IIE Transactions (2012) **44**, 88–106

ISSN: 0740-817X print / 1545-8830 online
DOI: 10.1080/0740817X.2011.588994

Large-scale order batching in parallel-aisle picking systems

SOONDO HONG, ANDREW L. JOHNSON* and BRETT A. PETERS

Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843, USA
E-mail: ajohnson@tamu.edu

Received March 2010 and accepted March 2011

This article discusses an order batching formulation and heuristic solution procedure suitable for a large-scale order picking situation in parallel-aisle picking systems. Order batching can decrease the total travel distance of pickers not only through reducing the number of trips but also by shortening the length of each trip. In practice, some order picking systems retrieve 500–2000 orders per hour and include ten or more aisles. The proposed heuristic produces near-optimal solutions with run times of roughly 70 s in a ten-aisle system. The quality of the solutions is demonstrated by comparing with a lower bound developed as a linear programming relaxation of the batching formulation developed in this article. A simulation study indicates that the proposed heuristic outperforms existing methods described in the literature or used in practice. In addition, the resulting order picking operations are relatively robust to picker blocking.

Keywords: Warehouse, order batching, large-scale optimization

1. Introduction

Distribution Centers (DCs) play a critical role in achieving the mission of the supply chain by helping to absorb market fluctuations and cost variations and by improving customer service. To remain competitive, warehousing and distribution entities are forced to reduce investment and operational cost while satisfying increasing customer demands for smaller and more diverse orders (Napolitano, 2008). DCs face a critical operational issue when retrieving small-sized, broken-case orders. This order picking problem involves determining the set of orders, i.e., a batch, that a picker will retrieve and the route through the facility that the picker will take to pick that batch. The traditional single-order picking mode of operation can require a significant number of trips and result in high operational cost. In contrast, a batch order picking strategy can group orders to reduce the number of order picking trips required and better utilize labor resources to reduce the operational cost. Thus, an efficient order batching algorithm can have a significant impact on operational costs in an order picking environment that requires the retrieval of a large number of small orders. A variety of strategies can be used to batch orders, each leading to a different tour length through the facility. Consequently, it is important to consider the interaction between the order batching and picker routing strategies in order to minimize the total travel distance required to collect a set of orders.

The task of batching orders includes identifying batches and selecting routes. The computational difficulty is mainly due to the combinatorial number of potential batches. The route selection problem is typically computationally easy. The routing problem in rectangular parallel-aisle systems can be optimally solved with polynomial complexity (Ratliff and Rosenthal, 1983). Furthermore, pickers prefer heuristic routing methods (De Koster *et al.*, 1999; Gademann and Van de Velde, 2005), which can be computationally simpler than the optimal routing method. In contrast, the partitioning decision is a primary source of complexity for the batching problem. For example, when the number of orders is 100 and the capacity of the order picker is ten orders per trip, the number of possible combinations for batching the orders is 6.5×10^{85}. Because of this computational burden, only heuristic batching algorithms can solve large-sized problems in a timely manner (De Koster *et al.*, 1999). In addition, the complexity of the batching problem affects the evaluation of the solution quality. The performance of the various proposed methods for batching have not been demonstrated quantitatively in any practical sized problem because lower bound estimates were not previously available.

We are interested in picking systems that process 500–2000 orders in a 1-h time window. This problem size (500–2000 orders; i.e., 1000–4000 items per hour) is typical in the literature. For example, the order picking scenarios reported by Ruben and Jacobs (1999), Petersen (2000), Lieu (2005), and Gong and De Koster (2008) can manage 20 000 to 80 000 items daily (assume an 8-h day). The target

*Corresponding author

picking environment has one-way narrow aisles, and we assume that pickers use traversal routes through the warehouse. A traversal route is a path through the warehouse used by an order picker to collect an order or a batch of orders in which when the picker enters an aisle and the picker travels the full aisle and exits through the opposite end of the aisle from which he or she entered. This is necessary in warehouses with one-way aisle structures and common when 180° turns within an aisle are difficult or time consuming. Throughout most of the article we assume one-way narrow aisles as this is a typical setting for the batch picking problem where congestion is a concern and thus one-way travel is enforced; however, these methods can be extended to multi-directional travel with some increase in computational burden, as will be discussed in Section 6.3. We consider both sort-while-pick and pick-then-sort strategies and both random and class-based storage policies. This study aims to take advantage of the traversal routing method in developing a computationally efficient procedure to solve large-sized problems and determine a tight lower bound to evaluate performance.

We approach the batching problem by selecting an appropriate route, not by constructing a route, and derive a new batching procedure by first assigning orders to routes and then constructing batches within each route set. Even though the routing mechanism occupies a small portion of the computational time, it influences solution approaches for order batching algorithms. The direct assignment of orders to routes can improve the solution quality, reduce the computational time, and obtain a lower bound. Accordingly, we build an efficient heuristic procedure to pack batches from orders within routes. The traditional order batching algorithms build a route for a batch and calculate the route length, which we term a *construction-based routing method*. This route construction concept then guides the search procedure by narrowing order-to-batch assignments to identify batches with potentially shorter routes.

This study makes three contributions. First, we demonstrate a large-scale, near-optimal order batching procedure for parallel-aisle picking systems. The environments cover both narrow-aisle and wide-aisle systems and are extendible to other layouts using traversal routing methods. Second, we introduce a new order batching formulation and relevant relaxation models utilizing a bin packing problem. We solve the bin packing problem more efficiently for large-sized instances compared to a batching problem, even though a complexity analysis categorizes both problems as difficult. Third, the proposed algorithm is compared with available heuristic algorithms in terms of both the total travel distance and the total travel time. A narrow-aisle environment produces picker blocking, and its impact on the order picking throughput can be significant (Gue *et al.*, 2006). Thus, shortest routing distance does not guarantee a shortest retrieval time. We conduct a simulation study to evaluate the performance of the proposed algorithm considering picker blocking.

The remainder of the article is organized as follows. In Section 2, we review related studies regarding order batching algorithms in parallel-aisle picking systems. Sections 3 and 4 provide details about the new formulation and the relaxed models. Section 5 describes a heuristic batching procedure based on the relaxation model. Section 6 discusses the computational experiments and comparison results. We conclude with directions for future research activities.

2. Related literature

This research focuses on proximity batching algorithms that identify orders to be picked together for the purpose of reducing travel distances. The primary objective is to identify orders requiring items stored in close locations within parallel-aisle picking system. Prior work concerning proximity batching algorithms for parallel-aisle picking systems can be categorized into (i) seed heuristics; (ii) saving heuristics; (iii) metaheuristics; and (iv) optimal approaches.

De Koster *et al.* (1999) conducted a comparison study of seed and saving algorithms and concluded that the best seed algorithms combine three control factors: (i) select the seed order as the order that must visit the largest number of aisles; (ii) choose the next order to minimize the number of additional aisles; and (iii) cumulatively update the seed information based on orders in the seed. Alternatively, the same paper developed the saving algorithm, which is a modified Clarke–Wright method (Clarke and Wright, 1964; De Koster *et al.*, 1999), in which a saving list is updated until there is no remaining saving pair. De Koster *et al.* (1999) found the saving algorithm to be preferable to the seed algorithm. Our independent analysis confirms that seed and saving algorithms can analyze large-sized problems. However, the solution quality of these methods is uncertain in medium to large-sized problems, because a provable optimal solution cannot be identified and good lower bound estimates have not previously been reported in the literature.

Hsu *et al.* (2005) proposed a metaheuristic approach using a genetic algorithm to batch orders to minimize the total travel distance. The problem complexity of the genetic algorithm strongly depended on the number of batches, the number of orders, and the number of aisles. They conducted tests on ∼300 orders to generate ∼40 batches in a five-aisle warehouse, and the obtained computational performance was that their algorithm required ∼2500 s on such problems. It is not clear whether the proposed genetic algorithm can solve large-scale problems, because the algorithm appears to be inefficient even for medium-sized problems with low routing complexity.

An optimal approach solves the batching and routing problem exactly through a mixed-integer programming model (Gademann *et al.*, 2001; Gademann and Van de Velde, 2005). Gademann *et al.* (2001) presented a

branch-and-bound solution for a wave picking environment, where a large number of orders were partitioned into multiple batches to minimize the maximum route length. Gademann and Van de Velde (2005) developed a branch-and-price formulation for the sort-while-pick order picking strategy. The authors presented two important findings: (i) the number of aisles and the number of batches significantly impacted the computational time; and (ii) the average time to identify an optimal solution was very short compared to the time necessary to verify its optimality. An epsilon-optimal approach may seem appropriate; however, prior to the work presented in this article, good lower bounds were not available in large-scale problems. Despite enhanced branch-and-price methods, Gademann and Van de Velde (2005) were only able to solve problem sizes of ~30 orders and ~ 8 batches. We infer and confirm with our own experiments that exact methods based on a branch-and-bound approach face a limitation in scalability of the number of orders and batches.

Reviewing the available methods, we identified two critical issues. First, all approaches are implemented to obtain a solution with a partitioning first, routing second method. The route construction procedure is necessary and follows after a partitioning decision because the route length varies according to pick locations in a batch. However, the complex partitioning problem requires the construction of a large number of combinations of order-to-batch assignments. Second, there is no research on lower bound algorithms for a large-scale problem within the batching literature. Heuristic algorithms only demonstrate their improvement relative to random batching strategy or prior batching algorithms. Without a lower bound, we cannot quantify the performance of the heuristics in absolute terms.

3. Route selecting order batching model

3.1. *Problem definition*

We consider an order picking environment similar to those described in Petersen (2000) and Gong and De Koster (2008). The order profile assumes an average order size is two lines per order and 1080 orders arrive per hour. Figure 1 shows a ten-aisle bin-shelving order picking system with a narrow parallel-aisle configuration and two cross-aisles located in the front and back of the layout connecting the parallel aisles. A Loading/Unloading (L/U) station is located in front of the left-most aisle. There are 40 storage locations per aisle in which order pickers retrieve items. The height of the shelves does not impact the travel length, and we assume that it does not impact the pick time. To collect a batch, a picker starts from the L/U station, traverses all necessary aisles taking a one-way traversal route without making U-turns within an aisle, and returns to the L/U station. In other words, pickers pass through an aisle if they enter an aisle. However, they need not

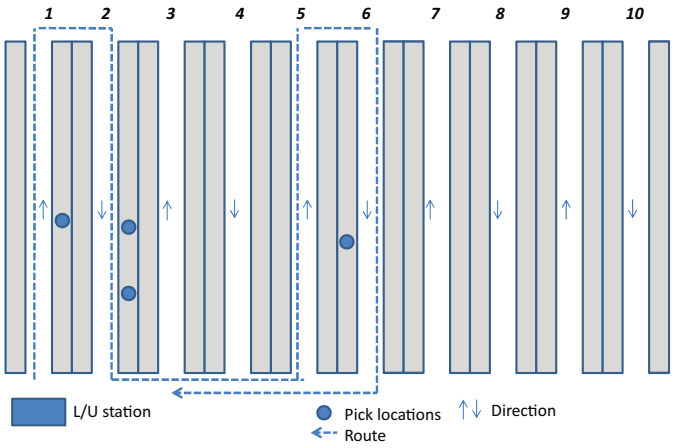

Fig. 1. A ten-aisle order picking system. (Color figure available online.)

traverse every aisle. Furthermore, each aisle should be passed in a fixed direction to prevent pickers from being blocked in an aisle by pickers approaching from the opposite direction; i.e., one-way traversal routing (Gue *et al.*, 2006). We discuss a more common routing method, two-way traversal routing, in Section 6.3. An order picker can carry ten bins to pick ten different orders and has a distance-dependent travel time but a constant walking speed and pick time per item. In determining batches, we ignore blocking delays and minimize total retrieval distance. For more detail on blocking, see the Conclusions and the Appendix.

3.2. *Formulation*

We formulate a new order batching model that takes advantage of the traversal routing method. Traversal routing is a popular heuristic routing method in industry and the academic literature (De Koster *et al.*, 1999; Petersen, 2000; Gue *et al.*, 2006; Gong and De Koster, 2008) and in warehouses with narrow-aisle and/or one-way aisle restrictions. Using traversal routing methods means that all possible routes can be constructed from the warehouse layout. Thus, given a batch, we can select a best-fit route as a matching problem which we term the Route Selecting order Batching (RSB) model.

The formulation is flexible enough to handle both sort-while-pick and pick-then-sort operational strategies. $CAPA$ represents the capacity of the cart. Q_o denotes the capacity of the cart that order o consumes. In the case of sort-while-pick order picking strategy, $CAPA$ is measured in units of orders; thus, Q_o has a value of one. For the case of the pick-then-sort picking strategy, $CAPA$ is measured in units of items; thus, Q_o becomes the number of items in order o. OA_{oa} is the set of aisles that must be visited to gather the items in order o. Route information and length are initially constructed for all routes r in the route set R. Route information is expressed with the aisle incidence

(RA_{ra}) and the route length is LT_r. Given one-way traversal routing of pickers, for pick areas of size $|A| = 2, 4, 6, 8, 10$, and 12, where A is the set of aisles, the size of route set, $|R|$, is 1, 4, 12, 33, 88, and 232, respectively. Though the size of the route set increases exponentially, for reasonable sized problems, e.g., ten aisles, there are only 88 potential routes. We define a set of batches, B, initially $|B| = |O|$ allowing the possibility for each single order to be a separate batch. In a solution, if batch b in B is set to include an order, batch b is active. We formulate the RSB model to determine whether batch b is active, indicated by BV_b, if order o is assigned to batch b indicated by X_{ob}, and to find the route of batch b, indicated by Y_{br}.

Indices and parameters

B, b = the set for batches and its index $b \in B$.
O, o = the set of orders and its index $o \in O$.
A, a = the set of aisles and its index $a \in A = \{1, \ldots, |A|\}$.
R, r = the set of routes and its index $r \in R$.
Q_o = the number of line items in order o.

$$OA_{oa} = \begin{cases} 1 & \text{if order } o \text{ passes through aisle } a \ (= \text{order } o \text{ has at least one pick in aisle } a), \\ 0 & \text{otherwise.} \end{cases}$$

LT_r = the length of route r.

$$RA_{ra} = \begin{cases} 1 & \text{if route } r \text{ passes through aisle } a, \\ 0 & \text{otherwise.} \end{cases}$$

$CAPA$ = the capacity of a cart.

Decision variables

$$X_{ob} = \begin{cases} 1 & \text{if order } o \text{ is assigned to batch } b, \\ 0 & \text{otherwise.} \end{cases}$$

$$Y_{br} = \begin{cases} 1 & \text{if batch } b \text{ takes route } r, \\ 0 & \text{otherwise.} \end{cases}$$

$$BV_b = \begin{cases} 1 & \text{if batch } b \text{ is active,} \\ 0 & \text{otherwise.} \end{cases}$$

Formulation

$$\text{(RSB) min} \quad \sum_{b \in B} \sum_{r \in R} LT_r Y_{br}, \tag{1}$$

Subject to

$$\sum_{b \in B} X_{ob} = 1, \qquad \forall o \in O, \tag{2}$$

$$\sum_{o \in O} Q_o \times X_{ob} \le CAPA, \qquad \forall b \in B, \tag{3}$$

$$X_{ob} \le BV_b, \qquad \forall o \in O, \forall b \in B, \tag{4}$$

$$\sum_{r \in R} Y_{br} \le BV_b, \qquad \forall b \in B, \tag{5}$$

$$X_{ob} \times OA_{oa} \le \sum_{r \in R} RA_{ra} Y_{br}, \quad \forall a \in A, \forall o \in O, \ \forall b \in B, \tag{6}$$

$$X_{ob} = \{0, 1\} \quad \forall o \in O, \forall b \in B,$$

$$Y_{br} = \{0, 1\} \quad \forall b \in B, \forall r \in R,$$

We want to minimize the total travel distance (1). The basic function of the given formulation is to partition orders into batches. An order cannot be separated into multiple batches and all orders should be assigned to batches (2) and a batch should not exceed the capacity constraint of the cart (3). Constraints (4) and (5) are not necessary but are used to calculate the number of batches required. The maximum number of batches is limited to be the number of orders. BV_b is active if at least one order is assigned to batch b (4). A batch must have exactly one route if BV_b is active (5). The aisle incidence vector of route r to which batch b is assigned should contain the aisle incidence vector of orders in batch b (6). Below, we introduce a route-bin packing formulation by focusing on first identifying the routes.

4. Route-bin packing reformulation

This section develops two relaxation models for RSB model, both of which can serve as lower bounds for the model as shown in Fig. 2. The RSB model simplifies the batching problem; however, it still contains partitioning Constraints (2), which have been proven to make the traditional problem NP-complete (Ruben and Jacobs, 1999; Gademann *et al.*, 2001). However, we can postpone the partitioning stage and develop a Route-bin Packing Problem (RPP) to assign orders directly to routes. This reformulation allows us to construct a lower bound, but we will still need additional relaxations to solve large-sized problems.

4.1. *The RPP*

In this section, we simplify RSB by removing the batching variables to develop a new partitioning problem. By skipping the partitioning stage, we relax the batching problem to obtain an assignment of orders to routes and the number of routes required to retrieve orders. Then within route types, we can identify batches similar to a generic bin-packing problem. We term this the RPP. Here, we reuse two decision variables, X_{ob} and Y_{br}, introduced in Section 3.2. Using the following two equations, $x_{or} = \sum_{b \in B} X_{ob} \times Y_{br}$ and $y_r = \sum_{b \in B} Y_{br}$, we define x_{or}, indicating that order o is assigned to route r and y_r is the count of batches taking route r. Note that we can derive these variables from the RSB formulation, and they are also the decision variables in the RPP formulation.

Decision variables

$$x_{or} = \begin{cases} 1 & \text{if order } o \text{ is assigned to route } r, \\ 0 & \text{otherwise.} \end{cases}$$

y_r = the number of batches assigned to route r.

$$\text{(Basic RPP) min} \quad \sum_{r \in R} LT_r \times y_r, \tag{7}$$

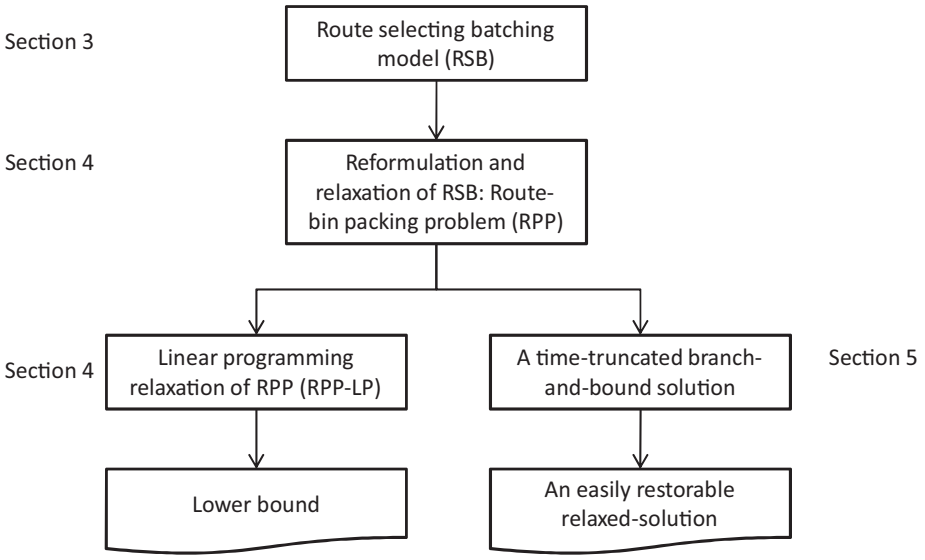

Fig. 2. Relationship among models.

Subject to

$$\sum_{r\in R} x_{or} = 1, \qquad \forall o \in O, \tag{8}$$

$$\sum_{o\in O} Q_o x_{or} \le CAPA \times y_r, \qquad \forall r \in R, \tag{9}$$

$$x_{or} \times OA_{oa} \le RA_{ra} y_r, \qquad \forall o \in O, \forall a \in A, \forall r \in R, \tag{10}$$

$$x_{or} = \{0, 1\} \qquad \forall r \in R, \forall o \in O,$$

$$y_r = \{0, 1, 2, \ldots, \} \qquad \forall r \in R.$$

The objective is to minimize the sum of the length of assigned routes (7). We assign all orders to exactly one route (8). The capacity of the assigned routes r should be greater than or equal to the total quantity of items to be picked (9). The aisle incidence vector of route r should contain the aisle incidence vector of each order o that has been assigned to route r (10).

Based on these two new variables (x_{or} and y_r), we derive three constraints (8), (9), and (10), using Gaussian elimination processes and Lagrangian relaxations from RSB (shown in more detail in Appendix A). We match a constraint specified by order o in Equation (2) to a constraint having the same order o in Equation (8). The Inequalities (9) and (10) also are valid after aggregating the constraints related to route r. Fundamentally, we aggregate Constraints (3) for batches using route r. We can replace batching index b with route index r by aggregating the constraints having the same route r; thus, constraints set (9) has no batch index. Given route r, Constraints (9) determine the number of required routes. We repeat the same process for Constraints (6) to obtain Constraints (10). Constraint set (10) ensures that route r can retrieve order o by comparing the aisle inclusion (indicator) parameters of route r and order o. Finally, we relax Constraints (4) and (5) and the result is RPP without batching variables (see Appendix A for the proof).

The number of constraints in the basic RPP formulation for constraint set (10) is $|O||A||R|$. We simplify it in a pre-processing step as follows:

Step 1. For each r in R, we evaluate whether order o is covered by route r and, if so, include order o in set O_r.

Step 2. Then for o in $O \backslash O_r$, x_{or} is set to zero because route r does not cover order o.

Thus, constraint set (11) is constructed, which has no more than $|O||R|$ constraints. Relaxing Constraint (10) to constant (11) reduces the complexity of the formulation with only a minimal expansion of the solution space. When solving RPP we use the formulation:

$$\text{(RPP) min} \quad \sum_{r\in R} LT_r \times y_r,$$

Subject to Equations (8) and (9),

$$x_{or} = 0, \quad \forall o \in O \backslash O_r, \forall r \in R, \tag{11}$$

4.2. *Linear programming relaxation of RPP*

We derive a lower bound algorithm by relaxing the integer restrictions within RPP. This Linear Programming (LP) relaxation of RPP provides a weak lower bound. To strengthen the lower bound, we add valid inequalities based on the original set of Constraints (10) and implement it by

enforcing y_r to be equal to maximal x_{or} for route r as shown in Equation (12).

$$\text{(RPP-LP)} \min \sum_{r \in R} LT_r \times y_r,$$

Subject to Equations (8), (9), and (11),

$$\begin{aligned} x_{or} &\le y_r, && \forall o \in O_r, \forall r \in R, \\ 0 &\le x_{or} \le 1 && \forall r \in R, \forall o \in O, \\ 0 &\le y_r && \forall r \in R, \end{aligned} \tag{12}$$

Constraints (12) ensure that if any order o is assigned to route r then there is at least one batch within route r.

4.3. *Relationship and optimality*

We can also construct a simple lower bound by assuming that each order uses an optimal route (LT_o) and that each cart is fully loaded during each trip. We define the travel distance under this construction to be the Ideal Batching (IB) bound represented by Obj(*IB*):

$$\text{Obj}(IB) = \sum_{o \in O} 1/CAPA \times LT_o = \sum_{o \in O} LT_o/CAPA.$$

where Obj(*IB*) is equal to or less than Obj(*RPP-LP*) because RPP-LP without Constraints (11) and (12) is the formulation to find the travel distance under ideal batching.

For Obj(*RPP-LP*), Obj(*RPP*), and Obj(*RSB*), the following inequalities hold as a definition of relaxation:

$$\text{Obj}(IB) = \text{Obj}(RPP\text{-}LP) = \text{Obj}(RPP) = \text{Obj}(RSB).$$

The solution to RPP is optimal if Obj(*RPP*) = Obj (*restored batches from RPP solution*) because the upper bound is the same as the lower bound. The solution by RPP-LP is also optimal if the solution by RPP-LP is integral and Obj(*RPP-LP*) is equal to Obj (*restored batches from RPP-LP solution*).

5. A heuristic for route-packing based order batching procedure

This section describes a heuristic solution procedure to solve the batching problem based on the RPP formulation (called the RBP). We prefer the RPP model because batches can easily be constructed from the solution to RPP. However, since RPP is still computationally difficult, we consider two further computational improvements: (i) a partial route set and (ii) a truncated branch-and-bound approach. The proposed heuristic procedure is composed of three steps as described in Fig. 3. Step 1 identifies potential route sets. Step 2 solves the RPP model. Step 3 restores a feasible solution from the infeasible solution generated by the relaxed model. We discuss each step below.

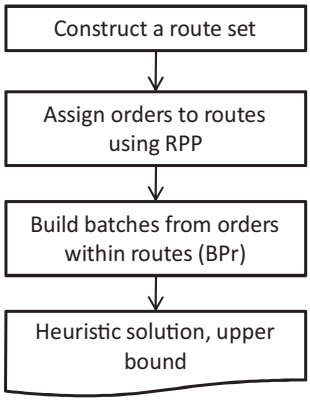

Fig. 3. A flowchart of route packing based order batching procedure.

Step 1. Construct a route set.

Before solving RPP, we must construct a set of candidate routes. Section 3.2 shows that $|R|$ increases exponentially as $|A|$ increases. Consequently, variables and constraints in the RPP formulation, including the route index, increase exponentially. We construct the set of routes in two steps. First, we select an elementary route set (R_e) to guarantee that each order can be picked using one of the routes in the route set. For order o, we check whether R_e has any route feasible for o; if not, we generate a shortest route for o and update $R_e \cup r$. The elementary route set is only part of the reduced route set (R_r) used in RPP.

Second, we will also consider combined route set (R_c) because these routes will be useful when the number of orders assigned to a route do not divide evenly into the batch size. To generate the combined route set, we employ the Clark and Wright II algorithm (CW II; Clarke and Wright, 1964; De Koster *et al.*, 1999). The modified CW II algorithm constructs routes with relatively short travel distances. As part of the CW II algorithm, we specify a composite level, indicating the maximum number of routes covered by a combined route. The composite level is a trade-off; a lower level reduces the number of composite routes, but may deteriorate solution quality. A higher level is necessary when the number of aisles is large or the ratio of the number of elementary routes to the number of orders is large. Our experiments indicate stable performance in terms of both computational time and solution quality when the value is between three and five. The details of the route-set selection procedure follow. Note this procedure significantly reduces the number of routes generated and makes the RPP method useful for larger problem sizes.

Figure 4 illustrates the route construction step. Assume that the number of aisles is six and six orders are given. In this aisle configuration, there are 12 different routes

Incidence vectors of orders	Incidence vectors of Elementary route set	Incidence vectors of Combined route set
$\mathbf{o_1}$: {1,0,0,0,0,0}	$\mathbf{e_1}$: {1,1,0,0,0,0}	$\mathbf{c_1}$: {1,1,1,1,0,0}
$\mathbf{o_2}$: {1,1,0,0,0,0}	$\mathbf{e_2}$: {0,0,1,1,0,0}	
$\mathbf{o_3}$: {0,0,1,1,0,0}	$\mathbf{e_3}$: {1,0,0,1,0,0}	
$\mathbf{o_4}$: {1,0,0,1,0,0}	$\mathbf{e_4}$: {1,1,1,1,0,0}	
$\mathbf{o_5}$: {1,1,0,0,0,0}		
$\mathbf{o_6}$: {1,0,1,1,0,0}		

Fig. 4. An example of elementary route set and combined route set.

Route-set selection procedure
1. Initialize O = all orders, $R_e = \{\}$, $R_c = \{\}$
2. Construct R_e
   For $o = 1$ to $|O|$
     If R_e does not include an optimal route for order o
       Generate route r of o
       $R_e = R_e \cup \{r\}$
     End if
   End for
3. Construct R_c from R_e using a route composition algorithm
     Set the composite level C
     Do
     Calculate the saving s_{ij} for all possible route pairs i, j in $R_e \cup R_c$
     Sort the saving in decreasing order.
     Do
       Select the pair with the non-selected highest saving; in the case of a tie, select a random pair
       If the pair does not violate composite level C Com bine both 'routes' to form a new element r in R_c
     While (remaining pair in the saving or any composite candidate)
   While (all r's in R_e have not been included in R_c)
4. $R_r = R_e \cup R_c$

available. From the orders to be picked, we construct the elementary route set as $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4\}$. For four elementary routes, CW II creates c_1 when the composite level is four. R_r becomes $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4\}$ because c_1 is already a route in R_e.

Step 2. Assign orders to routes using RPP.

This step solves RPP using an Inter Programming (IP) solver with a time-truncated branch-and-bound method. Gademann and Van de Velde (2005) indicated that the branch-and-bound approach to solving the batching formulation converges to a near-optimal solution quickly and that most of the computational time is spent validating the optimality of the solution. Because RPP considers a simpler set of potential routes the computational time will be faster, but we still need to truncate the search with a time-limitation.

Step 3. Build batches from orders within routes.

In this step, BP_r constructs batches with routes using the order-to-route assignment information. After the batches are constructed, we must merge the residual orders into additional batches. The BP_r sub-procedure is solved differently depending on the sortation strategy.

Decision variable

z_b = the number of batches generated from route r.

$$(\mathrm{BP}_r)\ \min \quad \sum_{b \in B_r} z_b, \tag{13}$$

subject to

$$\sum_{b \in B_r} x_{ob} = 1, \qquad \forall o \in O, \tag{14}$$

$$\sum_{o \in O} Q_o x_{ob} \le CAPA \times z_b, \qquad \forall b \in B_r, \tag{15}$$

$$x_{ob} = \{0, 1\} \qquad \forall b \in B_r, \forall o \in O,$$

$$z_b = \{0, 1.\} \qquad \forall b \in B_r.$$

1. *Sort-while-pick strategy.*

In this case, since the size of a batch is based on the number of orders, not items, we can solve BP_r using a greedy algorithm. In other words, Q_o is one and $CAPA$ equals the capacity of the cart in terms of number of orders. We assign orders to batches on a first-come, first-serve basis. When a batch is full, we start a new batch. This greedy procedure obtains an optimal solution to BP_r in the sense that it finds a solution with the minimum number of batches. Figure 5 illustrates a procedure to cluster ten orders into two five-order batches where $y_r = 2$. Then the orders are grouped into batches b_1 and b_2.

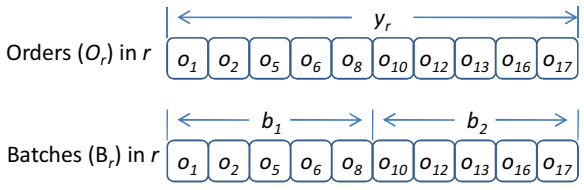

Fig. 5. Batches b_1 and b_2 are constructed by grouping y_r orders assigned to route r. (Color figure available online.)

Note that we can use the routes from the combined route set to handle residual orders from the elementary route sets. Thus, the remaining residual analysis is typically trivial under a sort-while-pick strategy.

2. *Pick-then-sort order picking strategy*.

In the pick-then-sort, strategy, we define the capacity of the cart in terms of items. Furthermore, orders can have multiple items. Thus, assigning orders to batches using a greedy algorithm produces a poor solution. Instead, we solve IP formulation BR_r shown above, which is small and relatively easy to solve, using an IP solver to allocate orders to batches more efficiently while maintaining an item limit on the cart, *CAPA*. We merge any orders remaining into new batches. When there are residual batches of size less than half of *CAPA*, we apply the CW II algorithm to merge these remaining batches.

6. Experimental results

This section reports the computational results and discusses insights from the computational studies. We first test the performance of the proposed heuristic on different problem sizes and number of aisles under the assumption of using one-way traversal routing. We then extend the experiments to the two-way traversal routing method.

6.1. *Experiment design*

In addition to evaluating the performance of the presented algorithm (i.e., RBP: the heuristic route selection–based batching algorithm) by comparing it with a lower bound algorithm, we also compare it with the following batching algorithms.

FCFS: partition orders into batches based on a first-come, first-serve policy.
Seed: the seed algorithm in De Koster *et al.* (1999): (i) update the seed as an order is added to the seed; (ii) select a seed having the largest number of aisles; (iii) choose the order minimizing the number of additional aisles.
CW II: the Clarke and Wright algorithm (II) in De Koster *et al.* (1999). See Appendix B for more detail.

We consider both sort-while-pick and pick-then-sort strategies with *CAPA* of 10 orders and 30 items, respectively. We determine the item locations by the within-aisle class-based storage policy where A:B:C ratio is 0.7:0.2:0.1. Furthermore, class A, B, and C items are stored in aisles 1 to 2, 3 to 4, and 5 to 10, respectively. Next, we compare the random policy and a variant class-based storage policy. The number of orders in an instance is fixed and ranges from a medium scale (360 orders) to a large scale (2180 orders), which we modify from the literature (Ruben and Jacobs, 1999; Petersen, 2000; Gong and De Koster, 2008), and an industrial case study (Lieu, 2005). We consider a ten-aisle picking system (see Appendix C for further extensions). The number of items in an order is also determined by a simple density function where $p(1) = 0.5/0.95$, $p(n) = (1/2)(n-1) - 1/2n)/(0.95)$ when $n = 2, \ldots, 10$, and $p(n) = 0$ otherwise. This order size distribution generates a result similar to the small picking example presented in Frazelle (2002). The average order size is 2.02 items. In addition, we test a variation of the order size = 6. The time to travel the length of one pick-face is one time unit. The time to travel the length and the width of the aisle is 21 and two time units, respectively. The L/U station is located in front of the left-most aisle. To combine routes in the route set selection stage, we set the composite level to three routes. In sum, the experiments consider the following profiles (Table 1). Each experiment repeats for 20 random instances.

Table 1. Experiment profiles and parameters

Profiles	*Parameters*
Algorithms	FCFS, Seed, CW II, RBP
Sorting strategy	Sort-while-pick, pick-then-sort
Storage policy	Class 1, Class 2, Random
The number of orders	360, 720, 1080, 1440, 1800, 2160
The number of aisles	10, 20, 30, 40
Order size	Default, Uniform (3, 9)

In discussing the performance of the algorithms, we use the following notations throughout this section:

LB: linear relaxation of RPP (RPP-LP).
IB: ideal batching model.
Obj: objective value of an algorithm.
ObjL: objective value of RPP; L stands for a lower bound.
ObjU: objective value of restored solution of RPP; U stands for an upper bound.
CPU: computational time in seconds.
LU gap: gap between an objective function value and the RPP-LP objective function value expressed as a percentage (= (an objective function value − LB)/(LB) × 100%).

We implement the analysis with the mixed-integer programming formulations developed above using the ILOG CPLEX Callable Library C API 11.0.4. The data set generator and comparison algorithms are developed using the C language. To test the computational performance, we ran executable files on a Windows NT-based server system with Windows Vista (Xeon 2.66 Ghz CPU, 12 GB memory). While compiling the CPLEX source, we used the stand-alone dynamic-linked library. We disabled both the branch-and-cut option and the heuristic search option to evaluate the exact computational time. While solving RPP and BP_r, we set the time limit to 60 s (i.e., the truncated branch-and-bound method). Instead of the optimal solutions, we evaluated solutions of the RBP by comparing with

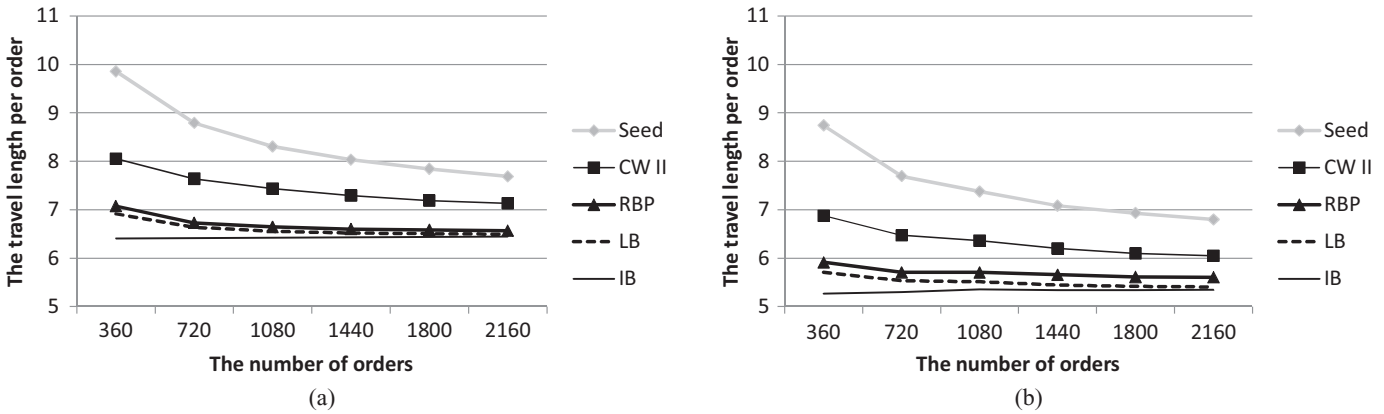

Fig. 6. The average travel length per order with the one-way traversal routing method: (a) sort-while-pick strategy and (b) pick-then-sort strategy.

their LP lower bound generated with a full route set. Note that RPP-LP does not require the time limit and that BP_r is only applicable for the pick-then-sort strategy.

6.2. *Experimental results*

6.2.1. *Computational time and the total travel distance*

The performance of the proposed RBP method is compared to FCFS, seed, CWII, and the LB to understand the relative performance. Because these problems are computationally difficult, we calculated the total travel distance, the run time, and the percentage deviation from the lower bound as shown in Table 2. The RBP produces near-optimal solutions within about 2 min and outperforms the Seed and CW II algorithms. Moreover, the RBP improvement over alternative methods is larger for scenarios in which the number of orders is smaller.

Specifically, in the sort-while-pick strategy, the Seed algorithm requires a run time of 0.2 s. However, the Seed algorithm's LU gap is between 15.50 and 29.87%. CW II has a shorter total travel distance but requires a longer computational time, which is also noted by De Koster *et al.* (1999). As the problem size increases, its computational time increases exponentially. When the number of orders is 2160, it takes on average 137.30 s. RBP demonstrates a considerable improvement in travel distance. The LU gap ranges from 1.07 to 2.26% when the computational time is limited to 60 s, whereas the best approach identified in De Koster *et al.* (1999), CW II, showed a gap ranging from 8.96 to 14.14%.

The computational results for the pick-then-sort strategy show similar results. The LU gap of RBP in a pick-then-sort strategy is larger than the sort-while-pick strategy. The increase in the gap stems partly from the LB estimation and is not only the result of RBP's performance. Inevitably, RBP produces some batches that are not filled to capacity because of irregular order sizes. Thus, the solution quality by RBP deteriorates.

While the computational time of RBP and CW II is almost equal under the sort-while-pick strategy, the run time of RBP increases under the pick-then-sort strategy because the batch packing stage is computationally intensive using the IP bin-packing algorithm. However, the run times are still shorter than 150 s for all cases. Note that for both RPP and BP_r the time limit for the branch-and-bound procedure is 60 s and there are multiple iterations of BP_r.

The Seed and CWII algorithms depend on having a large number of orders to improve performance. When the number of orders is relatively small (360 or 720), the Seed and CW II algorithms experience a large LU gap. Thus, the benefits of RBP are significant for large-sized problems but are even more prominent when the number of orders is small.

6.2.2. *The average travel length per order*

The average travel length per order is another metric we can use to evaluate the performance of various batching methods, assuming that all orders construct similar numbers of batches. With this objective, a large-sized batching problem is preferred since larger problems can produce more efficient batches and thus reduce trip distance. The previous methods developed for batching demonstrate a significant improvement in average travel length per order as shown in Fig. 6. The improvement declines as the number of orders increases. When the number of orders increases from 1800 to 2160, there are minimal gains in throughput of the order picking system. In all cases, RBP dominates the other heuristics in solution quality with very small gaps compared to both LB and IB.

Appendix C summarizes the other experimental results. RBP demonstrates consistent performance over order picking profiles, varying number of aisles and alternative storage policies.

6.2.3. *Overall results*

We analyzed the experimental results using Analysis of Variance (ANOVA) to detect the significance of the order

Table 2. Computational results over different algorithms

Sort strategy	*Number of orders*	*FCFS*		*Seed*			*CW II*			*RBP*				*LB*		*IB*
		Obj	*LU gap (%)*	*Obj*	*CPU*	*LU gap (%)*	*Obj*	*CPU*	*LU gap (%)*	*ObjL*	*ObjU*	*CPU*	*LU gap (%)*	*Obj*	*CPU*	*Obj*
Sort-while-pick	360	5923	58.0	3549	0.0	29.9	2899	0.4	14.1	2547	2547	11.5	2.3	2489	0.8	2306
	720	11 893	59.8	6332	0.0	24.5	5502	5.0	13.1	4845	4845	40.3	1.3	4780	1.8	4616
	1080	17 915	60.5	8970	0.1	21.1	8033	16.2	11.9	7177	7177	56.9	1.3	7081	2.7	6939
	1440	23 961	60.8	11 573	0.1	18.9	10 505	39.1	10.6	9505	9505	60.3	1.2	9388	3.6	9256
	1800	29 990	61.0	14 123	0.1	17.1	12 943	75.7	9.5	11 849	11 849	60.3	1.2	11 710	4.6	11 587
	2160	36 034	61.1	16 606	0.2	15.5	15 412	137.3	9.0	14 183	14 183	60.4	1.1	14 032	5.7	13 916
Pick-then-sort	360	4646	55.7	3147	0.0	34.7	2477	0.5	17.0	2129	2129	17.5	3.4	2056	4.9	1897
	720	9343	57.4	5539	0.0	28.1	4659	4.8	14.5	4108	4108	67.1	3.0	3983	12.0	3814
	1080	14 127	57.8	7968	0.1	25.3	6869	14.7	13.3	6137	6161	75.3	3.3	5955	12.9	5783
	1440	18 832	58.3	10 199	0.1	23.1	8927	33.7	12.1	8076	8145	96.5	3.7	7844	18.1	7690
	1800	23 523	58.5	12 477	0.1	21.9	10 980	62.2	11.2	10 025	10 101	105.0	3.5	9750	22.8	9615
	2160	28 258	58.7	14 684	0.2	20.5	13 065	104.1	10.7	12 002	12 109	140.5	3.6	11 672	27.7	11 551

Table 3. ANOVA results

	Degree of freedom	*Sum of squares*	*Mean of squares*	*F*	*Pr(>F)*	*Significance*
Algorithms	2	999 780 000	499 890 000	53.1	0.000	0.001
Sorting strategy	2	58 200 000 000	29 100 000 000	3093.8	0.000	0.001
Storage policy	2	79 006 000 000	39 503 000 000	4199.8	0.000	0.001
Number of orders	5	72 401 000 000	14 480 000 000	1539.5	0.000	0.001
Number of aisles	3	70 209 000 000	23 403 000 000	2488.1	0.000	0.001
Order size	1	9995 900 000	9995 900 000	1062.7	0.000	0.001
Residuals	2264	21 295 000 000	9406 000			

profile characteristics and the relevant parameters using the software R version 2.11.1. In the one-way ANOVA test, we excluded FCFS and focused on the relationship between non-random algorithms. The results in Table 3 indicate that all six profiles have a statistical significance of less than 0.001. For the algorithm profile, we additionally conducted a pairwise comparison using a standard t test statistic over the null hypotheses: (i) H_0: RBP $\geq$ CW II and (ii) H_0: RBP $\geq$ Seed. The type-I error rates of multiple testing are ≤ 0.021 and $\leq 6.4 \times 10^{-12}$, respectively, using a pooled standard deviation P-value adjustment method, the Holm–Bonferroni method (Holm, 1979). Our results confirm that the RBS outperforms both the CW II and Seed algorithms but that CW II is more competitive.

6.3. *Application in wide-aisle picking systems*

We can extend the proposed framework in this article to apply to operations with two-way wide-aisle pick areas. Industry uses the wide-aisle picking system to reduce picker blocking or to accommodate storage/retrieval vehicles. The two-way traversal routing commonly appears in the literature (Ruben and Jacobs, 1999; Petersen, 2000; Gong and De Koster, 2008).

6.3.1. *Two-way traversal routing method*

For this case, pickers have greater flexibility in route selection. Consider constructing an extended route set R based on a two-way traversal routing method. The number of unique routes required grows quickly in the number of aisles. For example, for $|A| = 2, 4, 6, 8, 10, 12$, the corresponding number of routes is 1, 7, 31, 127, 511, 2047. The number of routes for any even value of $|A|$ is calculated as:

$$L(A) = {}_{|A|}C_2 + {}_{|A|}C_4 + {}_{|A|}C_6 + .. + {}_{|A|}C_{|A|}, \quad \text{where } |A| = 2, 4, \ldots, 12$$

6.3.2. *Computational results*

In Table 4, the previous four methods for batching are used in two-way traversal routing situations. Furthermore, Figure 7 compares the average travel length per order in ten-aisle picking system. The impact of optimally batching is more significant as the routing methods become more complex. RBP benefits are even more pronounced for the two-way traversal routing method. The RBP route set includes a smaller proportion of the total number of possible routes to attempt to balance performance with computation time. This is the primary source of the deterioration of the performance for both RBP and the lower bound estimates.

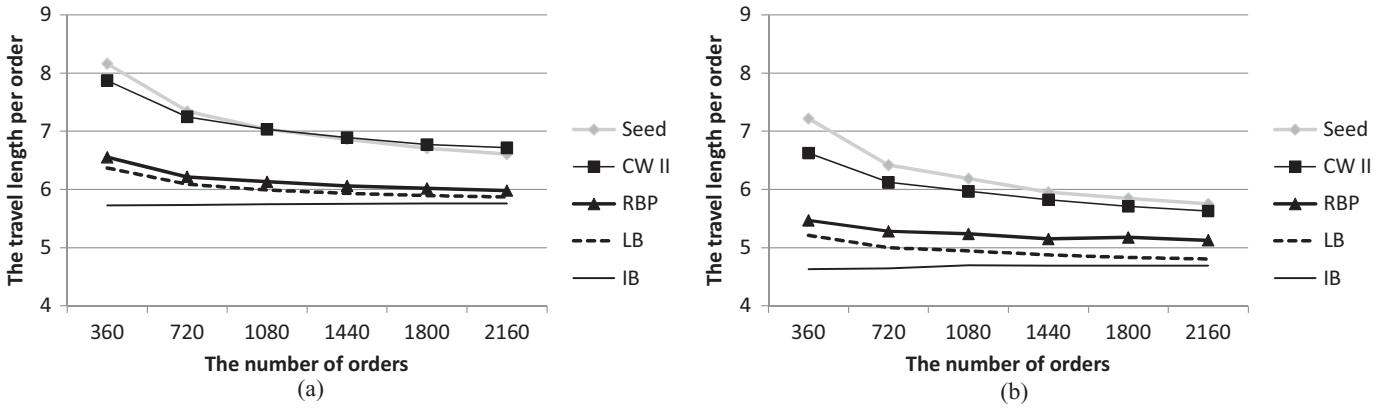

Fig. 7. The average travel length per order with the two-way traversal routing method: (a) sort-while-pick strategy and (b) pick-then-sort strategy.

Table 4. Computational results with the two-way traversal routing method in the ten-aisle picking system

Sort strategy	*Number of orders*	*FCFS*		*Seed*			*CW II*			*RBP*				*LB*		*IB*
		Obj	*LU gap (%)*	*Obj*	*CPU*	*LU gap (%)*	*Obj*	*CPU*	*LU gap (%)*	*ObjL*	*ObjU*	*CPU*	*LU gap (%)*	*Obj*	*CPU*	*Obj*
Sort-while-pick	360	5385	57.4	2939	0.0	22.0	2833	0.4	19.1	2360	2360	30.5	2.8	2293	64.8	2063
	720	10 808	59.4	5287	0.0	17.1	5219	4.2	16.0	4477	4477	60.2	2.0	4385	120.0	4129
	1080	16 242	60.2	7597	0.0	14.8	7598	13.6	14.8	6622	6622	60.3	2.3	6470	185.3	6207
	1440	21 717	60.7	9884	0.1	13.6	9923	31.4	13.9	8729	8729	60.4	2.1	8544	258.8	8286
	1800	27 203	61.0	12 077	0.1	12.1	12 187	63.9	12.9	10 834	10 834	60.6	2.0	10 614	422.9	10 365
	2160	32 726	61.3	14 274	0.2	11.2	14 506	111.4	12.6	12 925	12 925	60.8	1.9	12 680	429.9	12 444
Pick-then-sort	360	4244	55.8	2598	0.0	27.8	2386	0.5	21.4	1969	1969	50.7	4.7	1876	1267.5	1666
	720	8489	57.6	4622	0.0	22.1	4408	4.9	18.3	3803	3803	60.8	5.4	3599	6833.2	3343
	1080	12 837	58.4	6682	0.1	20.0	6446	17.5	17.1	5654	5654	64.6	5.5	5342	13 546.4	5070
	1440	17 132	59.0	8577	0.1	18.1	8385	42.8	16.2	7400	7417	79.9	5.3	7023	19 910.8	6752
	1800	21 426	59.4	10 528	0.1	17.4	10 283	85.3	15.4	9256	9314	98.8	6.7	8695	16 521.8	8436
	2160	25 744	59.7	12 424	0.2	16.4	12 169	146.7	14.7	11 040	11 073	127.1	6.2	10 382	24 644.2	10 137

7. Conclusions

This article introduces a RSB, its bound model (RPP-LP), and a heuristic solution procedure (RBP) to solve large-scale order batching problems. The special structure of RSB is exploited in developing the formulations and the solution procedure. We found that RBP produces near-optimal solutions in a narrow-aisle order picking system, where the number of aisles is ten and the number of orders is 2180. The computational time required was 60–80 s on average with a maximum of 140 s. We found that the methods continue to perform well in larger problems. The solution quality was demonstrated by comparing with a tight lower bound developed from the proposed model.

Determining the number of orders in a batching cycle is important, because a larger number of orders may lead to a shorter travel distance. However, a large number of orders will not always be beneficial since the order picking operation becomes less responsive. The time saved in picking is less than that spent waiting to gather orders to form better batches; thus, the lead time to the customer increases. Regardless of the method of creating batches, the performance of the LB describes the minimal travel distance per order. When the number of orders is 2160, the gap between LB and IB is less than 1% for one-way traversal routing and 2.5% for two-way traversal routes as shown in Fig. 6, Table 2, Fig. 7, and Table 4, showing that IB becomes tighter as the number of orders increases. Furthermore, the IB per order does not reduce when the number of orders is greater than 1440. This observation indicates that a near-optimal batching solution to the 50 000-order batching problem might produce minimal gains compared to the 2160-order batching problem with respect to the average travel length per order. Some order picking scenarios reported in the literature (Ruben and Jacobs, 1999; Petersen, 2000; Lieu, 2005; Gong and De Koster, 2008) manage 20 000 to 80 000 items daily. Gathering orders for a day would increase the lead time to the customer by a day, and in industries with customer service requirements the savings in order picking do not justify this increase in lead time.

In addition, the procedure developed in this article contributes to efficient and effective DC design and operation, where both space utilization and operational throughput are critical. A narrow-aisle picking area in a DC is advantageous in terms of space utilization, but it produces more picker blocking (Gue *et al.*, 2006; Napolitano and Gross & Associates, 2003). In narrow-aisle picking systems, the shorter travel length does not guarantee a shorter retrieval time due to picker blocking (Gue *et al.*, 2006). Thus, we conducted a simulation study to quantify the effect on picker blocking on the various batching algorithms reported in Appendix D. We find that RBP is relatively robust to picker blocking, while Seed and CW II produce very poor results under heavy congestion. These results emphasize the need to select a batching algorithm that reduces travel distance and does not create excessive picker blocking. Solutions by RBP shorten the total travel distance to near-optimal solutions and are robust to picker blocking.

The proposed RBP batching method could easily be added to a standard warehouse management software package. This would be a powerful tool for the warehouse manager to improve operational performance. However, we note that integrating this batching approach with slotting or warehouse design algorithms provides added benefits in warehouse performance. Because the algorithm can handle even very large sets of orders, the manager has the flexibility to set the wave size to optimize other performance criteria, where wave picking is a method of dividing and staggering order releases with the purpose to minimize the maximum lead time of any order (Gademann *et al.*, 2001). The analysis described in this article could be performed once for each wave.

A variety of direct extensions of RBP are also possible. We showed that the RBP framework can be extended to wide-aisle picking systems where pickers use two-way traversal routes. A multiple cross-aisle system (Roodbergen and De Koster, 2001) and a two-block warehouse (Le-Duc and De Koster, 2007) can also be modeled using RBP. In these systems, it is possible to enumerate available or preferred routes (R) and to define matching relationships between routes and orders (O_r) for general situations. In fact, whenever a warehouse manager can construct a preferred route set (R), RBP can solve the problem for general warehouse designs with only slight modifications.

Future research should extend this work to more routing methods and to explicitly account for picker blocking. RBP is a key enabler when developing an efficient batching algorithm with different routing methods (Section 6.3). Our experimental results also show that using the RBP method for batching can significantly reduce picker blocking. However, productivity loss by picker blocking remains a concern. Explicitly modeling and controlling picker blocking when constructing batches and routing pickers offers potential for important improvements by reducing total retrieval time in order-picking operations.

References

Clarke, G. and Wright, J.W. (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, **12**, 568–581.

De Koster, R., Van der Poort, E.S. and Wolters, M. (1999) Efficient order-batching methods in warehouses. *International Journal of Production Research*, **37**(7), 1479–1504.

Frazelle, E. (2002) *World-Class Warehousing and Material Handling*, McGraw-Hill, New York, NY.

Gademann, N. and Van de Velde, S. (2005) Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, **37**(1), 63–75.

Gademann, N., Van den Berg, J. and Van der Hoff, H. (2001) An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE Transactions*, **33**(5), 385–398.

Gong, Y. and De Koster, R. (2008) A polling-based dynamic order picking system for online retailers. *IIE Transactions*, **40**(11), 1070–1082.

Gue, K.R., Meller, R.D. and Skufca, J.D. (2006) The effects of pick density on order picking areas with narrow aisles. *IIE Transactions*, **38**(10), 859–868.

Holm, S. (1979) A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, **6**(2), 65–70.

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.

Le-Duc, T. and De Koster, R. (2007) Travel time estimation and order batching in a 2-block warehouse. *European Journal of Operational Research*, **176**(1), 374–388.

Lieu, C.C.A. (2005) Impact of inventory storage and retrieval schemes on productivity. M.B.A. and M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Napolitano, M. (2008) Sitting tight—2008 warehouse/DC operations survey results. *Logistics Management*, **47**(11), 47–50.

Napolitano, M. and Gross & Associates. (2003) *The Time, Space and Cost Guide to Better Warehouse Design*, The Distribution Group, Ogden, UT.

Petersen, C.G. (2000) An evaluation of order picking policies for mail order companies. *Production and Operations Management*, **9**(4), 319–335.

Ratliff, H.D. and Rosenthal, A.S. (1983) Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, **31**(3), 507–521.

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.

Ruben, R.A. and Jacobs, F.R. (1999) Batch construction heuristics and storage assignment strategies for walk/ride and pick systems. *Management Science*, **45**(4), 575–596.

Appendix A: Formulation of basic RPP from RSB

The basic RPP can be derived from RSB. In particular, each constraint in the basic RPP is derived from a constraint of RSB, or becomes a constraint aggregating relevant constraints in RSB.

1. Objective function.

$$\sum_{b\in B}\sum_{r\in R} LT_r\, Y_{br} = \sum_{r\in R} LT_r \sum_{b\in B} Y_{br}$$

By definition, $\sum_{b\in B} Y_{br} = y_r$.

$$= \sum_{r\in R} LT_r \times y_r.$$

2. Constraints (8).

From Equation (2):

$$\sum_{b\in B} X_{ob} = 1, \qquad o \in O,$$
$$X_{ob_1} + X_{ob_2} + \ldots + X_{ob_{|b|}} = 1, \qquad o \in O,$$

Since $\sum_{r\in R} Y_{br} = 1$.

$$X_{ob_1} \times \sum_{r\in R} Y_{b_1 r} + X_{ob_2} \times \sum_{r\in R} Y_{b_2 r} + \ldots + X_{ob_{|b|}} \times \sum_{r\in R} Y_{b_{|b|} r} = 1, \qquad o \in O,$$
$$\sum_{r\in R} X_{ob_1} \times Y_{b_1 r} + \sum_{r\in R} X_{ob_2} \times Y_{b_2 r} + \ldots + \sum_{r\in R} X_{ob_{|b|}} \times Y_{b_{|b|} r} = 1, \qquad o \in O,$$
$$\sum_{r\in R} (X_{ob_1} \times Y_{b_1 r} + X_{ob_2} \times Y_{b_2 r} + \ldots + X_{ob_{|b|}} \times Y_{b_{|b|} r}) = 1, \qquad o \in O,$$

By definition, $\sum_{b\in B} X_{ob} \times Y_{br} = x_{or}$.

$$\sum_{r\in R}\sum_{b\in B} (X_{ob} \times Y_{br}) = 1, \qquad o \in O,$$
$$\sum_{r\in R} x_{or} = 1 \qquad o \in O.$$

3. Constraints (9).

From Equation (3): Assume that all of the b have at least one order.

$$\begin{cases} \sum\limits_{o\in O} Q_o \times X_{ob} \le CAPA, & b \in B_1 = \left\{b \mid Y_{br_1} = 1, b \in B\right\}, \\ \sum\limits_{o\in O} Q_o \times X_{ob} \le CAPA, & b \in B_2 = \left\{b \mid Y_{br_2} = 1, b \in B\right\}, \\ \quad \vdots & \\ \sum\limits_{o\in O} Q_o \times X_{ob} \le CAPA, & b \in B_{|R|} = \left\{b \mid Y_{br_{|R|}} = 1, b \in B\right\} \end{cases}$$

By definition, $Y_{b_r r} = 1$.

$$\begin{cases} \sum\limits_{o\in O} Q_o \times X_{ob} \times Y_{br_1} \le CAPA \times Y_{br_1}, & b \in B_1, \\ \sum\limits_{o\in O} Q_o \times X_{ob} \times Y_{br_2} \le CAPA \times Y_{br_2}, & b \in B_2, \\ \quad \vdots & \\ \sum\limits_{o\in O} Q_o \times X_{ob} \times Y_{br_{|R|}} \le CAPA \times Y_{br_{|R|}}, & b \in B_{|R|} \end{cases}$$

Aggregate constraints indexed by r. The new constraints become weaker; thus, the new model becomes a relaxation of the original constraints.

$$\begin{cases} \sum\limits_{b\in B_1} \sum\limits_{o\in O} Q_o \times X_{ob} \times Y_{br_1} \le \sum\limits_{b\in B_1} CAPA \times Y_{br_1}, \\ \sum\limits_{b\in B_2} \sum\limits_{o\in O} Q_o \times X_{ob} \times Y_{br_2} \le \sum\limits_{b\in B_2} CAPA \times Y_{br_2}, \\ \quad \vdots \\ \sum\limits_{b\in B_{|R|}} \sum\limits_{o\in O} Q_o \times X_{ob} \times Y_{br_{|R|}} \le \sum\limits_{b\in B_{|R|}} CAPA \times Y_{br_{|R|}}, \end{cases}$$

$$\sum_{o\in O} Q_o \times \sum_{b\in B_r} X_{ob} \times Y_{br} \le CAPA \times \sum_{b\in B_r} Y_{br}, \qquad r \in R$$

By definition, $\sum\limits_{b\in B} X_{ob} \times Y_{br} = x_{or}$ and $\sum\limits_{b\in B} Y_{br} = y_r$.

$$\sum_{o\in O} Q_o \times x_{or} \le CAPA \times y_r, \qquad r \in R$$

4. Constraints (10).

From Equation (6): Assume that all b have at least one order.

$$\begin{cases} X_{ob} \times OA_{oa} \le \sum\limits_{r\in R} RA_{ra} Y_{br}, & b \in B_1 = \left\{b \mid Y_{br_1} = 1, b \in B\right\}, a \in A, \\ X_{ob} \times OA_{oa} \le \sum\limits_{r\in R} RA_{ra} Y_{br}, & b \in B_2 = \left\{b \mid Y_{br_2} = 1, b \in B\right\}, a \in A, \\ \quad \vdots & \\ X_{ob} \times OA_{oa} \le \sum\limits_{r\in R} RA_{ra} Y_{br}, & b \in B_{|R|} = \left\{b \mid Y_{br_{|R|}} = 1, b \in B\right\}, a \in A, \end{cases}$$

By definition, $Y_{b_r r} = 1$.

$$\begin{cases} X_{ob} \times OA_{oa} \le RA_{r_1 a}, & b \in B_1, a \in A, \\ X_{ob} \times OA_{oa} \le RA_{r_2 a}, & b \in B_2, a \in A, \\ \quad \vdots & \\ X_{ob} \times OA_{oa} \le RA_{r_{|R|} a}, & b \in B_{|R|}, a \in A, \end{cases}$$

Since $Y_{br} \ge 0$, inequalities hold.

$$\begin{cases} X_{ob} \times OA_{oa} \times Y_{br_1} \le RA_{r_1 a} \times Y_{br_1}, & b \in B_1, a \in A, \\ X_{ob} \times OA_{oa} \times Y_{br_2} \le RA_{r_2 a} \times Y_{br_2}, & b \in B_2, a \in A, \\ \quad \vdots & \\ X_{ob} \times OA_{oa} \times Y_{br_{|R|}} \le RA_{r_{|R|} a} \times Y_{br_{|R|}}, & b \in B_{|R|}, a \in A, \end{cases}$$

Aggregate constraints indexed by r. The new constraints become weaker; thus, the new model becomes a relaxation of the original constraints.

$$\begin{cases} \sum\limits_{b \in B_1} X_{ob} \times OA_{oa} \times Y_{br_1} \le \sum\limits_{b \in B_1} RA_{r_1 a} \times Y_{br_1}, & a \in A, \\ \sum\limits_{b \in B_2} X_{ob} \times OA_{oa} \times Y_{br_2} \le \sum\limits_{b \in B_2} RA_{r_2 a} \times Y_{br_2}, & a \in A, \\ \quad \vdots & \\ \sum\limits_{b \in B_{|R|}} X_{ob} \times OA_{oa} \times Y_{br_{|R|}} \le \sum\limits_{b \in B_{|R|}} RA_{r_{|R|} a} \times Y_{br_{|R|}}, & a \in A, \end{cases}$$

$$\begin{cases} OA_{oa} \times \sum\limits_{b \in B_1} X_{ob} \times Y_{br_1} \le RA_{r_1 a} \times \sum\limits_{b \in B_1} Y_{br_1}, & a \in A, \\ OA_{oa} \times \sum\limits_{b \in B_2} X_{ob} \times Y_{br_2} \le RA_{r_2 a} \times \sum\limits_{b \in B_2} Y_{br_2}, & a \in A, \\ \quad \vdots & \\ OA_{oa} \times \sum\limits_{b \in B_{|R|}} X_{ob} \times Y_{br_{|R|}} \le RA_{r_{|R|} a} \times \sum\limits_{b \in B_{|R|}} Y_{br_{|R|}}, & a \in A, \end{cases}$$

$$OA_{oa} \times \sum_{b \in B_r} X_{ob} \times Y_{br} \le RA_{ra} \times \sum_{b \in B_r} Y_{br}, \qquad r \in R, a \in A,$$

By definition, $\sum\limits_{b \in B} X_{ob} \times Y_{br} = x_{or}$ and $\sum\limits_{b \in B} Y_{br} = y_r$.

$$OA_{oa} \times y_{or} \le RA_{ra} \times y_r, \qquad r \in R, a \in A.$$

Appendix B: Clarke and Wright II algorithm

Step 1. Obtain the distance saving s_{ij} for all possible order pairs i, j when two orders are grouped, given the capacity of the pick device.
Step 2. Sort the savings in decreasing order.
Step 3. Select the pair with the highest savings. In the case of a tie, select a random pair.
Step 4. Combine both orders to form a new cluster, if allowed by the pickers' capacity. If not, choose the next combination on the list and repeat Step 4.
Step 5. If not all order combinations have been include in a route, proceed with Step 1. In the calculation, all clusters are considered as orders. Otherwise, finish.

Appendix C: Computational performance over other order picking profiles

1. *The number of aisles.*

Table A1 compares the CW II and RBP over different numbers of aisles. The cardinality of the route set strongly dependent on the number of aisles. RPP-LP can only solve ~14-aisle or smaller instances. Thus, Table A1 does not include LB results and LU gaps. Instead, we use the following comparison:

RBP/CW: the ratio of ObjU to the objective function value of CW II. This measure is used where a lower bound is impossible.

RBP still dominated CW II in RBP/CW, but RBP required a long computational time as the number of aisles increased. Note that we set the composite level to five when the number of aisles = 40.

The route reduction step is not effective in the 40-aisle instance. As the number of routes increased, we modulated the truncation time limit to produce good solutions; specifically, 120 s, 180 s, and 240 s were allowed for 20-aisle, 30-aisle, and 40-aisle instances. However, despite this increase in the truncation limit, RBP's performance suffered loss in the objective values. Figure A1 illustrates the variations of the average travel length over different algorithms with respect to the number of aisles. The performance gap between CW II and RBP has not been widened as shown in Fig. A1 when the number of aisles is 40.

2. *Storage policy.*

Table A2 includes the test results with different storage policies. Picking systems can operate under different storage patterns or storage policies. As orders were scattered more evenly, all algorithms had longer travel distances. In particular, the computational time of RBP lengthens. The storage policy has an impact on the route set of RBP. An increase in the number of uniformly stored items produces more elementary routes. Thus, the elementary route set becomes larger, and the number of combined routes also increases. A larger route set results in longer computational time.

3. *Order size = Uniform[3,9].*

Table A3 includes the test results when the order size is determined using a uniform distribution, Uniform [min, max] = Uniform [3, 9]. Other profiles are the same as the narrow-aisle sort-while pick strategy when a regular class-based storage policy is employed. A similar result has been observed.

Appendix D. Impacts on picker blocking in narrow-aisle configuration

In narrow-aisle picking systems, the shorter travel length does not guarantee a shorter retrieval time due to picker blocking (Gue *et al.*, 2006). Thus, we conduct a simulation study to quantify the effect on picker blocking on the various batching algorithms. Two situations were considered: light congestion and heavy congestion. A light congestion environment is defined as the number of orders in a time window = 1080 orders; four time windows; pick : walk time ratio = 5:1; five pickers; setup time per batch = 120; and cart capacity = ten orders or 20 items. A heavy congestion environment was defined as above except: pick : walk time

Table A1. The experimental results with the variation of the number of aisles

		CW II		*RBP*				
Number of orders	*Number of aisles*	*Obj*	*CPU*	*ObjL*	*ObjU*	*CPU*	*Number of routes*	*RBP/CW*
1080	10	8033.3	15.6	7175.0	7175.0	56.7	40.4	0.89
	20	12 492.8	17.0	10 647.5	10 647.5	121.0	147.2	0.85
	30	16 614.3	17.2	14 379.6	14 379.6	242.9	254.4	0.87
	40	20 517.8	18.7	18 418.0	18 418.0	366.8	342.4	0.90
2160	10	15 412.0	141.5	14 186.6	14 186.6	60.5	47.8	0.92
	20	23 365.4	129.5	20 287.7	20 287.7	123.1	214.1	0.87
	30	31 102.9	147.8	26 587.4	26 587.4	253.5	393.4	0.85
	40	37 971.8	142.0	33 637.3	33 637.3	394.3	552.9	0.89

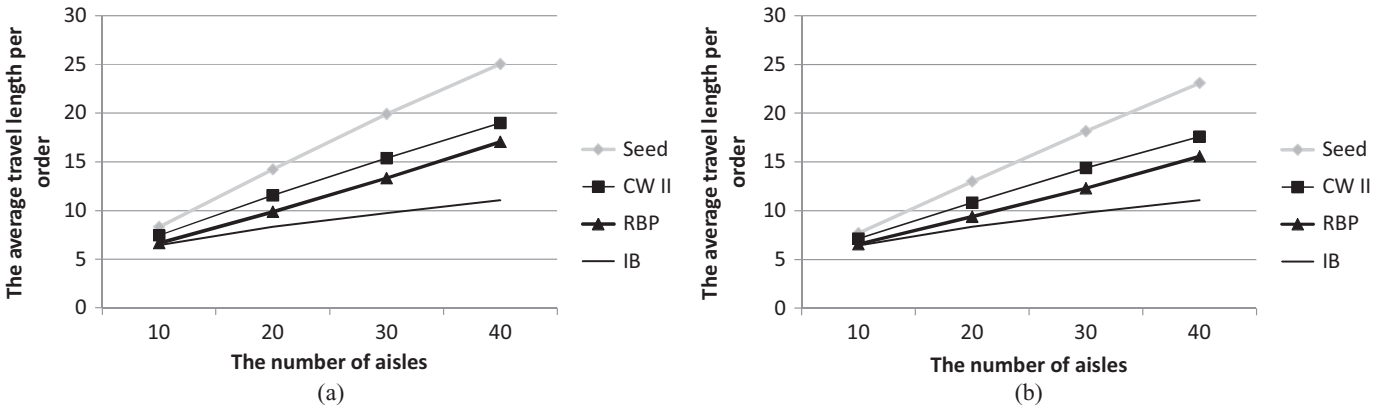

Fig. 1. The average travel length per order over the variation of the number of aisles, (a) the number of orders = 1080: and (b) the number of orders = 2160.

Table A2. The experimental results with the variation of storage policies

		CW II		*RBP*				
Number of orders	*Number of aisles*	*Obj*	*CPU*	*ObjL*	*ObjU*	*CPU*	*Number of routes*	*RBP/CW*
ABC	10	18 000.4	140.8	16 181.4	16 181.4	60.7	63.7	0.90
=0.5:0.3:0.2	20	28 926.6	130.9	24 043.8	24 043.8	128.4	340.5	0.83
Random	10	22 125.6	121.3	19 310.4	19 310.4	60.9	83.0	0.87
storage	20	37 872.4	126.9	34 535.9	34 535.9	150.0	554.5	0.91

Table A3. Experimental results with order size = Uniform[3,9]

	FCFS		*Seed*			*CW II*			*RBP*				*LB*		
Number of orders	*Obj*	*LU gap (%)*	*Obj*	*CPU*	*LU gap (%)*	*Obj*	*CPU*	*LU gap (%)*	*L ObjL*	*U ObjU*	*CPU*	*LU gap (%)*	*Obj*	*CPU*	*IB Obj*
360	7999.0	49.35	5415.8	0.01	25.19	4724.0	0.43	14.24	4116.7	4116.7	25.22	1.59	4051.3	0.42	3909.3
720	15 993.4	50.33	9943.6	0.02	20.11	8934.3	5.44	11.09	8015.9	8015.9	49.02	0.90	7943.5	0.87	7812.3
1080	24 072.7	50.74	14 328.5	0.05	17.25	13 009.6	18.51	8.86	11 934.8	11 934.8	56.93	0.65	11 857.0	1.51	11 725.6
1440	32 115.6	50.83	18 602.2	0.09	15.12	17 085.4	45.33	7.58	15 870.3	15 870.3	59.94	0.50	15 790.4	2.19	15 660.1
1800	40 127.3	50.94	22 819.8	0.15	13.72	21 093.6	75.18	6.66	19 770.4	19 770.4	60.39	0.42	19 688.3	2.87	19 561.4
2160	48 193.6	50.98	27 005.7	0.21	12.51	25 141.7	127.23	6.03	23 721.8	23 721.8	60.50	0.40	23 626.8	3.85	23 491.9

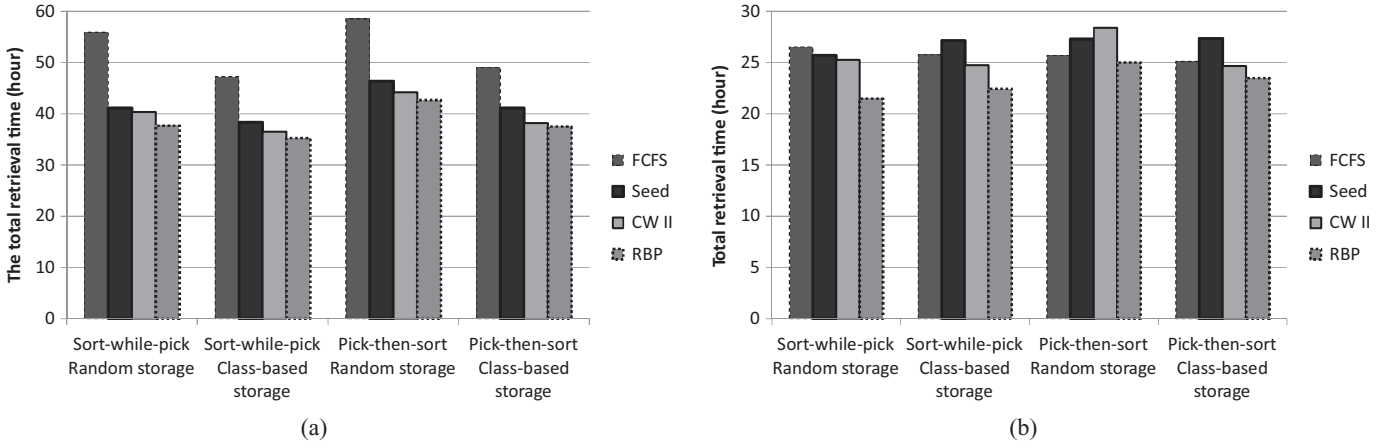

Fig. 2. Total retrieval time comparison via a simulation study: (a) light congestion case and (b) heavy congestion case.

ratio = 10:1; 15 pickers; and cart capacity = 25 orders or 50 items.

Figure A2 depicts the comparison of the total retrieval time. RBP is relatively robust to picker blocking situation, whereas Seed and CW II produce poor results under heavy congestion. This result emphasizes the importance of picker blocking and selecting a batching algorithm that not only reduces travel distance but does not create excessive picker blocking.

Biographies

Soondo Hong obtained his B.S. and M.S. in Industrial Engineering from POSTECH, South Korea, and his Ph.D. from the Department of Industrial and Systems Engineering at Texas A&M University. His research interests include cross-training based operations management and robust logistics operations in manufacturing, warehousing, software, and service industries. He is a member of the INFORMS.

Andrew L. Johnson is an Assistant Professor in the Department of Industrial and Systems Engineering at Texas A&M University. He obtained his B.S. in Industrial and Systems Engineering from Virginia Tech and his M.S. and Ph.D. from the H. Milton Stewart School of Industrial and Systems Engineering at Georgia Tech. His research interests include productivity and efficiency measurement, warehouse design and operations, material handling and mechanism design. He is a member of the INFORMS, National Eagle Scout Association, and German Club of Virginia Tech.

Brett A. Peters is Professor and Head of Industrial and Systems Engineering at Texas A&M University. His education includes a B.S.I.E. from the University of Arkansas and M.S. and Ph.D. in Industrial Engineering from the Georgia Institute of Technology. His primary research area is in facility design and material handling systems. His work has been funded by a variety of government and industry entities. He served as President of the College-Industry Council on Material Handling Education in 2002 and 2003. He was named a 2004 College of Engineering Faculty Fellow at Texas A&M, received the Outstanding Young Engineering Alumni Award from the University of Arkansas in 2005, and was named a Fellow by the Institute of Industrial Engineers in 2010. He has been Department Head at Texas A&M since 2004.