IX. ORDER BATCHING WITH TIME CONSTRAINTS IN A PARALLEL-AISLE WAREHOUSE: A MULTIPLE-POLICY APPROACH

Soondo Hong

Department of Industrial Engineering, Pusan National University, Pusan, 609-735, Korea

Andrew L. Johnson

Department of Industrial and Systems Engineering
Texas A&M University
College Station, TX 77843, USA

Brett A. Peters

Department of Industrial and Manufacturing Engineering
University of Wisconsin - Milwaukee
Milwaukee, WI 53201, USA

Abstract

A commitment of delivery time is critical in some online businesses (De Koster, 2003). An important challenge to meeting customers' needs is timely order picking which is also relevant to worker safety, item freshness, overall operational synchronization, and reduced overtime. We analyze an order batch picking situation where a trip is constrained by vehicle capacity and must be completed within a specified time. We develop a model which partitions orders to batches to minimize the total travel time such that each trip meets the orders' time constraints and capacity limit, and also determines a suitable operational policy for each batch. Each policy is characterized by routing method, travel speed, capacity, and pick time. The proposed batching model can simultaneously group orders and can select a best policy among possible policy choices

for each batch. To solve the proposed batching procedure, an exact algorithm is implemented based on a branch-and-price method. Our multiple-policy approach experiences 2.1~7.0% reductions in retrieval time compared to a best single-policy approach. The experimental results emphasize that when time constraints are enforced in order batching, a multiple-policy is preferable to a single-policy approach, because allows additional flexibility.

1 Introduction

Today, warehouses play a critical role in supply chains (Tompkins et al. 2003). Warehouse operations are becoming more diverse; however, order picking is the most labor-intensive, and consumes a significant portion of operational costs (Frazelle 2002). Batch picking, where order pickers consolidate multiple orders in a single trip, improves picking throughput by reducing the number of trips. Typically, the batch size is determined to utilize the full capacity of a picking vehicle, which maximizes throughout.

We are interested in situations where orders have due dates they must be picked by. In this setting picking vehicles may not be fully loaded if the entire batch cannot be collected within the specified time limits. Two examples in the literature are an online order fulfillment situation that must meet customer promise dates (De Koster (2003) and the large online retailers that guarantee a fast delivery schedule to obtain a competitive marketing advantage (Gong and De Koster 2008). An example of the latter is Amazon.com, who gives its customers a shipping lead time guarantee, based on the option, "Amazon Prime", whereby membership guarantees two-day shipping at no additional fee (Pace 2009).

Food retailers, such as Peapod and Webvan, depend on specialized frozen and refrigerated warehousing (Maloney 2007), and the order picking time may be limited by safety regulations controlling the operational hours spent by order pickers in refrigerated climates (Platoni 2001). Time restricted order picking may also be when managerial rules force retrieval cycles to synchronize with the next operation.

Even though these time-critical operational issues and marketing strategies strongly rely on the stability and performance of the warehouse operations, the most expansive warehouse operation, order picking, has been not well studied from the perspective of the time constraint and its performance. In a preliminary study we grouped orders using traditional approaches (De Koster et al. 1999, Gademann and Van de Velde 2005), and found that a time-constraint batching approach suffered from a significant loss of productivity due to the increased number of trips.

Fundamentally, a traditional approach tends to assume a unique capacity, a unique travel speed, and a unique route method, where the latter represents a mechanism to organize a trip for a batch. Under these restrictions and two constraints (i.e., time and capacity), it is difficult to find batches that both utilize capacity and meet the time

constraints. Thus, typically the addition of a time constraint leads to small and more batches, requiring additional labor and higher operational costs.

We present an efficient and robust batching method to solve the time-constrained-order batching problem. In practice, warehouse managers face multiple operational policy decisions, where a policy is characterized by routing method, travel speed, pick time per item, and capacity. Note these parameters do not change independently that for example the routing method will affect the travel speed. Typically a manager selects S-shape routes for pickers' convenience. Our study allows such multiple operational policies, which we term "multiple-policy".

The purpose of our study is to: 1) define a multiple-policy order batching problem with capacity and time constraints and develop an appropriate model; and 2) based on the formulation, provide an adapted exact branch-and-price solution. The proposed order batching with time constraints (OBT) problem incorporates the policy selection problem with the order batching problem.

The remainder of the paper is organized as follows. Section 2 reviews order batching models and algorithms. Section 3 defines the order batching situation. We describe the characteristics of an operational policy, how to approximate retrieval times, and a succinct formulation. Section 4 presents a construction-based order batching algorithm and a branch-and-price solution. Section 5 summarizes the experimental results under the assumption of multiple-policy order picking. We conclude with our insights and suggestions for future research in Section 6.

2 Literature Review

This study is interested in an order batching problem in parallel-aisle warehouses. An order batching model is a special case of the vehicle routing problem (VRP) (Gu et al. 2007). Thus, several models and solutions are available based on VRP formulation. First, we summarize a route method, a significant consideration in order batching, since it determines route length. Second, we review available batching algorithms. Last, we summarize some issues relevant to the time-constrained order batching.

2.1 Routing Methods

The routing method tool used by pickers to construct a trip path over a given batch or order consists of an optimal algorithm and heuristic algorithms. Optimal routing of order pickers is a special case of the travelling salesman problem (Ratliff and Rosenthal 1983). Ratliff and Rosenthal (1983) present a polynomial timed dynamic model to solve the order picking problem in a parallel- aisle warehouse. However, due to route complexity, heuristics routing is often preferred for practical purposes; Hall (1993) concludes that S-shape and Largest gap strategies are reasonable.

2.2 Order Batching Algorithms

The extensive literature on batching algorithms for parallel-aisle picking systems can be categorized as: 1) optimal approach; 2) meta-heuristic; 3) seed heuristic; and 4) saving heuristic. An optimal approach for order batching is to solve the batching and routing problem exactly through a mixed integer programming model (Gademann and Van de Velde 2005). The authors extend Gademann et al. (2001) by developing a branch-and-price formulation for the sort-while-pick order picking strategy. They evaluate a trip length using an optimal routing method, and point out that the optimal route is impractical due to route complexity.

De Koster et al. (1999) conduct a comparison study of seed and saving algorithms and conclude that the best seed algorithms combine three control factors: select the seed order as the order that must visit the largest number of aisles; choose the next order to minimize the number of additional aisles; and cumulatively update the seed information based on orders in the seed. Alternatively, in the same paper the savings algorithm (the modified Clarke and Wright method) is developed in which a savings list is updated until no savings pair remains. They conclude that the savings algorithm is preferable to the seed algorithm.

Hsu et al. (2005) propose a meta-heuristic approach, a genetic order batching algorithm, to minimize the total travel distance. The picking distance is calculated through S-shape strategy. Each route is constrained by the capacity of the order picking vehicle.

2.3 Issues

In reviewing the extant literature, we find that only distance has been considered as a measure of a solution quality. The distance-based approach itself can be extended to a time-based batching model. If the order picker travels at a constant speed and the retrieval time is constant regardless of batching, then a distance-based objective gives the same solution as a time-based objective. However, if the time-based batching formulation is further constrained to enforce each order is retrieved within in a limited time, the required travel distance will increase.

3 Order Batching Model with Time-Constraints

This section presents an order picking situation where a time constraint is enforced and multiple-policy is allowed.

3.1 Problem Definition

We consider a parallel-aisle warehouse where pickers ready at the loading/unload station and circumnavigate parallel aisles to retrieve items. At a pick location, pickers place the

items needed in one vehicle and travel to the next pick location. This is repeated until their pick lists are complete, and then return to the station. The order size is relatively small compared to the cart capacity; thus, batch picking is adopted to improve order picking throughput by combining many orders in one trip. A set of orders is available at a given time, and pickers must complete each trip within the time length specified. We assume that the number of pickers at a loading station is greater than the number of batches read for picking at any point in time. The pickers who do not participate in picking operation can be assigned to other operations such as sorting or packing. We assume the aisle width is designed to be wide enough that congestion is not a concern.

An operational policy specifies a routing method and a batch size. This study considers three routing methods: optimal routing method, S-shape routing method, and traversal routing method. Note a traversal routing method is distinguished from an S-shape routing method in that traversal routing does not allow a U-turn in an aisle. Dependent on routing methods, travel speed can vary. For example, optimal routing can decrease a travel speed because a picker requires additional care to follow the specified route. Batch size can vary with different pick time. If a picker is more careful loading the picking vehicle additional capacity can be obtained.

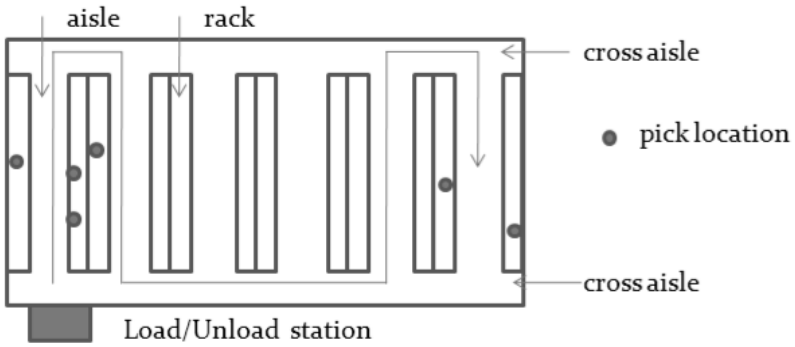

Figure 1: A narrow-aisle system and a routing example (Gademann and Van de Velde 2005).

3.2 Estimating Retrieval Time

Previously, Tompkins et al. (2003) described the time components of order picking operations, and Gademann and Van de Velde (2005) described the time components of batch picking. We integrate their work and calculate the retrieval time for operation policy v having four components: operation policy, route, item, and distance, according to dependency:

- Policy setup time (OT_v, operation policy set time): e.g. training time
- Route setup/finishing time (RT_v, loading/loading time per trip): e.g. pick list pick-up time, unload time(one-time unload)
- Pick dependent time (PT_v, pick time per time): e.g. pick time, search time,

acceleration/deceleration time, unloading time (item by item)

- Travel distance dependent time (WT_v, walk time per pick face): e.g. travel distance, acceleration/deceleration time (# of travel aisles)

We assume that the components are deterministic and linearly dependent on operation policy, the number of routes, the number of picked items, and travel length. Next, we develop a linear equation to estimate the retrieval time for policy v:

The retrieval time relevant to policy v =
OT_v
+ (RT_v + PT_v * the number of picked items + WT_v * the number of pick faces) // route 1
+ (RT_v + PT_v * the number of picked items + WT_v * the number of pick faces) // route 2
…
+ (RT_v + PT_v * the number of picked items + WT_v * the number of pick faces) // route n

3.3 Mathematical Model

We formulate a batching model as a set partitioning problem to account for the potential of multiple-policies. Initially, we assume that S_v possible batches are available for each policy v. For batch s using policy v, the retrieval time is d_{vs}. Batch s of policy v contains multiple orders, which we express with an incidence vector (a_{ovs}). Some operation policy may require an initial setup cost (OT_v). This time does not impact the retrieval time of a batch, but it does increase the overall retrieval time of orders. For example, if a policy uses a different picking vehicle, time must be allotted for the picker to set up the vehicle.

Indices and Parameters

V = the set of all possible policies

v = index regarding policies

S_v = the set of all possible batches in policy v

s = index regarding batches

O = the set of all possible orders

o = index regarding batches

OT_v = the setup time for policy v

a_{ovs} = the incidence column vector for batch s containing order o of policy v

d_{vs} = the retrieval time to pick all items of orders in batch s via policy v

Decision Variables

$$Z_v = \begin{cases} 1 & \text{if policy } v \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{sv} = \begin{cases} 1 & \text{if batch } s \text{ of policy } v \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Formulation

$$\text{(OBT)} \quad \text{Min} \quad \sum_{v \in V} \sum_{s \in S_v} d_{vs} Y_{vs} + \sum_{v \in V} OT_v Z_v \tag{1}$$

$$\text{s.t} \quad Z_v \geq Y_{vs}, \qquad \forall\, s \in S_v, \forall\, v \in V, \tag{2}$$

$$\sum_{v \in V} \sum_{s \in S_v} a_{ovs} Y_{vs} = 1, \qquad \forall\, o \in O, \tag{3}$$

$$Z_v \in \{0,1\}, \qquad \forall\, v \in V, \tag{4}$$

$$Y_{vs} \in \{0,1\}, \qquad \forall\, s \in S_v, \forall\, v \in V, \tag{5}$$

The objective function (1) minimizes the total travel time. Constraints (2) ensures if a policy v is used for a batch s, then the indicator variable, z_v, is set to one to indicate the setup for that policy is required. Constraints (3) ensure that each order is picked exactly one time. Constraints (4) and (5) require that the decision variables take binary values.

The OBT is a form of the Location-Routing problem (LRP) with multi- constraints (see Berger *et al.* (2007)). Although LRP selects facility locations and organizes their delivery routes, OBT selects operation policies and develops batches.

4 Branch-and-Price Algorithm

We apply a branch-and-price algorithm to optimally solve the OBT formulation. The branch-and-price algorithm is a well-known exact method to solve VRP problems (Toth and Vigo 2002, Berger et al. 2007). We build a construction algorithm, a master problem solution, sub problem solution, and branching strategies. The prior work by Gademann and Van de Velde (2005) and Berger et al. (2007) cannot be applied directly to our formulation. Gademann and Van de Velde do not consider multiple policies. The multiple-policy approach impacts the master problem algorithm and the branch-and-pricing strategy. In addition, our sub problem considers both sort-while-pick and pick-then-sort strategies, where the item-based quantity constraint of the pick-then-sort strategy requires a new algorithm for the sub problem. Although our formulation shares a

similar structure to the location-routing approach (Berger et al. 2007), the sub-problems in our formulation has a unique structure which has not appeared in the VRP literature.

4.1 Construction Algorithm

Generating an initial feasible solution is critical to the performance of a branch-and-price algorithm. The initial feasible solution defines the first upper bound, which if tight, decreases unnecessary exploration of the branch and bound tree. Typically heuristic algorithms are used to build initial solutions.

We propose a modified Clarke and Wright algorithm (CW) (Clarke and Wright (1964), De Koster et al. (1999)). Our CW algorithms is an adaptation of CW(i) and includes logic to deal with both policy selection and batching as follows:

CW1: extension of CW(i) by De Koster et al. (1999)

Step 1: calculate the saves s_{vij} for all possible order pairs i,j, given the capacity and time limit of the policy v

Step 2: sort the savings in decreasing sequence

Step 3: select the pair with the highest savings. In the case of a tie, select a random pair

Step 4. Three cases can happen:

- Neither of the orders has been included in an existing route and the remaining capacity of the vehicle is sufficient for both orders: include both orders in a new route.
- Exactly one order has been included in an existing route of a policy. If the other order fits in this route, add it to the route; if not, proceed with step 5
- Both orders have been included in an existing route: proceed with step 5

Step 5: select the next order combination from the list and repeat step 4 until all orders have been included in a route

We also develop a similar extension for CW(ii).

CW2: extension of CW(ii) by De Koster et al. (1999)

Step 1: calculate the saves s_{vij} for all possible order pairs i,j, given the capacity and time limit of policy v

Step 2: sort the savings in decreasing sequence

Step 3: select the pair with the highest savings; in the case of a tie, select a random pair

Step 4. merge the two orders i.e., 'clusters' into a new cluster, if it is feasible for the capacity and the time; if not, choose the next combination on the list

Step 5: if all order combinations have not been included in a route, proceed with step 1; in the calculation, all clusters are considered as orders; otherwise: go to end

4.2 Master Problem

The master problem optimally assigns of orders to batches. This can be formulated as a set partition problem.

$$\text{(MP)} \quad \text{Min} \quad \sum_{v\in V}\sum_{s\in S_v} d_{vs}Y_{vs} + \sum_{v\in V} OT_v Z_v$$

$$\text{s.t} \quad Z_v - \sum_{s\in S} a_{ovs}Y_{vs} \geq 0, \qquad \forall\, v\in V, \forall\, o\in O \tag{6}$$

$$\sum_{v\in V}\sum_{s\in S_v} a_{ovs}Y_{vs} \geq 1, \qquad \forall\, o\in O \tag{7}$$

$$0 \leq Z_v \leq 1, \qquad \forall\, v\in V, \tag{8}$$

$$0 \leq Y_{vs} \leq 1, \qquad \forall\, s\in S_v, \forall\, v\in V, \tag{9}$$

The OPT formulation is a set partition problem with location constraints, thus an Integer Program (IP). To solve the OPT problem, a linear programming relaxation of the problem is used. Equations (8) and (9) define Z_v and Y_{vs} as continuous variables. Further, constraint (3) relaxes equation (7) from requiring strict equality to a greater than or equal condition. This problem is a set covering problem. The LP relaxation provides solution that is far from feasible in the OBT formulation. Thus, we modify the constraints (6) is a rank-1 Chvatal cuts (Berger et al. 2007) developed from equations (2).

4.3 Sub Problem

There are two kinds of dual variables from the master problem associated with constraints (6) and (7): for orders o , π_o , and for policies v, μ_{ov}. For each sub problem, new columns are generated if the dual variables from the master problem indicated a

negative reduced cost. The dual variables update the reduced cost of a batch s associated with policy v. We define the reduced cost as:

$$d'_{vs} = d_{vs} - \sum_{o \in O} a_{ovs}(\pi_o - \mu_{ov}) \tag{10}$$

Indices and Parameters

Q_v = the maximum capacity for one route when policy v is used

T_v = the maximum time length for one route when policy v is used

R = the set of all possible routes

c_{ijv} = the travel cost from item i to item j by policy v

RT_v = the route setup time for policy v

PT_v = the unit picking time of policy v

WT_v = the travel speed of policy v

q_o = the number of items in order o

Decision Variables

$$x_{ij} = \begin{cases} 1 & \text{if a route has an arc from item } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_o = \begin{cases} 1 & \text{if a route contains order o} \\ 0 & \text{otherwise} \end{cases}$$

The reduced cost is reflected in the objective function of the sub problem. The three factors appearing in the objective function are: 1) setup time, 2) picking dependent time, and 3) travel distance dependent time. Without considering dual variables, the objective function can be expressed as:

$$RT_v + PT_v \sum_{o \in O} q_o y_o + WT_v \sum_{i \in N} \sum_{j \in N} c_{ijv} x_{ij} \tag{11}$$

Based on the dual variables, we derive an updated objective function. As described above, the dual variables are indexed as orders and picking modes. For simplicity, we derive the objective function as a value dependent on order quantity:

$$\begin{aligned} RT_v + PT_v \sum_{o \in O} q_o y_o + WT_v \sum_{i \in N} \sum_{j \in N} c_{ijv} x_{ij} - \sum_{o \in O} a_{ovs}(\pi_o - \mu_{ov}) y_o \\ = RT_v + \sum_{o \in O} (PT_v q_o - a_{ovs}(\pi_o - \mu_{ov})) y_o + WT_v \sum_{i \in N} \sum_{j \in N} c_{ijv} x_{ij} \end{aligned} \tag{12}$$

Thus, the sub problem of policy v generates a batch (or batches) to minimize the equation (13).

$$\text{(SP)} \quad \text{Min } RT_v + \sum_{o \in O} \left(PT_v q_o - a_{ovs} \left(\pi_o - \mu_{ov} \right) \right) y_o + WT_v \sum_{i \in N} \sum_{j \in N} c_{ijv} x_{ij} \tag{13}$$

We modify the branch-and-bound approach in Gademann et al. (2005) and adapt an active node search. Each level matches an order. The level contains two branches: including the current level and excluding the current node. To improve search performance, we consider only negative weighted orders. We sort the result in a decreasing order. Because we know the remaining capacities and all worst possible negative weight, we can forecast the most negative value at a node. Based on the current n^{th} objective value, if the minimal forecast objective value is greater than n^{th} objective value, we do not need to continue the branch step. We use the following lower bound forecasting logic:

1) Define the remain capacities for time and quantity
2) Find (new dual-picking times)/order quantity for all orders
3) Sort the values in increasing order
4) Compute minimal (maximal possible quantity)
    a. When the remaining quantity is less than the candidate order quantity, consider the possible portion.

4.4 Branching Rules

Our problem contains two branching stages: 1) we determine which policies are acceptable for the optimal solution, and 2) we fix the order pairs for each policy. For the strategies decision we follow Berger et al. (2007). To branch on an order pair, we adopt Gademann et al. (2005).

To branching on a policy we consider fractional values for Z_p and we apply the 0-1 branching. When $Z_p > 0.001$, we set Z_p to 1. For $Z_p \leq 0.001$, we set Z_p to 0 and set all routes Y_{op} regarding Z_p to 0.

For Y_{op}, we design two branching strategies: 1) we apply the most common approach by Gademann et al. (2005) to consider the branch on the most fractional node and correct the more factional pair into an integer value, 0 or 1; and 2) we also consider the branch on the smallest order size, because a smaller order has a larger number of batch combinations than a larger order.

4.5 Overall Iteration

The branch-and-price procedure requires a step-by-step approach. First, we run the construction algorithm to obtain an upper bound (UP). Second, we execute a column generation step. Then, we get a lower bound (LP). If |UP-LP|= 0, we stop. The current

solution becomes an optimal solution. If not, we conduct a branch-and-price step utilizing branching rules and column generation steps.

5 Experimental Results

We report the computational results and discuss insights from the computational studies. We first test the computational performance of the proposed heuristic and branch-and-price algorithms with two policies. Our experiment then focuses on a picking situation where four extreme policies are prepared. Last, we discuss a situation where a limited number of policies are possible.

We implement the proposed system using the C language. The master problem is implemented using the ILOG CPLEX Callable Library C API 11.0.4. To test the computational performance, the executable files are run on a Window NT-based server system with Windows Vista (Xeon 2.66 Ghz CPU, 12 GB memory). We use a modified order picking profile as in Gademann et al. (2005) and Le-Duc (2005).

5.1 Computational Performance

We use the following profiles:

- Warehouse layout: 6 aisles (height=30, width=5)
- Operation policy 1. OT_1=0, RT_1=0, PT_1=8, WT_1=1, Q_1= 8 or 15
- Operation policy 2. OT_2=0, RT_2=0, PT_2=4, WT_2=1, Q_2= 6 or 12
- Time constraint: 350 or 400
- Storage strategy: random or class (within-aisle class based strategy, 2 aisles:4 aisles= 70:30)
- Routing: optimal method
- The proportion of orders (o) to items (i): 20:24, 20:36, 15:60, and 20:72

The test sets are generated randomly. For each order set, we generate 10 problem runs per each instance:

- ub: objective value by a construction algorithm
- opt: objective value by a branch-and-price algorithm
- diff %: (ub-opt)/ub*100
- note: the number of branched nodes
- cpu: run time in seconds

Construction Algorithm

Table 1 compares construction algorithms in terms of the quality of the objective value. CW2 dominates CW1 and first-come first-served (FCFS) batching algorithms. The

CW2 algorithm accounts for the cost savings at each iteration, therefore the candidates at the top of the list are more likely to improve the solution. The computational time is negligible because of the problem size.

Table 1: Computational results of conduction algorithms.

(Q_1,Q_2)	Instances	Construction algorithms (ub)		
		CW2	CW1	FCFS
(8,6)	20:24 class storage	538.8	660.4	940.0
	20:24 random storage	595.4	742.6	1229.6
	20:36 class storage	1660.0	1749.8	2483.8
	20:36 random storage	1957.0	2095.8	3167.0
	15:60 class storage	2305.8	2309.8	2810.2
	15:60 random storage	2950.6	2976.0	3613.2
	20:72 class storage	2305.8	2309.8	2810.2
	20:72 random storage	2950.6	2976.0	3613.2
(15,12)	20:24 class storage	541.8	749.2	1031.2
	20:24 random storage	607.6	729.0	1179.6
	20:36 class storage	919.8	1050.2	1553.8
	20:36 random storage	993.0	1242.0	1892.0
	15:60 class storage	1650.8	1720.6	2148.8
	15:60 random storage	1939.2	2013.2	2760.0
	20:72 class storage	1951.2	2038.0	2822.8
	20:72 random storage	2217.2	2323.6	3582.2

Branch-and-Price Algorithm

Table 2 summarizes the computational results by the branch-and-price algorithm. We compare two branching rules: 1) a most fractional node first rule (Fraction first), and 2) a smallest order size node first rule (Smallest order). Commonly, the branch-and-price algorithm for the location-routing problem works well branching on the most fractional node (Berger et al. 2007), whereas our results exhibit different patterns. When the order size is small, branching on the most fractional node performs well. However, when the order size is large and the items are stored randomly, branch on the smallest order works better. Pairs with small order sizes have a better chance to lead to cost efficient batches.

Table 2: Computational results of the branch-and-price algorithm.

(Q_1,Q_2)	Instances	B&P rules						
		Objective values			Fraction first		Smallest order	
		ub	opt	diff %	nodes	cpu	nodes	cpu
(8,6)	20:24 class storage	538.8	492.0	9.51	262.1	53.7	565.4	78.9
	20:24 random storage	595.4	554.6	7.36	173.9	35.2	400.2	70.1
	20:36 class storage	1660.0	1560.0	6.41	4347.3	717.6	5445.1	655.2
	20:36 random storage	1957.0	1898.6	3.08	2337.2	367.1	2828.6	472.1
	15:60 class storage	2305.8	2228.4	3.47	48.0	7.5	57.8	7.3
	15:60 random storage	2950.6	2871.4	2.76	151.6	21.0	61.0	10.1
	20:72 class storage	2305.8	2228.4	3.47	48.0	7.5	57.8	7.3
	20:72 random storage	2950.6	2871.4	2.76	151.6	21.0	61.0	10.1
(15,12)	20:24 class storage	541.8	522.0	3.8	264.5	59.4	494.1	102.5
	20:24 random storage	607.6	563.0	7.9	99.5	33.5	376.6	68.6
	20:36 class storage	919.8	818.0	12.4	733.4	153.1	2484.7	438.3
	20:36 random storage	993.0	922.8	7.6	4291.6	765.0	2402.9	411.2
	15:60 class storage	1650.8	1537.2	7.4	905.4	204.4	1439.9	330.1
	15:60 random storage	1939.2	1835.6	5.6	223.8	38.2	330.6	52.2
	20:72 class storage	1951.2	1812.8	7.6	19127.0	3755.5	31925.4	5899.5
	20:72 random storage	2217.2	2052.8	8.0	9953.3	2764.2	8095.8	1748.2

5.2 Single-Policy Versus Multiple-Policy

An order picker can take different route methods according to the batch. On the operational level, we can design a multiple-policy operation situation. We provide the following rationales:

1) If an order picker takes more time, he/she can load more items
2) When a simpler routing method is employed under the same routing length, a travel time can be shortened
3) Even though an optimal routing method is inconvenient, sometimes the reduction in distance outweighs the additional cognitive processing time.

We repeat the previous experiment with the same order picking profile. First, we consider various storage methods: random, within-aisle class, and across aisle class. Second, we define four operation modes summarized in Table 3. The four policies are named normal policy, heavy-load policy, fast-pick policy, and a short-path policy.

Table 1: Possible operation policies (V).

Type	PT_v	WT_v	Q_v	Routing method
Normal (A)	10	2.7	10	S-shape
Heavy load (B)	10	3.3	15	S-shape
Fast (C)	10	2.25	10	Traversal
Short (D)	10	3	10	Optimal

- Storage strategy: random, within-aisle class based strategy (The products in the

first 2 aisle account for 70% of the demand whereas the remaining 4 aisles contain product making up only 30% of the demand), and across-aisle class-based strategy (The products in the 10 pickfaces closest to the bottom cross aisle in each aisle account for 70% of the demand whereas the remaining 20 pick faces contain product making up only 30% of the demand)

- diff %: (obj of a single policy –opj by multiple-policy)/obj of a single policy *100

Table 4 summarizes the results. Our approach generates 2.1~7.0% total retrieval time reduction according to the time length compared to the shortest single-policy.

Table 2: Comparison of single-policy and multiple-policy with the branch-and-price algorithm.

Order type	Storage policy	T_v	Single policy				OBT (OPT)					
								Diff (%)				
			A	B	C	D	Obj	A	B	C	D	Min
12:29	Random	920	1553	1508	1415	1492	1373	11.6	8.9	3.0	8.0	3.0
		800	1553	1627	1415	1492	1373	11.6	15.6	3.0	8.0	3.0
		700	1555	1743	1415	1492	1373	11.7	21.2	3.0	8.0	3.0
	Within-Aisle	920	1405	1380	1278	1346	1242	11.6	10.0	2.8	7.7	2.8
		800	1405	1424	1278	1346	1242	11.6	12.8	2.8	7.7	2.8
		700	1405	1472	1278	1346	1242	11.6	15.6	2.8	7.7	2.8
	Across-Aisle	920	1509	1463	1415	1386	1321	12.5	9.7	6.7	4.7	4.7
		800	1509	1532	1415	1386	1321	12.5	13.8	6.7	4.7	4.7
		700	1509	1676	1415	1386	1321	12.5	21.2	6.7	4.7	4.7
15:60	Random	920	3515	3092	3152	3242	2927	16.7	5.3	7.1	9.7	5.3
		800	3515	3370	3152	3242	2949	16.1	12.5	6.4	9.1	6.4
		700	3580	3403	3152	3243	2991	16.5	12.1	5.1	7.8	5.1
	Within-Aisle	920	3102	2716	2756	2923	2616	15.7	3.7	5.1	10.5	3.7
		800	3102	2866	2756	2923	2617	15.6	8.7	5.0	10.5	5.0
		700	3103	2923	2756	2923	2635	15.1	9.9	4.4	9.9	4.4
	Across-Aisle	920	3425	3013	3152	3009	2838	17.2	5.8	10.0	5.7	5.7
		800	3425	3237	3152	3009	2844	17.0	12.2	9.8	5.5	5.5
		700	3434	3733	3152	3009	2896	15.7	22.4	8.1	3.8	3.8
15:90	Random	920	5787	4993	5281	5310	4744	18.0	5.0	10.2	10.7	5.0
		800	5787	5490	5281	5310	4909	15.2	10.6	7.0	7.6	7.0
		700	5851		5281	5310	4975	15.0		5.8	6.3	5.8
	Within-Aisle	920	5071	4386	4574	4766	4161	17.9	5.1	9.0	12.7	5.1
		800	5071	4509	4574	4766	4169	17.8	7.6	8.9	12.5	7.6
		700	5084	4312	4574	4766	4220	17.0	2.1	7.8	11.5	2.1
	Across-Aisle	920	5639	4864	5281	4912	4552	19.3	6.4	13.8	7.3	6.4
		800	5639	5191	5281	4912	4607	18.3	11.3	12.8	6.2	6.2
		700	5659		5281	4912	4721	16.6		10.6	3.9	3.9

6 Conclusion

This study presents an efficient, robust framework for the time-constrained order batching problem and finds solutions using exact solution approaches. The time-based order batching typically develops solutions that meet both capacity and time constraints, but are very costly. Thus, we develop a model which jointly selects an operating policy and defines batches. We adapt branch-and-price methods to identify exact solution.

Time constrained order picking a characteristic of many picking operations. Our approach selects the best policy and associated batching decisions. We believe that the proposed approach can improve productivity in order batching situation through obtaining robustness by policy selection.

We implemented a branch-and-price method to determine the best policies and associated set of batches. To construct an initial solution, we extended the order batching version of the Clarke and Wright algorithm (De Koster et al. 1999). The results showed the operational improvement over a single policy-based approach. However, computational performance remains an outstanding issue.

A warehouse management system (WMS) and the experiences of warehouse managers contribute to developing the best policies. WMS can be used to track the performances of order pickers, picking vehicles, and the deployment of forklifts, totes, mobile devices and carts, thereby analyzing their performance parameters. By analyzing these histories, warehouse managers can better include more accurate information in for the parameters used on our proposed model.

Acknowledgements

This Colloquium is a direct result of the work of many individuals associated with this and prior Colloquia. This work was supported by Pusan National University Research Grant, 2014.

References

[1] Berger, R.T., Coullard, C.R., and Daskin, M.S., 2007. Location-routing problems with distance constraints. *Transporation Science,* 41 (1), 29-43.

[2] 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.

[3] De Koster, R., 2003. Distribution strategies for online retailers. *Engineering Management, IEEE Transactions on,* 50 (4), 448-457.

[4] De Koster, R., Van Der Poort, E.S., and Wolters, M., 1999. Efficient orderbatching methods in warehouses. *International Journal of Production Research,* 37 (7), 1479-1504.

[5] Frazelle, E., 2002. *World-class Warehousing and Material Handling.* New York. McGraw-Hill.

[6] 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.

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

[8] Gu, J., Goetschalckx, M., and Mcginnis, L.F., 2007. Research on warehouse operation: a comprehensive review. *European Journal of Operational Research,* 177 (1), 1-21.

[9] Hong, S., Johnson, A.L., and Peters, B.A., 2012. Batch picking in narrow-aisle order picking systems with consideration for picker blocking. *European Journal of Operational Research,* 221 (3), 557-570.

[10] 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.

[11] Le-Duc, T., 2005. *Design and control of efficient order picking processes*. Ph.D. dissertation. Erasmus University.

[12] Maloney, D., 2007. *Express lane* [online]. http://www.dcvelocity.com/articles/20070801verticalfocus/ [Accessed Access Date 2011].

[13] Pace, M., 2009. *Amazon Prime squeezes already struggling rivals* [online]. http://blog.compete.com/2009/01/20/amazon-prime-holiday-shopping/ [Accessed Access Date 2011].

[14] Platoni, K., 2001. *The last mile* [online]. http://www.eastbayexpress.com/eastbay/the-last-mile/Content?oid=1065476 [Accessed Access Date 2011].

[15] 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.

[16] Tompkins, J.A., Bozer, Y.A., and Tanchoco, J.M.A., 2003. *Facilities Planning.* 3rd. Hoboken, NJ. J. Wiley.

[17] Toth, P. and Vigo, D., 2002. *The vehicle routing problem.* Philadelphia. Society for Industrial and Applied Mathematics.