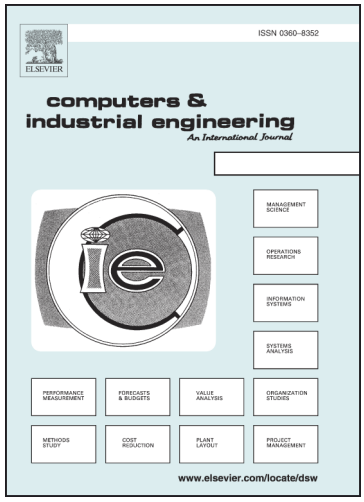
Accepted Manuscript



Integrated on-line scheduling of order batching and delivery under B2C e-commerce

Jun Zhang, Xuping Wang, Kai Huang

PII: S0360-8352(16)30017-1
DOI: http://dx.doi.org/10.1016/j.cie.2016.02.001
Reference: CAIE 4247

To appear in: *Computers & Industrial Engineering*

Received Date: 30 June 2015
Revised Date: 30 November 2015
Accepted Date: 1 February 2016

Please cite this article as: Zhang, J., Wang, X., Huang, K., Integrated on-line scheduling of order batching and delivery under B2C e-commerce, *Computers & Industrial Engineering* (2016), doi: http://dx.doi.org/10.1016/j.cie.2016.02.001

Integrated on-line scheduling of order batching and delivery under B2C e-commerce

Jun Zhang[1,3], Xuping Wang[1,21], Kai Huang[3]

[1] *Institute of Systems Engineering, Dalian University of Technology, Dalian, China*

[2] *School of Business, Dalian University of Technology, Panjin, China*

3 DeGroote School of Business, McMaster University, Hamilton, Canada

Abstract:

Order batching is a general method of grouping a set of orders into several sub sets, i.e., batches. As many warehouses outsource order delivery to a Third Party Logistics (3PL) provider in B2C e-commerce, small lot-size orders arriving dynamically within a certain time period should be grouped into batches and packed up before a fixed departure time. Research on on-line order batching problems, however, seldom takes delivery constraints into consideration. This paper studies the integrated on-line order batching and distribution scheduling problem in which the maximal number of orders has to be completed before the vehicles' departure time in the shortest service time. Several novel rule-based solutions are proposed, including the formation of batches, which are assigned to appropriate pickers without any information on the arrival times of future orders. Moreover, the solutions define which orders are urgent and should be picked up directly, and which ones can be satisfied later. The solution algorithms are evaluated through a series of experiments. It is demonstrated that these algorithms can lead to a substantial increase of the number of delivered orders, which reveals the importance of integrating order batching with delivery.

Keywords: on-line order batching; order batching and delivery; rule-based solutions; urgency degree

[1] wxp@dlut.edu.cn

Integrated on-line scheduling of order batching and delivery under B2C e-commerce

Abstract:
Order batching is a general method of grouping a set of orders into several sub sets, i.e., batches. As many warehouses outsource order delivery to a Third Party Logistics (3PL) provider in B2C e-commerce, small lot-size orders arriving dynamically within a certain time period should be grouped into batches and packed up before a fixed departure time. Research on on-line order batching problems, however, seldom takes delivery constraints into consideration. This paper studies the integrated on-line order batching and distribution scheduling problem in which the maximal number of orders has to be completed before the vehicles' departure time in the shortest service time. Several novel rule-based solutions are proposed, including the formation of batches, which are assigned to appropriate pickers without any information on the arrival times of future orders. Moreover, the solutions define which orders are urgent and should be picked up directly, and which ones can be satisfied later. The solution algorithms are evaluated through a series of experiments. It is demonstrated that these algorithms can lead to a substantial increase of the number of delivered orders, which reveals the importance of integrating order batching with delivery.

Keywords: on-line order batching; order batching and delivery; rule-based solutions; urgency degree

1 Introduction

With the development and wide-spread use of mobile technology, customers can shop anytime and anywhere through a business-to-consumer (B2C) e-commerce shopping platform. However, with the big boom of B2C e-commerce, small lot-size, high frequency, and dynamic arrival of customer orders make order picking and delivery difficult to implement. In order to accelerate the whole order fulfillment process, orders should be picked and delivered to customers in a very short lead time. It is therefore critical to integrate these two operations. This paper focuses on the on-line order batching strategy in relation to the picking process, where orders become available dynamically over time and the known orders can be grouped into several batches based on existing batching rules. Moreover, as many warehouses outsource order delivery to a Third Party Logistics (3PL) provider, orders should be picked and packed up before a fixed departure time. Motivated by these changes, we study the integrated on-line order batching and delivery scheduling (IOOPDS) problem with a fixed delivery departure time.

In the B2C on-line scheduling model, information such as arrival time and customer demand may not be known until the order's release. Algorithms for the model have to form batches and assign them to appropriate pickers with a fixed departure time in an on-line environment. The objective is to deliver a maximal number of orders within the shortest service time. Four main decision issues need to be solved: 1) when should arrival orders begin to be grouped into batches; 2) which orders should be assigned to the same batch; 3) which orders are urgent and should be satisfied quickly, and which ones can be satisfied later; and 4) how should the batches be assigned to available pickers. However, departure time, multiple pickers, and the on-line environment all challenge classical batching and assigning approaches. For ease of implementation, rule-based solutions are proposed to solve the complex problems: 1) hybrid time window batching rules

based on urgency degree and similarity degree; and 2) assigning rules based on departure time and balance strategy.

The remainder of this paper is organized as follows. In section 2, we provide a literature review of the integrated on-line order picking and distribution scheduling problem. In section 3, we establish an IOOPDS optimization model to maximize delivered orders in minimal service time with regard to multiple order-pickers. To solve this problem, rule-based batching and assigning solutions are proposed in section 4. In section 5, several examples are tested to verify the effectiveness of the proposed model and algorithms for IOOPDS. The paper concludes with a summary and an outlook on further research topics in section 6.

2. Literature survey

The integrated on-line order picking and distribution scheduling problem (IOOPDS) has some similar features with the integrated production and distribution scheduling problem (IPDS) as both contain production and distribution stages. Many scholars have studied IPDS in various aspects. Chen (Chen, 2010) classified existing IPDS models into five major classes: machine configurations, order parameters, delivery characteristics, number of customers, and objective functions. In the IOOPDS model of our paper, the number of customers is multiple and the objective function is to maximize the number of delivered orders within the shortest service time. Therefore we analyze the other three classes of IPDS. The existing results for the three classes of IPDS are summarized in Table 1. IPDS models can be separated into six classes by delivery characteristics: individual and immediate delivery (Liu and Cheng, 2002; Mastrolilli, 2003; Dawande et al. ,2006), batch delivery to a single customer by direct shipping method (Chang and Lee, 2004; Chen and Pundoor, 2009; Cheng et al., 2015), batch delivery to multiple customers by direct shipping method (Chen and Lee, 2008; Li and Vairaktarakis, 2007), batch delivery to multiple customers by routing method (Geismar et al., 2008; Chen and Vairaktarakis, 2005; Gao et al., 2015), delivery with time windows (Low et al., 2014; Low et al., 2013), and fixed delivery departure dates (Li et al., 2006; Hajiaghaei-Keshteli et al., 2014; Hajiaghaei-Keshteli and Aminnayeri, 2014; Agnetis et al., 2014).

The machine configuration contains single, parallel, and bundling, while the processing method includes direct processing and batching processing. For the direct processing method, the production time of order j is carried out with a constant time like p_j or with a constant processing rate R. For the batching production, the processing time of the batch b_k is set to be $\max\{p_j, j \in b_k\}$, $\sum_{j \in b_k} p_j$ or a constant processing rate R. In this work, the delivery method is fixed delivery departure dates and the machine configuration is parallel (where order pickers are regarded as machines). However, the order parameter of IOOPDS is different from IPDS. In IPDS, the processing time is generally constant, while the processing time optimization in order picking scheduling is an NP-hard problem. Furthermore, effective on-line order picking scheduling should be considered in B2C e-commerce, which requires defining which customer order should be satisfied directly and which one should be satisfied later.

As IPDS models do not address order batching problems, the reviews of order batching need to be discussed. Order batching is the key strategy of order picking optimization (Henn, 2012). Most previous researchers have focused on off-line order batching strategies, and only a few studies on on-line strategies have been conducted. Off-line batching policies can be divided into five types: priority rule-based algorithms (Gibson and Sharp, 1992), seed algorithms (Ho and Tseng, 2006), saving algorithms (Clarke and Wright, 1964), meta-heuristic algorithms (Henn and Schmid, 2013; Matusiak et al., 2014), and data mining approaches (Chn and Wu, 2005; Hsieh and Huang, 2011).

In this paper, we focus on an on-line batching problem with a fixed departure time. The existing results of the problem are summarized in Table 2. On-line batching solutions can be divided into two classes. First, fixed time window batching assigns all orders arriving during a particular time interval to batches (Henn, 2012; Bukchin et al., 2012). Second, variable time window batching (Chew and Tang, 1999; Le-Duc and De Koster, 2007; Xu et al., 2014) has a picker wait until a particular number of orders arrive and then assigns the items of these orders into batches. Only one picker is considered in the on-line order batching problem and no due time constraints are considered. A few scholars have taken into account the due time to avoid production or delivery tardiness in the off-line environment. Tsai et al. (2008) attempted to solve a batch picking model that considered earliness and tardiness penalty using a multiple genetic algorithms method for obtaining the best possible batch picking plans. Henn and Schmid (2013) indicated that customer orders have to be completed by certain due dates in order to avoid shipment or production delays. They presented how meta-heuristics can be used to minimize the total tardiness for a given set of customer orders. Additionally, Amir et al. (2013) proposed a novel solution approach in order to minimize tardiness which consists of four phases. However, all of these models are analyzed in an off-line environment and only one picker is employed.

In this paper we study a particular case of IOOPDS, where the orders' departure time is known beforehand. In practice, many warehouses outsource the transportation to a 3PL provider, which means the orders' departure and delivery time are determined by the 3PL (Cheng et al., 2015). Orders should be picked and packed up before the departure time. Different from the research noted above (Tsai et al., 2008; Henn and Schmid, 2013; Amir et al., 2013), we study the on-line order picking scheduling, taking both a fixed time window and a variable time window into consideration. Moreover, as previous on-line strategies assume only one picker, which limits the exploration of the picking and distrubution efficiency, we expand the problem with the constraints of multiple pickers.

3 Problem description and formulation

3.1 Problem description

The integrated on-line order picking system and the distribution system are shown in Figure 1 (separated into order picking system and distribution system). In an on-line order picking system, customer orders arrive dynamically over time. Orders that arrive over a period of time are batched by existing batching rules, then the generated batches are queued and waited to be assigned to available pickers based on existing assigning rules. In the distribution system, picked orders are packed up and delivered to customers based on each destination's departure time. The objective of the IOOPDS system, therefore, is to minimize the service time and make sure more orders could be completed before departure time. As we know, orders that belong to the same destination are delivered together in the same vehicle, and that each vehicle can deliver only once per day. We can simplify the IOOPDS system into a batch generation system and a batch operation system with the departure time constraint.

In a batch generation system, customer orders arrive according to a stochastic process and batches are generated by batching rules. The point in time when a customer order becomes available is called the arrival time. The entering time is the point in which the generated batches enter into batch operation system. The batch time can be determined as the length of the time period between the earliest order's arrival time and entering time. Therefore, the first two decision points of IOOPDS are: 1) when should arrival orders enter into batch operation system; 2) which orders should be assigned to the same batch.

In a batch operation system, the generated batches are operated by available order pickers based

on existing assigning rules. The point in time when a batch is assigned to an available picker to process is called the starting time. The waiting time of a batch released to order picker can be calculated as the time period between the entering time and starting time. The service time of a batch is the time period that includes walking time, picking time, and packing time. It is clear that a different assigning method would cause a different waiting time and service time. Completion time is the point in time when the order picking for a batch is finished. If the completion time is earlier than the departure time, the order can be distributed and its lead time is the time period between departure time and completion time. If the completion time is later than the departure time, the order is left in the distribution center and waits to be distributed the next day. Delay time is the time period between completion time and departure time. The second decision problem we need to address in this paper is the way that batches are queued and then assigned to pickers in order to deliver more orders before departure time and to balance the picker's workload.

In summary, IOOPDS under B2C environment deals with the following questions: In an on-line order picking system with a fixed departure time, how should the orders be grouped into batches, and how should the batches be assigned to available pickers in order to minimize service time and to maximize the number of delivered orders?

3.2 IOOPDS optimization model

The IOOPDS problem in this paper is based on the following assumptions:

(1) The layout of picking area is shown in Figure 2, which is a common single-block warehouse with two cross-aisles.

(2) The picking route strategy is S-shape (Henn, 2012), and the service time includes walking time, picking time, and packing time. Similar to the setting of existing literatures, we assume that the storage policy is random.

(3) Customer orders are unknown in advance but become available over time. A hybrid time window batching policy which considers departure time is used. One customer order must be incorporated into one batch.

(4) Only when a previous batch is finished can an order picker handle the next one. We assume that there are a finite number of pickers.

(5) Vehicle departure time is the time when 3PL's vehicle departs the distribution center. All the vehicles leave once per day. If the vehicle departs before all orders are loaded, the left orders can be processed next day.

The off-line version optimization model for the IOOPDS problem is presented in order to analyze the structure of the problem.

These parameters are used:

N	A set of orders, $N = \{1, 2, ..., n\}$;
M	A set of batches, $M = \{1, 2, ..., m\}$;
O	A set of destinations, $O = \{1, 2, ..., o\}$;
L	A set of order pickers, $L = \{1, 2, ..., l\}$;
t_i^{arrive}	Arrival time of customer order i to the order batching system, $i \in N$ where $0 \le t_i^{arrive} \le t_{i+1}^{arrive}$;
t_h^{depart}	Vehicle departure time of destination h, $h \in O$, where $0 \le t_h^{depart} \le t_{h+1}^{depart}$;
t^{pack}	Packing time;
q_i	Number of items of order i, $i \in N$;

Q_b	Maximal number of items in a batch;
v_{travel}	Travel speed, the distance order picker can cover per time unit;
v_{pick}	Picking speed, the number of items order picker can search and pick per time unit;
z_{ih}	1 if order i is allocated to destination h, 0 otherwise. Note that z_{ih} is given parameter in the optimization model.

The model uses the following decision variables:

$t_j^{service}$	Service time of batch j, $j \in M$;
t_j^{start}	Starting time of batch j;
$t_i^{complete}$	Completion time of order i;
$T(J')_k^{complete}$	Picker k's completion time of batches J', where $J' = \{1,2,...,j-1\}$, $k \in L$;
dis_j	Distance function of the picking tour for batch j, which is decided by the routing method;
R	Number of effective orders that could be delivered to customers on time;
x_{ij}	1 if order i is assigned to batch j, 0 otherwise;
y_{jk}	1 if batch j is assigned to available order picker k, 0 otherwise;
r_i	1 if order i is delivered to its customer on time.

Therefore the IOOPDS problem can be modeled as follows:

Minimize $\sum_{j=1}^{M} t_j^{service}$ (1)

Maximize $\sum_{i=1}^{N} r_i$ (2)

Subject to：

$$\sum_{j=1}^{M} x_{ij} = 1, \quad \forall i \in \{1,...,n\} \tag{3}$$

$$\sum_{k=1}^{L} y_{jk} = 1, \quad \forall j \in \{1,...,m\} \tag{4}$$

$$\sum_{i=1}^{N} q_i x_{ij} \le Q_b, \quad \forall j \in \{1,...,m\} \tag{5}$$

$$t_j^{service} = dis_j / v_{travel} + \sum_{i=1}^{n} x_{ij} \cdot q_i / v_{pick} + t_{pack}, \quad \forall j \in \{1,...,m\} \tag{6}$$

$$t_j^{start} \ge \max_{i \in \{1,...,n\}} \{x_{ij} \cdot t_i^{arrive}\}, \quad \forall j \in \{1,...,m\} \tag{7}$$

$$T(J')_k^{complete} = \max\{y_{jk}(t_{j'}^{start} + t_{j'}^{service})\}, \ \forall k \in \{1,...,l\}, j' \in J' \tag{8}$$

$$t_j^{start} \ge \min_{k \in \{1,...,l\}} T(J')_k^{complete}, \quad \forall j \in \{1,...,m\} \tag{9}$$

$$t_i^{complete} = \sum_{j=1}^{M} x_{ij} \cdot t_j^{start} + \sum_{j=1}^{M} x_{ij} \cdot t_j^{service}, \quad \forall i \in N \tag{10}$$

$$\sum_{h=1}^{O} z_{ih} = 1, \quad \forall i \in N \tag{11}$$

$$r_i = \begin{cases} 1 & (\sum_{h=1}^{O} z_{ih} \cdot (t_h^{depart} - t_i^{complete}) \geq 0 \\ 0 & otherwise \end{cases} \quad (12)$$

In this model we formulate two objective functions to coordinate the picking efficiency with on time delivery: one is to minimize the service time of all batches, and the second is to guarantee maximal number of delivered orders. In the objective functions, (1) minimizes the whole service time of customer orders and function (2) maximizes the number of delivered orders. Eq. (3) ensures the assignment of each customer order to exactly one batch, and (4) ensures the assignment of each batch to exactly one order picker. Furthermore, inequality (5) guarantees that the capacity of the picking device is not violated. Eq. (6) is the expression of service time, which is the sum of walking time, picking time, and packing time. Walking distance function dis_j is decided by the S-shape routing method. The starting time of each batch is a critical variable which significantly impacts completion time and number of delivered orders. Constraints (7)-(9) indicate the starting time of batch j. Inequality (7) indicates that a batch can be started when all orders assigned to this batch are known. As multiple pickers are considered in this article, (8) expresses the time when picker k finishes the top $j-1$ batches assigned to him. Inequality (9) ensures the starting time is no earlier than the minimal $T(J')_k^{complete}$ per picker. In other words, batch j can only be started once there is an available picker. Eq. (10) is the completion time of order i. Constraints (11)-(12) belong to the delivery constraints. Eq. (11) ensures each order belongs to exactly one destination and (12) indicates r_i is a binary decision variable, equals to 1 if completion time is earlier than departure time. Therefore $\sum_{i=1}^{N} r_i$ represents the number of orders that can be delivered.

4 Rule-based solutions of IOOPDS

The novel rule-based solutions of IOOPDS are proposed in order to form batches and assign them to appropriate pickers without any information on the arrival times of future orders. We distinguish two decision points and the basic principle of the solutions in section 4.1. Batching rules and assigning rules are explained in section 4.2 and 4.3.

4.1 Basic principle

The flowchart of rule-based solutions is shown in Figure 3, and it contains decision points, batching rules, and assigning rules. There is a sequence of orders $1,...,n$ with different arrival times $t_1^{arrive},...,t_n^{arrive}$. The algorithm has to generate batches without having complete information on the arrival times of future orders. The decision points of the algorithm combine the classical ideas of the fixed time window batching rule and of the variable time window batching rule. It is therefore named the hybrid time window batching rule, which contains the time interval constraint ($t = t^{end}$) and the item quantity constraint ($q_R \geq Q_{\max}$). Both of the two constraints can activate order batching process. Given this, the decision points in time t can be distinguished as follows:

Type A: A set of unprocessed orders exists and the capacities exceed $Q_{\max}$. We should decide to batch these orders and the entering time of the batches is the instant time t.

Type B: A set of unprocessed orders exists and the time t exceeds t^{end}. We should decide to batch these orders and the entering time of the batches is t^{end}.

We can see that at each decision point, the algorithm should determine a solution of the off-line

order batching problem. Since the off-line order batching problem is NP-hard, we propose a heuristic batching rule H_B to generate batches at the decision points. Since different orders have different departure times, we should distinguish if the arrival time is close to the departure time. If so, we can design the order urgency. If there is no urgent order in the unprocessed orders, a batching rule based on similarity degree H_{B1} is proposed. Once it contains an urgent order in the unprocessed orders, a batching rule based on urgency degree H_{B2} is proposed. The solution of H_B (H_{B1}, H_{B2}) contains a set of batches $B(t)$ with different service times and different departure times. When the batches enter into the batch operation system, an assigning rule is needed to assign each batch to a picker to avoid delayed delivery. When there are several idle pickers, the rule should decide to which picker each batch should be assigned. When all the pickers are busy, the rule should insert the batches into the waiting queue and rearrange the queue. The assigning rule is named H_A. Algorithm A summarizes the on-line principle:

Algorithm A. Basic principle of the integrated on-line order picking and distribution algorithm.

Decision Point A and B (at time t)

Input the unprocessed orders into batch generation system;

If there are no urgent orders, **then**

Generate a set of batches $B(t)$ by means of batching rule H_{B1};

Else

Generate a set of batches $B(t)$ by means of batching rule H_{B2};

End if

Input $B(t)$ into batch operation system;

Schedule the batches based on assigning rule H_A

4.2 Batching rule H_B

Seed algorithms (*Seed*) (Ho and Tseng, 2006) and recalculation saving algorithms (*C&W(ii)*) (De Koster et al., 1999) are two high efficiency batching algorithms. For both algorithms, the similarity degree between two orders is used to evaluate if the two orders should be batched. Moreover, *Seed* and *C&W(ii)* have similar calculation steps: order selection and order addition. Therefore, *Seed* and *C&W(ii)* are chosen to batch a collected order. However, hybrid time window batching rules based on similarity degree ignore the orders' departure time. Except for similarity degree, we should identity an order′s urgency degree as well, which expresses the urgency towards departure time. The greater the urgency degree, the earlier the order should enter into the batch operation system and be completed before departure time. Taking both the similarity degree and urgency degree into consideration, improved batching rules versions of *Seed* and *C&W(ii)* are designed, called *UrgentSeed* and *UrgentC&W(ii)*.

The expressions of similarity degree are as follows: 1) Picking-aisle similarity ratio as the similarity degree for *Seed* is designed as $simi_{ij} = \left|Aisle_i \cap Aisle_j\right| / \left|Aisle_i \cup Aisle_j\right|$. $\left|Aisle_i \cap Aisle_j\right|$ represents the number of identical picking aisles between orders i and j. $\left|Aisle_i \cup Aisle_j\right|$ is the total number of picking aisles that the picker needs to visit if combining orders i and j. 2) Time saving value for *C&W(ii)* is designed as $simi_{ij} = t_i^{service} + t_j^{service} - t_{ij}^{service}$, where $t_i^{service}$ and $t_j^{service}$ are service time of orders i and j if both orders are collected individually. Parameter $t_{ij}^{service}$ represents service time when combining two orders i and j in one route.

The urgency degree of each order is defined as:

$$urgent_i = \begin{cases} 1/t_i^{left} & t_i^{left} \in [t_{adv}, t_{pack}] \\ 0 & otherwise \end{cases}, \text{ and } t_i^{left} = \sum_{h=1}^{O} t_h^{depart} z_{ih} - t_i^{arrive}.$$

t_{adv} is lead time of the urgent order. t_{pack} is pack up time of the urgent order. If t_i^{left} is between t_{adv} and t_{pack}, we identify order i as an urgent order. If $t_i^{left} < t_{pack}$, order i will be left to process next day.

4.2.1 Batching rule based on similarity degree H_{B1}

If there are no urgent orders arriving in the unprocessed orders, a batching rule based on similarity degree H_{B1} is proposed. Set the unprocessed orders that arrive during this period as OS, and the detailed rules are as follows:

Step 1: Similarities calculation. If there is more than one order in OS, calculate $simi_{ij}$ between each order in OS. Else move to Step 6.

Step 2: Order selection. Pick orders i and j with the maximal $simi_{ij}$ from OS, calculate the volume of orders i and j (Q_i and Q_j).

Step 3: Orders addition. Batch orders based on the following constraints:

1) If $Q_i + Q_j < Q_b$, then combine $\{O_i, O_j\}$ to form O_l, delete O_i, O_j from OS and add O_l to OS. Move back to *Step 1*.

2) If $Q_i + Q_j = Q_b$, then combine $\{O_i, O_j\}$ to form a batch, delete O_i, O_j from OS. Move back to *Step1*.

3) If $Q_i + Q_j > Q_b$, move on.

Step 4: Select other orders to generate batch. Select an order from O_i, O_j with larger volume (for instance O_i). Calculate $simi_{im}$ between O_i and other orders in OS. If there exists O_m containing sub-maximal $simi_{im}$ with O_i where $Q_i + Q_m \le Q_b$, then combine $\{O_i, O_m\}$ to form a batch, delete O_i, O_m from OS. Move back to *Step1*. Else move on.

Step 5: Generate batch directly. Batch O_i directly and delete O_i from OS. Move back to *Step 1*.

Step 6: Output the order batching results $B(t)$.

4.1.2 Batching rule based on urgency degree H_{B2}

If there are urgent orders arriving in the unprocessed orders, a batching rule based on urgency degree H_{B2} can be proposed, which considers both batching efficiency and departure time. Set the unprocessed orders that arrive during this period as OO, and the detailed rules are as follows:

Step 1: Order selection. If there is more than one order in OO, rank the orders in the order of urgency degree and set the order, which contains largest urgency degree, as O_{seed}, and the volume as Q_{seed}. Set urgent orders of OO as OO_urgent, where $i' \in OO\_urgent$ and $urgent_{i'} > 0$. Else move to *Step 5*.

Step 2: Order addition. If $OO\_urgent \neq \varnothing$, calculate similarities between O_{seed} and other orders in OO_urgent. Else calculate similarities between O_{seed} and other orders in $\{OO - OO\_urgent\}$. Choose order i with the maximal $simi_{seed\,i}$ and the volume is Q_i. Set the two combined orders' service time as $service\_time(O_{seed}, O_i)$ and entering time as $enter\_time(O_{seed}, O_i)$. If the following departure constraint is satisfied,

$service\_time(O_{seed}, O_i) + enter\_time(O_{seed}, O_i) <= \min(depart\_time(O_{seed}, O_i))$, then

1) If $Q_{seed} + Q_i < Q_b$, then combine $\{O_{seed}, O_i\}$ to replace O_{seed}, delete O_i and original O_{seed} from OO, add updated O_{seed} to OO. Move back to *Step 2*.

2) If $Q_{seed} + Q_i = Q_b$, then combine $\{O_{seed}, O_i\}$ to form a batch, delete O_{seed} and O_i from OO. Move back to *Step 1*.

3) If $Q_{seed} + Q_i > Q_b$, move on.

Step 3: Select other orders to generate batch. If O_i cannot satisfy departure constraint, we select other orders. Calculate the similarities between O_{seed} and other orders. If there is O_m containing sub-maximal $simi_{seed\,m}$, which satisfies both departure and capacity constraints, then combine $\{O_{seed}, O_i\}$ to form a batch, delete O_{seed} and O_m from OO, and move back to *Step 1*. Else move on.

Step 4: Generate batch directly. If there are no orders satisfy both departure and capacity constraints, batch O_{seed} directly and delete O_{seed} from OO. Move back to *Step 1*.

Step 5: Output the order batching results $B(t)$.

4.3 Assigning rule H_A

The batch assigning problem is similar to the berth allocation problem (which focuses on scheduling ships from anchorage to available berths). Pratap et al. (Pratap et al., 2015) solved the berth allocation problem by considering both operating time of the ships unloading and priority of ships customer. Motivated by Pratap et al. (Pratap et al., 2015), we present the assigning rule H_A, which is based both on the service time and urgency degree (priority of batch) of each batch. Assigning batches with consideration of the urgency degree would allow for more orders to be delivered in one day. Assigning batches to idle pickers whose service time is minimal can minimize batch waiting time and balance their workload. The concrete rule can be summarized as follows:

Step 1: We suppose that there are l pickers in the batching operating system, which work independently.

Step 2: If all the pickers are busy, the batches need to generate a queue and wait. Generated batches queue for being serviced by the pickers based on the descending order of urgency degree. We should note that only one queue is allowed.

Step 3: If there is only one idle picker, the maximal urgency degree batch in the queue can be serviced.

Step 4: If there is more than one available picker, we first assign the maximal urgency degree batch to the picker whose workload is minimal.

5. Simulation experiments and analysis

In order to verify the effectiveness of the proposed model and algorithms for IOOPDS, several examples are tested in this section. The performance of the proposed methods (*Seed*, *UrgentSeed*, *C&W(ii)*, *UrgentC&W(ii)*) is compared to *FCFS*, which provides an upper bound. *Seed* and *C&W(ii)* are two traditional algorithms based only on batching rule H_{B1} and assigning rule H_A, named similar algorithms. *UrgentSeed* and *UrgentC&W(ii)*, based on batching rules H_{B1}, H_{B2} and assigning rule H_A, are called urgent algorithms.

5.1 Experiment parameters

The warehouse parameters are those frequently used in experiments described in the literature

(Gademann and Velde, 2005; Henn, 2010), which are shown in Table 3. The warehouse contains 900 identical storage locations and 90 storage locations are distributed on both sides of an aisle. Pickers retrieve items from the right and left side of an aisle simultaneously. The depot is on the leftmost aisle. The length of the aisle is 45m and the width between two aisles is 5m. A picker's horizontal and vertical walking speed is 48m/min. The constant pickup speed is 6 items/min and pack up time is 3min.

In order to analyze the quality of the proposed algorithms, several problem parameters are varied. For the total number of customer orders *n*, arriving within a time period of 2 hours, the value 600 is considered. The inter-arrival times (the time between the arrival of customer order) *i* with *i*+1 are exponentially distributed with the arrival rate λ, where for *n*=600, $\lambda = 5$. The capacity Q_b is set to 30, 45 and 60. The average number of orders to generate batches R values 50, 75, and 100. In other words, for different capacities, average {5, 7.5, 10}, {3.3, 5, 6.7}, {2.5, 3.75, 5} batches can be generated per time interval. For a customer order, the quantity per order is uniformly distributed in {1, 2, ..., 5}, therefore average quantity per order is $\bar{q} = 3$. The fixed interval time is set to be $t_b = R/\lambda$ and $Q_{\max} = R \cdot \bar{q}$. The starting time of the system, which means first order's arrival time, is zero. Orders will be delivered to three different destinations, where departure time separately is 60, 90, and 120, determined by the 3PL. As we can see, 9 problem classes are generated. The corresponding problem parameters can be considered as typical for real-world applications such as e-supermarket and also from the literature (Henn et. al 2010, Henn, 2012).

The computations for the 9 problem classes are carried out on a Pentium processor 2.53GHz and 2.0GB RAM. The algorithms have been implemented with MATLAB R2011b.

5.2 Experimental results analysis

5.2.1 Experimental results of service time and number of completed orders

In a batch generation system, we can analyze the service time and number of completed orders under different fixed time interval (t_b)/ maximal item quantity ($Q_{\max}$) and different device capacity (Q_b) (as shown in Table 4). For all *FCFS*, similar algorithms and urgent algorithms, a larger $t_b/Q_{\max}$ obtains a relatively smaller service time under the same Q_b, and a larger Q_b performs better under the same $t_b/Q_{\max}$. It is mainly because a larger $t_b/Q_{\max}$ and Q_b can increase batching efficiency and decrease service time per order. However, along with the increase of Q_b, the improvement of service time gradually decreases.

For similar algorithms, *Seed* and *C&W(ii)* get smaller service time than *FCFS*. Impro[1] gradually decrease with the increasing of Q_b under the same $t_b/Q_{\max}$. For example, Impro[1] of Seed drops from 9.40% to 1.54% with the increasing of Q_b when $t_b/Q_{\max}$=10/150. Furthermore, *C&W(ii)* performs better than *Seed* except for the situations Q_b=60, where the number of orders per batch is large. From the literature of De Koster (De Koster et al., 1999), it can be seen that the differences between *C&W(ii)* and *Seed* are mainly determined by the device capacity Q_b. If the capacity is large, then seed algorithms are preferred. The same results apply to the on-line order picking system proposed in this paper. The number of completed orders is 459 for all three algorithms; there are 141 orders which arrive later than their departure time and are passed to the next day.

Urgent algorithms complete fewer orders, compared with similar algorithms and *FCFS*. Except for the 141 passed orders, any other orders that cannot be completed before departure time are

passed to the next day as well if the left time is less than pack time. In addition, compared with *UrgentC&W(ii)*, *UrgentSeed* is also preferred if the capacity is large. Furthermore, the service time per order of urgent algorithms is larger than the one of similar algorithms *and* Impro[2] gradually decrease with the increasing of Q_b under the same $t_b / Q_{\max}$. In urgent algorithms, urgency degree is prior to similarity degree. To guarantee more orders to be completed before departure time, fewer urgent orders are batched together based on the departure constraint. Overall, it can be concluded that the batching scale effect decreases and the service time per order becomes higher than similar algorithms.

5.2.2 Experimental results of delivered orders

When generated batches enter into a batch operation system and are assigned to pickers, the number of delivered orders can be calculated. For certain, the more pickers we employ, the shorter the waiting time will be and the more orders can be delivered. Moreover, when the amount of employees reaches a certain upper bound, no waiting time is needed at all. Significantly, the upper bound of picker number is reached when t^{wait}=0. To retain consistency, *FCFS*'s upper bound of picker number is used to analyze the performance. The upper bound of picker number is gradually {15, 12, 9} when $t_b / Q_{\max}$=10/150, {16, 10, 8} when $t_b / Q_{\max}$=15/225, {18, 13, 10} when $t_b / Q_{\max}$=20/300 under different Q_b.

The results of delivered orders under the upper bound of picker number are shown in Table 5. The tendency of delivered orders and delivery rate remain the same for all five algorithms: smaller $t_b / Q_{\max}$ and smaller Q_b obtain relatively more delivered orders. As smaller $t_b / Q_{\max}$ and Q_b lead to earlier entering time and smaller service time per batch, more orders can be picked up and delivered before departure time under the upper bound of picker number. For *Seed* and *C&W(ii)*, delivery rates range from 48.67% to 57.33%. Compared with *FCFS*, similar algorithms' improvement of delivery rates is no more than 6.50%, which means that algorithms based on similarity degree have no significant improvement on delivery rates. Urgent algorithms' delivery rates improve markedly, however, ranging from an increase of 6.38% to 24.17%. It is concluded, therefore, that urgent algorithms can lead to a substantial increase of the number of delivered orders, which reveals the importance of integrating order batching with delivery.

In general, the two objectives, service time and number of delivered orders, are both very significant for B2C companies. From section 5.2.1 and 5.2.2, we can see that there is a substantial improvement in number of deliveries but no improvement in service time. As we know, similar algorithms focus on similar degree which contributes more towards service efficiency. However, as we discuss above, urgent algorithms pay more attention to urgency degree. Few urgent orders are batched together to guarantee delivery efficiency, which lowers the batching scale effect and increases service time. Although service time has no improvement, urgent algorithms deliver more orders in one day by employing the same number of pickers.

5.2.3 Experimental results under different number of pickers

In subsection above, the *FCFS*'s upper bound of picker number is used to analyze all algorithms. However the number of pickers has a great effect on delivery efficiency. In this part, we choose the problem class $t_b / Q_{\max}$=15/225 and Q_b=45 to analyze delivery rate, waiting time, delay time, and other factors under a different picker number. As *FCFS*'s upper bound is 10, the picker number is set to be {5, 6, 7, 8, 9, 10} in this subsection. The results of the problem class $t_b / Q_{\max}$ =15/225 and Q_b=45 under different picker number is shown in Table 6. As we can see, these five indexes have the same trends with the increase of picker number for both similar and urgent

algorithms: 1) more orders can be delivered and less delayed orders exist; 2) orders have a shorter wait time for available pickers to process them and so there will be a shorter delay time; 3) *Rate* will reach an upper bound when picker number reaches a certain value.

Viewed from *Rate* point, when picker number ranges from 5 to 7, the improvements of *Seed*(*C&W(ii)*) and *UrgentSeed*(*UrgentC&W(ii)*) are 11.00%(11.67%) and 18.60%(21.83%). However, the improvements drop to 2.33% (2.17%) and 4.67% (1.33%) when picker number ranges from 7 to 10. At this time, it makes little sense to improve delivery rate through hiring more pickers. Taking urgency degree into consideration, the improvement of *Rate* increases remarkably with increasing picker number. When the picker number is 5, the difference between *Seed*'s(*C&W(ii)*'s) *Rate* and *UrgentSeed*'s(*UrgentC&W(ii)*'s) *Rate* is 0(-0.16%). When the picker number is 10, the gap becomes 10%(9.16%). Furthermore, urgent algorithms greatly decrease delayed orders and the delay time. When 10 pickers are employed, although *Seed(C&W(ii))* completes more than 150(146) orders than *UrgentSeed(UrgentC&W(ii))*, they are delayed and the delay time is more than 2215.56(2044.86). However, delayed orders of *UrgentSeed* and *UrgentC&W(ii)* reach to zero. In addition, the waiting time of urgent algorithms is larger than the one of similar algorithms.

In summary, the extensive numerical experiments carried out have produced several findings:

1) Taking urgency degree into consideration, although the service time per order is larger and the number of completed orders is smaller, it is much more meaningful to increase the number of delivered orders.

2) To obtain the maximal number of delivered orders, smaller $t_b / Q_{\max}$ and smaller Q_b are preferred, while for minimal service time, larger $t_b / Q_{\max}$ and larger Q_b are better.

3) Hiring more pickers is helpful to improve delivery rate and reduce waiting time, but the improvement range gradually decreases and reaches to zero.

4) Taking urgency degree into consideration, the improvements of *Rate* increase remarkably with increasing picker number. When the picker number is 10, the difference reaches a maximal value of 10% between *UrgentSeed* and *Seed*, and 9.16% between *UrgentC&W(ii)* and *C&W(ii)*.

6. Conclusion and future scope

This paper considers the integrated on-line order batching and distribution problem in order to deal with picking scheduling under a fixed vehicle departure time. The challenge is to transform dynamically arriving orders into picking batches and assign them to appropriate pickers such that the maximal number of orders can be delivered within a minimal service time. The novel solutions are named *Seed*, *UrgentSeed*, *C&W(ii)* and *UrgentC&W(ii)*, which combine existing on-line picking rules and off-line batching methods. Through extensive numerical experiments it is observed that urgent algorithms can lead to a substantial increase of the number of delivered orders, which reveals the importance of integrating order batching with delivery.

The three main contributions of this paper are: 1) we study the integrated on-line order picking and distribution problem through the two objectives of minimizing the whole service time and maximizing the number of delivered orders; 2) except for similarity degree, urgency degree is formulated to express an order′s urgency between arrival time and departure time; and 3) new assigning rules based on departure time and balance strategy are established to increase delivery efficiency and balance workload.

In this paper, we assume that B2C companies outsource the transportation to a 3PL provider and

that no consideration of vehicle routing scheduling is needed. However, many companies have recently started to establish their own transportation network. Further research should investigate integrated on-line order batching and distribution scheduling by both optimizing picking processing and vehicle routing.

Acknowledgements

The authors appreciate the constructive suggestions provided by the editor and two anonymous referees. This work is supported by National Natural Science Foundation of China by Grant (Nos. 71471025, 71171029).

References

Agnetis A., Aloulou M.A., Fu L. 2014. Coordination of production and interstage batch delivery with outsourced distribution. European Journal of Operational Research, 238(1): 130-142.

Amir H.A., Shahrooz T., Pezhman G., Muhamad Z.M.S., Kuan Y.W. 2013. Order batching in warehouses by minimizing total tardiness: a hybrid approach of weighted association rule mining and genetic algorithms. The Scientific World Journal, vol. 2013, Article ID 246578, 13 pages.

Bukchin Y., Khmelnitsky E., Yakuel P. 2012. Optimizing a dynamic order-picking process. European Journal of Operational Research, 219(2): 335–346.

Chang Y.C., Lee C.Y. 2004. Machine scheduling with job delivery coordination. European Journal of Operational Research, 158(2): 470–487.

Chen M.C., Wu H.P. 2005. An association-based clustering approach to order batching considering customer demand patterns. OMEGA, 33(4): 333–343.

Chen Z.L. 2010. Integrated production and outbound distribution scheduling: review and extensions. Operations Research, 58(1):130-148.

Chen Z.L., Pundoor G. 2009. Integrated order scheduling and packing. Production and Operations Management, 18(6): 672-692.

Chen Z.L., Vairaktarakis G.L. 2005. Integrated scheduling of production and distribution operations. Management Science, 51(4): 614–628.

Chen B., Lee C. Y. 2008. Logistics scheduling with batching and transportation. European Journal of Operational Research, 189(3): 871–876.

Cheng B.Y., Li K., Hu X.X. 2015. Approximation algorithms for two-stage supply chain scheduling of production and distribution. International Journal of Systems Science: Operations & Logistics, 2(2): 78-89.

Chew E.P. and Tang L.C. 1999. Travel time analysis for general item location assignment in a rectangular warehouse. European Journal of Operational Research, 112(3): 582–597.

Clarke G. and Wright J.W. 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12(4): 68–581.

Dawande, M., Geismar H.N., Hall N.G., Sriskandarajah C. 2006. Supply chain scheduling: Distribution systems. Production and Operations Management, 15(2): 243–261.

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.

Gao S., Qi L., Lei L. 2015. Integrated batch production and distribution scheduling with limited vehicle capacity. International Journal of Production Economics, 160: 13-25.

Geismar H. N., Laporte G., Lei L., Sriskandarajah C. 2008. The Integrated Production and Transportation Scheduling Problem for a Product with a Short Lifespan. INFORMS Journal on Computing, 20(1): 21–33.

Gibson D.R and Sharp G.P. 1992. Order batching procedures. European Journal of Operational Research, 58(1): 57–67.

Hajiaghaei-Keshteli M., Aminnayeri M., Fatemi Ghomi S.M.T. 2014. Integrated scheduling of production and rail transportation. Computers & Industrial Engineering, 74: 240-256.

Hajiaghaei-Keshteli M., Aminnayeri M. 2014. Solving the integrated scheduling of production and rail transportation problem by Keshtel algorithm. Applied Soft Computing, 25: 184-203.

Henn S., Schmid V. 2013. Metaheuristics for order batching and sequencing in manual order picking systems. Computers & Industrial Engineering, 66(2): 338-351.

Henn S. 2012. Algorithms for on-line order batching in an order picking warehouse. Computers & Operations Research, 39(11): 2549-2563.

Ho Y.C. and Tseng Y.Y., 2006. A study on order-batching methods of order-picking in a centre with two cross-aisles distribution. International Journal of Production Research, 44(17): 3391–3417.

Hsieh L.F. and Huang Y.C. 2011. New batch construction heuristics to optimize the performance of order picking systems. International Journal of Production Economics, 131(2): 618-630.

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.

Li C. L., Vairaktarakis G. 2007. Coordinating production and distribution of jobs with bundling operations. IIE Transactions, 39(2): 203–215.

Li K. P., Ganesan V. K., Sivakumar A. I. 2006. Scheduling of single stage assembly with air transportation in a consumer electronic supply chain. Computers & Industrial Engineering, 51(2): 264–278.

Liu, Z., T. C. E. Cheng. 2002. Scheduling with job release dates, delivery times and preemption penalties. Information Processing Letters, 82(2): 107–111.

Low C.-R., Chang C.M., Li R.K., Huang C.L. 2014. Coordination of production scheduling and delivery problems with heterogeneous fleet. International Journal of Production Economics, 153: 139-148.

Low C.-R., Li R.K., Chang C.M. 2013. Integrated scheduling of production and delivery with time windows. International journal of production research, 51(3): 897-909.

Mastrolilli, M., 2003. Efficient approximation schemes for scheduling problems with release dates and delivery times. Journal of Scheduling, 6(6): 521–531.

Matusiak M., De Koster R., Kroon L., Saarinen J. 2014. A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. European Journal of Operational Research, 236(3): 968–977.

Pratap S., Nayak A., Cheikhrouhou N., Tiwari M.K. 2015. Decision support system for discrete robust berth allocation. IFAC-PapersOnLine, 48(3): 875-880.

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

Tsai C.Y., Liou J.J.H, Huang T.M. 2008. Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. International Journal of Production Research, 46(22): 6533–6355.

Xu X.H., Liu T., Li K.P., Dong WH. 2014. Evaluating order throughput time with variable time window batching. International Journal of Production Research, 52(8): 2232-2242.

Table 1. The existing results for the three classes of IPDS

Literature	Delivery characteristic	Machine configuration	Order parameter	
			Processing method	Processing time
Liu and Cheng, 2002	Individual and immediate delivery	Single	Direct processing	p_j
Mastrolilli, 2003		Parallel	Direct processing	p_j
Dawande et al., 2006		Parallel	Direct processing	p_j
Chen and Pundoor, 2009	batch delivery to a single customer by direct shipping method	Single	Direct processing	p_j
Chang and Lee, 2004		Single/Parallel	Direct processing	p_j
Cheng et al., 2015		Parallel	Batch processing	$\max\{p_j, j \in b_k\}$
Chen and Lee, 2008	batch delivery to multiple customers by direct shipping method	Single	Direct processing	p_j
Li and Vairaktarakis,2007		Bundling	Direct processing	p_j
Geismar et al., 2008	batch delivery to multiple customers by routing method	Single	Direct processing	p_j
Chen and Vairaktarakis, 2005		Single/Parallel	Direct processing	p_j
Gao et al., 2015		Single	batch processing	$\sum_{j \in b_k} p_j$
Low et al., 2014	delivery with time windows	Single	Batch processing	Constant rate R
Low et al., 2013		Single	Direct processing	Constant rate R
Li et al., 2006	fixed delivery departure dates	Single	Direct processing	Constant rate R
Hajiaghaei-Keshteli et al., 2014		Single	Direct processing	Constant rate R
Hajiaghaei-Keshteli and Aminnayeri, 2014		Single	Direct processing	Constant rate R
Agnetis et al., 2014		Single	Direct processing	p_j

Table 2. The existing results of on-line order batching problem with due time

Literature	On-line Vs Off-line	Fixed time window batching	Variable time window batching	Due time	One Picker
Henn, 2012	On-line	√			√
Bukchin et al., 2012	On-line	√			√
Chew and Tang, 1999	On-line		√		√
Le-Duc and De Koster, 2007	On-line		√		√
Xu et al., 2014	On-line		√		√
Tsai et al., 2008	Off-line			√	√
Henn and Schmid, 2013	Off-line			√	√
Amir et al., 2013	Off-line			√	√
Our paper	On-line	√	√	√	Multi-pickers

Table 3. The warehouse parameters

Parameter	Value
no. of aisles	10
no. of cells on each side of an aisle	45
no. of storage locations	900
length of a cell	1m
center-to-center distance between two aisles	5m
picker's travel speed	48m/min
pickup speed	6 items/min
Pack up time	3min
storage policy	Random-based
t_b / $Q_{\max}$	10/150, 15/225, 20/300
Q_b	30, 45, 60
departure time	60, 90, 120

Table 4. Service time and number of completed orders under different fixed time interval (t_b)/ maximal item quantity ($Q_{\max}$) and different device capacity (Q_b)

$t_b / Q_{\max}$		10/150			15/225			20/300		
Q_b		30	45	60	30	45	60	30	45	60
FCFS	Service time	963.88	749	630.25	938.79	715.42	611.5	933.92	714.5	601.54
	Completed orders	459			459			459		
Similar algorithms										
Seed	Service time	873.29	734.38	620.54	830.17	680.33	590.21	805.21	660.92	580.46
	*Impro[1]	9.40%	1.95%	1.54%	11.57%	4.90%	3.48%	13.78%	7.50%	3.50%
	Completed orders	459			459			459		
C&W(ii)	Service time	808.00	694.04	626.71	777.29	665.58	597.17	761.88	649.17	591.79
	*Impro[1]	16.17%	7.34%	0.56%	17.20%	6.97%	2.34%	18.42%	9.14%	1.62%
	Completed orders	459			459			459		
Urgent algorithms										
Urgent Seed	Service time	857.92	749.13	697.75	776.92	669.46	608.42	642.29	550.42	488.79
	*Impro[2]	-19.29%	-23.87%	-36.54%	-16.41%	-22.40%	-28.23%	-6.12%	-10.80%	-12.03%
	Completed orders	378			369			345		
Urgent C&W(ii)	Service time	832.96	733.42	681.21	734.96	630.21	575.29	648.00	564.25	521.83
	*Impro[2]	-22.90%	-25.98%	-29.59%	-17.94%	-18.10%	-20.16%	-13.49%	-15.98%	-17.66%
	Completed orders	385			368			344		

*Impro[1]. represents the improvement of service time per order compared *Seed(C&W(ii))* with *FCFS*. *Impro[2]. represents the improvement of service time per order compared *UrgentSeed(UrgentC&W(ii))* with *Seed(C&W(ii))*

Table 5. The results of delivered orders under the upper bound of picker number

t_b / Q_{max}		10/150			15/225			20/300		
Q_b		30	45	60	30	45	60	30	45	60
Picker no.		15	12	9	16	10	8	18	13	10
FCFS	Delivered orders	323	317	302	323	313	300	323	307	281
	*Rate	53.83 %	52.83 %	50.33 %	53.83 %	52.17 %	50.00 %	53.83 %	51.17 %	46.83 %
Similar algorithms										
Seed	Delivered orders	325	317	302	323	313	300	323	311	292
	*Rate	54.17 %	52.83 %	50.33 %	53.83 %	52.17 %	50.00 %	53.83 %	51.83 %	48.67 %
	*Impro^3	0.62 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	1.30 %	3.91 %
C&W(ii)	Delivered orders	344	317	302	323	313	300	323	311	295
	*Rate	57.33 %	52.83 %	50.33 %	53.83 %	52.17 %	50.00 %	53.83 %	51.83 %	49.17 %
	*Impro^3	6.50 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	1.30 %	4.98 %
Urgent algorithms										
Urgent Seed	Delivered orders	378	377	371	369	369	360	345	345	345
	*Rate	63.00 %	62.83 %	61.83 %	61.50 %	61.50 %	60.00 %	57.50 %	57.50 %	57.50 %
	*Impro^4	14.02 %	15.92 %	18.60 %	12.47 %	15.18 %	16.67 %	6.38 %	9.86 %	15.36 %
Urgent C&W(ii)	Delivered orders	385	385	375	368	368	368	344	344	344
	*Rate	64.17 %	64.17 %	62.50 %	61.33 %	61.33 %	61.33 %	57.33 %	57.33 %	57.33 %
	*Impro^4	11.92 %	21.45 %	24.17 %	13.93 %	17.57 %	22.67 %	6.50 %	10.61 %	16.61 %

*Rate. represents the delivery rate, which means number of delivered orders/total number.

*Impro^3. represents the improvement of **Rate* compared *Seed(C&W(ii))* with *FCFS.*

*Impro^4. represents the improvement of **Rate* compared *UrgentSeed(UrgentC&W(ii))* with *Seed(C&W(ii)).*

Table 6. The results of the problem class $t_b / Q_{\max}$ =15/225 and Q_b =45 under different picker number

Seed algorithms						
Index	Picker no.	Delivered orders	Delay orders	Rate	Waiting time	Delay time
Seed	5	229	230	38.17%	718.25	7125.06
	6	277	182	46.17%	289.03	3842.33
	7	295	164	49.17%	126.28	2765.81
	8	309	150	51.50%	53.84	2370.19
	9	309	150	51.50%	26.23	2283.89
	10	309	150	51.50%	6.24	2215.56
Urgent Seed	5	229	140	38.17%	756.89	2003.99
	6	311	58	51.83%	329.68	374.47
	7	341	28	56.83%	149.24	108.06
	8	362	7	60.33%	85.71	43.82
	9	367	2	61.17%	47.16	14.96
	10	369	0	61.50%	24.13	0.00
C&W(ii) algorithms						
Index	Picker no.	Delivered orders	Delay orders	Rate	Waiting time	Delay time
C&W(ii)	5	230	229	38.33%	595.97	6063.03
	6	284	175	47.33%	227.36	3278.61
	7	300	159	50.00%	87.61	2474.61
	8	304	155	50.67%	41.08	2207.11
	9	313	146	52.17%	13.88	2094.89
	10	313	146	52.17%	0.00	2044.86
Urgent C&W(ii)	5	229	139	38.17%	559.86	1417.76
	6	327	41	54.50%	234.87	202.54
	7	360	8	60.00%	103.31	23.25
	8	368	0	61.33%	45.57	0.00
	9	368	0	61.33%	21.25	0.00
	10	368	0	61.33%	0.00	0.00

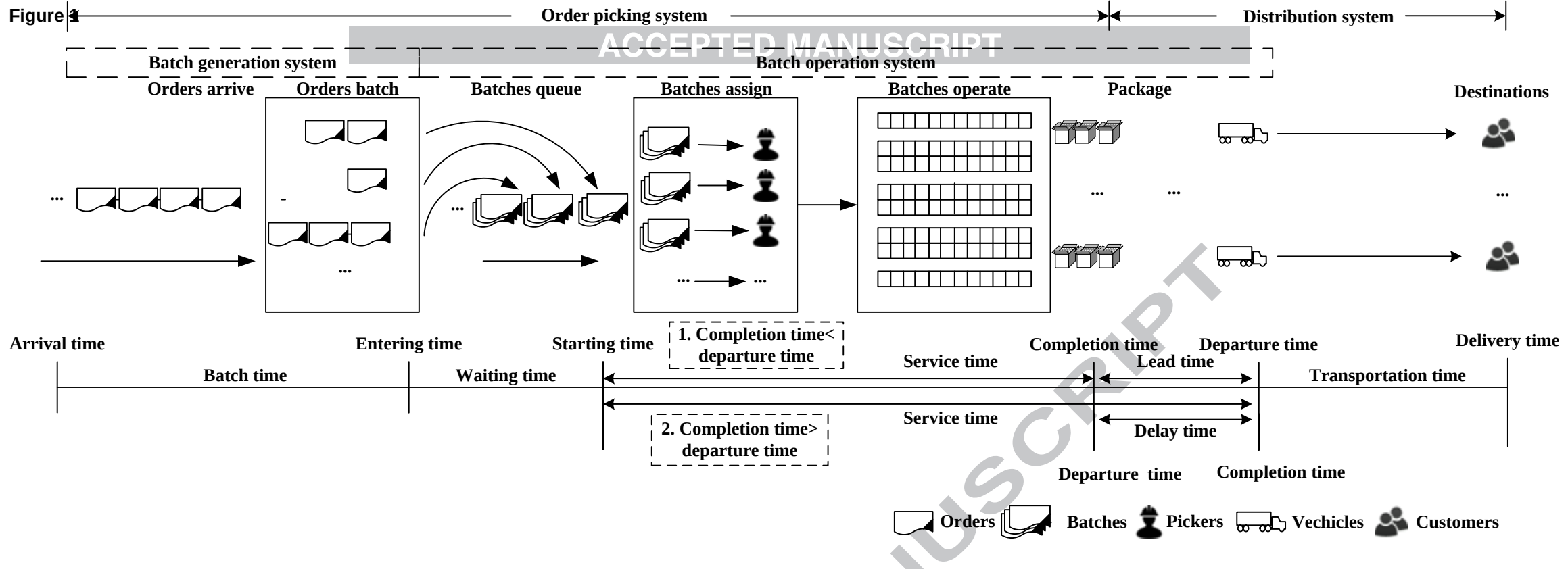


Figure 2

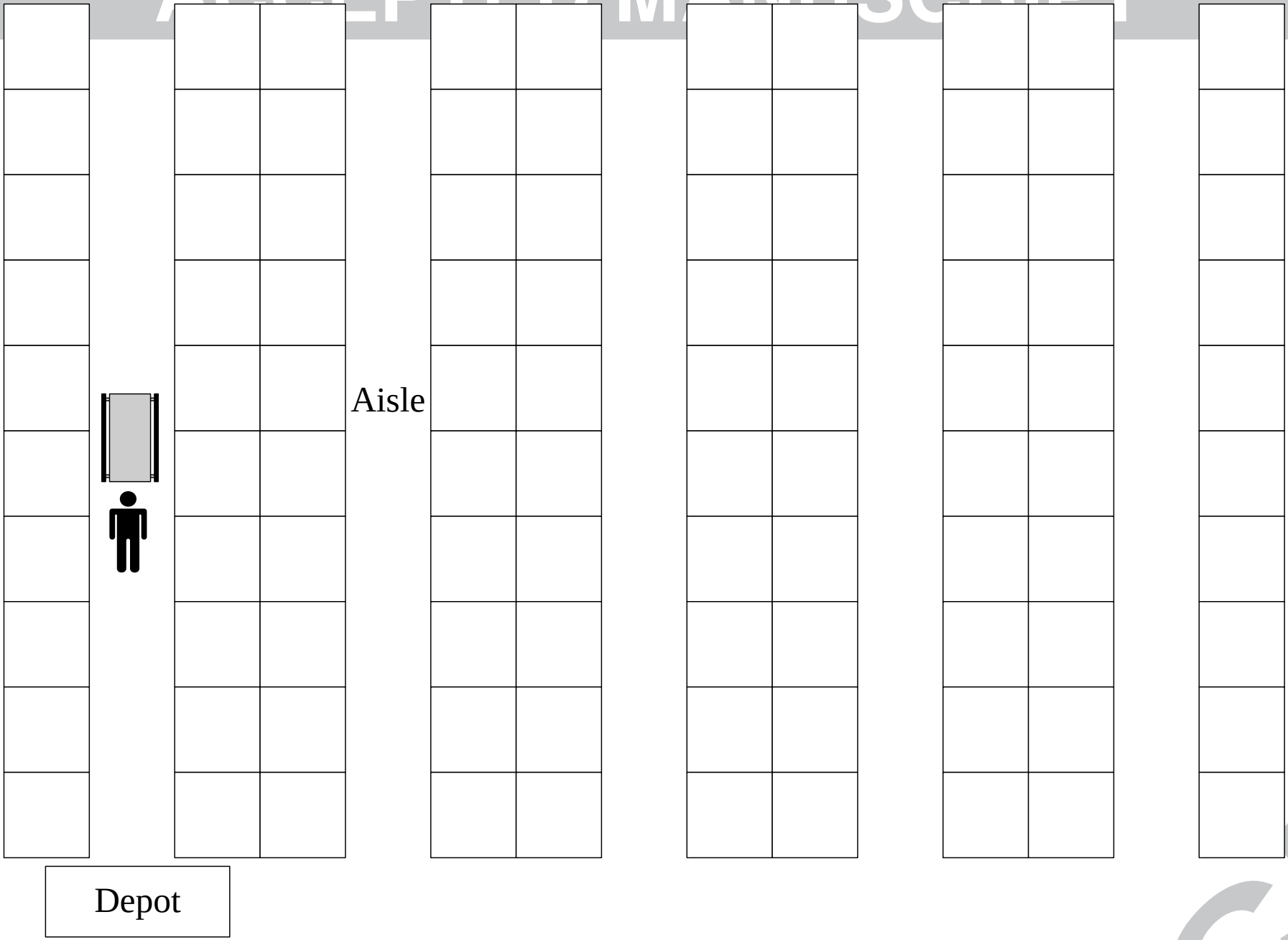
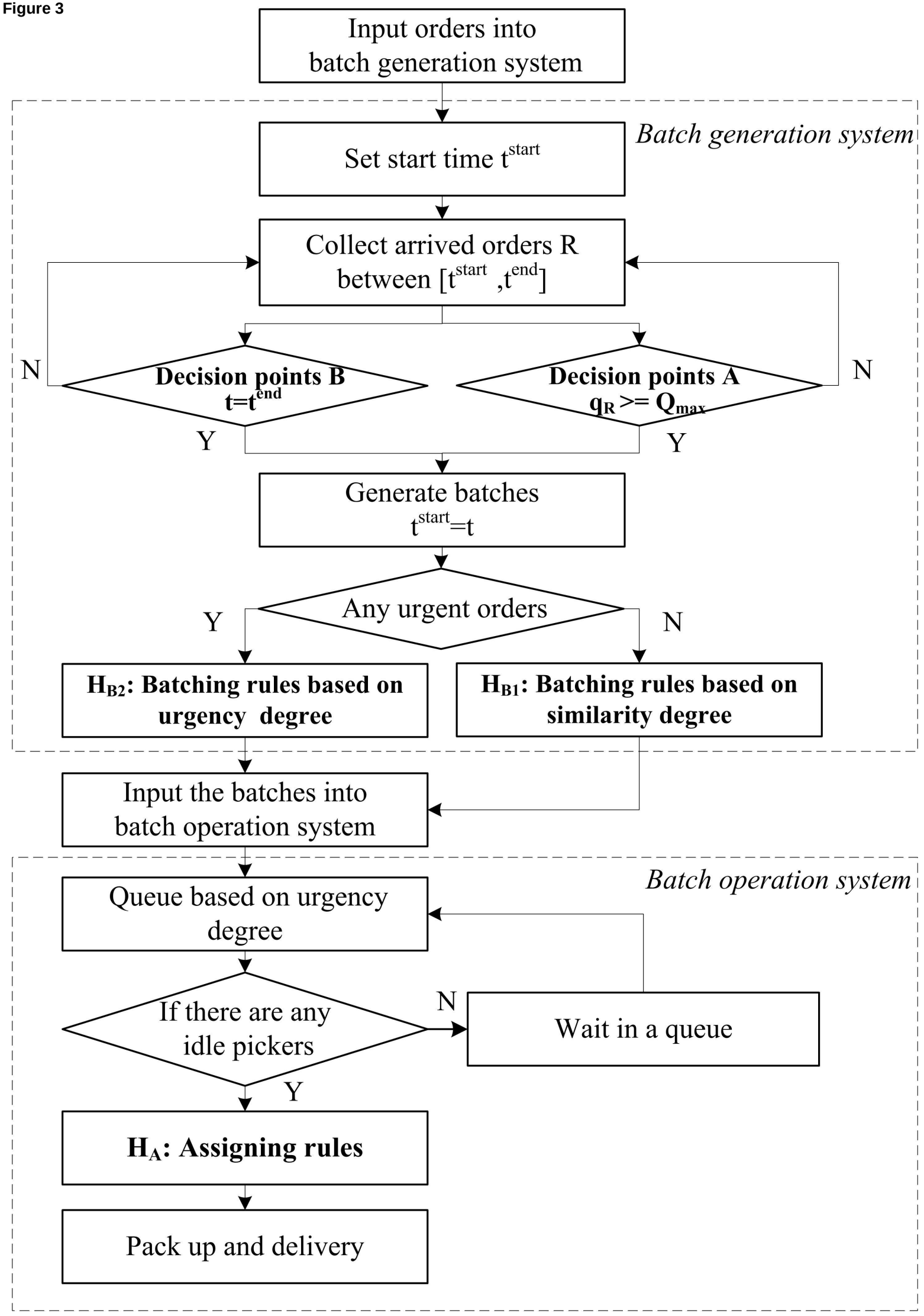


Figure 3


Input orders into batch generation system
Batch generation system
Set start time t^{start}
Collect arrived orders R between $[t^{start}, t^{end}]$
Decision points B $t=t^{end}$
Decision points A $q_R >= Q_{max}$
N
Y
N
Y
Generate batches $t^{start}=t$
Any urgent orders
Y
N
H_{B2}: Batching rules based on urgency degree
H_{B1}: Batching rules based on similarity degree
Input the batches into batch operation system
Batch operation system
Queue based on urgency degree
If there are any idle pickers
N
Wait in a queue
Y
H_A: Assigning rules
Pack up and delivery

- This paper deals with the integrated on-line scheduling of order batching and delivery problem.
- Two objectives including the number of delivered orders and service time are considered.
- Several novel rule-based solutions are proposed, including decision points, batching rules and assigning rules.
- The solutions define which orders are urgent and should be picked up directly, and which ones can be satisfied later.
- The results show that considering order's urgency degree is significant to increase the number of delivered orders.