The simultaneous reshuffle and yard crane scheduling problem in container terminals

Jiao Zhao, Lixin Tang*

The Logistics Institute, Northeastern University, Shenyang, 110819, China.
Email: Lixintang@mail.neu.edu.cn, jzhao@tli.neu.edu.cn

* corresponding author

Abstract

This paper studies the simultaneous reshuffle and yard crane scheduling problem in the yard of container terminals, which is one of the key logistics problems affecting the operation efficiency in container terminals. Although the reshuffle scheduling problem has received many research attention due to its importance, the problem without considering yard crane is not very practical, which is because the shuffle process needs yard crane to carry out, and the improper yard crane scheduling will weaken the optimized shuffle planning, therefore the reshuffle and yard crane scheduling should be considered simultaneously. The problem in this paper is to choose appropriate objective locations for the reshuffled containers and obtain the picking up sequence for all the containers on the yard crane, with the objective to reduce the sum of the total completion time for all the containers and the reshuffling time for all the reshuffled containers. A mixed integer model is proposed for the problem. A Particle Swarm Optimization (PSO) algorithm is employed to obtain near optimal solutions of the problem. The experiment results show that the solutions of the simultaneous scheduling problem are better than the results obtained by solving the reshuffle scheduling problem and the yard crane scheduling problem, respectively.

Keywords: yard crane, reshuffle, PSO, simultaneous scheduling problem

1. Introduction

Container trade worldwide increases steadily in recent years, and the container yard have to stack more and more containers. To cope with the growth of container trade, it is important to utilize the limited equipments and resources, for example, yard cranes and yard space. Because the limit of the yard storage space and the difference between the arrival sequence and retrieve sequence of containers, the reshuffle often happened. When ships with unloaded containers arrive at container terminals, the containers have to be unloaded from the ships by quay cranes, and transported to yard and waiting for external truck to take away. If the container to be retrieved earlier is under some containers that will be retrieved latter, these upper containers have to be moved to other positions, and then the container can be retrieved, it is said that reshuffle happened. The reshuffle operation is unproductive moves and should be avoided.

Figure 1 is an example of a container block with a yard crane. A tier is consisted by the containers with the same height, and a column is consisted by the containers with the same width. A container position means a slot. A bay only can be served by one yard crane in the same time because of the width limit of the bay.

The reshuffle scheduling problem and the yard crane scheduling problem are often researched separately in the pervious literatures, it is not very practical because the reshuffle operation are carried out by yard cranes. In the simple yard crane scheduling problem, reshuffles between containers may often happen, it is because that the retrieve sequence is decided without considering the feasibility in practice(Ng and Mak, 2005). For example, the real storing location of retrieved container is under some other containers which is retrieved latter than this container. Therefore, the study on the simultaneous reshuffle and yard crane scheduling problem is helpful to improve the productivity of
container terminals.

In this paper, we model the reshuffle and yard crane scheduling problem in container terminals to minimize the sum of the total completion time for all the containers and the reshuffling time which helps to improve the efficiency for container terminals. The solution of the reshuffle and yard crane scheduling problem will decide the sequence for all the containers to be retrieved and the completion time for each container, as well as the position that each blocking container to be moved to. A PSO algorithm is developed to solve the problem.

This paper is organized as follows. Section 2 gives a brief review of previous literatures related to reshuffle scheduling problem and yard crane scheduling problem in container terminals. The problem description is given in Section 3, and the model of the simultaneous reshuffle and yard crane scheduling problem is presented in this section. Section 4 describes the PSO algorithm for the simultaneous scheduling problem. Section 5 reports the experiments results of the simultaneous scheduling problem, as well as the results of the reshuffle scheduling problem and the yard crane scheduling problem solved separately. Finally, Section 7 gives the conclusions.

2. Literature Review

Due to the significance of reshuffle in container terminals, it has received much research attention. Kim (1997) estimates the expected number of reshuffles when picking up a random container and the total number of reshuffles to retrieve all the containers in a bay with given configuration, and he proposed a methodology to obtain the results. The weights of containers are considered by Kim et al. (2000) to decide the storage location for each arrival export container. In their research, containers are divided into three kinds, heavy, medium and light, and the heavier containers should be loaded onto ships earlier. A dynamic programming model with the objective to minimize the expected number of reshuffles for loading operation is presented in their paper. Wan et al. (2009) studies the assignment of storage locations to containers in a stack with the objective to minimize the total number of reshuffle.

There are many algorithms to solve the reshuffle scheduling problem, for example, Kim and Hong (2006) propose a branch and bound algorithm and a heuristic algorithm ENAR to minimize the number of reshuffles and determine the storage positions for reshuffled containers. Kang et al. (2006) present a method based on a simulated annealing search to obtain a good stacking strategy for containers with uncertain weight information, and experiments results show that the method is more effectively to reduce the number of reshuffles compared to the traditional same-weight-group-stacking strategy. The reshuffling index heuristic is proposed by Murty et al. (2005) to determine a storage position with the objective to minimize the possible caused number of reshuffle. The scheduling problem of equipment in yard have been researched by Kim and Kim (2003), Laik and
There are several literatures that both considered the scheduling sequence and reshuffle scheduling. Meisel and Wichmann researched the container sequencing problem for quay cranes with internal reshuffles for unloading containers in the arriving ship (2010). The simultaneous stowage and load planning for a container ship with container rehandle in yard stacks is researched by Imai et al. (2006), but they considered the stowage and load planning, the crane scheduling has not been researched.

Yard crane is one of the most important equipment to handle containers to storage in the yard or transfer the containers onto trucks to be transported. Ng and Mak (2005) studies the scheduling problem with a single yard crane, and there are some certain containers with different ready times have to be processed. The handling sequence of all the containers processed are needed to determined to minimize the total waiting time of the external trucks.

3. Problem Description and Formulation

3.1. Problem Description

In this problem, the configuration of the stack is known in advance, it means that the indexes of column and tier for each container is obtained. The elements to be determined are the retrieval sequence for all the containers in a certain stack and the objective locations for reshuffled containers, with the objective to minimize the sum of the total completion time for all the containers and the reshuffling time for all the blocking containers. An example of a container stack with a yard crane is shown in Figure 2.

In previous researches, the reshuffle scheduling problem and the yard crane scheduling problem are often studied separately. In the yard crane scheduling problem, the situation of reshuffle is usually ignored (Ng and Mak, 2005) or transformed into certain time and did not decide the retrieval sequence for containers, and in the reshuffle scheduling problem, the retrieval sequence for containers are considered as known (Wan et al., 2009). In this paper, we study the two problems simultaneously and decided the retrieval sequence for all the containers and the objective locations for the reshuffled containers, which not only make the total completion time for all the containers earliest, but also minimize the reshuffling time for all the blocking containers.

![Figure 2: An example of a container stack with a yard crane](image)

3.2. The model for the simultaneous reshuffle and yard crane scheduling problem

The reshuffle scheduling problem in container terminals was studied by Wan et al. (2009), in which the retrieve sequence was assumed as given information; and the yard crane scheduling problem has been researched by Ng and Mak (2005), and the reshuffle has not been considered in their paper.
Considering on the relationship between the above two problem and the models of the above two literatures, we formulate the model of the simultaneous scheduling problem, and the following symbols are used for defining the parameters and variables.

Parameters:

- \( N \) – The number of containers will be retrieved;
- \( C \) – The number of columns in a stack;
- \( P \) – The number of tiers in a stack;
- \( i,j \) – The index of container;
- \( d_{ij} \) – The traveling time for yard crane from container \( i \) to container \( j \);
- \( h_i \) – The handling time of container \( i \);
- \( r_i \) – The ready time of external truck for container \( i \);

Decision variables:

- \( t_i \) – The completion time of container \( i \);
- \( s_i \) – The retrieving sequence of container \( i \), \( s_i = 1, \ldots , N \);
- \( X_{ij} \) = \begin{cases} 1 & \text{if container } i \text{ is handled before container } j \\ 0 & \text{otherwise} \end{cases}
- \( X_{i, j, c, p} \) = \begin{cases} 1 & \text{if container } j \text{ is at position } p \text{ of column } c \text{ when retrieving container } s_i \\ 0 & \text{otherwise} \end{cases}
- \( u_{i,j} \) = \begin{cases} 1 & \text{if the column index of container } j \text{ is no less than that of container } s_i \\ 0 & \text{otherwise} \end{cases}
- \( v_{i,j} \) = \begin{cases} 1 & \text{if the column index of container } j \text{ is no greater than that of container } s_i \\ 0 & \text{otherwise} \end{cases}
- \( z_{i,j} \) = \begin{cases} 1 & \text{if containers } j \text{ and } s_i \text{ are in the same column} \\ 0 & \text{otherwise} \end{cases}
- \( y_{i,j} \) = \begin{cases} 1 & \text{if container } j \text{ is shuffled in the retrieval of container } s_i \\ 0 & \text{otherwise} \end{cases}
- \( w_{i,j,k} \) = \begin{cases} 1 & \text{if containers } j \text{ and } k \text{ are shuffled when retrieving container } s_i \\ 0 & \text{and container } k \text{ is at a higher position than container } j \text{ before reshuffling} \\ \text{otherwise} \end{cases}

Based on the definition of the parameters and variables, we give the model as follows:

\[
\begin{align*}
\text{Min} & \quad \sum_{i=1}^{N} t_i + \sum_{i=j=1}^{N} \alpha y_{i,j} \\
\text{s.t.} & \quad r_i + h_i \leq t_i, \quad i = 1, 2, \ldots , N \\
& \quad t_i - t_j \geq d_{ij} - (1 - X_{ij})M, \quad i, j = 1, 2, \ldots , N \quad \text{and} \quad i \neq j \\
& \quad X_{ij} + X_{ji} = 1, \quad i, j = 1, 2, \ldots , N \quad \text{and} \quad i \neq j \\
& \quad X_{ij} \in [0,1], \quad i, j = 1, 2, \ldots , N \quad \text{and} \quad i \neq j \\
& \quad \sum_{j \neq i \neq s_i} X_{ij} = N - s_i, \quad i, j = 1, 2, \ldots , N \quad \text{and} \quad i \neq j
\end{align*}
\]
Formula (1) is the objective to minimize the sum of the total completion time for all the containers and the total time of reshuffle. Constraints (2) is the definition of variable $t_i$, and describes the relationship between the completion time, ready time and handling time of each container.
Constraints (3), (4) and (5) define variable \( X_{ij} \), and guarantee the precedence relationship between different containers. Constraints (6) give the relationship between \( X_{ij} \) and \( s_i \), and it defines the retrieving sequence for container \( i \). Constraints (7) and (8) defined the variable \( u_{n,j} \), if \[ \sum_{c \in C} \sum_{p \in P} c x_{c,i,p} + \sum_{c \in C} \sum_{p \in P} c x_{c,j,p} + j \geq 1 \], and constraints (7) make \( u_{n,j} \) to be 1. If \[ \sum_{c \in C} \sum_{p \in P} c x_{c,i,p} - \sum_{c \in C} \sum_{p \in P} c x_{c,j,p} < 0 \], and constraints (8) make \( u_{n,j} \) to be 0. Constraints (9) and (10) defined the variable \( v_{n,j} \), if \[ \sum_{c \in C} \sum_{p \in P} c x_{c,i,p} - \sum_{c \in C} \sum_{p \in P} c x_{c,j,p} \geq 0 \], and constraints (9) make \( v_{n,j} \) to be 1. If \[ \sum_{c \in C} \sum_{p \in P} c x_{c,i,p} - \sum_{c \in C} \sum_{p \in P} c x_{c,j,p} < 0 \], and constraints (10) make \( v_{n,j} \) to be 0. Constraints (11) defined the variable \( z_{n,j} \), if container \( s_i \) and container \( j \) are in the same column, then \( z_{n,j} = 1 \). Constraints (12), (13) and (14) defined the variable \( y_{n,j} \), if \( z_{n,j} = 1 \) and container \( j \) is laid higher than container \( s_i \), then constraints (12) make \( y_{n,j} = 1 \). If \( z_{n,j} = 0 \), it means that container \( j \) and container \( s_i \) are in the different column, then constraints (13) make \( y_{n,j} = 0 \). If \( z_{n,j} = 1 \) and container \( j \) is laid lower than container \( s_i \), then constraints (14) make \( y_{n,j} = 0 \). Constraints (15) ensure that each container can only be at one slot. Constraints (16) ensure that each slot can only be occupied by at most one container when retrieving any container. Constraints (17) ensure that any one container can not float in air, there must be one or more container below it or it touch the ground. Constraints (18) ensure that one container must be reshuffled to another column which is not the column the container is at when retrieving container \( s_i \). Constraints (19) - (22) defined the variable \( w_{n,i,k} \). Constraints (23) describe the height relationship of containers \( j \) and \( k \) when retrieving container \( s_i \). Constraints (24) and (25) describe the relationship of variables \( x_{n,j,c,p} \) and \( y_{n,j} \). Constraints (26) and (27) assign known values to variable \( x_{n,j,c,p} \). Constraints (27) means that the container that has not to be reshuffled, it will stay at the initial position until being retrieved. Constraints (28) and (29) are binary constraints.

Now we take an example in the literature of Ng and Mak (2005) and give locations for each container to illustrate the simultaneous reshuffle and yard crane scheduling problem. For example, there are six containers have to be retrieved from the yard and transported by external trucks, and one yard crane is available in the stack. The bay number of these containers and the arrival time of external trucks waiting for these containers are shown in Table 1.

<table>
<thead>
<tr>
<th>Container no.</th>
<th>Location (column, tier)</th>
<th>( r_i )</th>
<th>( h_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2,2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2,1</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3,1</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1,2</td>
<td>15</td>
<td>4</td>
</tr>
</tbody>
</table>

A feasible solution of the example is shown in Table 2.

<table>
<thead>
<tr>
<th>Container no.</th>
<th>Sequence on crane</th>
<th>Reshuffle or not</th>
<th>The location after reshuffle (column, tier)</th>
<th>( t_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>no</td>
<td>/</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>yes</td>
<td>(4,1)</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>no</td>
<td>/</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>no</td>
<td>/</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>yes</td>
<td>4,1</td>
<td>24</td>
</tr>
</tbody>
</table>
4. PSO algorithm

PSO (Particle swarm optimization) is first proposed by Eberhart and Kennedy (1995), which is a kind of stochastic and swarm intelligence-based optimization algorithm, and with the advantage of fast convergence speed and the less setting parameters. PSO algorithm is effective to solve scheduling problems (Ching et al., 2007), and the simultaneous reshuffle and yard crane scheduling problem is a practical scheduling problem in container terminals. The practical problem usually has to be solved quickly, and the PSO algorithm is with the advantage of converging fast. Therefore, we adopt PSO algorithm to solve the simultaneous scheduling problem in this paper.

The common formulations for updating velocity and position is as bellowed:

\[
\begin{align*}
    v_{ik}^{t+1} &= v_{ik}^t + c_1 r_1 (x_{pbest_i}^k - x_{ik}^t) + c_2 r_2 (x_{gbest_k}^t - x_{ik}^t) \\
    x_{ik}^{t+1} &= x_{ik}^t + v_{ik}^{t+1}
\end{align*}
\]

In the formulations (30) and (31) \(v_{ik}^t\) and \(x_{ik}^t\) represent the velocity and the position of the \(i\)th particle on \(k\)th dimension in the \(t\)th iteration, respectively. \(c_1\) and \(c_2\) are acceleration weights, \(r_1\) and \(r_2\) are generated in \([0,1]\) randomly. \(x_{pbest_i}^t\) is the best position of the \(i\)th particle on \(k\)th dimension in the \(t\)th iteration, and \(x_{gbest_k}^t\) is the best position of the global best particle on \(k\)th dimension in the \(t\)th iteration, from above we can see that, PSO is concerned to the own position of the particle and the positions of the whole swarm.

4.1. Initial particles generation and solution representation

We generate the initial population randomly to assure the diversity of the particles.

The problem in this paper is to decide the retrieval sequence of all the containers and the objective locations of the reshuffled containers in the stack. The retrieval sequence of all the containers can be done independently. Therefore in the proposed PSO we use a particle to represent the retrieval sequence of all the containers. Because there are \(N\) containers in total, we define a particle as an \(N\)-dimensional vector. The \(i\)th element of the \(N\)-dimensional vector, \(X_i\), indicates the retrieval sequence of container \(i\).

Corresponding to each particle is an retrieval sequence of containers. We now present a method to obtain a retrieval sequence of containers. Let \(R\) be the set of values of \(X_i\) in the particle. We rank the containers in \(R\) in non-decreasing order of their value of \(X_i\), and then obtain a set \(R'\) and the retrieval sequence of containers.

With the obtained retrieval sequence a simple heuristic is proposed to decide the objective locations for all the reshuffled containers. The detail procedure is as follows.

**Step 1**: \(i=1\). For the container \(i\) in \(R'\), if there is any container \(j\) \((j=1,2,\ldots,N\ and \ j\neq i)\) which is in the same column with container \(i\) and is in the upper tier than container \(i\), go to step 2.

**Step 2**: If there are some empty locations, select one empty location randomly and lay container \(j\) there. Otherwise, find one container of which the retrieval sequence is bigger than container \(j\), randomly, and lay the reshuffled container on this container, then the second reshuffle of container \(j\) can be avoided.

**Step 3**: If there is any container \(k\) \((j=1,2,\ldots,N\ and \ j\neq i)\) which is in the same column with container \(i\) and is in the upper tier than container \(i\), go to step 2; Otherwise, go to step 4.

**Step 4**: If all containers in \(R'\) are considered, stop; Otherwise, \(i+1\) and go to step 1.
4.2. Velocity updating strategy

The major merit of PSO algorithm is that it has very fast speed to converge, but accordingly the major fault of PSO algorithm is that it is easy to fall into the local optimal solution. This is because particle only flies towards the best position it has reached and the global best position of all the particles. The problem in this paper is for scheduling and it is suitable to be solved by PSO algorithm. Nevertheless the problem itself is hard to solve and the short of PSO, it is difficult to find the better solutions by the standard PSO algorithm. For this reason, Shi and Eberhart (1998) proposed the idea of inertia weight, and modified the velocity update formulation as below:

\[ v_{ik}^{t+1} = w_{ik}^{t} + c_1 r_1 (x_{pbest}^{t} - x_{ik}^{t}) + c_2 r_2 (x_{gbest}^{t} - x_{ik}^{t}) \]  

(32)

In the formulation (32), the coefficient \( w^i \) is the inertia weight, and \( w^i = \frac{w_{t_{w}}^1 - t}{w_{t_{w}}^1} (w_{t_{w}}^1 - w_{t_{w}}^1) + w_{t_{w}}^1 \), where \( w_{t_{w}}^1 \) and \( w_{t_{w}}^1 \) are the biggest and the smallest value of the inertia weight. In this paper, we adopt formula (32) to update the velocity in PSO algorithm.

Generally, PSO algorithm converge fast, but it is easy to be trapped into a local optimum. This is because each particle flies based on its reached best position and the global best position of all the particles. We add a disturbance strategy to the PSO algorithm to overcome this drawback. We initialize \( \lambda \) particles of the particles when the best solution has not been updated after \( \gamma \) iterations.

The detailed method is as followed: when the best solution has not been updated after some iterations, we select several particles randomly, and initialize them again. If the new solution of the particle is better than its current solution, accept the new position value of the particle, and update the best position of the particle; otherwise, remain the current position of the particle.

4.3. Construction of PSO algorithm

In this paper, the PSO procedure to solve the simultaneous reshuffle and yard crane scheduling problem is as below:

**Step1.** Initialize L particles as a swarm. Set iteration number \( \tau = 1 \).

**Step2.** For \( l = 1, 2, \ldots, L \), encode a set of container \( R \), and obtain the retrieval sequence for all the containers in the stack. Then find the shuffled containers and reshuffling time based on the retrieval sequence.

**Step3.** For \( l = 1, 2, \ldots, L \), calculate the fitness value, which is equal to that the sum of the completion time for all the containers and the total reshuffling time.

**Step4.** Update pbest which is the best position of every particle.

**Step5.** Update gbest which is the best position of the swarm.

**Step6.** Update the velocity and the position of each particle.

**Step7.** If \( \tau = \gamma \), select \( \lambda \) particles, and initialize them. If the new solution of the particle is better than its current best solution, update the best position of the particle.

**Step8.** If the stopping criterion is met, when \( \tau = T \), stop. Otherwise, \( \tau = \tau + 1 \) and return to step 2.

5. Computation Experiments

In this section, the PSO algorithm is coded by C++ language. Experiments are carried out on a personal computer with Intel core 2 CPU running at 2.83 Ghz.
All the experiment data are generated based on Ng and Mak (2005) and Wan et al. (2009). The retrieval time for one container ranges from 2 to 4 min, the traveling time between two container is from 0 to 10 min, the ready time for each container generated from an exponential distribution with a mean of 120/N (N is the container number) min randomly. The stacks are of 6 columns in all the experiments, and tier ranges from 2 to 5. The number of container in the stacks is not less than 0.2(column-1)tier+1, 0.5(column-1)tier+1 and 0.8(column-1)tier+1, respectively. The locations of each container is generated randomly, and not floating, if any, to touch the ground or the lower containers which is at the same column. In our PSO algorithm, we set γ = 3 and λ = 3 based on experiments.

In order to test the performance of our proposed simultaneous scheduling problem, we first solve the yard crane scheduling problem presented by Ng and Mak (2005) using CPLEX 11.0 software to obtain the total completion time for all the containers, which is expressed as F1. Second, we solve the reshuffle scheduling problem described by Wan et al. (2009) using CPLEX 11.0 software, and then get the number of reshuffle, in order to compare with the results of our proposed simultaneous scheduling problem, we multiply the number of reshuffle by a certain value which is received by the practical time for one single reshuffle in container terminal) and transfer it into values by units of time, and we use F2 to indicate the result. At last, F1 plus F2 is the sum of the total completion time for all the containers and the total reshuffling time, which is recorded as F.

The results of the simultaneous scheduling problem in this paper is solved by PSO algorithm, and we record it as F'. Then we compare F' with the value of F, which is obtained by solving the yard crane scheduling problem and the reshuffle scheduling problem separately. The detail is shown in Table 3.

Table 3: Comparison results between the separately solved problem and simultaneous scheduling problem

<table>
<thead>
<tr>
<th>N</th>
<th>Stack size</th>
<th>YCS model (F1 and cpu time)</th>
<th>MRIP model (F2 and cpu time)</th>
<th>F</th>
<th>PSO (F' and cpu time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C 1 1</td>
<td>6-2-3</td>
<td>27, 0.25</td>
<td>0, 0.50</td>
<td>27</td>
<td>27, 0.015</td>
</tr>
<tr>
<td>C 1 2</td>
<td>6-2-6</td>
<td>90, 0.75</td>
<td>2, 0.25</td>
<td>98</td>
<td>94, 0.031</td>
</tr>
<tr>
<td>C 1 3</td>
<td>6-2-9</td>
<td>192, 6.26</td>
<td>3, 0.75</td>
<td>204</td>
<td>192, 0.079</td>
</tr>
<tr>
<td>C 2 1</td>
<td>6-3-4</td>
<td>44, 0.26</td>
<td>0, 0.26</td>
<td>44</td>
<td>44, 0.015</td>
</tr>
<tr>
<td>C 2 2</td>
<td>6-3-8</td>
<td>161, 1.01</td>
<td>2, 0.26</td>
<td>169</td>
<td>167, 0.063</td>
</tr>
<tr>
<td>C 2 3</td>
<td>6-3-9</td>
<td>212, 10.00</td>
<td>3, 1.75</td>
<td>224</td>
<td>220, 0.063</td>
</tr>
<tr>
<td>C 3 1</td>
<td>6-4-5</td>
<td>69, 0.25</td>
<td>1, 0.25</td>
<td>73</td>
<td>73, 0.031</td>
</tr>
<tr>
<td>C 3 2</td>
<td>6-4-9</td>
<td>196, 8.50</td>
<td>2, 0.25</td>
<td>204</td>
<td>200, 0.078</td>
</tr>
<tr>
<td>C 3 3</td>
<td>6-4-11</td>
<td>282, 368.11</td>
<td>5, 4.25</td>
<td>302</td>
<td>298, 0.11</td>
</tr>
<tr>
<td>C 4 1</td>
<td>6-5-6</td>
<td>90, 3.25</td>
<td>1, 0.28</td>
<td>94</td>
<td>90, 0.046</td>
</tr>
<tr>
<td>C 4 2</td>
<td>6-5-9</td>
<td>175, 6.00</td>
<td>3, 0.75</td>
<td>187</td>
<td>187, 0.079</td>
</tr>
<tr>
<td>C 4 3</td>
<td>6-5-11</td>
<td>270, 351.52</td>
<td>3, 1.50</td>
<td>282</td>
<td>280, 0.109</td>
</tr>
</tbody>
</table>

From the results in Table 3, we can see that the solutions of the simultaneous scheduling problem are better than or equal to the solutions of the reshuffle scheduling problem and the yard crane scheduling problem solved separately, and the computation time for the simultaneous scheduling problem is less than 1s for all the problems by PSO algorithm. It shows that PSO algorithm is effective to solve the simultaneous scheduling problem.

6. Conclusions

The efficiency of large container terminals greatly depends on the effectiveness of resource schedule. Reshuffle is a key factor to block the efficiency of the container terminals, and the reshuffle operation is executed by yard crane, the performance of reshuffle is influenced by the yard crane scheduling. This paper formulates a integer-programming model about the simultaneous reshuffle and yard crane scheduling problem in container terminals. A PSO algorithm is used to solve this simultaneous
scheduling problem. Computation experiments indicate that the simultaneous scheduling problem can obtain more better solution than the reshuffle scheduling problem and the yard crane scheduling problem solved separately.

Acknowledgements

This research is partly supported by State Key Program of National Natural Science Foundation of China (71032004), the Fundamental Research Funds for the Central Universities (N090104002, N100704002).

References


Laik, N. and Hadjiconstantnou E. (2008), Container assignment and yard crane deployment in a container terminal: a case study, Maritime Economics & Logistics 10: 90-107


Wan, Y.W., Liu, J.Y. and Tsai, P.C. (2009), The assignment of storage locations to containers for a container stack, Naval Research Logistics 56(8): 699-713.
