.0242	66	Mean	0.126 (5.177)	64.0 (97%)	29.40	0.138 (5.681)	62.2 (94%)	42.90	**0.140 (5.767)**	**62.0 (94%)**	40.50
			S.D.	0.003	0.00	0.49	0.004	0.40	2.66	0.007	0.00	2.54
5	0.0265	68	Mean	0.097 (3.676)	64.2 (94%)	52.60	**0.103 (3.895)**	**62.0 (91%)**	65.90	**0.103 (3.885)**	**62.0 (91%)**	65.30
			S.D.	0.001	0.46	0.49	0.002	0.40	2.70	0.003	0.49	3.20
6	0.0162	111	Mean	0.055 (3.411)	107.5 (97%)	103.80	**0.058 (3.600)**	**104.2 (94%)**	119.50	**0.059 (3.609)**	104.7 (94%)	122.00
			S.D.	0.001	0.47	1.11	0.001	0.38	5.26	0.001	0.44	7.01
7	0.1714	21	Mean	0.249 (1.452)	20.5 (98%)	14.70	**0.277 (1.616)**	20.3 (97%)	27.80	0.272 (1.588)	**20.1 (96%)**	27.80
			S.D.	0.016	0.30	0.64	0.014	0.46	0.60	0.021	0.30	0.60
8	0.0878	41	Mean	0.176 (2.005)	37.1 (90%)	29.60	**0.214 (2.433)**	**37.0 (90%)**	46.60	**0.214 (2.441)**	**37.0 (90%)**	46.30
			S.D.	0.009	0.30	0.49	0.010	0.00	1.50	0.010	0.00	0.78
9	0.0419	86	Mean	0.077 (1.836)	82.3 (96%)	49.70	**0.082 (1.964)**	**79.3 (92%)**	71.20	0.081 (1.941)	79.4 (92%)	71.40
			S.D.	0.001	0.46	0.90	0.003	0.64	3.84	0.003	0.49	3.17
10	0.0735	49	Mean	0.176 (2.398)	47.0 (96%)	44.50	0.192 (2.617)	47.0 (96%)	65.00	0.196 (2.674)	47.0 (96%)	64.90
			S.D.	0.006	0.00	0.50	0.007	0.00	2.24	0.002	0.00	1.64
11	0.0576	66	Mean	0.095 (1.653)	63.0 (95%)	85.70	0.102 (1.765)	62.9 (95%)	100.40	0.102 (1.766)	62.9 (95%)	99.80
			S.D.	0.002	0.00	1.19	0.002	0.30	3.17	0.002	0.30	2.60
12	0.0342	111	Mean	0.056 (1.647)	105.7 (95%)	159.00	0.060 (1.762)	103.8 (94%)	176.00	0.059 (1.722)	104.0 (94%)	177.20
			S.D.	0.001	0.46	1.67	0.001	0.40	8.17	0.001	0.00	6.16
13	0.2545	22	Mean	0.263 (1.032)	21.0 (95%)	21.20	0.292 (1.146)	21.0 (95%)	34.30	0.295 (1.161)	21.0 (95%)	34.30
			S.D.	0.007	0.00	0.60	0.018	0.00	0.78	0.019	0.00	1.00
14	0.2800	20	Mean	0.375 (1.338)	19.0 (95%)	21.90	0.423 (1.512)	19.0 (95%)	35.30	0.439 (1.569)	19.0 (95%)	35.70
			S.D.	0.021	0.00	0.54	0.019	0.00	1.00	0.037	0.00	1.10
15	0.0682	85	Mean	0.077 (1.124)	82.0 (96%)	67.80	0.086 (1.254)	78.6 (92%)	86.80	0.084 (1.230)	78.6 (92%)	88.00
			S.D.	0.003	0.00	0.98	0.003	0.49	3.06	0.002	0.49	2.28
16	0.0836	67	Mean	0.125 (1.499)	62.4 (93%)	64.90	0.139 (1.667)	62.1 (93%)	86.00	0.138 (1.647)	62.0 (93%)	87.30
			S.D.	0.004	0.49	0.70	0.006	0.30	1.61	0.003	0.00	2.45
17	0.0853	68	Mean	0.092 (1.047)	64.2 (97%)	126.10	0.098 (1.117)	63.9 (97%)	144.10	0.098 (1.110)	63.5 (96%)	137.10
			S.D.	0.002	0.44	2.15	0.003	0.29	2.78	0.001	0.44	2.27
18	0.0527	110	Mean	0.056 (1.061)	105.6 (96%)	220.20	0.059 (1.113)	103.8 (94%)	241.40	0.059 (1.119)	103.7 (94%)	243.80
			S.D.	0.001	0.46	1.66	0.001	0.40	5.89	0.001	0.46	8.98

Note: the numbers in bold indicate the best value of the four policies, and numbers in the parenthesis denote the ratio of the corresponding value to that of FIFS.

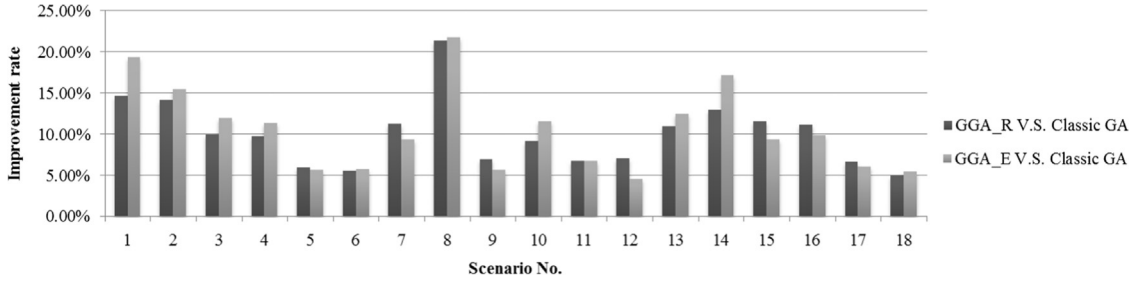

Fig. 6. The relative improvement rates of the GGAs vs. the classic GA.

(2) Waiting: a picker waits before processing the next container. The waiting rate is expressed as 1−(a pick's waiting time in the simulation model/total simulation time).

Table 5 lists the differences among these methods at the 95% confidence level. Except for the waiting rate in Zone 1, the classic GA produces the highest idle and waiting rates in each zone. All of the blocking rates are 0 in Zone 5, because Zone 5 is the final zone in which blocking does not exist. GGA_E and GGA_R strive to balance the workload to reduce blocking; thus, the fulfillment times of the picking line by using either GGA_E or GGA_R are evidently shorter than those obtained using the classic GA. Based on the these results, the GGA significantly outperforms the classic GA in the experiment, and the proposed order batching heuristic algorithm can be used for effectively solving OBPs for pick-and-pass systems.

5. Conclusions

This paper proposes an order batching approach for a pick-and-pass warehousing system to minimize the number of batches and balance the workload of the line as much as possible so as to increase the throughput of the system. Most studies on OBP for

Please cite this article as: Pan JC-H, et al. Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.05.004

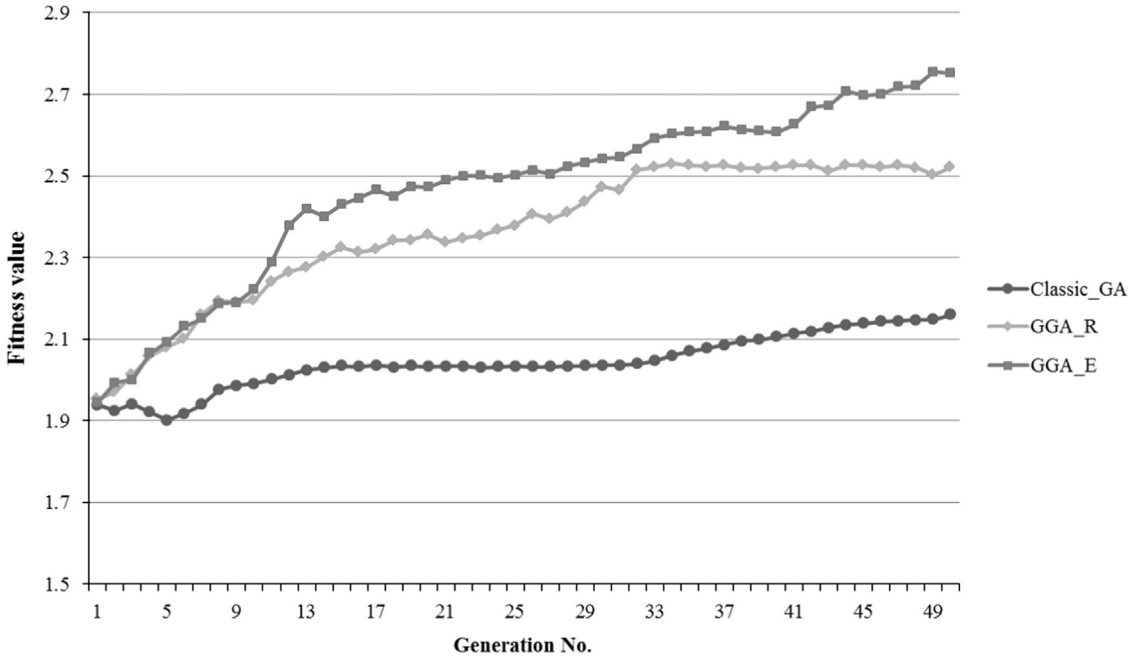

Fig. 7. A performance comparison of Classical GA, GGA_R and GGA_E for Scenario 2 in the experiment.

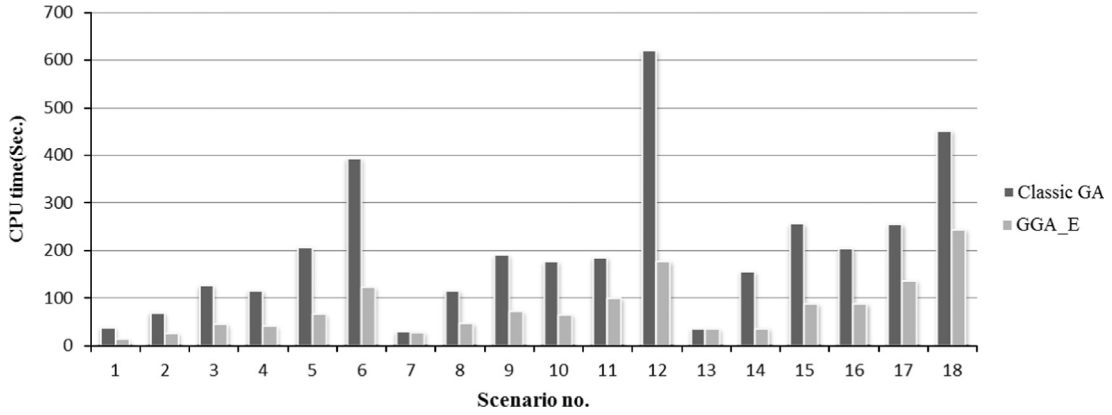

Fig. 8. The comparison of CPU time between GGA_E and classic GA.

Table 5
The blocking rate, waiting rate and fulfill time generated by simulation for the three polices with Scenario 2 at 95% confidence level.

Policy		Zone 1		Zone 2		Zone 3		Zone 4		Zone 5		Overall Picking line
		Blocking	Waiting	Blocking	Waiting	Blocking	Waiting	Blocking	Waiting	Blocking	Waiting	Fulfill time (Unit)
Classic GA	Mean	34.37%	5.94%	29.66%	10.65%	23.98%	16.33%	15.79%	24.51%	0.00%	40.31%	207.800
	S.D.	0.013	0.017	0.010	0.021	0.008	0.018	0.009	0.018	0.000	0.011	3.868
	Confidence interval	[33.4%, 35.3%]	[4.7%, 7.2%]	[28.9%, 30.4%]	[9.1%, 12.2%]	[23.4%, 24.6%]	[15.0%, 17.7%]	[15.1%, 16.5%]	[23.1%, 25.9%]	[00.0%, 00.0%]	[39.5%, 41.1%]	[204.88, 210.72]
GGA_R	Mean	30.84%	7.55%	27.43%	10.26%	22.41%	15.27%	14.08%	23.61%	0.00%	37.68%	197.800
	S.D.	0.03	0.02	0.02	0.02	0.02	0.02	0.01	0.01	0.00	0.01	2.926
	Confidence interval	[28.8%, 32.9%]	[5.9%, 9.2%]	[25.7%, 29.2%]	[8.6%, 11.9%]	[20.9%, 23.9%]	[13.7%, 16.8%]	[13.3%, 14.9%]	[22.5%, 24.7%]	[00.0%, 00.0%]	[37.3%, 38.1%]	[195.59, 200.01]
GGA_E	Mean	27.31%	7.83%	23.71%	10.60%	19.59%	14.72%	13.98%	20.33%	0.00%	34.31%	188.800
	S.D.	0.030	0.018	0.019	0.015	0.012	0.011	0.009	0.011	0.000	0.008	2.227
	Confidence interval	[25.0%, 29.6%]	[6.4%, 9.2%]	[22.3%, 25.2%]	[9.5%, 11.7%]	[18.7%, 20.5%]	[13.9%, 15.6%]	[13.3%, 14.7%]	[19.5%, 21.1%]	[00.0%, 00.0%]	[33.7%, 34.9%]	[187.12, 190.48]

pick-and-pass systems have considered only the minimization of the number of batches formed, or equivalently, the maximization of the number or total size of the items picked; however, without the inclusion of the line balancing consideration among the zones on the line, this criterion alone may be insufficient for improving the throughput of the line since the batches may necessitate

waiting for the items to be picked between zones. The advantage of a picking line balancing can ensure that all orders are fulfilled on time. For the associated grouping problem, the GGA is more suitable compared with the classic GA because the encodings of the GA used are not adapted to the cost function for optimization [7]. Thus, the GGA-based heuristic approach proposed in this paper simultaneously considers the total number of batches formed and line balancing to reduce the order fulfillment time. The results of the numerical experiment indicate that the throughput of the proposed heuristic method is statistically superior to that of the order batching heuristic based on the classic GA for such a pick-and-pass system.

Reference

[1] Bozer YA, Kile JW. Order batching in walk-and-pick order picking systems. International Journal of Production Research 2008;46(7):1887–909.
[2] Brynzer H, Johansson MI. Design and performance of kitting and order picking systems. International Journal of Production Economics 1995;41(1-3):115–25.
[3] Chen MC, Wu HP. An association based clustering approach to order batching considering customer demand patterns. Omega 2005;33(4):333–43.
[4] De Koster MBM, Van Der Poort ES, Wolters M. Efficient order batching methods in warehouses. International Journal of Production Research 1999;37(7):1479–504.
[5] De Koster R, Le-Duc T, Roodbergen KJ. Design and control of warehouse order picking: a literature review. European Journal of Operation Research 2007;182 (2):481–501.
[6] De Koster R. Performance approximation of pick-to-belt order picking systems. European Journal of Operational Research 1994;72(3):558–73.
[7] Falkenauer E. Genetic algorithms and grouping problems. New York: John Wiley & Sons Ltd;; 1998.
[8] Flexsim simulation software user guide 4. Flexsim Software Products, Inc.; 2007.
[9] Frazelle EH, Sharp GP. Correlated assignment strategy can improve any order-picking operation. Industrial Engineering 1989;21(4):33–7.
[10] Gademann AJRM (NOUD), Van Den Berg Jeroen P, Van Der Hoff Hassan H. An order batching algorithm for wave picking in a parallel-aisle warehouse. IIE Transactions 2001;33(5):385–98.
[11] Gagliardi JP, Ruiz A, Renaud J. Space allocation and stock replenishment synchronization in a distribution center. International Journal of Production Economics 2008;115(1):19–27.
[12] Henn S, Schmid V. Metaheuristics for order batching and sequencing in manual order picking systems. Computers and Industrial Engineering 2013;66(2):338–51.
[13] Henn S, Wäscher G. Tabu search heuristics for the order batching problem in manual order picking systems. European Journal of Operational Research 2012;222(3):484–94.
[14] Henn S. Algorithms for on-line order batching in an order picking warehouse. Computers and Operations Research 2012;39(11):2549–63.
[15] Henn S. Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. Flexible Services and Manufacturing Journal 2012:1–29.
[16] Ho YC, Su TS, Shi ZB. Order-batching methods for an order-picking warehouse with two cross aisles. Computers and Industrial Engineering 2008;55 (2):321–47.
[17] Holland JH. Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press; 1975.
[18] Hong S, Johnson AL, Peters BA. Large-scale order batching in parallel-aisle picking systems. IIE Transactions 2012;44(2):88–106.
[19] Hsieh LF, Huang YC. New batch construction heuristic to optimize the performance of order picking system. International Journal of Production Economics 2011;131:618–30.
[20] Hsu CM, Chen KY, Chen MC. Batching orders in warehouses by minimizing travel distance with genetic algorithms. Computers in Industry 2005;56 (2):169–78.
[21] Hwang H, Kim DG. Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system. International Journal of Production Research 2005;43(17):3657–70.
[22] Hwang HC, Chang SY. Order consolidation for batch processing. Journal of Combinatorial Optimization 2005;9(1):121–38.
[23] Jane CC, Laih YW. A clustering algorithm for item assignment in a synchronized zone order picking system. European Journal of Operational Research 2005;166(2):489–96.
[24] Jewkes E, Lee C, Vickson R. Product location, allocation and server home base location for an order picking line with multiple servers. Computers & Operations Research 2004;31(4):623–6.
[25] Lam CHY, Choy KL, Ho GTS, Lee CKM. An order-picking operations system for managing the batching activities in a warehouse. International Journal of Systems Science 2014;45(6):1283–95.
[26] Rachamadugu R, Talbot, B. Improving the equality of workload assignments in assembly lines. International Journal of Production Research 1991;29 (3):619–33.
[27] Maloney D. Seeing the light. Modern Materials Handling 2000;55(9):49–53.
[28] Melacini M, Perotti S, Tumino A. Development of a framework for pick-and-pass order picking system design. International Journal of Advanced Manufacturing Technology 2011;53(9):841–54.
[29] Muppani (Muppant) VR, Adil GK. Efficient formation of storage classes for warehouse storage location assignment: simulated annealing approach. Omega 2008;36(4):609–18.
[30] Pan CH, Liu SY. A comparative study of order batching algorithms. Omega 1995;23(6):691–700.
[31] Pan JCH, Wu MH. A study of storage assignment problem for an order picking line in a pick-and-pass warehousing system. Computer & Industrial Engineering 2009;57(1):261–8.
[32] Parikh PJ, Meller RD. Selecting between batch and zone order picking strategies in a distribution center. Transportation Research Part E 2008;44 (5):696–719.
[33] Petersen CG. Considerations in order picking zone configuration. International Journal of Operations & Production Management 2002;27(7):793–805.
[34] Tho LD, De Koster Rene MBM. Travel time estimation and order batching in a 2-block warehouse. European Journal of Operational Research 2007;176 (1):374–88.
[35] Vroblefski M, Brown EC. A grouping genetic algorithm for registration area planning. Omega 2006;34(3):220–30.
[36] Whitt W. The queuing network analyzer. Bell System Technical Journal 1983;62(9):2779–815.
[37] Won J, Olafsson S. Joint order batching and order picking in warehouse operations. International Journal of Production Research 2005;43(7):1427–42.
[38] Yu M, De Koster R. The impact of order batching and picking area zoning on order picking system performance. European Journal of Operation Research 2009;198(2):480–90.
[39] Elsayed EA. Algorithms for optimal material handling in automatic warehousing systems. International Journal of Production Research 1981;19 (1):525–35.
[40] Gademann N, van de Velde S. Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Transactions 2005;37(1):63–75.

Please cite this article as: Pan JC-H, et al. Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.05.004