A HYBRID FLOW SHOP SCHEDULING MODEL FOR LOADING OUTBOUND CONTAINERS IN CONTAINER TERMINALS

Qingcheng ZENG
Lecturer
Transport and Logistics College
Dalian Maritime University
1, Linghai Road, Dalian
116026 China
Fax: +86-411-84726756
E-mail: zqcheng2000@tom.com

Zhongzhen YANG
Professor
Transport and Logistics College
Dalian Maritime University
1, Linghai Road, Dalian
116026 China
Fax: +86-411-84726756
E-mail: yangzhongzhen@263.net

Abstract: This paper discusses the scheduling problem for loading outbound containers in container terminals. The problem is to determine a schedule that minimizes the makespan or time being taken to load a given set of outbound containers. An integrated model based on hybrid flow shop scheduling problem is developed. Two Metaheuristic algorithms are designed to solve the proposed model, the procedures are: initializing container sequence first; then allocating containers according to certain partitioning rule; and then improving the sequence by simulated annealing. Numerical experiments are conducted to test the performance of the proposed model and algorithms.

Keywords: Container terminal; Simulated annealing; Hybrid flow shop problem

1 INTRODUCTION

The trade globalization has significantly increased the demand of containerized marine transport. Container terminals become the important nodes of logistics network, which serve as hubs for the transshipment of containerized goods from ship to ship or from ship to other transport modes. With the increased volume of container traffic, mega container terminals, especially those in Asia, have been working at, or close to, their capacity. In addition, the rising competition among ports has compelled them to improve their service, which makes the efficiency of port operation an important factor to succeed in the fierce competition.

For most container terminals, there are mainly three types of equipments involved in the loading and unloading process, i.e., quay cranes (QCs), yard trailers (YTs) and yard cranes (YCs). Upon a ship’s arrival, QCs unload containers from or load containers onto the ship, and YTs move containers from quayside to storage yard and vice versa. At the storage yard, YCs perform the loading and unloading of yard trailers.

Scheduling the operation order of QCs, dispatching YTs to containers, allocating the optimal storage location for each container, and dispatching YCs to YTs in storage yard are major problems for the optimization of loading/unloading process in the terminals. Evidently, these problems are interrelated. Here, we focus on the integrated and coordinated scheduling of loading outbound containers.

This paper is organized as follows. In Section 2, we give a brief review of previous works. A model based on hybrid flow shop problem is formulated in Section 3. A procedure for calculating makespan lower bounds is developed in section 4. Two metaheuristic algorithms to solve the problem is described in Section 5. Numerical examples are used to test the performance of the algorithm in Section 6. And then conclusions are given in Section 7.
2 LITERATURE REVIEW

Issues related to container terminal operations have gained attention and have been extensively studied recently due to the increased importance of marine transport systems. Here, we provide a brief review of existing studies related to the operation scheduling optimization in container terminals.

The QCs are the main bottleneck of the efficiency of the container terminals, and their operation plan determines the turnaround time of a ship in the terminals. Daganzo (1989, 1990) suggested an algorithm for determining the number of QCs assigned to ship-bays of multiple vessels. Kim et al. (2004) developed a mixed-integer programming model considering various constraints related to the operation of QCs, and proposed a heuristic search algorithm to solve the problem.

Yard vehicles are used to transfer containers between the quay and the yard. Most of Studies about routing problem in container terminals are focus on automated guided vehicle (AGV) and straddle carrier. Evers et al. (1996) developed a hierarchical AGV control system by using semaphores. Liu et al. (2002) compared different AGV dispatching rules in container terminals. Vis et al. (2005) develop a heuristic based on the maximum flow problem to determine the fleet size of AGVs. Kim et al. (1999) developed models and algorithms to optimize the routing of straddle carrier. Considering YTs, Nishimura et al. (2005) proposed ‘dynamic routing’ trailer assignment method, and developed a heuristic algorithm to solve the problem.

The scheduling of YCs determines the efficiency of the terminal to a great extent. Research focused on scheduling of YCs has been conducted widely. Zhang et al. (2002) discussed the dynamic crane deployment problem, he formulated the problem as a mixed integer programming model and solved it by Lagrangean relaxation. Linna et al. (2003) proposed an algorithm and a mathematical model for the optimal yard crane deployment problem. Kim et al. (2003) developed a dynamic programming model to optimize the receiving and delivery operations of outside trucks, and derived the decision rule by learning based method. Ng WC et al. (2005) examined the problem of scheduling multiple yard cranes to perform a given set of jobs with different ready times in a yard zone with only one bi-directional traveling lane.

The operations in terminals have the features of multi-objectives, uncertainty, and complexity. Most of the existing literatures focus on a special sub-process. Very little of the existing literature focuses on the integrated scheduling problem of various types of handling equipment used in container terminals. Bish (2003) provided models and algorithms to integrate several sub-processes; the problem is (i) to determine a storage location for each unloaded container, (ii) to dispatch vehicles to containers, and (iii) to schedule the loading and unloading operations on the cranes, so as to minimize the maximum time it takes to serve a given set of ships. And developed a heuristic algorithm based on formulating the problem as a transshipment problem. Kang et al. (2006) proposed a model to schedule YTs routing and to dispatch vehicles to containers and developed a genetic algorithm to solve the model, so as to minimize the totally idle time of the related equipment and YTs travel time. Chen et al. (2006) proposed an integrated model based on hybrid flow shop problem to schedule different equipments in container terminals, he designed tabu search algorithm to solve the model. In his algorithm, the operation orders of all three stages are optimized, thus the computation time is long. These models and algorithms improve the coordination and integration of the operation scheduling in container terminals. However, how to tackle the complex constraints
and interrelation; how to improve the computation efficiency; and how to realize coordination among different sub-process are the problems that have not been solved well.

Due to the strong interdependence of the different sub-process, an integrated model is developed in order to: (1) coordinate the scheduling of various types of equipment at the same time so as to achieve a high level integrated optimization; (2) minimize the makespan or time being taken to load a given set of outbound containers.

3. PROBLEM DESCRIPTION AND MODEL FORMULATION

3.1 Hybrid flow shop scheduling problem

The hybrid flow shop scheduling problem (HFSS) can be stated as follows. Consider a set $J = \{1, 2, ..., n\}$ of $n$ jobs, each is to be processed in $S$ consecutive stages. Stage $s$ has a set of $M(s)$ identical machines, with $M(s) = \{m_s\}$, $s = 1, 2, ..., S$. In each stage $s$, there are $m_s \geq 1$ parallel identical machines, with $m_s \geq 2$ for at least one stage, $s = 1, 2, ..., S$.

Let $p_{is}$ be the processing time of job $i$ at stage $s$. Each machine can process only one job once. Since all machines at each stage are identical and preemptions are not allowed, to define a schedule, it suffices to specify the completion times for all tasks. Let $C_{is}$ be the ending time of the $s$th stage of job $i$. Therefore, the HFSS is to find a schedule to minimize the maximum completion time $C_{\text{max}}$, with $C_{\text{max}} = \max C_{is}$.

The HFSS has recently received attention because of its importance from both theoretical and practical points of view. Lin and Liao (2003) presented a case study in a two-stage hybrid flow shop with sequence-dependent setup time and dedicated machines. Kurz and Askin (2003) explored three kinds of heuristics for flexible flow lines with sequence-dependent setup times. Jin et al. (2006) discussed the three stages HFSS; and proposed two metaheuristic algorithms first sequence and then allocate jobs to machines based on a particular partition of the shop.

3.2 Scheduling problem for loading outbound containers

Container terminal operations can be divided into two parts based on the operations on vessels: loading outbound containers and discharging inbound ones. The process of loading outbound containers involves three stages: yard cranes will pick up the desired containers from yard blocks and load them onto the yard trailers, then yard trailers transport the containers to quay cranes, and the quay cranes finally load the containers onto the vessels. The loading process in container terminals is similar to HFSS, in order to formulate the problems, the definitions and principles are stated as follows:

For loading outbound containers, each container must undergo three handling operations: a transfer operation within storage yard done by YCs; a transfer operation of a container onto the ship; a transfer operation between QCs and YCs done by YTs. And there are three different sets of machines: QCs, YCs, and YTs. Thus, a job can be defined as a complete loading process for a container.

Comparing with the classical hybrid flow shop scheduling problem, our problem has several
unique characteristics as follows:

- **Job precedence constraints**: For loading, containers in the hold must precede the containers on the deck of the same vessel.
- **Blocking**: There is limited or no buffer at all between two successive machines, therefore, blocking happens when the buffer is full. For example, when the YT carries a container to a QC that is handling another container, the YT cannot unload the container.
- **Setup times**: In container terminals, there is empty movement when a crane or a YT moves between two containers. For example, once a YT carries an outbound container to a QC, it has to make an empty trip to the storage yard in order to proceed next container. We denote it as **setup time**.

Due to the characteristics, the scheduling problem for loading outbound containers is formulated as HFSS problem. The objective is 1) to assign each operation to a machine 2) to sequence the assigned operations on each machine, therefore, to minimize the makespan of the loading operations.

### 3.3 Model formulation

In order to formulate the scheduling problem for loading outbound containers, the following parameters and decision variables are defined:

**Problem parameters:**

- $N$: the set of all containers (jobs);
- $n$: the number of containers;
- $i, j$: container index;
- $s$: stage index, $s = 1, 2, 3$;
- $m$: machine index;
- $m_s$: the number of machines at stage $s$;
- $M_{is}$: the set of machines to process container $i$ at stage $s$;
- $E_m$: the set of containers that might be processed on machine $m$;
- $B$: the set of pairs of containers between which there is precedence relationship, when container $i$ must precede container $j$;
- $p_{is}$: processing time of container $i$ at stage $s$;
- $w_{gs}$: setup time between container $i$ and $j$ at stage $s$;
- $G$: a sufficiently large constant.
- $H$: a sufficiently large constant.

**Decision variables:**

- $x_{ism}$ = 1, if operation of container $i$ at stage $s$ is assigned to machine $m$; 0, otherwise;
- $y_{ijsm}$ = 1, if operation of container $i$ and $j$ at stage $s$ are assigned to the same machine $m$; 0, otherwise;
- $z_{ijsm}$ = 1, if operation of container $i$ immediately precedes $j$ on machine $m$ at stage $s$;
- $t_{is}$: the starting time of container $i$ at stage $s$;
- $C_{max}$: the completion time of the last container.
The scheduling model can be formulated as follows:

\[
\text{Min } C_{\text{max}} = \text{Max}(t_i + p_i) \\
\text{s.t. } \sum_{m \in M_i} x_{ism} = 1, \forall i \in N, \forall s \in \{1,2,3\} \\
t_{is} \geq 0, \forall i \in N, \forall s \in \{1,2,3\} \\
t_{is} + p_{is} \leq t_{(s+1)}, \forall i \in N, \forall s \in \{1,2,3\} \\
y_{ijsm} = y_{jism}, \forall i,j \in E_m, \forall s \in \{1,2,3\}, \forall m \in M_{ij} \\
y_{ijsm} \leq 0.5(x_{ism} + x_{jsm}), \forall i,j \in E_m, \forall s \in \{1,2,3\}, \forall m \in M_{ij} \\
\sum_{j \in E_m} z_{jism} \leq 1, \forall i \in E_m, \forall s \in \{1,2,3\}, \forall m \in M_{ij} \\
\sum_{j \in E_m} z_{jism} \leq 1, \forall i \in E_m, \forall s \in \{1,2,3\}, \forall m \in M_{ij} \\
x_{ism} \leq 0.5(z_{jism} + z_{jism}) \leq x_{ism} - 0.5, \forall i,j \in E_m, \forall s \in \{1,2,3\}, \forall m \in M_{ij} \\
t_{(s+1)} + s_{js} \leq t_{js} + H(1 - z_{jism}), \forall i,j \in E_m, \forall s \in \{1,2,3\}, \forall m \in M_{ij} \\
t_{is} \leq t_{js}, \forall i,j \in B, \forall s \in \{1,2,3\} \\
x_{ism}, y_{ijsm}, z_{jism} = 0 or 1, \forall i,j \in N, \forall s \in \{1,2,3\}, \forall m \in M_{ij}
\]

The objective function (1) is to minimize the makespan. Constraints (2) guarantee that each operation must be processed by exactly one machine. Constraints (3) ensure that each operation begins after time zero. Constraints (4) ensure that the order of operation for each container is respected. Constraints (5) ensure that the order of operation for each container is respected. Constraints (5) and (6) ensure that \(y_{ijsm} = y_{jism} = 1\) when \(x_{ism} = x_{jsm} = 1\). Constraints (7) and (8) ensure that each operation has at most one predecessor and successor on machine \(m\). Constraints (9) are flow balance constraints, ensure that each container is processed in well defined sequences. Constraints (10) determine the starting time of the operation and define the blocking constraints. Constraints (11) determine the set of pairs of jobs between which there is precedence relationship. Constraints (12) are simple binary constraints.

It is well known that the HFSS problem is NP-hard, thus, scheduling problem for loading outbound containers is also NP-hard. It is doomed unable to obtain optimal solutions for large-scale problems. Hence, metaheuristic algorithms are required to obtain near-optimal solutions efficiently. Here, simulated annealing algorithm, one of the commonly used metaheuristics, is used to solve the proposed problem.

### 4. LOWER BOUND

To determine the effectiveness of the heuristics, we need to compare the makespan obtained by the proposed metaheuristics algorithms with the optimal one obtained by the solution of above mixed-integer program. However, finding the optimal makespan requires the solution of a mixed-integer program, which can be quite time-consuming even for medium-sized problems. Therefore, the optimal solution is measured against well-defined lower bounds.
Lower bound problem is widely discussed in HFSS problem. Here, we derive the lower bound on the optimal makespan based on the method proposed by Stantos (1995).

Let $LS(i,s)$ denote the left-hand side sum of processing time from stage 1 to $s-1$ for job $i$, and $RS(i,s)$ the right-hand side sum of processing time from stage $s+1$ to $S$ for job $i$. $LS(i,s)$ and $RS(i,s)$ are given by

\begin{align}
LS(i,s) &= \begin{cases} 
\sum_{l=1}^{s-1} p(i,l), & s > 1 \\
0, & s = 1
\end{cases} \\
RS(i,s) &= \begin{cases} 
\sum_{l=s+1}^{S} p(i,l), & s < 1 \\
0, & s = S
\end{cases}
\end{align}

$JL(k,s)$ is the $k$th value in the ascending order list of $LS(i,s)$ for all jobs in stage $s$, and $RL(k,s)$ is the $k$th value in the ascending order list of $RS(i,s)$ for all jobs in stage $s$. Based on these denotations, Stantos proposed stage-based lower bounds, $lb_s$ and global lower bound, $glb$ for HFSS problem as follows:

\begin{align}
lb_s &= \frac{1}{m_i} \left[ \sum_{k=1}^{m_i} JL(k,s) + \sum_{j=1}^{n} p(i,s) + \sum_{k=1}^{m_i} JR(k,s) \right], s = 1,2,\ldots,S \\
glb &= \max_{0 \leq s \leq S} \{lb_s\}
\end{align}

During the container loading process, a setup time is needed after each operation is finished. Therefore, the makespan lower bound must account for these setup times. Considering that the container sequence for each machine and the exact values of the setup times are not known, we develop a notation $SET^{s,i}_j$ to obtain the minimum possible setup time for each container.

For each container $i$, let $SET^{s,i}_j$ denote the minimum time required to set up the container immediately preceding container $i$ at stage $s$, $SET_{ij}$ denote setup time between container $i$ and container $j$ at stage $s$.

\begin{align}
SET^{s,i}_j &= \min_j SET_{ij}
\end{align}

Therefore, the stage-based lower bounds, $LB_s$ and global lower bound, $LB$ for scheduling problem for loading outbound containers can be formulated as follows:

\begin{align}
LB_s &= \frac{1}{m_i} \left[ \sum_{k=1}^{m_i} JL(k,s) + \sum_{j=1}^{n} p(i,s) + \sum_{k=1}^{m_i} JR(k,s) + \sum_{i=1}^{n} SET^{s,i}_j \right], s = 1,2,3
\end{align}
\[ LB = \max_{1 \leq s \leq 3} \{LB_s\} \quad (19) \]

5 METAHEURISTICS ALGORITHMS

5.1 Solution process
In this section, we propose two approaches based on job partition and simulated annealing. In the approaches, the scheduling problem for loading outbound containers is partitioned into \( s \) parallel machines at first. Then, for a given initial job sequence, the containers are allocated to the parallel machines in stage 1 according to the FAM or LFM rule. For \( s \geq 2 \), containers are allocated to machines according the FCFS rule. Finally, the initial container sequence is improved by simulated annealing (SA). Therefore, we label them SA-FAM-FCFS and SA-LFM-FCFS, where the ‘FAM’ refers to first available machine, ‘LFM’ refers to last free machine, and ‘FCFS’ refers to first come first served.

According to the FAM rule, the container is assigned to the first machine that becomes available, i.e., the machine that finishes the job previously assigned to it first. According to the LFM rule, the container is assigned to the last machine that becomes available, i.e., the machine with the smallest idle time among all available machines. Because our scheduling problem has no intermediate storage, those containers whose handling operations finished earlier will be delivered to the next stage as soon as possible. The FCFS rule is applied when building the container sequences in the stages other than the first stage. Precedence constraints are considered before inserting a container and applying the FCFS rule, so as to ensure that a feasible schedule can be obtained.

5.2 SA algorithm
SA is based on the simulation of the annealing of solids and applied to solving large-scale combinatorial optimization problems. A standard SA procedure begins by generating an initial solution randomly. At each stage, the new solution taken from the neighborhood of the current solution is accepted as the new current solution if it has a lower or equal cost; if it has a higher cost it is accepted with a probability that decreases as the difference in the costs increases and as the temperature of the method decreases. The temperature is periodically reduced by a temperature scheme, it moves gradually from a relatively high value to near zero as the method progresses. Therefore, at the start of SA most worsening moves are accepted, but at the end only improved ones are likely to be accepted. The method converges to a local optimum as the temperature tends zero. The SA based algorithm for scheduling problem for loading outbound containers is shown in Fig.1.

5.3 Encoding method
To use SA algorithm solving the scheduling problem, an encoding method to represent the feasible solutions is used. The feasible solutions for the problem are coded into strings of integer numbers. Each string represents a loading sequence, and its length is equal to the number of outbound containers.

5.4 Generation mechanism of neighborhood solution
To implement the SA algorithm, we need to generate a sequence of iterations, of which each is composed of changing the current solution in a designed way to create a neighborhood solution. The general mechanism of neighborhood solution deployed here is as follows: randomly select two containers in the string then swap the loading order between the two containers.
5.5 Acceptance criterion for the neighborhood solution

Once a neighborhood solution is generated, the following criterion is adopted to judge whether to accept it or not:

Let \( \Delta = f(s) - f(s_0) \) \hspace{1cm} (20)

Where, \( s_0 \) denotes the current solution and \( s \) denotes the neighborhood solution generated from current solution. \( f(\ast) \) denotes the objective function value.

A random number \( r \) in \( (0, 1) \) is generated from a uniform distribution, \( T \) represents the current temperature. If

\[ r \leq e^{-\Delta/T} \] \hspace{1cm} (21)

Then the neighborhood will be accepted as the current solution, else, the current solution will remain unchanged.

---

**Diagram: SA based algorithm for scheduling loading outbound containers in container terminals**

- **Initialize the loading sequence for outbound containers**
- **Neighborhood solution to loading sequence**
- **Dispatch containers to YCs according to FAM or LFM rule**
- **Dispatch containers to YT\(s\) and QC\(s\) according to FCFS rule, obtain \( C_{max} \)**
- **Acceptance criterion**
  - Satisfied:
    - **YES**
    - **Stopping criterion**
      - Satisfied:
        - **YES**
          - **The end**
      - Satisfied:
        - **NO**
          - **Temperature updating scheme**
  - **NO**
    - **Replace**

---

**Fig.1 SA based algorithm for scheduling loading outbound containers in container terminals**
5.6 Temperature updating scheme
The temperature is updated by the following formula:

\[ T_{i+1} = \frac{T_i}{1 + \beta T_i}, i = 1, \ldots, K - 1 \]  

(22)

Where \( \beta \) is the rate parameter in terms of the initial temperature \( T_i \), stopping temperature \( T_k \) and iteration number \( K \):

\[ \beta = \frac{T_i - T_k}{(K - 1)T_i T_k} \]  

(23)

6. NUMERICAL TESTS
To assess the solution quality and efficiency of the proposed Metaheuristic algorithms, experiments are conducted and analyzed. The algorithms are implemented in Matlab7.0 and run on personal computer with 1.7GHZ processor and 512 MB random access memory.

6.1 Problem settings
Data generator is developed to produce instance sets with specific characteristics. The parameters for terminal layout are based on the data received from Port of Dalian. Different scenarios are constructed by changing the number QCs, YCs, and YTs. Details are as follows:
- There are 16 different scenarios (according to the number of containers and equipments) to be test.
- The processing time of QCs are generated from uniform distribution of \( U(100,180) \), and the processing time of YCs follows the uniform distribution of \( U(40,70) \)
- Storage location in yard for each container is selected randomly. The storage location determines the processing time for YTs.
- The setup times in stage 1 and stage 2 are determined by the storage location for each container, and the setup times in stage 3 depend on the ship stowage plan.
- Precedence relations among containers are considered.

6.2 Performance of the proposed algorithms
The results are compared with the global lower bound. The relative deviation (RD) is calculated by the following formula:

\[ RD = \frac{C_{\max} - LB}{LB} \times 100 \]  

(24)

Where \( C_{\max} \) is the makespan obtained by the proposed algorithms and LB is its lower bound.

Two SA based heuristics are compared in terms of both solution quality and efficiency. These two heuristics are defined as:
- SA1: Initialize container sequence first, then allocate containers according to FAM-FCFS rule, and improve the sequence by SA, namely, SA-FAM-FCFS.
- SA2: Initialize container sequence first, then allocate containers according to LFM-FCFS rule, and improve the sequence by SA, namely, SA-LFM-FCFS.

The experimental results are summarized in Table 1, which includes the RD of the makespan obtained from the SA based algorithms and their computing times. The results illustrate the effective of both SA1 and SA2. On the whole SA2 performs better than SA1, this is because
that the LFM rule considers two stages and tries to improve the future assignment of the currently unscheduled containers whereas the FAM rule only considers a single stage. The relative deviation (RD) increases with the increase of container number. This is because that not only the lower bound but also the makespan obtained by SA1/SA2 deteriorates as the container number increases.

### Table 1 Performance comparison of SA1 with SA2

<table>
<thead>
<tr>
<th>Problem size</th>
<th>RD(%)</th>
<th>CPU times (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QC</td>
<td>YC</td>
<td>YT</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>200</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>200</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>400</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>400</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>500</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>500</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>600</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>600</td>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>

Moreover, experiments are conducted to compare our method SA1 with method proposed by Chen (2006). In his paper, Chen designed Tabu search algorithm (TA1) to solve the HFSS in container terminals. Result is shown as Table 2. It shows that RD obtained by SA1 is more than that of TA1; and the CPU times is less than TA1. This is because that TA1 method tries to optimize the operation order of all three stages; but SA1 method only optimizes the loading order, the operation order in each stage is determined by dispatching rules. Therefore, the computation time of SA1 is less than that of TA1; but the solution quality of SA1 is less than that of TA1. With the increasing of container ship size, the number of loading or unloading containers often reaches to several thousands. Thus, considering the computation time, SA1 is more suitable to practical scheduling problem than TA1.

### Table 2 Performance comparison of SA1 with method proposed by Chen

<table>
<thead>
<tr>
<th>Problem size</th>
<th>RD(%)</th>
<th>CPU times (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QC</td>
<td>YC</td>
<td>YT</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>200</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>400</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>500</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

Based on the results of the 16 sets of tests, Fig.2 shows the difference of RD between SA1
and SA2. We can find that the difference of RD between SA1 and SA2 decreases with the increase of containers number. The explanation for this is that the ratio between the number of containers and the number of equipments increases with the increase of containers. It is expected that the dispatching of containers to equipments, one of the two aspects for the scheduling problem, becomes less important.

![Relative deviation of the SA1 and SA2](image)

**Fig.2 Relative deviation of the SA1 and SA2**

7. CONCLUSIONS

In this paper, an integrated scheduling model for loading outbound containers in container terminals is developed; the model is an extension of the hybrid flow shop scheduling problem. A procedure for calculating makespan lower bounds is presented, then two Metaheuristic algorithms (SA-FAM-FCFS and SA-LFM-FCFS) based on job partition and simulated annealing are designed. Numerical tests are provided to illustrate the efficiency of the proposed metaheuristic algorithms. Results show that SA-LFM-FCFS performs better than SA-FAM-FCFS. In addition, the assignment rules (such as LFM and LFM) are important for the scheduling problem and helpful for the improvement of the solution.

The proposed model involves the problem of determining the loading sequence, scheduling and dispatching various kind of equipment (QCs, YCs, YT) simultaneously. Therefore, comparing with models that optimize different equipments respectively, it can improve the coordination among different equipments, and enhance the integration of operation scheduling in container terminals.

REFERENCES


Kim Kap Hwan.(2004) A crane scheduling method for port container terminals, European Journal of operational research, 156, 752-768
Nishimura Etsuko, Imai Akio, Stratos.(2005) Yard trailer routing at a maritime container terminal, Transportation Research Part E, 41, 53-76