A Model and Efficient Heuristics of Order-Picking with Replenishment in a Warehouse

Jie GONG, Dept. of Precision Engineering, The University of Tokyo
Jun OTA, Dept. of Precision Engineering, The University of Tokyo
Hirofumi TAMURA, Murata Systems LTD., Kyoto, Japan
Toshimitsu HIGASHI, Murata Machinery LTD., Inuyama, Japan

Abstract—In this paper, given the orders and stock in a warehouse, we present heuristics which aim to solve the order-picking with replenishment problem. An agent appointment problem is described with respect to appointing agents to replenish or pick separately at the same period. It aims to reduce the interactions between agents. The performances of the heuristics are evaluated.

Keywords: warehouse management, order-picking, dispatching rules, agent appointment

1. INTRODUCTION

Warehouse management aims to increase the efficiency of warehouse operations and provide reliable customer service. The main operations for warehouse include receiving, put-away, order-picking, and shipping. Some study has shown that order-picking accounts for as much as 55% of the total warehouse operating cost [1].

Nowadays, warehouse is divided down into a reserve area and a small fast picking area (forward area). The smaller the picking area, the lower average picking travel time will be. And with the trend of shrinking of forward area, the replenishment activity also becomes frequent and crucial. Furthermore, in order to deal with some urgent orders, it is necessary to replenish products during a busy picking period. The interesting decision problems such as the timing of the replenishment are becoming more and more popular [2].

Warehouse design is always divided into tactical or operational level [3]. In a tactical level, it is important to decide how many products are placed in the forward area, and how many products are stored in reserve area. The decisions concerning the problems described are commonly called the forward-reserve problem [4]. In this problem, regular replenishment quantity from the reserve to the forward area is concerned.

In the operational level, picking problem has been addressed as a routing method problem which aims to sequence the items on the pick list to ensure a good route [5] [6] [7]. Such a picking problem can be represented as a form of Travelling Salesman Problem (TSP) and solved with exact algorithms [8]. In these studies, one important assumption is: the stock (product quantity in the storage location) is unlimited, and there is no need to replenish the product supplies.

From the traditional studies, order-picking is the most cost activity in a warehouse. And the replenishment activity also becomes crucial. However, replenishment problem was only referred in tactical level such as forward-reserve problem. In the operational level, there are few studies referring the order-picking with replenishment. Thus, in this research, we aim to model and present efficient heuristics to solve order-picking with replenishment in the operational level.

The order-picking problem with replenishment is different from the simple order-picking problem because of the existence of replenishment. The main obstacle is the constraint that the picking agent cannot finish operation until the shortage products are replenished.

The agents do multiple picking or replenishing cycles (trips) wherein they must begin and end the trips in the shed or exit of AS/RS. Given a set of agents and two groups of generated routes (trips) which represent the order-picking operation (PT) and replenishing operation (RT) (Figure 1), we propose a heuristic aiming to decrease the maximum operational time (makespan).
The rest of this paper proceeds as follows. Section 2 gives an overview of order-picking with replenishment problem and then describes this problem in detail. Section 3 discusses the methodology of the research while Section 4 presents the experiment results. Finally, Section 5 concludes the paper.

2. PROBLEM STATEMENT

An overview of the warehouse picking problem is first given. The products are stored in stationary storage locations distributed across the warehouse. Each location holds quantities of unique type of products and has a unit quantity called case. For one storage location, there is a capacity such as 60 cases. If the quantity of products exceeds the capacity, the unloading time of exceeding part will be twice than normal.

Customer order lists and the stock-sheet (Figure 2) serve as inputs to the picking problem whereas the resulting sequence of picking or replenishing activities for the agents (agent schedule) as the output (Figure 3).

<table>
<thead>
<tr>
<th>Orders</th>
<th>Stock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Location</td>
</tr>
<tr>
<td>AA</td>
<td>1</td>
</tr>
<tr>
<td>BT</td>
<td>7</td>
</tr>
<tr>
<td>CY</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>EF</td>
<td>50</td>
</tr>
</tbody>
</table>

Figure 2 Orders and stock as input

<table>
<thead>
<tr>
<th>Agent</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Location</td>
</tr>
<tr>
<td>A1</td>
<td>7, 13, 18, exit; 18, 26, 33, exit; 13, 8, 56, exit;</td>
</tr>
<tr>
<td>A2</td>
<td>3, 22, shed; 5, 19, 35, shed; 35, 67, 87, shed;</td>
</tr>
<tr>
<td>A4</td>
<td>65, 77, shed; 4, 46, 48, 49, shed; 35, 67, 87, shed; 55, 78, 79, shed;</td>
</tr>
</tbody>
</table>

Figure 3 Agent Schedule as output

The orders placed on the warehouse should be picked by the agents from the storage locations and transferred to the shed where the orders will be loaded into trucks for delivery to the customers. An agent will start from the shed and pick up the products location to location. If the quantity reaches the capacity (60 cases) of the agent, the agent will return to the shed and load the products. Such a circle route can be called a picking trip.

As the products are picked, the stock will decrease, and it is necessary to replenish the shortage products to the storage locations. Similar to the picking process, the agent will upload the shortage products from exit of AS/RS, and replenish the products location to location. When the agent is empty, the agent will return to the AS/RS and upload products for next replenishing round. Such a circle route can be called a replenishing trip.

The replenishing trip and picking trip can be considered as the unit of replenishing operations and picking operations. We should schedule the agents to finish order-picking and replenishing operations as soon as possible.

For the shortage products, the picking agents cannot finish operation until the products are replenished. Before the replenishments coming, agents need to wait and the waiting time must be considered as delay time to implement in the model. In the practical problem, it is also necessary to consider other delays which are caused by interactions between the agents such as loading queue and collision avoidance.

The problem is presented as following, let:

\[ RT = \{r_1, r_2, ..., r_m\} \] a set of replenishment trips generated from the orders and stock (Figure 2).

\[ PT = \{p_1, p_2, ..., p_n\} \] a set of picking trips generated from the orders.

\[ A = \{a_1, a_2, ..., a_k\} \] a set of agents.

\[ m = \text{total number of RT to be assigned} \]

\[ n = \text{total number of PT to be assigned} \]

\[ k = \text{total number of agents} \]

\[ l = \text{number of agents to be appointed to replenish at beginning} \]

\[ c_i = \text{time cost of trip } i \]

\[ x_j^i = \begin{cases} 
1, & \text{if route } j \text{ is assigned to agent } i \\
0, & \text{otherwise} 
\end{cases} \]

\[ \mu_i = \sum_{j=1}^m c_j x_j^i \] ideal makespan of agent \( i \) (makespan of agent \( i \) without delay time).

\[ \mu_i' = \text{makespan of agent } i \text{ (makespan of agent } i \text{ with delay time) } \]

\[ M = \{\mu_1', \mu_2', ..., \mu_k'\} \] a set of agent makespans

Objective:

Minimize: max\( (M) \)

3. RESEARCH METHODOLOGY

This problem just described is so complex that it is not easily modelled and solved in one stage. To simplify it, the problem can be broken down into stages as shown in Figure 4.

Figure 4 Multistage solution to the picking problem
Given the orders and stock, Route Generate aims to create a set of RT, and a set of PT. In this paper, we applied a heuristic based on the simplest and extensive $S$-shape routing method.

Agent Appointment means that the agents need to be appointed to do replenishing or picking before the agents start working.

Routing Assignment is the most important stage in this problem. The generated trips are assigned to the agents. Then, we can get the ideal $makespans$ of agents without considering the delay time caused by agent interactions.

In a practical problem, the operational cost will be variable due to the agent interactions. Delay processing aims to deal with the interactions between the agents.

3.1. Route Generation and $S$-shape heuristic

Many studies apply the efficient routine method to generate picking route [3]. In this paper, we apply the $S$-shape heuristic based on the wide-practiced method which is called $S$-shape routing method. In the traditional $S$-shape routing method, the agent enters an aisle from one end and leaves the aisle from the other end like the shape of an $S$ (Figure 5). The heuristic we applied here is a modified $S$-shape routing method. The connected route between two locations in a trip is modified to be the minimal length path, even though the agent pairs up the product from one storage location by next location whose sequence was determined by the $S$-shape. If the picked product quantity is filled to the agent capacity, the agent will return to shed directly.

![Figure 5 S-shape for the warehouse](image)

S-shape Heuristic:
1. Search the shortage products based on the orders and stock.
2. Get the items of replenishment for each shortage product.
3. Generate RT,
   1. Generate the replenishing location list based on S-shape.
   2. Set the exit of AS/RS as the current location.
   3. If there are still unprocessed locations for routing, go to 2.4; else go to 3.
   4. Go to the first unprocessed location in the location list
   a. If the available route slack (The agent capacity minus the current quantity of items contained in the route) \( \geq \) current supply of the location, then add this location and set it as the current location. Repeat 2.4.
   b. Else if the route slack < current supply of the node, then fill the route with part of the supply equal to the route slack and set the available supply of location as the original location supply minus the original route slack.
3. Generate PT,
   1. Generate the picking list based on S-shape.
   2. Set the shed as the current location.
   3. If there are still unprocessed locations for routing, go to 3.4; else STOP.
   4. GO to the first unprocessed location in the location list
   a. If the available route slack (The agent capacity minus the current quantity of items contained in the route) \( \geq \) current supply of the location, then add this location and set it as the current location. Repeat 3.4.
   b. Else if the route slack < current supply of the location, then fill the route with part of the supply equal to the route slack and set the available supply of location as the original location supply minus the original route slack.
3.2. Agent Appointment

Next stage is Agent Appointment. There is a set of agents $A=\{a_1, a_2, \ldots, a_k\}$. After setting replenishment agent numbers $l (l \leq k)$, we can divide $A$ to a set of replenishing agents $ReA=\{ra_1, ra_2, \ldots, ra_l\}$ and a set of picking agents $PiA=\{pa_1, pa_2, \ldots, pa_{k-l}\}$.

The following conditions are also imposed on the Agent Appointment:
- The replenishment agent number $l$ is decided before the agents start working.
- If replenishment is finished, and there are still picking operations remaining, $ReA$ will convert to do order-picking.
- $ReA$ starts work from the exit of AS/RS. $PiA$ starts work from the shed. All of agents start work at the same time.

3.3. Route Assignment and FIFO heuristic

The operation assignment problem, which is similar to the job shop scheduling problem being well-studied in Flexible Manufacturing System (FMS), can be solved in a variety of approaches. For example, the currently popular methods like Simulated Annealing or Tabu based on the neighbourhood search. Furthermore, Genetic Algorithms (GA) and other methods such as neural networks can also be used to solve such problem. These methods, although effective, generally take a long computational time. In this problem, we need a faster way to assign the trips to the agents since leave an extension to the next stage dealing with the congestion between agents. Thus we select the dispatching rules as the basic heuristic.
In Route Assignment stage, RT and PT will be assigned to the agents, and for a given ReA, it will do replenishing firstly, and after finishing the replenishing operation at the last shortage location it will being assigned the picking trips, move to the first picking location to begin its picking operation. Figure 6 shows an example of route assignment. Our goal is to minimize the maximum the operation time (makespan).

<table>
<thead>
<tr>
<th>ReA</th>
<th>pt</th>
<th>pt</th>
<th>pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>PiA</td>
<td>pt</td>
<td>pt</td>
<td>pt</td>
</tr>
<tr>
<td>PiA</td>
<td>pt</td>
<td>pt</td>
<td>pt</td>
</tr>
</tbody>
</table>

Figure 6 A route assignment example

In order to assign the RT and PT to the agents, we implement FIFO heuristic based on "dispatching rules" in the area of job shop scheduling [9]. FIFO means the first trip generated in RG stage will be assigned firstly. Figure 7 provides the algorithmic structure of the FIFO heuristic.

![Algorithmic structure of FIFO heuristic](image_url)

Next, we will describe the heuristic in detail:

1. **Step 1**: Set the first generated replenishing trip in RT to be the current trip and the first ReA to be the current agent.
2. **Step 2**: Assign the current trip to the current agent.
   - Let the next earliest generated trip to be the new current trip. Let the ReA which has the shortest operation time to be the current agent.
   - IF there are still unassigned trips in RT, GOTO step 2 ELSE GOTO step 3.
3. **Step 3**: Set the first generated picking trip in PT to be the current trip, and assign the current trip to the current agent.
   - Let the next earliest generated trip to be the new current trip. Let the agent which has the shortest operation time to be the current agent.
   - IF there are still unassigned trip in PT, GOTO step 4. ELSE STOP.

3.4. **Delay Processing**

After assigning the RT and PT to the agents, we can get ideal makespans without considering the interactions between agents. In the practical problem, it is necessary to compose the delay time caused by interactions. There are several types of delays an agent may experience.

- **Loading queue**: The delay causes when agents have the demand of unloading or loading products in the same storage location. These interactions are not only variable but also unpredictable and can cause an agent to diverge from its base time significantly. If the two agents demand to do same operations (replenishing or picking) in the same location in the same period, the first arrived agent will do firstly, and the second one will wait until the first one leaves. The waiting time is called loading queue time.

- **Replenishment**: The delay causes when a picking agent will wait for the replenishment in the shortage storage location until the replenishing agent finishes replenishing the shortage. The waiting time is called replenishing waiting time. Those interactions can also cause significant delay in a bad agent appointment condition when the agents appointed to replenish are few and cannot finish replenishing shortage before the picking agents pick the corresponding products.

- **Punishment loading**: The delay causes when products quantity after replenishing exceeds the capacity of storage location. The loading time of exceeding part will become twice than normal. This type of delay is named punishment loading time.

- **Collision avoidance delays**: The delay occurs when an agent takes extra time to manipulate in order to avoid colliding with another agent.

Normally, collision avoidance delays are of secondary concern. This is because in practice, the number of agents is much less than the number of locations. And unless numerous agents constantly load items in the same area at the same time, these delays are going to remain small compared to the total delay.
operation time. Thus, we ignore the collision avoidance delays and focus on the other three delays.

4. RESULTS

We evaluate the performances of FIFO heuristic with different agent appointment in computer experiments. The Table I shows the environment of simulator.

<table>
<thead>
<tr>
<th>Table 1 Environment of simulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent Speed</td>
</tr>
<tr>
<td>Loading/unloading speed</td>
</tr>
<tr>
<td>Punishment loading speed</td>
</tr>
<tr>
<td>Average stock</td>
</tr>
<tr>
<td>Average picking items</td>
</tr>
<tr>
<td>Location size</td>
</tr>
<tr>
<td>Location number</td>
</tr>
<tr>
<td>Location capacity</td>
</tr>
<tr>
<td>Agent capacity</td>
</tr>
<tr>
<td>Loading time at Shed or exit of AS/RS</td>
</tr>
</tbody>
</table>

Figure 9 shows the average values of makespan (in seconds) and with 4 agents, 8 replenishment trips and 16 picking trips. The appointed ReA number is varied by $l=4, 3, 2, 1$ which present 4/0, 3/1, 2/2, 1/3 patterns with respect to the ratio of ReA number to PiA number all the computation time is less than 500 ms under the environment of Pentium4 1.2GHz with 1GB RAM. The results show that we can achieve significant improvement while ReA number is 4. It means that the pattern which all the agents replenish at first, and then convert to order-picking can show the best performance. Compared with 3/1, 2/2, 3/1 pattern, 4/0 is improved by 3.2%, 8.4% and 39%. The figure also indicates that as the number of ReA increases, the makespan will increase as well.

![Figure 9](image-url)  
Figure 9 Maximum operation time (makespan)

In the Figure 9, while there is only one ReA appointed (1/3 pattern), the performance is much worse than others. The reason can be found in the Figure 10 and Figure 11 which indicate the replenishing waiting time and loading queue time. It proves that a bad agent appointment can cause significant delays. As the replenishment waiting time increases, the operation time in one picking location will also extend. Thus, the loading queue time tends to increase as well with the increase of PiA number. Figure 12 indicates the punishment loading delays. As the increase of PiA number, we can achieve improvement of reducing the punishment loading time. Punishment loading time can not be improved any more if the parameters are fixed in the 4/0 pattern. But for the 3/1, 2/2 and 1/3, it is possible to minimize the punishment loading time by some scheduling methods. In this circumstance, the 3/1, 2/2 and 1/3 patterns have the potential to achieve makespan improvement and the efficiency will be totally increased.

![Figure 10](image-url)  
Figure 10 Replenishment waiting delays

![Figure 11](image-url)  
Figure 11 Loading queue delays

![Figure 12](image-url)  
Figure 12 Punishment loading time
5. DISCUSSION & CONCLUSION

In this paper, we introduced a model of the problem of order-picking with replenishment in a warehouse. Then FIFO heuristic based on dispatching rules was presented. The heuristic was applied for assigning some generated replenishing and picking trips to the appointed agents. And this heuristic was proved to be efficient to solve the order-picking with replenishment in an operational level. The results showed that if we appoint all the agents to replenish at first, the makespan would be best because of the fewer interactions between the agents, even though the punishment loading is longest. Furthermore, from the results of experiment proved that a bad agent appointment can cause significant delay. Nevertheless, because appointing the agent can decrease the punishment loading time, we will focus on reduce the punishment loading by modify the RT and PT sequence for the future work.

6. REFERENCES