Available online at www.sciencedirect.com
ScienceDirect
Computers & Industrial Engineering 56 (2009) 979–992
computers & industrial engineering
www.elsevier.com/locate/caie

Maximizing the number of dual-cycle operations of quay cranes in container terminals

Haipeng Zhang [a], Kap Hwan Kim [b,*]

[a] Department of Industrial Engineering, Pusan National University, Room #10605, Building 10, Pusan National University, Busan 609-735, Republic of Korea
[b] Department of Industrial Engineering, Pusan National University, Room #10619, Building 10, Pusan National University, Busan 609-735, Republic of Korea

Available online 17 September 2008

Abstract

An important objective of all schedules in a port is to minimize turnaround time of vessels. An operation schedule for Quay Cranes (QCs) can significantly affect the turnaround time of a vessel. Recently dual cycling techniques have already been used for reducing the number of operation cycles of QCs in some advanced container terminals. This study attempts to minimize the number of operation cycles of a QC for discharging and loading containers in a ship-bay, which is equivalent to maximizing the number of dual cycle operations. A formulation in QC scheduling problems is proposed as a mixed integer programming model. A hybrid heuristic approach is proposed to solve this model. The algorithm applied in this study takes two types of sequencing into account, i.e. inter-stage sequencing (hatch sequencing) and intra-stage sequencing (stack sequencing in the same hatch). This approach hybridizes a certain reconstructive Johnson's rule with an effective local search method. Finally, certain data in real cases are used to evaluate the effectiveness of the proposed approach.

Keywords: Dual cycling; Quay crane scheduling; Container terminals

1. Introduction

Recently, more and more people recognize the importance of logistic business via container terminals. Especially in Asia, for example in Hong Kong, Busan, Shanghai, and Shenzhen, their governments have been constantly made an investment to enlarge the throughput of their own sea ports during the past several years, even though all these cities are geographically closed to each other. As a result of increasing competitions between container ports, improving the efficiency in container terminals has become an important and immediate challenge for all managers in order to gain higher competitiveness.

One of the most important performance measures in container terminals is the turnaround time of vessels, which consists of the discharging and the loading operation times. In most container ports, there are three

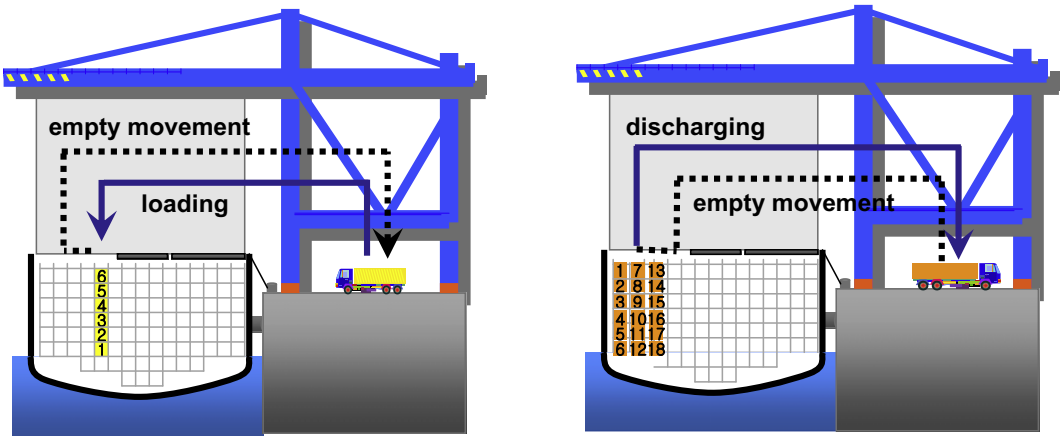
* Corresponding author. Tel.: +82 51 510 2419; fax: +82 51 512 7603.
E-mail addresses: hpzhang@pusan.ac.kr (H. Zhang), kapkim@pusan.ac.kr (K.H. Kim).

0360-8352/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2008.09.008

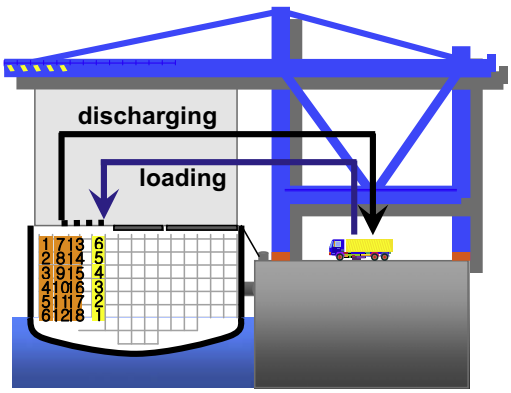
main types of equipments involved in the loading and discharging process, i.e., Quay Crane (QC), Yard Truck (YT), and Yard Crane (YC). For an example of typical working flow during the loading operation for outbound containers, a YC will pick up a desired container from a container block and load it onto a YT waiting aside. The YT will then transport the container to a QC, which will finally load the container onto the containership. In reality, the QC schedule can significantly affect the turnaround time of a vessel, because QCs are the most expensive single unit of handling equipment in port container terminals, and for this reason, one of the key operational bottlenecks at ports is quay crane availability (Crainic & Kim, 2007). Hence, some researchers have put great efforts into the QC scheduling problems.

An operational method to improve the productivity of QCs is the dual cycle operation in discharging and loading operations. However, most of container terminals adopt the single cycle method for the QC operation in which the QC usually handles loading activities after all unloading tasks have been finished that means the QC's activities in those cases will make more empty movements compared to the dual cycling model as shown in Fig. 1. The dual cycling allows QC to discharge a container in the same cycle as a loading operation, thus doubling the number of QC tasks in one cycle, and decreasing the empty movements. Such a kind of efficiency improvement can observably reduce the ship turn-around time so as to increase the berth productivity. Some container terminals have already begun to use this strategy in practice.

Related to the discharging and the loading operations, Daganzo (1989) firstly studied the static and dynamic QC scheduling problems considering the environment of multiple container vessels. He proposed an effective algorithm for optimizing the number of cranes to assign to ship-bays. Peterkofsky and Daganzo (1990) developed a branch and bound solution method to minimize a weighted sum of the unloading times of



(a) Single loading (or discharging) leads to empty movement



(b) No empty movement for dual cycle

Fig. 1. Comparing the single cycle and dual cycle QC movement.

ships in their later research. Kim and Park (2004) discussed more practical QC scheduling in terms of assuming the time windows during which QCs are assigned to that vessel are given. Furthermore, they applied an effective heuristic search algorithm called GRASP (Greedy Randomized Adaptive Search Procedure) to the QC scheduling problem for overcoming the computational difficulty of the previous branch and bound method. Recently, Lee, Wang, and Miao (2008) adapted Genetic Algorithm to find an optimal handling sequence of holds for QCs, and also considered the case of interference between different QCs. Zhu and Lim (2005), Liu, Wan, and Wang (2006), and Moccia, Cordeau, Gaudioso, and Laporte (2006) addressed the quay crane scheduling problem. However, no study on quay crane scheduling problems considered the dual cycle operations. Some studies (Bish, 2003; Imai, Sasaki, Nishimura, & Papadimitriou, 2006; Kim, Kang, & Ryu, 2004) addressed sequencing loading and discharging operations of individual containers.

The dual (double) cycle operations have not been explicitly considered in the previous studies except Goodchild (2005), Goodchild and Daganzo (2006), Goodchild and Daganzo (2007). They suggested a scheduling method only for stacks under a single hatch cover for the first time and showed the Johnson's rule can be applied to sequencing discharging and loading tasks for stacks. They also provided a method to evaluate the effect of the dual cycle operations on the reduction in the number of cycles during the ship operation and also analyzed the impact of the dual cycle operations on the land side operations (Goodchild & Daganzo, 2007).

This study extends the study by Goodchild and Daganzo (2006) to the sequencing problem not only for stacks under a hatch cover but also for hatches. We reformulate the QC scheduling problem with dual cycles by a mixed integer programming model. We decompose the problem into two parts: inter-stage sequencing (hatch sequencing) and intra-stage sequencing (QC tasks sequencing in the same hatch). Moreover, a hybrid heuristic approach is proposed to solve the model. Our study compares solutions by the proposed approach with the optimal solutions by using data from real cases.

This paper is organized as follows. The next section gives a brief definition of the QC scheduling problem with dual cycles by using a simple typical example. Section 3 precisely defines the mathematical formulation of the QC scheduling problem with dual cycles. Section 4 proposes a hybrid heuristic approach combining with an effective local search procedure for solving the general model. Section 5 uses several real cases from a container terminal in Busan to validate the proposed approach, and further discusses the results of numerical experiment. Finally, we draw some conclusions in Section 6.

2. Problem description

In general, the goal of our research is to minimize the total number of cycles of QC activities on one ship instead of the traditional makespan. This substitution can be accomplished based on two assumptions as follows:

(A1) We consider the number of cycles as a proxy of the operation time for a QC to complete all the discharging and the loading operations in a ship-bay.
(A2) The operation of QCs is the bottleneck during the vessel operation and so the throughput rate of QCs determines the throughput rate of the integrated handling system consisting of QCs, transporters, and yard cranes.

2.1. Deck hatches

Regarding container vessels, there are always two or three hatch covers (which depend on the size of the bay) in each bay. Hatch covers can separate the stacks of containers into two or three parts above the deck and in the hold. The steel covers inevitably lead to some precedence constraints among different QC tasks as follows: (1) all the unloading activities under the deck (in the hold) cannot begin until all the unloading activities above (on the deck) are completed; (2) all the loading activities on the deck cannot begin until all the loading activities below (in the hold) are completed.

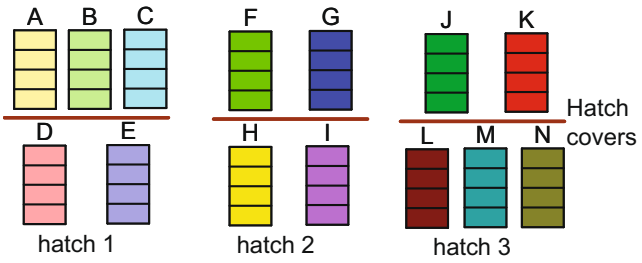


Fig. 2. Simple example of general case of stack positions.

Table 1
Data set of the simple example

Index of hatches	Index of stacks	Deck or hold	No. of containers to unload	No. of containers to load
1	A	D	4	3
	B	D	2	0
	C	D	5	7
	D	H	8	6
	E	H	4	6
2	F	D	5	5
	G	D	8	2
	H	H	2	6
	I	H	6	3
3	J	D	6	5
	K	D	1	5
	L	H	9	3
	M	H	5	7
	N	H	8	6

A simple example is illustrated to describe the characteristics and backgrounds of the problem, which is shown in Fig. 2. In this case, 14 stacks and 3 hatch covers in a vessel are supposed to be handled by a QC and the optimal loading and unloading sequence must be found. Table 1 gives the data corresponding to the simple case.

Following the situation of hatch covers discussed above, we can draw some precedence constraints according to the 3 hatches as follows: in hatch 1, unloading{A, B, C} → unloading{D, E} and loading{D, E} → loading{A, B, C}; in hatch 2, unloading{F, G} → unloading{H, I} and loading{H, I} → loading{F, G}; in hatch 3, unloading{J, K} → unloading{L, M, N} and loading{L, M, N} → loading{J, K}.

3. Mathematical formulation

To precisely formulate the mathematical model, some notations in Goodchild (2005) are used and additional notations are defined as follows:

Indices:

h: the index of hatches
i, j: the indices of stacks

Input parameters:

m: the number of hatches
n_i^u: the number of containers to unload from stack i

n_i^l: the number of containers to load into stack i
$\bar{u}_h$: the number of containers to unload from the hold of hatch h where: $\bar{u}_h = \sum_{i \in \bar{S}_h^u} n_i^u$
$\underline{u}_h$: the number of containers to unload from the deck of hatch h where:
$\underline{u}_h = \sum_{i \in \underline{S}_h^u} n_i^u$
$\bar{l}_h$: the number of containers to load into the hold of hatch h where: $\bar{l}_h = \sum_{i \in \bar{S}_h^l} n_i^l$
$\underline{l}_h$: the number of containers to load onto the deck of hatch h where: $\underline{l}_h = \sum_{i \in \underline{S}_h^l} n_i^l$

Sets

H: the set of hatches
S^u: the set of stacks for which unloading operation is planned
S^l: the set of stacks for which loading operation is planned
$\bar{S}_h^u$: the set of stacks, for which unloading operation is planned, in the hold of hatch h
$\bar{S}_h^l$: the set of stacks, for which loading operation is planned, in the hold of hatch h
$\underline{S}_h^u$: the set of stacks, for which unloading operation is planned, on the deck of hatch h
$\underline{S}_h^l$: the set of stacks, for which loading operation is planned, on the deck of hatch h

Procedure variables:

C_i^u: completion time of unloading all containers from stack i
C_i^l: completion time of loading all containers into stack i
$\bar{C}_h^u$: completion time of unloading all containers from the hold of hatch h where: $\bar{C}_h^u = \max_{i \in \bar{S}_h^u} \{C_i^u\}$
$\bar{C}_h^l$: completion time of loading all containers into the hold of hatch h where: $\bar{C}_h^l = \max_{i \in \bar{S}_h^l} \{C_i^l\}$
$\underline{C}_h^l$: completion time of unloading all containers from the deck of hatch h where: $\underline{C}_h^u = \max_{i \in \underline{S}_l^h} \{C_i^l\}$
$\underline{C}_h^l$: completion time of loading all containers onto the deck of hatch h where: $\underline{C}_h^l = \max_{i \in \underline{S}_h^l} \{C_i^l\}$

Decision variables:

X_{ij}: binary variable for permutation of unloading stacks, where:

$$X_{ij} = \begin{cases} 1, & \text{if unloading for stack j is performed immediately after unloading for stack i} \\ 0, & \text{otherwise} \end{cases}$$

Y_{ij}: binary variable for permutation of loading stacks, where:

$$Y_{ij} = \begin{cases} 1, & \text{if loading for stack j is performed immediately after loading for stack i} \\ 0, & \text{otherwise} \end{cases}$$

As the objective function, we use w to represent the number of cycles necessary to complete all the QC tasks, and M is an arbitrary large number. The whole mathematical model of the general case is written as the following:

$$\min \quad w \tag{1}$$
$$\text{s. t.} \quad w \geq \bar{C}_h^u \quad \forall h \in H \tag{2}$$
$$w \geq \bar{C}_h^l \quad \forall h \in H \tag{3}$$
$$w \geq \underline{C}_h^u \quad \forall h \in H \tag{4}$$
$$w \geq \underline{C}_h^l \quad \forall h \in H \tag{5}$$
$$\bar{C}_h^u \geq C_i^u \quad \forall i \in \bar{S}_h^u, \ \forall h \in H \tag{6}$$

$$\bar{C}_h^l \geq C_i^l \quad \forall i \in \bar{S}_h^l,\ \forall h \in H \tag{7}$$
$$\underline{C}_h^u \geq C_i^u \quad \forall i \in \underline{S}_h^u,\ \forall h \in H \tag{8}$$
$$\underline{C}_h^l \geq C_i^l \quad \forall i \in \underline{S}_h^l,\ \forall h \in H \tag{9}$$
$$C_i^u \geq \underline{C}_h^u + n_i^u \quad \forall i \in \bar{S}_h^u,\ \forall h \in H \tag{10}$$
$$C_i^l \geq C_i^u + n_i^l \quad \forall i \in S^u \cap S^l \tag{11}$$
$$C_i^l \geq \bar{C}_h^l + n_i^l \quad \forall i \in \underline{S}_h^l,\ \forall h \in H \tag{12}$$
$$C_j^u - C_i^u + M(1 - X_{ij}) \geq n_j^u \quad \forall i, j \in S^u \tag{13}$$
$$C_j^l - C_i^l + M(1 - Y_{ij}) \geq n_j^l \quad \forall i, j \in S^l \tag{14}$$
$$\sum\nolimits_{j \in S^u} X_{0j} = 1 \tag{15}$$
$$\sum\nolimits_{j \in S^u} X_{ij} = 1 \quad \forall i \in S^u \tag{16}$$
$$\sum\nolimits_{i \in S^u} X_{ij} = 1 \quad \forall j \in S^u \tag{17}$$
$$\sum\nolimits_{i \in S^u} X_{iT} = 1 \tag{18}$$
$$\sum\nolimits_{j \in S^l} Y_{0j} = 1 \tag{19}$$
$$\sum\nolimits_{j \in S^l} Y_{ij} = 1 \quad \forall i \in S^l \tag{20}$$
$$\sum\nolimits_{i \in S^l} Y_{ij} = 1 \quad \forall j \in S^l \tag{21}$$
$$\sum\nolimits_{i \in S^l} Y_{iT} = 1 \tag{22}$$
$$X_{ij} = (0, 1) \quad \forall i, j \in S_h \tag{23}$$
$$Y_{ij} = (0, 1) \quad \forall i, j \in S_h \tag{24}$$

In the notations of X_{0j}, Y_{0j}, X_{iT}, and Y_{iT}, subscripts 0 and T are used to represent the start and the completion of unloading and loading operations, respectively. Constraints (2)–(5) give the definition of the object function w, which is to minimize the make-span in the unit of the number of cycles for the QC. Constraints (6)–(9) ensure the concepts of the completion times. Considering only one hatch (see Fig. 3), constraint (10) ensures that the unloading activities in a hold cannot begin until all the unloading activities on the deck of the same hatch are completed. Constraint (11) ensures that the loading activities in a stack cannot begin until all the unload-

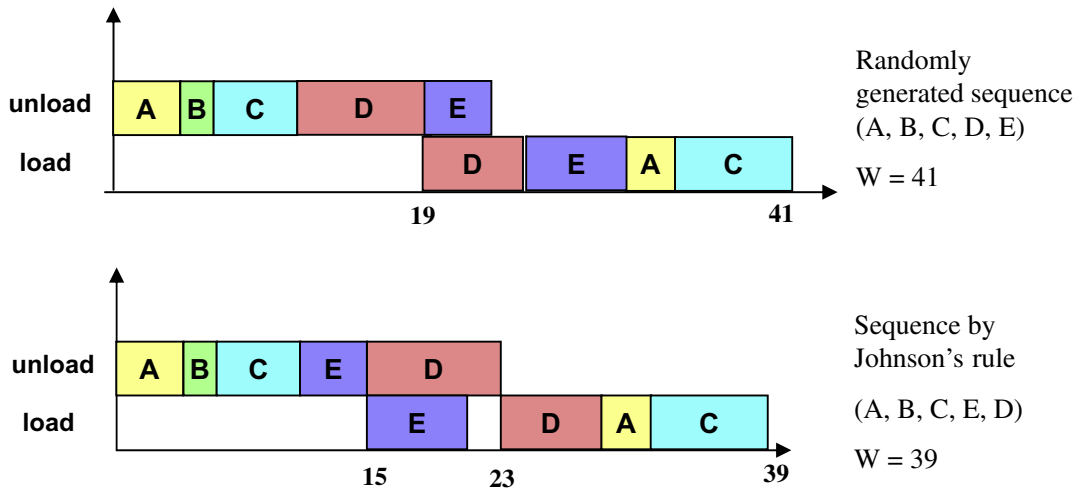


Fig. 3. Comparing two stack sequences in hatch 1.

ing activities in the same stack are completed, while constraint (12) ensures that in each hatch the loading activities on deck cannot begin until all the loading activities in hold of the same hatch are completed. Constraints (13) and (14) describe the definition of X_{ij} and Y_{ij}. Constraints (15)–(22) are used to define the sequence of unloading and loading tasks.

4. A heuristic algorithm

First, we decompose the whole QC scheduling into two parts: intra-stage optimization (sequencing all the stacks in one hatch) and inter-stage optimization (sequencing all the hatches).

4.1. Intra-stage optimization for stack sequencing

For example, if we consider the stacks belongs to the first hatch as shown in Table 1, then, as suggested by Goodchild (2005, 2006), we can find the followings: (1) only all the stacks in hold can be treated as the tasks of two-machine flow shop scheduling problem; (2) Johnson's rule (Baker, 1974) can solve to sequence the stacks in a hold; (3) the sequence of the stacks on the deck will not affect the solution.

Applying Johnson's rule, we can obtain an optimal stack sequence for hatch 1, comparing with another feasible solution in Fig. 3, the number of cycles was reduced from 41 to 39. We can get the optimal stack sequence for each hatch for the problem in Table 1 as shown in Fig. 4.

4.2. Inter-stage optimization for hatch sequencing

Suppose that the stack sequences of each hatch are optimized by using Johnson's rule. We illustrate an arbitrary hatch sequence in Fig. 5. The total number of cycles of the sequence in Fig. 5 is 84, and the sum of idle times between different hatches is 21. The main objective is to maximize the dual cycling movements. In other words, the optimal sequence for different hatches is the sequence of different hatches minimizing the number of single cycle operations. Changing sequence of hatches will lead to a more magnificent difference in the number of cycles comparing with the intra-stage optimization in Section 4.1.

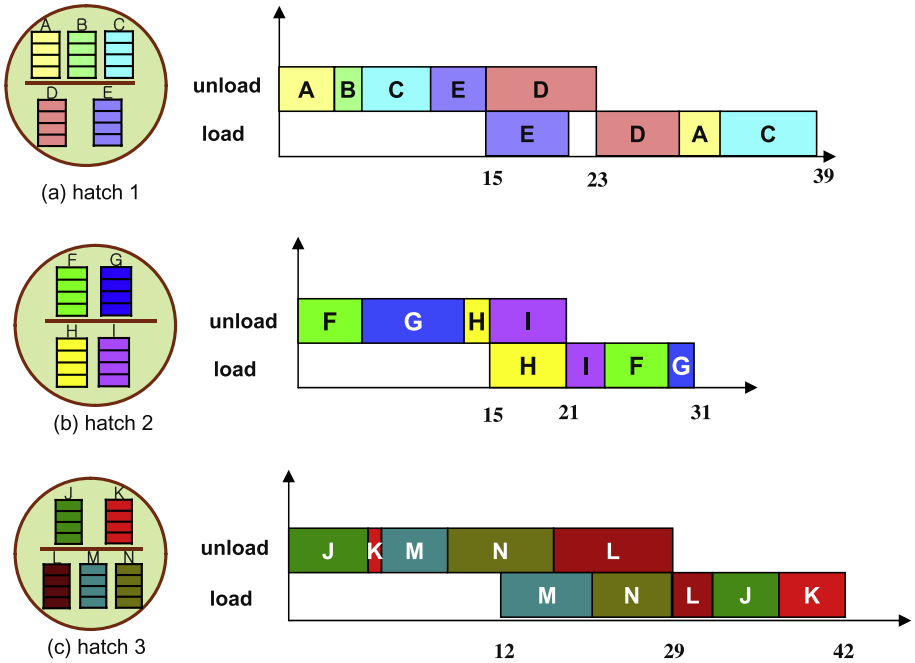


Fig. 4. Intra-stage optimization for stack sequencing.

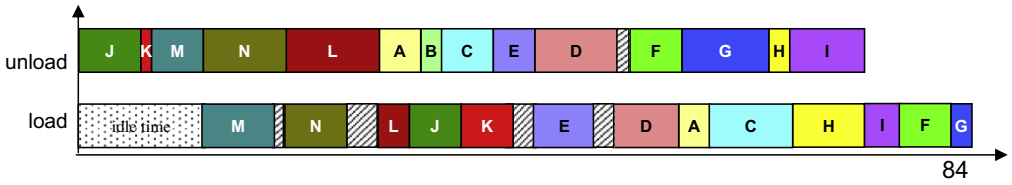


Fig. 5. Gantt chart of randomly generated sequence (hatch 3, hatch 1, hatch 2).

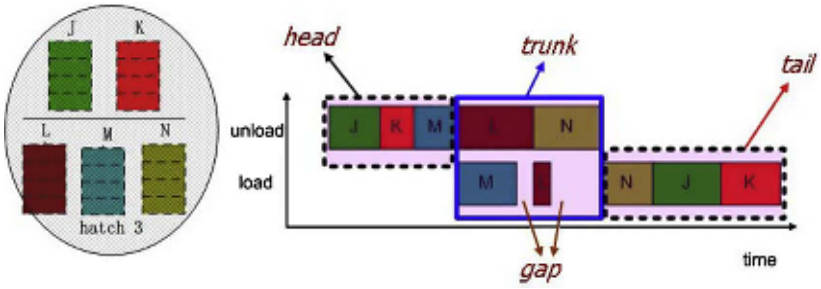


Fig. 6. Concepts of head, trunk, and tail in one hatch sequence.

We will use tasks in the third hatch for describing the proposed approach in this study (see Fig. 6). To clarify our reformative heuristic method, we precisely define several special concepts firstly i.e. *head*, *tail*, *trunk*, and *gap* for the further discussion.

Head: all the activities of QC tasks in the pane left side will be defined as the *head* of this hatch. The *head* of hatch consists of not only all the unloading activities above the hatch cover, but also the first unloading activity below the hatch cover.

Tail: all the activities of QC tasks in the pane right side will be defined as the *tail* of this hatch. The *tail* of hatch consists of not only all the loading activities above the hatch cover, but also some part of loading activity below the hatch cover.

Trunk: all the activities of QC tasks in the middle pane will be defined as the *trunk* of this hatch. The *trunk* is the ignorable part in the whole schedule during the hatch sequencing, in case of the *gap* inside of the *trunk* is relatively immutable due to the fixed loading and discharging plan.

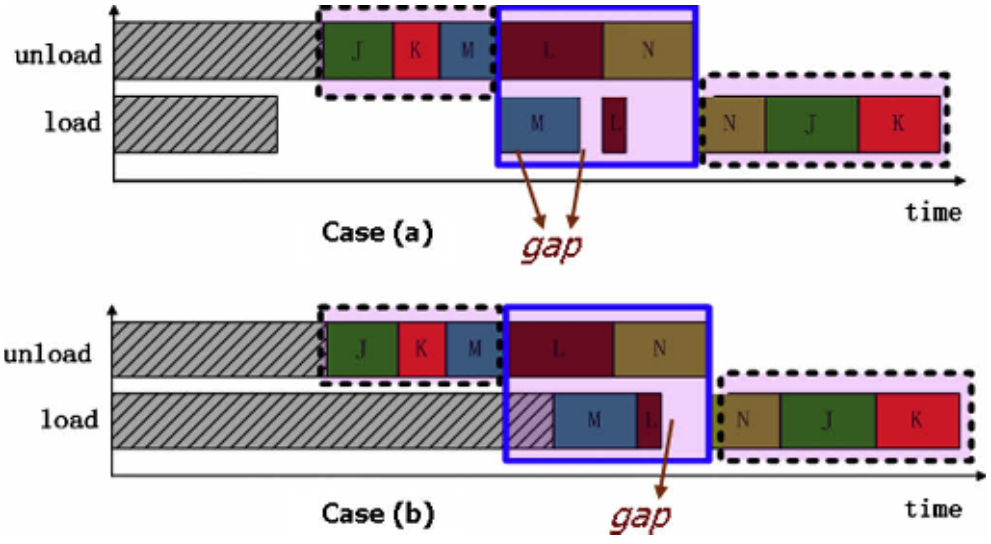


Fig. 7. Two cases of gap: case (a) unloading push; case (b) loading push.

Gap: the *gap* exists only in the trunk and is a little different from the idle time between different hatches. Even the *gap* in the *trunk* has already been minimized during the intra-stage optimization in some degree, and it still can be reduced further during the inter-stage optimization. Consider the exclusive two cases in Fig. 7. In case (a), the *gap* is not reduced even if the hatch is "pushed" by the predecessor on the unloading process; while in case (b) the *gap* will be reduced if the hatch is "pushed" by the predecessor on the loading process.

To utilize the observations in Figs. 6 and 7, it is necessary to analyze the property of the four concepts. All the activities belonging to the *trunk* are related to the containers loaded or unloaded in the hold, and this part

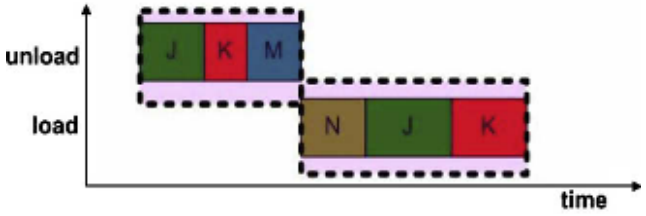


Fig. 8. Link up the *head* and *tail* after deleting the *trunk*.

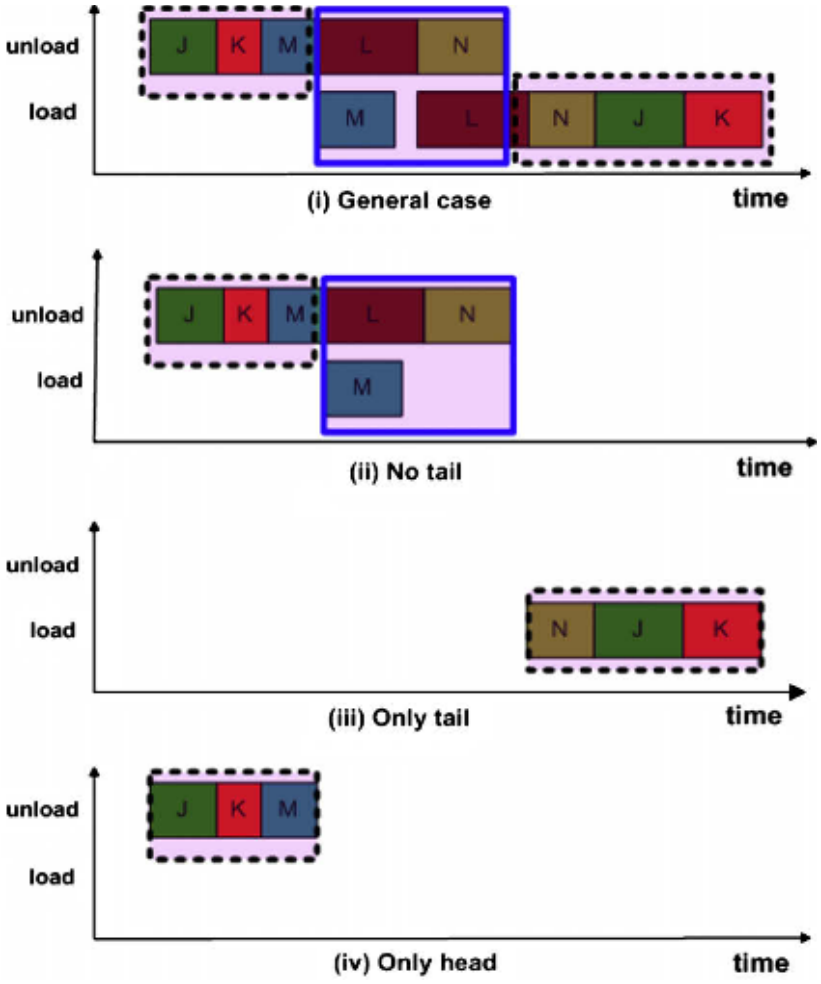


Fig. 9. Four special cases of the distribution of tasks in a hatch.

can not affect the make-span of the original problem during inter-stage optimization for hatch sequencing only except in case (b) of Fig. 7.

Hence, the *trunk* will be neglected for the time being during the inter-stage sequencing. Following the assumption, we reformulate the hatch task by deleting the *trunk*. Now, the *head* and the *tail* can be linked up together (see Fig. 8), and this reformulated hatch task can be treated as a typical task in the two-machine flow shop scheduling problem. In other words, Johnson's rule can be reused to obtain the optimal hatch sequence. Furthermore, depending on the different distribution of tasks in a hatch, some special cases must be additionally considered during the reformulating process of hatches (see Fig. 9). Note that the algorithm in this paper can be applied even to the special cases, because Johnson's rule can be applied even to the two-machine flow-shop problems, in which, for some jobs, either the processing time on machine 1 or the processing time on machine 2 is equal to zero.

To utilize the opportunity for possible reduction of gaps during inter-stage optimization, a special local search technique will be introduced in Section 4.3.

4.3. Gap-based neighborhood local search

In Fig. 7-(b), we have already observed that the reduction of gaps may affect the solution of hatch sequencing. Hence, this study proposes a local search, called "Gap-based neighborhood local search, to find the best hatch sequence. In Fig. 10, two hatches are included in the example, and their relevant heads and tails are (4, 6) and (8, 5), respectively. We can find the first solution in Fig. 10 by using the Johnson's rule. However, we can find the second solution, which is better than the first solution, by swapping the positions of the two hatches. We can see that the longer *tail* ($1 + 5 = 6$ units) of the preceding hatch schedule than the *head* ($2 + 2 = 4$ units) of the succeeding hatch schedule can make a positive reduction in the *gap*.

We can define a gap-based neighborhood block to be a list of adjacent hatch schedules in which, for every adjacent pair of hatch schedules, the head of the preceding hatch schedule is shorter than the tail of the succeeding hatch schedule (See Fig. 11). The length of each neighborhood block is from 2 to m, and we only conduct the neighbor swapping for the schedules of the first two hatches and those of the last two hatches to get two new solutions when the length of the block is longer than 2. Note that the proposed gap-based neighborhood local search is conducted only when the current solution does not reach the lower bound.

4.4. Hybrid heuristic approach

The proposed hybrid heuristic approach can be applied to the dual cycling QC scheduling problem as shown in Fig. 12.

For solving the simple case using the data in Table 1, firstly, we apply the Johnson's rule to the problem for each hatch (intra-stage sequencing) and get the solutions in Fig. 4. And then, we apply the Johnson's rule again for hatch sequencing (inter-stage sequencing) to obtain the solution in Fig. 5 which has the make-span of 84. And then we apply the gap-based neighborhood local search to obtain the solution in Fig. 13 whose

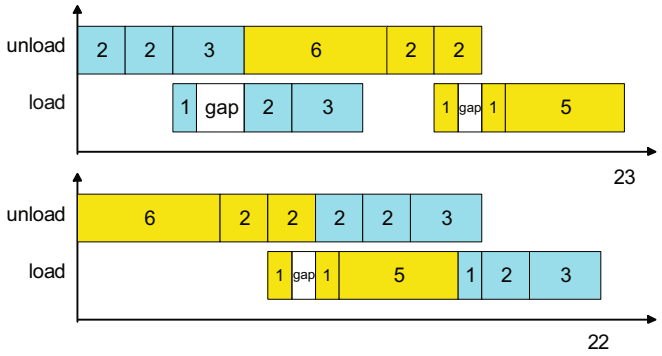


Fig. 10. An illustration of improving a solution by swapping.

Procedure: gap-based neighborhood local search
Input: a feasible hatch sequence
Output: improved hatch sequence
Begin
h ← 1
while h < m, do
If (tail-length of hatch *h*+1 > head-length of hatch *h*),
then append hatch h into the list of the neighborhood block (LNB):
h++;
else if LNB is not empty, then go out of the while loop
else h++
end
end
If LNB is not empty and LNB = (a_1, a_2, a_3, …, a_k), then
Exchange the positions of the a_1^{th} and the $(a_1+1)^{th}$ hatches and exchange the a_k^{th} and the $(a_k+1)^{th}$ hatches in the original hatch sequence and accept the new solution when the make-span is reduced.
end
Output: improved hatch sequence
end

Fig. 11. Procedure of gap-based neighborhood local search.

procedure: overall procedure of proposed hybrid heuristic approach
input: related data including total number of hatches *F*; containers' information $\overline{u}_{hc}$, $\underline{u}_{hc}$, $\overline{l}_{hc}$, $\underline{l}_{hc}$;
output: optimal QC schedule (loading and unloading sequences); total number of cycles
begin
h ← 0
for *k* ← 0 **to** *m* **do**
finding optimal sequence for all the stacks in *k*-th hatch;
// intra-stage sequencing by using Johnson's Rule
calculating the length of *head*, *trunk*, *tail* for *k*-th hatch;
reformulate a new *h*-th hatch by link up the *head* and *tail* of *k*-th hatch.;
h++
end
finding optimal sequence for all the reformulated hatches;
// inter-stage sequencing by using Johnson's Rule
calculating the total cycles of QC
while (current solution do not reach the lower bound)
improve the solution by gap-based neighborhood local search
output optimal QC schedule and total cycles
end

Fig. 12. Overall procedure for dual cycling QC scheduling using hybrid heuristic approach.

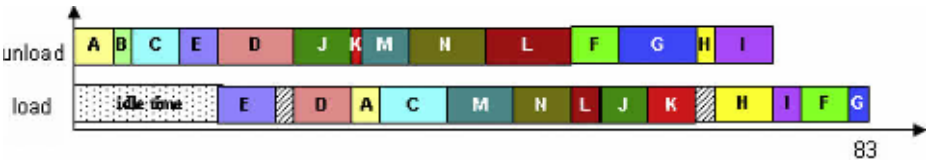


Fig. 13. Gantt chart of the final solution for the example.

make-span is 83. Note that the solution in Fig. 5 was improved by exchanging hatch 3 and hatch 1 in the sequence of Fig. 5.

5. Numerical experiment

In this paper, to prove the effectiveness of the approach proposed in this study, we used the stowage data for 68 ship bays for 10 vessels collected from a container terminal in Busan. The results of this application are summarized in Table 2. All experiments were run on a computer with an Intel(R) Pentium(R) D CPU 3.40 GHz and 1GB (DDR) of memory. We first test the problem using the CPLEX based on the proposed mathematical formulation in Section 3.

We could obtain the optimal solutions of simple cases in several minutes by using the CPLEX, whose size is smaller than 20 in the number of stacks in a ship-bay. Thus, we concluded that the maximum size

Table 2
Comparison of the experimental results

Vessel index	Vessel size (TEU)	No. of problems (bays) solved	Average No. of stacks in each bay	Maximum No. of stacks in a bay	Total No. of containers	No. of cycles in practical schedule	CPLEX		Proposed approach	
							No. of cycles	CPU-Time(s)	No. of cycles	CPU-Time(s)
1	–	1	14	14	146	–	89	347.14	89	0.63
2	3016	6	6	9	328	268	260	17.81	260	0.63
3	3177	4	10.3	12	183	179	141	105.92	141	0.94
4	3177	11	8.3	18	291	288	252	69.60	252	1.07
5	3177	4	8	20	202	156	129	713.92	129	1.09
6	3000	12	15.3	20	981	927	757	25372.18	757	1.25
7	6724	10	11.7	23	572	516	–	–	476	0.94
8	6724	4	21.5	26	730	614	–	–	505	0.78
9	6724	6	10.8	26	525	513	–	–	373	0.62
10	6724	5	21.8	30	285	277	–	–	256	0.78
11	6724	5	16.3	30	504	446	–	–	438	1.10

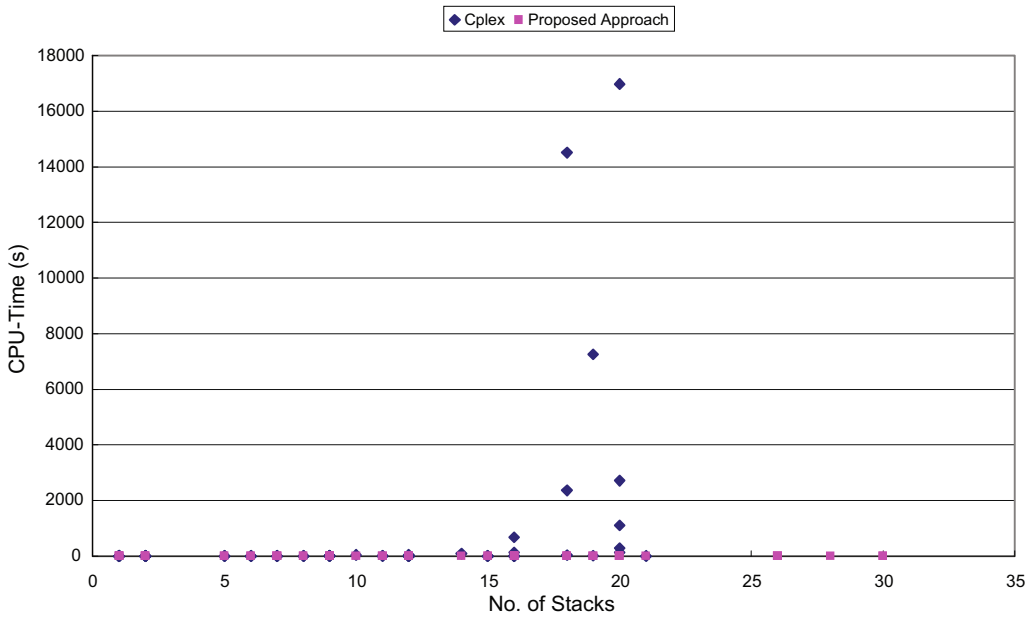


Fig. 14. The computational time for different number of stacks.

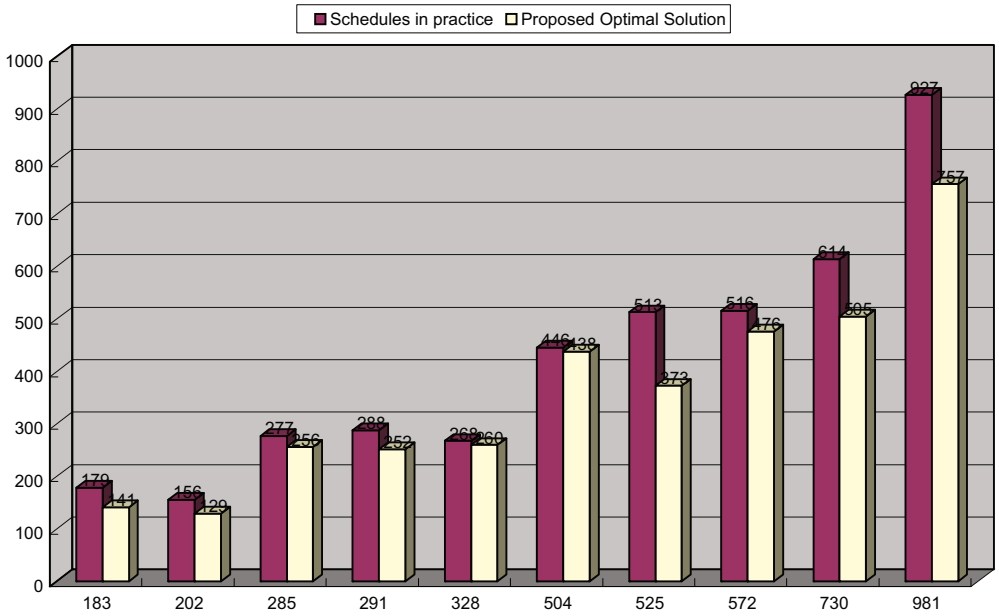


Fig. 15. The number of cycles for schedules in practice and those in this study.

of the problems for using the CPLEX is 20 in the number of stacks. Fig. 14 shows the distribution of the computational time for problems of different sizes. The computational time of the CPLEX increased rapidly as the number of stacks exceeds its limitation, 20. However, the computational time of the CPLEX was quite different for even several cases with the same number of stacks (for instance 20). Although the total number of stacks can definitely affect the number of variables in the optimization formulation, however, it was really not the only factor that determined the efficiency of the computation. For instance, the CPLEX could find the optimal solution easily in case that all tasks of one bay consist of only unloading activities.

Thirty eight problems from the first five vessels in Table 2 were solved by both the CPLEX and the heuristic approach in this study. For all the problems, we could obtain the optimal solution using a heuristic algorithm. We applied only the heuristic algorithm to solve the remaining 30 problems from the last five vessels that could not be solved using the optimization method. Instead of comparing the solutions by the heuristic algorithm with those by the optimization method, we compared those with schedules constructed by human planners and implemented in practice. We can see that the proposed approach significantly outperformed the human planners in all the cases by using the computational time shorter than two seconds to schedule all the bays in each vessel.

In general, the proposed heuristic approach could obtain near-optimal solutions in dual cycling QC scheduling problems. The computational time of the heuristic algorithm was not so sensitive to the size of the problem, which implies that this heuristic algorithm can be applied to solve QC scheduling problems for even larger vessels.

Fig. 15 illustrates the histogram that presents the improvement of the proposed approach compared to the schedules made by human planners. This result implies that the proposed approach can greatly improve the efficiency of QCs in ports. Also as the total number of containers for all QCs (including unloading and loading activities) becomes larger, the more improvement we can make by reducing the total number of cycles for QC operations.

6. Conclusions

The issue of dual cycling QC scheduling became an important issue for the efficient operation in modern container thermals. A mixed integer linear programming model was suggested for this problem. To solve this

model, the dual cycling QC scheduling problem was discomposed into two main phases, i.e. intra-stage optimization (sequencing all stacks in one hatch), and inter-stage optimization (sequencing all hatches). An effective gap-based local search technique has been proposed for combining it with some reformulated heuristic approaches to obtain optimal hatch and stack sequences in the QC scheduling.

Furthermore, some numerical experimental data from real cases have been used to prove the effectiveness of the proposed approach. The experimental results showed that the proposed approach found the optimal solution in most of all cases, and greatly outperformed the real schedules constructed by human planners in the current port. Moreover, considering the short computational time, it was found that the heuristic algorithm in this study is a promising approach to be applied in practice.

For future works, more practical considerations may be additionally included in this algorithm. Examples of the practical considerations are operation convenience by QC operators for a given sequence of operations, priority tasks requested by vessel operators, and precedence relationships among tasks because of physical constraints of tasks. Also, more accurate operation time may be considered for evaluating the efficiency of the QC operation.

Acknowledgments

This study was supported by the Korean Ministry of Education & Human Resources Development through the Second-Phase of Brain Korea 21 Project in 2007 and the Korean Government (MOEHRD) (The Regional Research Universities Program/Institute of Logistics Information Technology).

References

Baker, K. (1974). *Introduction to Sequencing and Scheduling*. New York: John Wiley.

Bish, E. K. (2003). A multiple-crane-constrained scheduling problem in a container terminal. *European Journal of Operational Research, 144*(1), 83–107.

Crainic, T. G., & Kim, K. H. (2007). *Intermodal Transportation. Hand books in Operations Research and Management Science, North-Holland*. The Netherlands: Amsterdam.

Daganzo, C. F. (1989). The crane scheduling problem. *Transportation Research Part B, 23*(3), 159–175.

Goodchild, A. V. (2005). *Crane Double Cycling in Container Ports: Algorithms, Evaluation, and Planning, PhD dissertation*. Berkeley: University of California.

Goodchild, A. V., & Daganzo, C. F. (2006). Double-Cycling Strategies for Container Ships and Their Effect on Ship Loading and Unloading Operations. *Transportation Science, 40*(4), 473–483.

Goodchild, A. V., & Daganzo, C. F. (2007). Crane double cycling in container ports: planning methods and evaluation. Transportation Research Part B, In press.

Imai, A., Sasaki, K., Nishimura, E., & Papadimitriou, S. (2006). Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. *European Journal of Operational Research, 171*(2), 373–389.

Kim, K. H., Kang, J. S., & Ryu, K. R. (2004). A beam search algorithm for the load sequencing of outbound containers in port container terminals. *OR Spectrum, 26*(1), 93–116.

Kim, K. H., & Park, Y. M. (2004). A crane scheduling method for port container terminals. *European Journal of Operational Research, 156*(3), 752–786.

Lee, D. H., Wang, H. Q., & Miao, L. (2008). Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E, 44*, 124–135.

Liu, J., Wan, Y. W., & Wang, L. (2006). Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Naval Research Logistics, 53*(1), 60–74.

Moccia, L., Cordeau, J. F., Gaudioso, M., & Laporte, G. (2006). A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics, 53*(1), 45–59.

Peterkofsky, R. I., & Daganzo, C. F. (1990). A branch and bound solution method for the crane scheduling problem. *Transportation Research Part B, 24*(1), 159–172.

Zhu, Y., & Lim, A. (2005). Crane scheduling with non-crossing constraint. *Journal of the Operational Research Society, 57*(12), 1464–1471.