TRANSPORT
ISSN 1648-4142 / eISSN 1648-3480
2016 Volume 31(4): 440–449
doi:10.3846/16484142.2014.994563

OPTIMIZATION ON THE CONTAINER LOADING SEQUENCE BASED ON HYBRID DYNAMIC PROGRAMMING

Zhan Bian, Qianqian Shao, Zhihong Jin

College of Transportation Management, Dalian Maritime University, Dalian, China

Submitted 9 May 2013; resubmitted 9 June 2013; accepted 20 August 2013;
first published online 14 January 2015

Abstract. Retrieving export containers from a container yard is an important part of the ship loading process during which arranging the retrieving sequence to enhance port efficiency has become a vital issue. This paper presents a two-phase hybrid dynamic algorithm aiming at obtaining an optimized container loading sequence for a crane to retrieve all the containers from the yard to the ship. The optimization goal is to minimize the number of relocation operations which has a direct impact upon container loading operation efficiency. The two phases of the proposed dynamic algorithms are designed as follows: at the first phase, a heuristic algorithm is developed to retrieve the containers subset which needs no relocation and may be loaded directly onto the ship; at the second phase, a dynamic programming with heuristic rules is applied to solve the loading sequence problem for the rest of the containers. Numerical experiments showed the effectiveness and practicability of the model and the algorithm by comparing with the loading proposals from an existing research and actual rules respectively.

Keywords: container terminal; loading sequence; container relocations; heuristic rules; dynamic programming.

Introduction

Over the past 20 years, container terminals have witnessed an increasing worldwide flow of containers and more popular larger-sized container ships. The competition among terminals has become prominent which makes the efficiency of port operation an important factor in the success of the fierce competition. Of all the popular measures of service performance, turnaround time, which is the average time a ship stays in a terminal, is the most important. One of the most effective methods for reducing the turnaround time is to improve the productivity of handing activities.

With the rapid development of container transport, the container yard has become an important place for exchanging and stocking containers. Most yards stack up containers utilize more and more precious space. Unlike many usual storage systems that are capable of providing random access to all stored items, only those located at the top are directly accessible to the yard cranes. In addition, containers have to be loaded onto ships according to the stowage plan, which specifies the location of each container on the ship, and thus largely determines the order the containers have to be retrieved onto the vessel. Extra movements, that waste time and money, occur when a container is due to be retrieved from the yard but is buried beneath other ones. Therefore, how to avoid or reduce the number of relocations to enhance the efficiency of handing activities has become a key issue for a container terminal.

There are certain publications that deal with the container relocation problem. Kim (1997) developed a method for estimating the number of relocations for import containers. This method also applies to other papers, e.g. Kim, K. H. and Kim, H. B. (1999) addressed the relationship between storage height and relocations for import containers, and formulated mathematical models under several arrival strategies, then applied the Lagrangian relaxation and sub-gradient optimization to solve for the best storage height. The container storage location has a direct impact on the follow-up operations. Kim *et al.* (2000) proposed a dynamic programming model to minimize the number of relocations considering the factor of container weight, and developed a decision tree to support real-time decisions. Yang and Kim (2006) suggested a genetic algorithm with simple heuristic rules to solve the dynamic location problem as well as the static one. To deal with the container retrieving problem during the loading process, Kim and Hong

Corresponding author: Zhihong Jin
E-mail: *jinzhihong@dlmu.edu.cn*

Taylor & Francis
Taylor & Francis Group

(2006) suggested a branch-and-bound algorithm for determining the locations of relocated blocks. Lee, Y. and Lee, Y.-J. (2010) developed a three-phase heuristic algorithm to minimize the number of relocations. Caserta *et al.* (2011) applied dynamic programming to get a proper solution. Xu *et al.* (2008) mainly considered the determination of relocated container locations to reduce the relocation rate in container yards. Yi *et al.* (2010) formulated a gambling model to address the same issue.

There are fewer related researches to optimize the container loading sequence. Wang *et al.* (2005) put forward a mathematic matrix model to deal with the problem. Zhu *et al.* (2010) proposed improvement strategies based on actual rules. Jin *et al.* (2011a, 2011b) applied a heuristic algorithm to solve the problem. Existing researches in this field are trying hard to improve the applicability of optimization methods. Nevertheless, the quality of obtained solutions declines significantly owing to the expansion of the problems scale. What is more, the solutions are unable to display the process of retrieving operations, which makes the algorithms lacking of practicability.

In this work, we develop a two-phase hybrid dynamic optimization algorithm for the container loading sequence problem. The optimization goal is to minimize the total number of relocations during the retrieving operation. With the combination of optimization algorithms and heuristic methods, we are able to obtain the optimal solution at a faster computing speed. Furthermore, the visualization of the loading sequence and shipping process makes the algorithms operatively.

This paper is divided into sections. Following this introduction, Section 1 defines the container loading sequence problem. An optimization model is formulated in Section 2. The framework and procedure of hybrid optimization algorithm are developed in Section 3. Numerical examples are used to test the performance of the proposed methods in Section 4. Conclusions are given in last section.

1. The Container Loading Sequence Problem

Given an initial layout of a bay in the yard (known as the yard plan) and a final layout of a bay on the containership (known as the stowage plan), the container loading sequence problem yields a container retrieval sequence that retrieves all the containers from the yard bay, one at a time in a specified order, such that the number of container relocations is minimized.

Fig. 1a shows the layout of a yard and the initial storage state of yard bay 5. Each number represents a slot, e.g. figure 43 represents the container which is stacked at the stack 4, tier 3, denoted by container 43. Fig. 1b illustrates the stowage plan of the containers in yard bay 5 of Fig. 1a. The containers are loaded onto bay 7 of the containership. Figures 2, 4, 6 of the 1st column represent the 1st, 2nd, 3rd tier of the hold respectively. Figures 82, 84 of the same column denote the 1st and 2nd tier above the deck respectively. Figures 2, 4, 6 of the 1st row indicate the 1st, 2nd, 3rd column of the portside respectively. Figures 1, 3, 5 of the same row describe the 1st, 2nd, 3rd column of the starboard respectively. Figure 43 means that container 43 of yard bay 5 is loaded to the 1st tier in the hold, the 1st column along the portside. If there is a × in the slot, it means the slot is under no consideration during the loading process.

Fig. 2 illustrates the loading sequence of the containers in yard bay 5. Figures in the slots represent the loading sequence, e.g., figure 1 means that container 43 is the first one to be retrieved and loaded onto the ship during the process.

2	3	15		7	
4	6	16	1	8	18
5	21	19	17	9	13
12	22	20	11	10	14

Fig. 2. The loading sequence of a yard bay

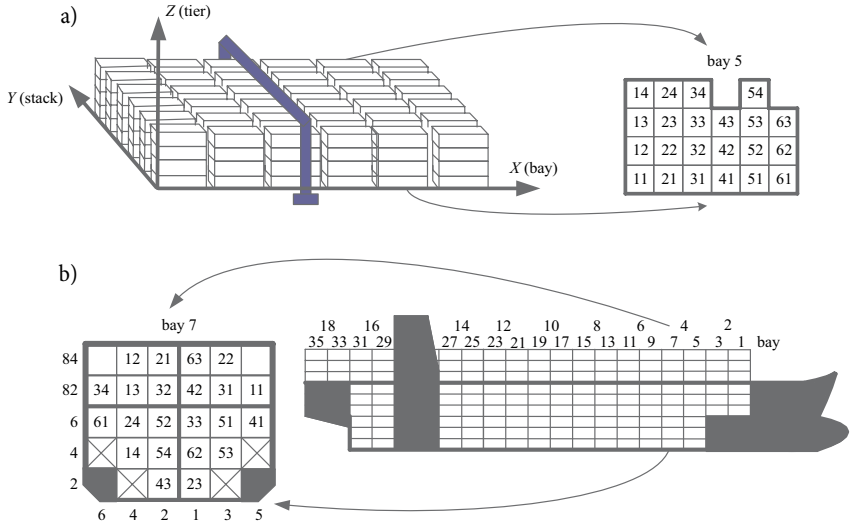

Fig. 1. Illustrations of the yard plan and stowage plan of a bay: a – the yard plan of bay 5; b – the stowage plan of yard bay 5

Two kinds of yard cranes are usually applied to container terminal operations, namely Rubber Tyred Gantry Crane (RTGC) and Rail Mounted Gantry Crane (RMGC). Their performances are summarized in Table 1. Besides, we focus on loading operations for only one bay in this research.

Table 1. The performance* of yard cranes

Device	Working span [stack]	Stacking height [tier]	Working capacity [TEU/bay]
RTGC	6	3~4	maximum 24
RMGC	10	6	maximum 60

**Data sources:* Dalian Container Terminal Co., Ltd.

Furthermore, during the retrieving process, the yard crane can only retrieve containers from the top of a stack, i.e. Late-In-First-Out (LIFO) rule. For the loading operation, the containers firstly have to be loaded onto the hold then onto the deck by the quay crane. Based on the industry practices and on the analysis of the actual conditions of the container terminal operations, basic assumptions used in this research are made as follows:

- a yard bay consists of several stacks and each stack permits several containers to stack up within the maximal tier (i.e., 6 tiers);
- only containers of same dimensions and of the same vessel are stacked within a yard bay – to simplify notations and explanations, it is assumed that all the containers are of the same length (i.e., 20-feet long);
- all the containers in the bay have accomplished the customs examination, i.e., there is no relocation that results from customs un-examination;
- without consideration of new containers arriving, the initial state of a yard bay is given and the stowage plan which is approved by the shipper is known;
- only at the moment when a container, which is not on the top tier, is to be retrieved, relocations of above containers within the bay occur;
- each relocation operation moves the top container to the empty slot of another stack, and the empty slot does not hang in the air;
- the initial yard bay has enough space to locate all containers relocated for retrieving a container;
- under the consideration of convenience and safety, relocation takes place in the same bay.

2. Notations and Model Development

It is obvious that the states of the yard storage and the ship stowage change once a container is retrieved and loaded onto the ship, therefore the loading process can be transformed into a dynamic one. Thus, we develop a two-phase algorithm consisting of a traverse heuristic and a hybrid dynamic programming to solve the problem in Section 3. In this section, the parameters and variables involved in the model are proposed in order to coincide with the algorithm, shown as Table 2.

The optimization goal of the loading sequence problem is to minimize the total number of relocations during the retrieving operation. Let M denote the total number of relocations. The objective function can be formulated as follows:

$$M = \min \sum_{k=1}^{N-l+2} \sum_{s=1}^{p_k} a_{ks} m_{ks}. \tag{1}$$

During each stage of the loading process, we should make sure that containers are not hung in the air when loaded on board. Eq. (2) ensures containers handled by the traverse heuristic are not hung in the air and Eq. (3) ensures containers satisfy the constraint during each sub-stage of the hybrid dynamic programming:

$$C_{lij} \le C_{li(j-1)}, \tag{2}$$

where: $l \in \{1,2,...,N\}$; $i \in \{1,2,...,I'\}$; $j \in \{2,...,J'\}$;

$$C_{ksij} \le C_{ksi(j-1)}, \tag{3}$$

where: $k \in \{1,2,...,N-l+2\}$; $s \in \{1,2,...,p_k\}$; $i \in \{1,2,...,I'\}$; $j \in \{2,...,J'\}$.

When a yard crane conducts a retrieving operation, only the container on the top tier can be retrieved. Eqs (4) and (5) ensure that during each sub-stage of the proposed algorithm, containers are retrieved to satisfy the constraint:

$$\prod_{j=1}^{k} D_{lij} \ne 0, \tag{4}$$

where: $D_{lik} \ne 0$; $l \in \{1,2,...,N\}$; $i \in \{1,2,...,I\}$;

$$\prod_{j=1}^{k} D_{ksij} \ne 0, \tag{5}$$

where: $D_{ksik} \ne 0$; $k \in \{1,2,...,N-l+2\}$; $s \in \{1,2,...,p_k\}$; $i \in \{1,2,...,I\}$.

At each sub-stage of the hybrid dynamic programming, only one relocation proposal is accepted. The constraint can be described as Eq. (6):

$$\sum_{s=1}^{p_k} a_{ks} = 1, \tag{6}$$

where: $\forall k$, $k \in \{1,2,...,N-l+2\}$.

And finally, decision variables should be assigned and represented by Eq. (7):

$$a_{ks} \in \{0,1\}, \tag{7}$$

where: $k \in \{1,2,...,N-l+2\}$; $s \in \{1,2,...,p_k\}$.

3. A Hybrid Optimization Algorithm Based on Dynamic Programming

In view of dynamic characteristics of container loading sequence problem, we suggest dynamic programming combined with heuristic rules to solve the problem. The optimality principle was first proposed by Bellman (1952), and later the theory of dynamic programming was developed (Bellman 1953, 1955), then applied to several research fields (Bellman 1965; Feldmann 1967; Li, Glazebrook 2010; Sanaye, Mahmoudimehr 2012).

Table 2. Definitions of parameters and variables

Name	Property	Definition
a_{ks}	Decision variable	Whether a relocation proposal is accepted as a result of state transition from stage $k-1$ to stage k under the condition of state s at stage k. $k \in \{1,2,...,N-l+2\}$; $s \in \{1,2,...,p_k\}$; $a_{ks} = \begin{cases} 0, \text{ the relocation proposal is not accepted;} \\ 1, \text{ the relocation proposal is accepted.} \end{cases}$
C_{lij}	Integer variable	Loading sequence of the container in column i, tier j on the ship after $l-1$ containers are loaded onto the ship. $l \in \{1,2,...,N\}$; $i \in \{1,2,...,I'\}$; $j \in \{1,2,...,J'\}$; $C_{lij} = \begin{cases} 0, \text{column i, tier } j \text{ has not been occupied yet;} \\ n, n \in \{1,2,...,l-1\} \text{ load a container to column } i, \text{ tier } j \text{ at the } n\text{th time;} \\ -1, \text{no container loaded to column } i, \text{ tier } j. \end{cases}$
C_{ksij}	Integer variable	The loading sequence of the container in column i, tier j on the ship under the condition of state s at stage k. $k \in \{1,2,...,N-l+2\}$; $s \in \{1,2,...,p_k\}$; $i \in \{1,2,...,I'\}$; $j \in \{1,2,...,J'\}$; $C_{ksij} = \begin{cases} 0, \text{column } i, \text{ tier } j \text{ has not been occupied yet;} \\ n, n \in \{1,2,...,k+l-2\} \text{ load a container to column } i, \text{ tier } j \text{ at the } n\text{th time;} \\ -1, \text{no container loaded to column } i, \text{ tier } j. \end{cases}$
D_{lij}	Integer variable	The serial number of the container in stack i, tier j in the yard bay after l – containers are loaded onto the ship. $l \in \{1,2,...,N\}$; $i \in \{1,2,...,I\}$; $j \in \{1,2,...,J\}$.
D_{ksij}	Integer variable	The serial number of the container in stack i, tier j in the yard under the condition of state s at stage k. $k \in \{1,2,...,N-l+2\}$; $s \in \{1,2,...,p_k\}$; $i \in \{1,2,...,I\}$; $j \in \{1,2,...,J\}$.
I	Constant	The number of stacks in a yard bay.
I'	Constant	The number of columns of the stowage plan.
J	Constant	The maximal tier of stacks in a yard bay.
J'	Constant	The number of tiers of the stowage plan.
k	Integer variable	The number of stages of the dynamic programming. $k \in \{1,2,...,N-l+2\}$.
l	Integer variable	The number of stages of the traverse heuristic. $l \in \{1,2,...,N\}$.
m_{ks}	Integer variable	The number of relocations as a result of state transition from stage $k-1$ to stage k under the condition of state s at stage k. $k \in \{1,2,...,N-l+2\}$; $s \in \{1,2,...,p_k\}$.
N	Constant	The total number of containers.
p_k	Integer variable	The total number of states at stage k. $k \in \{1,2,...,N-l+2\}$.

There are a few researches on container terminals using dynamic programming (Jin, Gao 2006; Lam *et al.* 2007; Alessandri *et al.* 2009; Jin *et al.* 2011a, 2011b; Meng, Wang 2011) of which various heuristic rules are generally applied to reduce the combinatorial complexity.

The hybrid dynamic programming consists of two phases, namely, the traverse phase and dynamic programming phase.

At the traverse phase, a traverse algorithm based on heuristic rules is developed to retrieve container subsets, which need no relocation directly onto the ship. Let D_l, C_l, and C denote the set of D_{lij}, the set of C_{lij} and the set of C_{ij} respectively. The concrete solution procedure of this method is explained as follows:

- *Step 1:* Set the stage number l as 1, the stack number i as 1, i.e., $l \leftarrow 1$, $i \leftarrow 1$. Turn to Step 2.
- *Step 2:* From the left to the right, compare the top non-zero figure in stack i of D_l with the figure of C, which corresponds with the bottom zero figure of C_l. Then turn to Step 3.
- *Step 3:* If the figures are equal, then retrieve the container corresponding with the figure and load it onto the ship, $l \leftarrow l + 1$, both C_l and D_l change into new states, then turn to Step 2; otherwise, turn to Step 4.
- *Step 4:* $i \leftarrow i + 1$, and turn to Step 5.
- *Step 5:* If $i \in \{1,2,...,I\}$, then turn to Step 2; otherwise, turn to Step 6.
- *Step 6:* If $l = N$, then the phase ends, output the result; otherwise, turn to Step 2.

The dynamic programming phase begins when the first relocation occurs and ends when all the containers

are loaded onto the ship. Let B denote the set of containers in the yard under the present state. TOP (B) is the list of containers in B which can be directly loaded onto the ship. $Ship(B,TOP(B))$ represents loading the containers of LC onto the ship. $destno(m)$ is the target stack with the lowest relocation cost for container m. Here, the relocation cost can be defined as follows:

$$Relocation\ cost = 1\,n_1 + 10\left|n_1 - n_2\right| + 100\,h, \tag{8}$$

where: n_1 is the serial number of the target stack; n_2 is the serial number of the retrieve stack; h is the height of the target stack; $n_1, n_2 \in \{1,2,...,I$; 1, 10, 100 represent the weights of distances between the target stack and the truck lane, between two stacks, and between two tiers respectively.

$Move$ (B,m,n) indicates moving container m to stack n. $C(B$ is the smallest number of relocations when the loading operation ends.

When $Ship(B,TOP(B))$ is done, we can obtain a new yard storage state presented by B', and obviously, $|B'| < |B|$; and $Move(B,m,n)$ turns $TOP(B) = \varnothing$ to $TOP(B) \neq \varnothing$. Therefore, $Ship(B,TOP(B))$ and $Move(B,m,n)$ can both change the state of B, and B is dynamically changing. The dynamic equation can be formulated as follows:

$$C(B) = \begin{cases} c_1, \text{ where } B=\varnothing; \\ c_2, \text{ where } TOP(B) \neq \varnothing; \\ c_3, \text{ where } TOP(B) = \varnothing;\ m_1,\ m_2: \text{containers to be relocated,} \end{cases} \tag{9}$$

where:

$$c_1 = 0;\ c_2 = C\left(Ship\left(B,TOP\left(B\right)\right)\right);$$

$$c_3 = \min\left(1 + C\left(Move\left(B,m_1,destno\left(m_1\right)\right)\right),1 + C\left(Move\left(B,m_2,destno\left(m_2\right)\right)\right)\right).$$

Regard each container loaded onto the ship as a stage. When relocation occurs, several new states will be generated. Thus, the number of states during the computing process will explosively increase. Considering the complexity, we suggest the following heuristic rules, shown as Fig. 3, and propose an example to explain the rules, shown as Fig. 4.

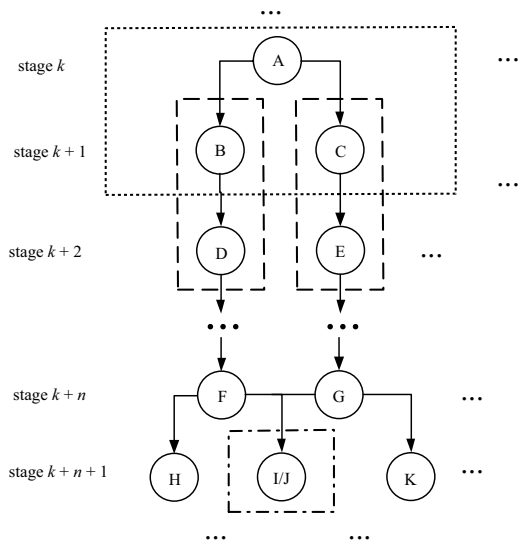

Fig. 3. The illustration of heuristic rules

Heuristic rules:

– *Rule 1* ⬚: if any relocation is needed, pick out the first and the second lowest cost choices, see the following description of the target stack choosing process (i.e., from stage k to stage $k+1$) in Fig. 4.
– *Rule 2* ⬚: if there are containers that can be loaded without relocation, only one new state should be generated from stage $k+1$ to stage $k+2$. It is discussed in detail by the process from stage $k+1$ to stage $k+2$ of the following example, shown in Fig. 4.
– *Rule 3* ⬚: if there are identical states in stage $k+1+n$, only one can be retained. As shown in Fig. 3, the state F of stage $k+n$ generates two new states H and I at stage $k+1+n$, and similarly, states J and K are generated by another state G, if I and J are exactly the same, then only one state can be kept at stage $k+1+n$, so we delete either of them.

Fig. 4 illustrates the state transition process of a certain stage of the example showed in Fig. 1. From the final

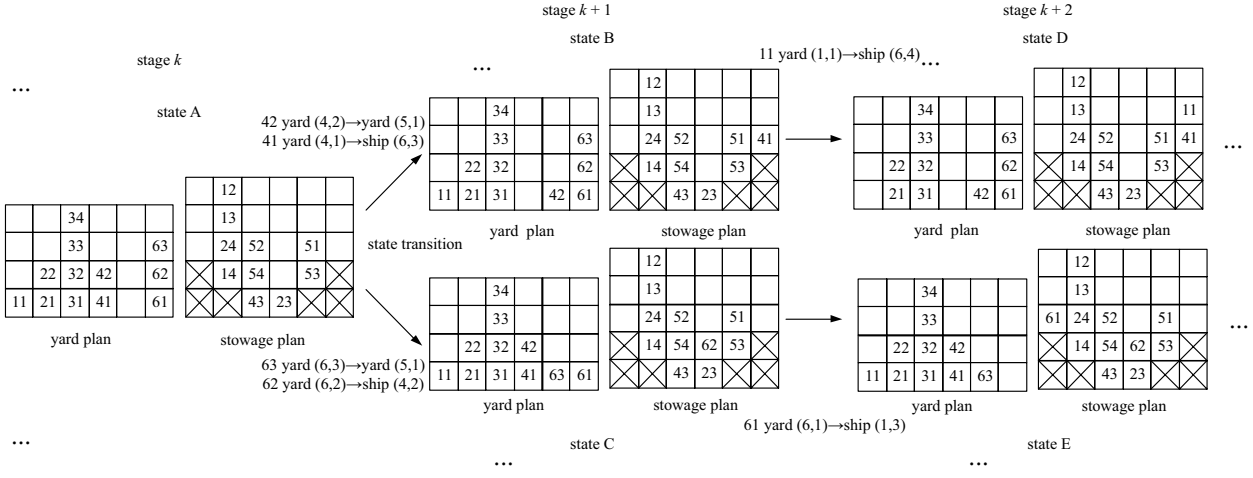

Fig. 4. The state transition of dynamic programming

stowage plan in Fig. 1b and the stowage plan of stage k, we can find 5 containers to be retrieved. These containers can be represented as the set of $\{61, 32, 62, 31, 41\}$, and the corresponding retrieve stacks of the yard plan can be represented as the set of $\{6, 3, 6, 3, 4\}$. It is easy to calculate that the number of containers located in the upper tier is 2, 2, 1, 3 and 1 respectively. Therefore, stack 6 and stack 4 are chosen as the retrieve stack, i.e., container 62 and container 41 are selected to be retrieved at stage $k + 1$. When retrieving container 62, container 63 should be moved to another stack first. According to Eq. (8) and Rule 1, we can find that stack 5 is of the lowest relocation cost, thus it is chosen to be the target stack. Similarly, stack 5 is chosen as the target stack for container 42. Then, two new states B and C are generated at state $k + 1$.

As for state B, according to the method used in the last paragraph, it is obvious that container 11 can be loaded without relocation, and based on Rule 2, only one new state should be generated for stage $k + 2$, shown as state D. Similarly, state E is the one and only state generated by state C for stage $k + 2$.

In order to generalize the proposed algorithm precisely, we describe it with the framework, shown as Fig. 5. It clearly shows that the algorithm begins with the Initialization, and after the traverse phase and then the hybrid dynamic programming phase, ends with the output of loading proposals and the number of relocations.

4. Numerical Experiments

In this section, we conduct computational examples to demonstrate the performance of the algorithm developed in this research. The proposed algorithm has been implemented by *Microsoft Visual C++* and run on a personal computer, which has a Core I5 CPU running at 2.50 GHz and with 4.0 GB memory. In all the cases, the containers are generated and randomly placed in the yard, subject to pre-determined number of rows, stacks, and maximum stack height, and stowage plan is generated in the same way.

Moreover, we compare the results of this research with others obtained under the following circumstances:

- actual scheduling rules: choose the stack with the fewest blocked containers as the retrieve stack, and choose the target stack randomly;
- improved heuristic rules: applied to an existing study (Zhu *et al.* 2010).

As mentioned before, container terminals always use two kinds of devices, RTGC and RMGC, to execute loading/unloading container operations. As is shown in Table 1, RTGC and RMGC are of different working capacity, so it is necessary to conduct numerical experiments respectively. Thus, the following session is divided into two parts, one is the numerical experiment part of the RTGC, and the other is of the RMGC.

Example 1 presents a small RTGC retrieving instance with only 17 containers spread over 6 stacks with

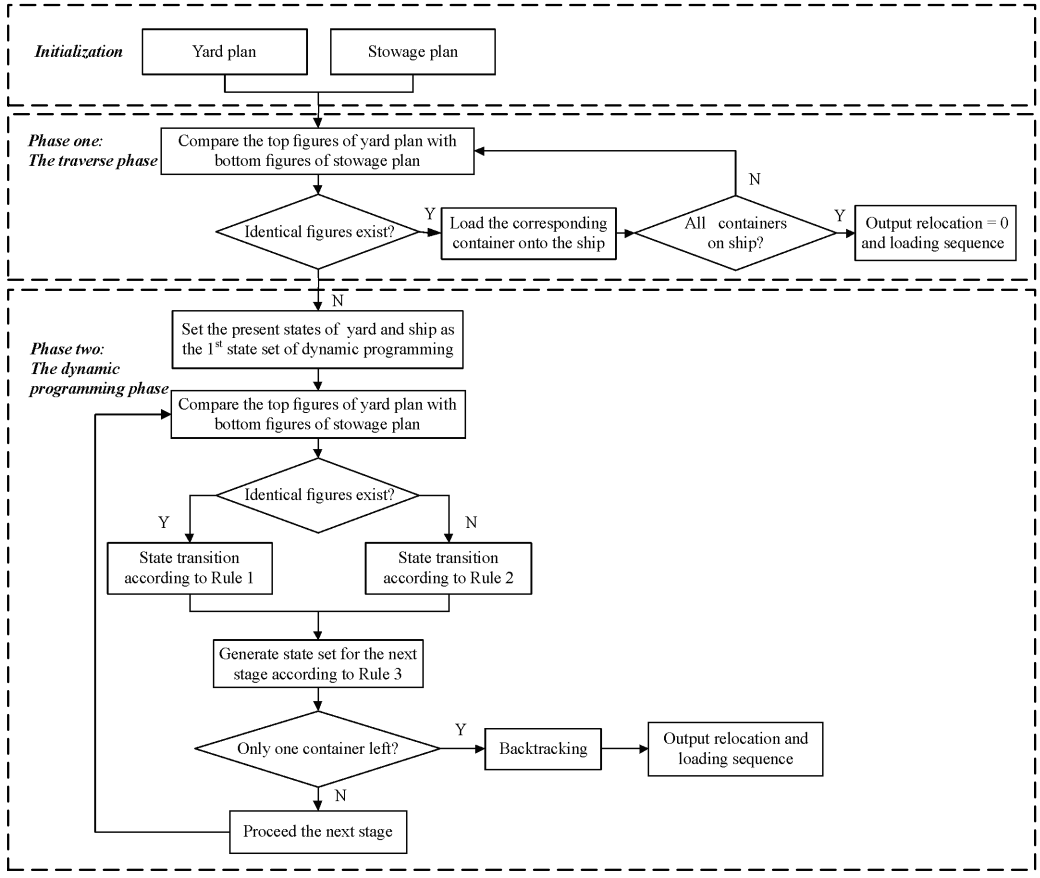

Fig. 5. The framework of hybrid dynamic programming

maximum height of 5, as shown in Fig. 6a. The corresponding stowage plan can be seen in Fig. 6b.

The instance was solved in 0.01 s with 6 relocations using actual scheduling rules and obtained the loading sequence, as shown in Fig. 7a. However, it took 0.02 s to execute improved heuristic rules and the number of relocations was 6. Fig. 7b illustrates the loading sequence of the algorithm.

The hybrid dynamic programming developed in this research consumed 0.05 s to complete, a litter longer than the other two different rules discussed above. With the same amount of relocations, known as 6, we finally came up with as many as 15 different loading proposals, which include the solutions under the two different circumstances. Furthermore, the relocation process was visualized so that it could directly conduct the operation site for the workers. Thus, workers can select the relatively easier proposals according to the situation. Table 3 shows the details of each procedure. Obviously, relocations occurred when containers 42, 32, 52, 62, 22, 13 were loaded onto the ship.

In order to test the efficiency of the proposed algorithm for solving RTGC operations comprehensively, we conducted 20 random experiments, setting the number of containers N as 20. Let A, B, C denote actual scheduling rules, improved heuristic rules and hybrid optimization algorithm proposed in this research respectively. We present the results in Table 4.

As can be seen, Table 4 illustrates the amount of relocations, number of proposals and CPU times respectively obtained by three different algorithms. We obtain 99 relocations altogether by A, 72 relocations by B, and 53 relocations by C. Therefore, the number of relocations by C decreases 46.5%, 26.4% respectively compared with A and B. It is particularly noteworthy that we obtain total 44 proposals as alternative choices by C. Owing to the lower complexities of A and B, the search spaces of the algorithms are smaller than that of C. As the result of that, it took 0.38 s, 0.66 s and 1.01 s respectively for three methods dealing with 20 instances, with 0.019 s, 0.033 s and 0.05 s on average respectively. In general, the hybrid optimization algorithm proposed in this research can solve RTGC loading sequence problem effectively.

a)

14					
13	23	33		53	
12	22	32	42	52	62
11	21	31	41	51	61

b)

08	13	12		33	23	62
06	11	22	53	14	52	61
04		21	32	42	51	
02			31	41		
	6	4	2	1	3	5

Fig. 6. The yard plan and stowage plan of a bay: a – the yard plan of a bay; b – corresponding stowage plan of the bay

a)

3					
17	12	4		9	
15	14	8	2	11	6
16	13	7	1	10	5

b)

3					
17	10	4		7	
15	14	6	2	9	12
16	13	5	1	8	11

Fig. 7. The loading sequences based on two different rules: a – actual scheduling rules; b – improved heuristic rules

In the following section, we executed 20 random experiments of RMGC loading sequence problem, setting the number of containers N as 50. The results are enumerated in Table 5.

It is generally known that the number of relocations increases with more containers in a bay. As Table 5 shows, figures in relocations columns are much larger than that in Table 4. The optimizations on the number of relocations and proposals are still conspicuous. We obtain 993 relocations altogether by A, 932 relocations by B, and 701 relocations by C. So the number of relocations by C decreases 29.4%, 24.7% respectively compared with A and B. We obtain 1006 proposals by C. Besides, it cost 0.039 s, 0.08 s and 0.2 s on average respectively to complete the 50 instances.

From different scaled experiments discussed above, we can come to a conclusion that the algorithm developed in this research can solve container loading sequence problems efficiently.

Conclusions

Achieving high yard operational efficiency is one of the most important tasks for managers in a container terminal system. In this research, scheduling problem for loading containers in container terminals was discussed. Combining the optimization algorithms with heuristic rules, we developed a two-phase hybrid dynamic algorithm, which aims to generate an optimal movement sequence for the crane to retrieve all the containers from a given yard to the ship. Meanwhile, a model is designed to minimize the total number of relocations.

The two-phase hybrid dynamic algorithm starts by generating an initial layout of the yard plan and stowage plan according to pre-determined elements. Phase one retrieves containers, which need no relocation onto the ship quickly. In the second phase, a hybrid algorithm is used to retrieve the rest of containers. The two phases are both iterative, and terminate when all container are loaded on the ship.

Numerical results showed that the two-phase hybrid dynamic algorithm is able to solve loading sequence instances of both RTGC and RMGC, which is within the range of real cases, and thus of practical use to the industry. The number of relocations is much smaller than that of actual scheduling rules and improved heuristic rules. Therefore, it is proved that the algorithm in this research can tackle the practical scheduling problem efficiently.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China (No 71172108), Dalian Science and Technology Project (No 2012A17GX125), Doctoral Program Foundation of Institutions of Higher Education of China (No 20122125110009) and Fundamental Research Funds for the Central Universities (No 3132013320).

Table 3. Details of loading sequence based on hybrid dynamic programming

Sequence	Container	State change		Illustration of state change	Relocation
		(yard→yard)	(yard→ship)		
1	42	(4,2)→(6,3)		container 42, from yard stack 4, tier 2 to yard stack 6, tier 3	√
2	41		(4,1)→(4,1)	container 41, from yard stack 4, tier 1 to ship column 4, tier 1	
3	42		(6,3)→(4,2)	container 42, from yard stack 6, tier 3 to ship column 4, tier 2	
4	14		(1,4)→(4,3)	container 14, from yard stack 1, tier 4 to ship column 4, tier 3	
5	33		(3,3)→(4,4)	container 33, from yard stack 3, tier 3 to ship column 4, tier 4	
6	32	(3,2)→(4,1)		container 32, from yard stack 3, tier 2 to yard stack 4, tier 1	√
7	31		(3,1)→(3,1)	container 31, from yard stack 3, tier 1 to ship column 3, tier 1	
8	32		(4,1)→(3,2)	container 32, from yard stack 4, tier 1 to ship column 3, tier 2	
9	53		(5,3)→(3,3)	container 53, from yard stack 5, tier 3 to ship column 3, tier 3	
10	52	(5,2)→(4,1)		container 52, from yard stack 5, tier 2 to yard stack 4, tier 1	√
11	51		(5,1)→(5,1)	container 51, from yard stack 5, tier 1 to ship column 5, tier 1	
12	52		(4,1)→(5,2)	container 52, from yard stack 4, tier 1 to ship column 5, tier 2	
13	23		(2,3)→(5,3)	container 23, from yard stack 2, tier 3 to ship column 5, tier 3	
14	62	(6,2)→(5,1)		container 62, from yard stack 6, tier 2 to yard stack 5, tier 1	√
15	61		(6,1)→(6,1)	container 61, from yard stack 6, tier 1 to ship column 6, tier 1	
16	62		(5,1)→(6,2)	container 62, from yard stack 5, tier 1 to ship column 6, tier 2	
17	22	(2,2)→(3,1)		container 22, from yard stack 2, tier 2 to yard stack 3, tier 1	√
18	21		(2,1)→(2,1)	container 21, from yard stack 2, tier 1 to ship column 2, tier 1	
19	22		(3,1)→(2,2)	container 22, from yard stack 3, tier 1 to ship column 2, tier 2	
20	13	(1,3)→(2,1)		container 13, from yard stack 1, tier 3 to yard stack 2, tier 1	√
21	12		(1,2)→(2,3)	container 12, from yard stack 1, tier 2 to ship column 2, tier 3	
22	11		(1,1)→(1,1)	container 11, from yard stack 1, tier 1 to ship column 1, tier 1	
23	13		(2,1)→(1,2)	container 13, from yard stack 2, tier 1 to ship column 1, tier 2	
Total relocations					6

Table 4. Comparisons of performances for RTGC instances

Test No	No of relocations			Optimization rate [%]		Total No of obtained proposals			CPU time [s]		
	A	B	C	(A–C)/A	(B–C)/B	A	B	C	A	B	C
1	8	6	4	50	33	1	1	4	0.04	0.05	0.05
2	5	5	3	40	40	1	1	1	0.03	0.03	0.04
3	3	1	1	67	0	1	1	1	0.01	0.03	0.04
4	3	3	2	33	33	1	1	2	0.01	0.03	0.05
5	6	3	3	50	0	1	1	2	0.02	0.02	0.04
6	3	3	3	0	0	1	1	1	0.01	0.03	0.05
7	8	6	5	38	17	1	1	2	0.01	0.03	0.05
8	2	2	2	0	0	1	1	2	0.02	0.04	0.04
9	7	7	5	29	29	1	1	1	0.03	0.04	0.10
10	2	0	0	100	–	1	1	1	0.01	0.03	0.04
11	6	4	4	33	0	1	1	6	0.02	0.03	0.08
12	3	3	2	33	33	1	1	1	0.01	0.03	0.04
13	8	7	5	38	29	1	1	10	0.03	0.04	0.10
14	2	1	1	50	0	1	1	2	0.02	0.03	0.04
15	5	3	3	40	0	1	1	2	0.02	0.04	0.04
16	9	6	1	89	83	1	1	1	0.02	0.02	0.04
17	3	3	2	33	33	1	1	1	0.02	0.03	0.05
18	4	4	2	50	50	1	1	1	0.01	0.03	0.04
19	4	1	1	75	0	1	1	1	0.02	0.04	0.04
20	8	4	4	50	0	1	1	2	0.02	0.04	0.04

Table 5. Comparisons of performances for RMGC instances

Test No	No of relocations			Optimization rate [%]		Total No of obtained proposals			CPU time [s]		
	A	B	C	(A–C)/A	(B–C)/B	A	B	C	A	B	C
1	21	19	13	38	32	1	1	3	0.05	0.10	0.12
2	18	19	14	22	26	1	1	4	0.04	0.07	0.10
3	25	20	11	56	45	1	1	49	0.04	0.09	0.24
4	19	17	15	21	12	1	1	229	0.03	0.09	0.80
5	24	20	16	33	20	1	1	1	0.05	0.09	0.14
6	30	26	21	30	19	1	1	34	0.06	0.12	0.20
7	20	21	17	15	19	1	1	3	0.04	0.07	0.10
8	20	17	13	35	24	1	1	6	0.05	0.08	0.10
9	16	13	13	19	0	1	1	3	0.04	0.05	0.08
10	17	18	15	12	17	1	1	4	0.04	0.08	0.10
11	23	25	18	22	28	1	1	7	0.05	0.13	0.10
12	16	13	15	6	–15	1	1	49	0.03	0.05	0.44
13	22	25	14	36	44	1	1	2	0.05	0.14	0.10
14	23	17	10	57	41	1	1	3	0.06	0.05	0.14
15	20	24	12	40	50	1	1	5	0.03	0.10	0.10
16	30	28	15	50	46	1	1	6	0.07	0.14	0.10
17	20	16	14	30	13	1	1	28	0.04	0.05	0.27
18	16	13	16	0	–23	1	1	10	0.04	0.05	0.14
19	34	27	19	44	30	1	1	2	0.09	0.12	0.30
20	21	19	17	19	11	1	1	4	0.04	0.08	0.10
21	23	24	15	35	38	1	1	2	0.05	0.10	0.10
22	15	14	13	13	7	1	1	64	0.02	0.08	0.40
23	17	20	17	0	15	1	1	3	0.03	0.07	0.08
24	17	17	15	12	12	1	1	3	0.02	0.05	0.08
25	25	22	14	44	36	1	1	5	0.06	0.09	0.09
26	20	18	13	35	28	1	1	1	0.04	0.07	0.10
27	20	20	15	25	25	1	1	8	0.04	0.09	0.10
28	17	15	13	24	13	1	1	18	0.03	0.06	0.14
29	23	17	12	48	29	1	1	1	0.05	0.08	0.12
30	10	11	10	0	9	1	1	8	0.02	0.05	0.08
31	18	20	16	11	20	1	1	19	0.03	0.07	0.16
32	12	14	10	17	29	1	1	36	0.02	0.07	0.30
33	20	16	13	35	19	1	1	12	0.03	0.07	0.12
34	10	10	9	10	10	1	1	22	0.01	0.05	0.30
35	21	18	15	29	17	1	1	7	0.03	0.07	0.08
36	16	15	13	19	13	1	1	207	0.04	0.07	0.62
37	17	20	15	12	25	1	1	31	0.03	0.10	0.50
38	11	9	9	18	0	1	1	2	0.01	0.07	0.10
39	15	19	13	13	32	1	1	1	0.01	0.07	0.07
40	22	19	13	41	32	1	1	6	0.03	0.09	0.14
41	20	19	14	30	26	1	1	16	0.03	0.07	0.12
42	20	19	16	20	16	1	1	4	0.04	0.07	0.08
43	27	23	18	33	22	1	1	7	0.07	0.13	0.10
44	19	21	15	21	29	1	1	15	0.04	0.08	0.14
45	16	17	13	19	24	1	1	28	0.03	0.06	0.50
46	26	21	10	62	52	1	1	1	0.05	0.08	0.14
47	16	18	15	6	17	1	1	16	0.02	0.05	0.14
48	16	17	12	25	29	1	1	6	0.02	0.06	0.10
49	22	19	13	41	32	1	1	3	0.05	0.08	0.12
50	27	23	14	48	39	1	1	2	0.06	0.15	0.20

References

Alessandri, A.; Cervellera, C.; Cuneo, M.; Gaggero, M.; Soncin, G. 2009. Management of logistics operations in intermodal terminals by using dynamic modelling and nonlinear programming, *Maritime Economics & Logistics* 11(1): 58–76. http://dx.doi.org/10.1057/mel.2008.24

Bellman, R. 1965. On the application of dynamic programing to the determination of optimal play in chess and checkers, *Proceedings of the National Academy of Sciences of the United States of America* 53(2): 244–247.

Bellman, R. 1955. Functional equations in the theory of dynamic programming. V. Positivity and quasi-linearity, *Proceedings of the National Academy of Sciences of the United States of America* 41(10): 743–746.

Bellman, R. 1953. Bottleneck problems and dynamic programming, *Proceedings of the National Academy of Sciences of the United States of America* 39(9): 947–951.

Bellman, R. 1952. On the theory of dynamic programming, *Proceedings of the National Academy of Sciences of the United States of America* 38(8): 716–719.

Caserta, M.; Voß, S.; Sniedovich, M. 2011. Applying the corridor method to a blocks relocation problem, *OR Spectrum* 33(4): 915–929. http://dx.doi.org/10.1007/s00291-009-0176-5

Feldmann, E. G. 1967. Dynamic program for drug quality, *Journal of the American Pharmaceutical Association* 7(6): 301–302.

Jin, C.; Gao, P. 2006. Container berth expansion planning with dynamic programming and fuzzy set theory, in *IEEE International Conference on Service Operations and Logistics, and Informatics, 2006 – SOLI'06*, 21–23 June 2006, Shanghai, China, 260–265. http://dx.doi.org/10.1109/SOLI.2006.328928

Jin, Z.; Lan, H.; Bian, Z.; Ji, M. 2011a. Optimization on containership loading scheduling based on actual constraints, *Journal of Dalian Maritime University* 37(1): 71–74. (in Chinese).

Jin, Z.-H.; Mao, J.; Li, N. 2011b. Scheduling of relocating containers within a bay in container yard based on hybrid dynamic programming, *Journal of Transportation Systems Engineering and Information Technology* 11(6): 131–136. (in Chinese).

Kim, K. H. 1997. Evaluation of the number of rehandles in container yards, *Computers & Industrial Engineering* 32(4): 701–711. http://dx.doi.org/10.1016/S0360-8352(97)00024-7

Kim, K. H.; Hong, G.-P. 2006. A heuristic rule for relocating blocks, *Computers & Operations Research* 33(4): 940–954. http://dx.doi.org/10.1016/j.cor.2004.08.005

Kim, K. H.; Kim, H. B. 1999. Segregating space allocation models for container inventories in port container terminals, *International Journal of Production Economics* 59(1–3): 415–423. http://dx.doi.org/10.1016/S0925-5273(98)00028-0

Kim, K. H.; Park, Y. M.; Ryu, K.-R. 2000. Deriving decision rules to locate export containers in container yards, *European Journal of Operational Research* 124(1): 89–101. http://dx.doi.org/10.1016/S0377-2217(99)00116-2

Lam, S.-W.; Lee, L.-H.; Tang, L.-C. 2007. An approximate dynamic programming approach for the empty container allocation problem, *Transportation Research Part C: Emerging Technologies* 15(4): 265–277. http://dx.doi.org/10.1016/j.trc.2007.04.005

Lee, Y.; Lee, Y.-J. 2010. A heuristic for retrieving containers from a yard, *Computers & Operations Research* 37(6): 1139–1147. http://dx.doi.org/10.1016/j.cor.2009.10.005

Li, D.; Glazebrook, K. D. 2010. An approximate dynamic programing approach to the development of heuristics for the scheduling of impatient jobs in a clearing system, *Naval Research Logistics* 57(3): 225–236. http://dx.doi.org/10.1002/nav.20395

Meng, Q.; Wang, T. 2011. A scenario-based dynamic programming model for multi-period liner ship fleet planning, *Transportation Research Part E: Logistics and Transportation Review* 47(4): 401–413. http://dx.doi.org/10.1016/j.tre.2010.12.005

Sanaye, S.; Mahmoudimehr, J. 2012. Minimization of fuel consumption in cyclic and non-cyclic natural gas transmission networks: assessment of genetic algorithm optimization method as an alternative to non-sequential dynamic programing, *Journal of the Taiwan Institute of Chemical Engineers* 43(6): 904-917. http://dx.doi.org/10.1016/j.jtice.2012.04.010

Wang, X.; Chen, H.-Y.; Wang, C.; *et al.* 2005. The algorithm of getting the reasonable harbor's shipping order, *Mathematics in Economics* 22(3): 284–290. (in Chinese).

Xu, Y.; Chen, Q.-S.; Long, L.; Yang, L.-Z.; Liu, L.-Y. 2008. Heuristics for container relocation problem, *Journal of System Simulation* 20(14): 3666–3669. (in Chinese).

Yang, J. H.; Kim, K. H. 2006. A grouped storage method for minimizing relocations in block stacking systems, *Journal of Intelligent Manufacturing* 17(4): 453–463. http://dx.doi.org/10.1007/s10845-005-0018-5

Yi, Z.; Li, B.; Li, X. 2010. Game heuristic optimization algorithm for reshuffle in container yards, *Journal of Shanghai Maritime University* 31(3): 47–51. (in Chinese).

Zhu, M.; Fan, X.; Cheng, H.; He, Q. 2010. Heuristics for export container loading sequence problem, *China Mechanical Engineering* 21(9): 1066–1070. (in Chinese).