TRANSPORT
ISSN 1648-4142 / eISSN 1648-3480
2018 Volume 33(2): 408–417
doi:10.3846/16484142.2016.1255993

SCHEDULING FOR YARD CRANES BASED ON TWO-STAGE HYBRID DYNAMIC PROGRAMMING

Zhan Bian[1], Qi Xu[2], Na Li[3], Zhihong Jin[4]

[1]*College of Business Administration, Capital University of Economics and Business, China*
[2, 3, 4]*College of Transportation Management, Dalian Maritime University, China*

Submitted 8 September 2013; resubmitted 14 November 2013, 17 February 2014, 14 December 2014; accepted 1 January 2015; published online 08 December 2016

Abstract. Making operational plans for Yard Cranes (YCs) to enhance port efficiency has become vital issues for the container terminals. This paper discusses the load-scheduling problem of multiple YCs. The problem is to schedule two YCs at different container blocks, which serve the loading operations of one quay crane so as to minimize the total distance of visiting paths and the make-span at stack area. We consider the container handling time, the YC visiting time, and the waiting time of each YC when evaluating the make-span of the loading operation by YCs. Both the container bay visiting sequences and the number of containers picked up at each visit of the two YCs are determined simultaneously. A mathematical model, which considers interference between adjacent YCs, is provided by means of time-space network to formulate the problem and a two-stage hybrid algorithm composed of greedy algorithm and dynamic programming is developed to solve the proposed model. Numerical experiments were conducted to compare performances of the algorithm in this study with actual scheduling rules.

Keywords: container terminals; load-scheduling; greedy algorithm; dynamic programming; two-stage hybrid algorithm.

Introduction

As a hub for container transportation, container terminal is an important logistics node, linking container transportation by water and on land. Port operations can be generally divided into two parts: the discharging operation during which containers are unloaded from containerships, and the loading operation during which containers are loaded onto ships. In order to improve service levels and economic benefits, container terminal has to make full use of various discharging and unloading terminal resources to conduct a reasonable scheduling job in order to improve the terminal's operational efficiency.

In most container terminals, a large portion of the turnaround time of a vessel is consumed by the two processes. One of the most effective methods for reducing the terminal turnaround time is to improve the productivity of handing activities. There are three main types of equipment involved in the discharging and loading process, i.e., Quay Crane (QC), Yard Truck (YT) and Yard Crane (YC). The planning process of containership operations consists of a berth scheduling, QC scheduling, and discharge/load sequencing. Through the process of berth scheduling, the berthing time and position of a containership are determined. During the QC scheduling process, the number of QCs to be assigned to the ship and the job sequence of each QC which indicates the order of the containers needed to be discharged/loaded from/to the ship by it are specified. For the discharging process, inbound containers are stacked one by one at a designated empty yard space without considering the attributes of each container. Hence, the YCs do not need to move much and the turnaround time mainly depends on the number of inbound containers. However, since the outbound containers are usually scattered in the container blocks and the containers picked up by YCs need to satisfy the job sequences of the QCs, the loading process becomes crucial in determining the turnaround time.

As the main handling equipment of container terminals, YCs play a vital role in terminal operation system and YCs route scheduling directly impacts the

Corresponding author: Zhan Bian
E-mail: *bianzhan1990@163.com*

Taylor & Francis
Taylor & Francis Group

discharging and unloading efficiency and the operation of other transportation equipment on port. Meanwhile, YCs route scheduling is also affected by other operational factors in the terminal system, such as the sequence of discharging and loading operations of QCs, the stacking location of containers on the yard, etc. Therefore, how to determine YCs routes so as to improve the utilization rate has become a hot topic in the study field of container terminals.

Several researches have addressed the YC scheduling problem. It was first proposed by Chung *et al.* (1988), then Kim and Kim (1997, 1999) formulated a mixed integer programming model for the routing problem of a single-YC loading export containers out of the stack area onto waiting YTs. Kim and Kim (2003) applied heuristic algorithms (genetic algorithm and beam search algorithm) to deal with the same problem. Their numerical experiments showed that the proposed beam search algorithm outperforms the genetic algorithm. Narasimhan and Palekar (2002) proved that the single-YC routing problem is NP-complete. A heuristic algorithm and an exact branch-bound algorithm for the problem were developed and tested by numerical experiments. The computational results showed that the heuristic algorithm is practical for the large-sized problems. Kim *et al.* (2003) studied the YC scheduling problem in another way by investigating the sequencing problem of the YTs. A simulation study was performed to compare the performance of the proposed solution methods. Considering the YC operation constraints, Kim and Park (2004) employed a branch-bound algorithm for YC scheduling problem and developed a greedy randomized adaptive search procedure to tackle computer solutions. Ng and Mak (2005a, 2005b) applied a branch-bound algorithm and heuristic algorithms to minimize the YC waiting time. Han (2005) discussed the optimal route of a YC while loading containers. He *et al.* (2007) proposed an integer programming and hill climbing algorithm to make the YC scheduling and storage space allocation an entire system.

However, the studies above only focus on the single YC scheduling problem in which only one YC serves one QC. In fact, due to the pursuit for higher port operation efficiency, it is the common practice in many container terminals that two or even more YCs are deployed to serve one QC. Therefore, the scheduling synchronization among YCs becomes more significant.

Zhang *et al.* (2002) addressed the deployment problem of YCs, in which yard block that each YC should be assigned to at each time period was determined. However, interference among YCs was not considered in this study, which indicates that YC scheduling problem could not be solved radically. Ng (2005) studied the scheduling problem of YCs considering interference and proposed a heuristic algorithm to minimize the make-span. However, the types of tasks to the loading operation were not restricted. Lee *et al.* (2006, 2007) applied the genetic algorithm and simulated annealing algorithm to solve the same problem. Wei *et al.* (2007) studied the YC scheduling in the quantitative operation condition. Han and Ding (2008) discussed problems of YCs allocation and optimization in loading/unloading process.

These recent researches generally solve multi-YCs scheduling problems under static conditions, which are all single-objective optimizations (either make-span or travel route), and rarely consider the work schedule of QCs, thus are not able to visualize scheduling details of YCs. In this paper, a time-space network is formulated to describe YCs travel route, and 'where to retrieve containers', 'how many containers to be retrieved', 'which YC to retrieve' are considered in the model.

This paper is divided into 6 sections. Following this introduction, Section 1 defines the YCs loading sequence problem. A mathematical model is formulated in Section 2. The framework and procedure of two-stage hybrid optimization algorithm are developed in Section 3. Numerical experiments are used to test the performance of the proposed method in Section 4. Conclusions are given in the last section.

1. YCs Load-Scheduling Problem

Fig. 1 shows a bird-eye's view of a typical container terminal and it briefly illustrates the loading operation in a container yard. As illustrated in Fig. 2, a block consists of several bays where dozens of containers scatter. A YC is composed of a gantry, a trolley and a spreader. When a YC lifts a container from the yard, the gantry first positions itself at the bay where the target container resides, positions its trolley over the target stack, lowers its spreader to hold onto the container, and then lifts it up. After lifting the container, the YC can lower the container onto a truck waiting at one end of the crane, or place the container on the top of another stack.

In order to determine the loading sequence, two documents are necessary. One is the work schedule for a QC (shown in Table 1), which shows the sequence of container groups and the number of containers a QC should pick up. A container group is defined as a collection of containers of the same length that will be loaded onto the same ship at the same destination port. The work schedule for QC A in Table 1 includes loading tasks to pick up sequentially 36 containers of group A, 23 containers of group C, 23 containers of group A, 24 containers of group B, 36 containers of group C, and 36 containers of group B.

The other document is the yard stowage map, which shows the distribution of containers of each container group in the yard as in Table 2. In the problem, the work schedule of the QC and the container yard stowage map are known beforehand. YC 1 and YC 2 are employed to serve QC A at the same time. They will perform the loading operation according to the work schedule of QC A together. It is obvious that different schedules of YCs will leads to different make-span of the loading process.

The following discusses how to determine routes of multiple YCs, which are assigned to the loading op-

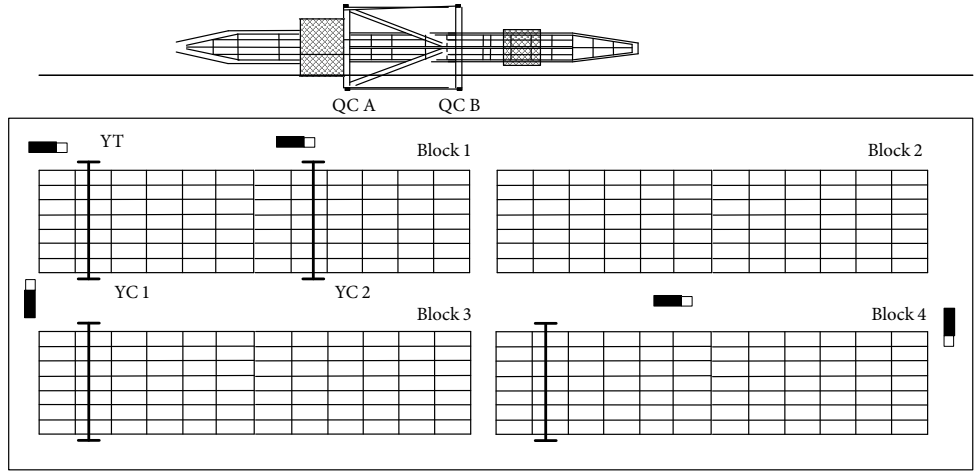


Fig. 1. A layout of a container-loading terminal

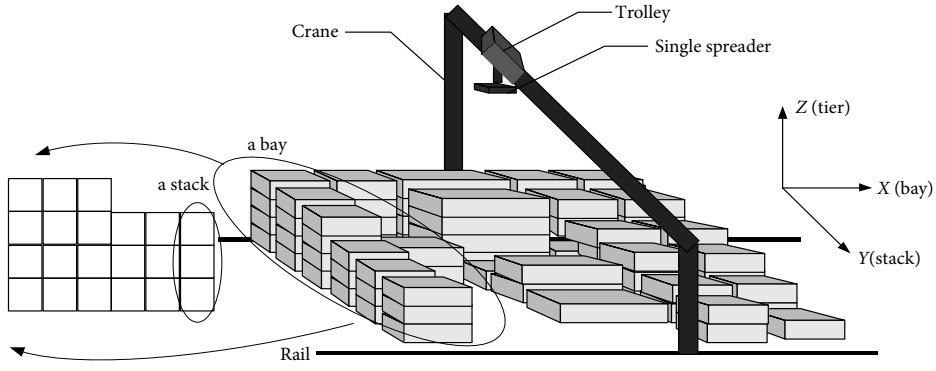


Fig. 2. Illustration of the YC and containers in a block of a yard

Table 1. The work schedule of QC A

Sequence	1	2	3	4	5	6
Group	A	C	A	B	C	B
Quantity	36	23	23	24	36	36

Table 2. The yard stowage map of export containers in container yard

Bay number	42	45	50	70	72	80	85
Group	B	A	C	B	A	C	B
Quantity	30	26	40	15	33	19	15

erations for one QC. The decision variables are the visiting sequence of bays of each YC and the number of containers, which each YC picks up at each bay during the loading operation. To deal with the visiting sequence of the two YCs is to find the visiting paths of the two YCs, which can be represented on double-layer time-space networks. Fig. 3 is the sample networks of YC1 and YC2 of the example above on which the character in each node (labelled by a square) is the decision variable representing whether the YC parks at the bay with the exception of a start node and a terminal node. The figure below the node with brackets represents the number of containers retrieved at the node. The x and y axes specify the sequence and bay respectively. The line between two nodes represents the visiting path of the YC. The detailed meaning of each character will be discussed in Section 2. For sequence 1, QC A should load 36 containers of group A, and there are two bays (i.e., bay 45 and bay 72) stacking containers of group A, so y_1 and y_2 are generated. It is remarkable that the nodes should be divided when containers of the same group spread over several bays in order to avoid loops (e.g. each node of bays 42, 70, 85 serving for sequence 6 is divided into 2 nodes, shown as $y_{15} \sim y_{20}$ in Fig. 3).

This approach will help YCs move, park and retrieve containers among different bays until the accumulative quantity matches the number of containers of a given sequence. For each sequence which is concerned, the nodes of the sequence will be divided $\left\lceil \frac{m}{n} \right\rceil - 1$ times while the quantity of bays to a specific container group and the number of YCs are m and n respectively.

Table 1. The work schedule of QC A

Sequence	1	2	3	4	5	6
Group	A	C	A	B	C	B
Quantity	36	23	23	24	36	36

Table 2. The yard stowage map of export containers in container yard

Bay number	85	80	72	70	50	45	42
Group	B	C	A	B	C	A	B
Quantity	15	19	33	15	40	26	30

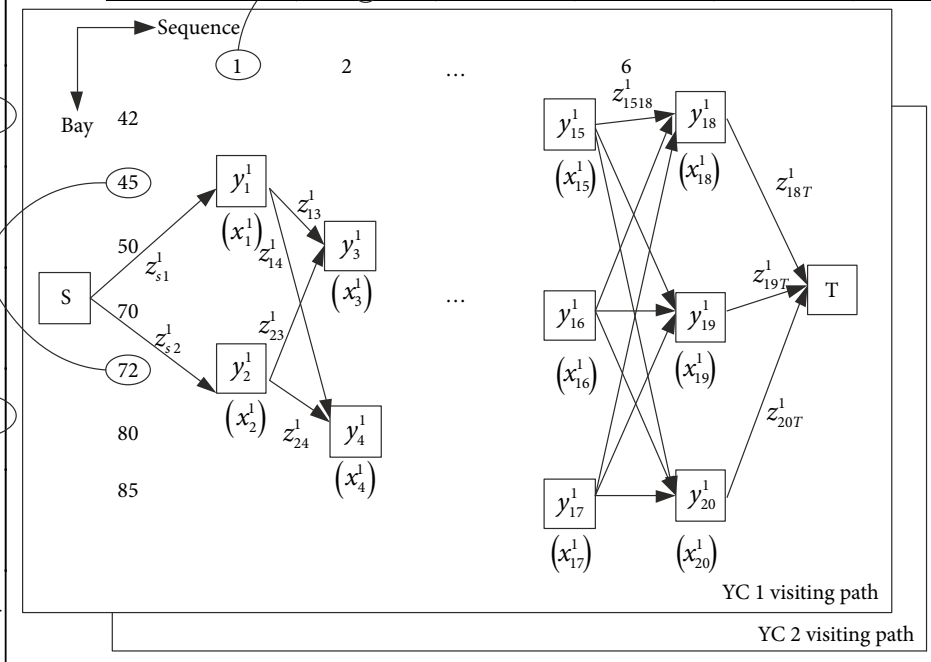


Fig. 3. The time-space networks of the visiting paths of two YCs

2. Mathematical Formulation

Since the loading sequences are distributed among the two YCs, making their working schedule dependent on one another, the schedules of the two YCs need to be coordinated to minimize the make-span. The following assumptions are introduced for the definition of the problem:

- two YCs serve one QC working for a vessel;
- crossover between YCs is not allowed during the loading process;
- the minimum space between adjacent YCs is required for avoiding the collision (in the paper, it was assumed to be one bay);
- the time required for a YC to load a container is assumed to be the same for all the containers despite the exact storage positions of individual containers;
- there is only one group of container stacked in one bay, which is the common practice in allocating space in the stack area of container terminals;
- all the containers for loading operations are located in one block or adjacent blocks in the travel direction.

To formulate the problem, the following notations will be used:

Parameters:

J – the set of bays;
I_j – the initial number of containers stacked at bay j;
P – the set of sequences;
q_p – the total number of containers loading in sequence p;
M – a sufficiently large constant;
w_n – the weight of the objective n;
R – the set of YCs;
L^r – the set of lines in the YC r network;
N^r – the set of nodes in the YC r network;
N^r_{ST} – the set of nodes in the YC r network, exclusive of the start node S and the terminal node T;
JN^r_j – the set of nodes within the limits of bay j in the YC r network;
PN^r_p – the set of nodes of sequence p in the YC r network;
CN^r_{pf} – the set of adjacent nodes of retrieving action f in sequence p in the YC r network;
CL^r – the set of crossed lines among the sequences or the divided nodes of one sequence in the YC r network;
c^r_{ij} – the travel distance from bay i to bay j in the YC r network;
(i, j) – the line between node i and node j;
$[j, k]$ – node j and node k;
$n^r_S = 1$; $n^r_T = -1$; 0 – otherwise.

Decision variables:

x^r_i – the number of containers retrieved at node i by YC r;
$y^r_i = 1$, if YC r parks at node i; 0 – otherwise;
$z^r_{ij} = 1$, if YC r moves from node i to node j; 0 – otherwise.

As the loading operation is assigned to two YCs, it is necessary to make them serve one certain sequence simultaneously and balance their workload to minimize the make-span. Besides, frequent moving of YCs will bring interrupt to the loading process and affect working performance. Thus, to reduce the parking frequency of YCs is also of great importance to reduce the time-consumption. What is more, retrieving patterns vary to satisfy the QC loading demands, which can low make-span with the shortest visiting path. Therefore, in this paper the three following objectives are focused: balancing workload of two YCs, minimizing the parking frequency during the loading process and minimizing the travel distance of two YCs.

The loading sequence problem can be formulated by time-space network as follows:

Object to:

$$\min w_1\left(\sum_{p\in P}\left|\sum_{i\in PN_p^1} x_i^1-\sum_{i\in PN_p^2} x_i^2\right|\right)+ w_2\left(\sum_{r\in R}\sum_{i\in N^r} y_i^r\right)+w_3\left(\sum_{r\in R}\sum_{(i,j)\in L^r} c_{ij}^r\cdot z_{ij}^r\right). \quad (1)$$

Subject to:

$$\sum_{\{j:(i,j)\in L^r\}} z_{ij}^r-\sum_{\{j:(j,i)\in L^r\}} z_{ji}^r=n_i^r \ \text{ for } \forall i\in N^r; \quad (2)$$

$$\sum_{\{i:(i,j)\in L^1\}} z_{ij}^1+\sum_{\{i:(i,j)\in L^2\}} z_{ij}^2\le 1 \ \text{ for } \forall j\in N^r; \quad (3)$$

$$\sum_{\{i:(i,j)\in L^1\}} z_{ij}^1+\sum_{\{i:(i,k)\in L^2\}} z_{ik}^2\le 1 \ \text{ for } \forall [j,k]\in CN_{pf}^r; \quad (4)$$

$$z_{ij}^1+z_{kl}^2\le 1 \ \text{ for } \forall (i,j),(k,l)\in CL^r; \quad (5)$$

$$\sum_{\{j:(i,j)\in L^r\}} z_{ij}^r\le M\cdot y_i^r \ \text{ for } \forall r\in R,\ \forall i\in N_{ST}^r; \quad (6)$$

$$x_i^r\le M\cdot y_i^r \ \text{ for } \forall r\in R,\ \forall i\in N_{ST}^r; \quad (7)$$

$$\sum_{r\in R}\sum_{i\in PN_p^r} x_i^r=q_p \ \text{ for } \forall p\in P; \quad (8)$$

$$\sum_{r\in R}\sum_{i\in JN_j^r} x_i^r=I_j \ \text{ for } \forall j\in J; \quad (9)$$

$$x_i^r\ge 0 \ \text{ for } \forall r\in R,\ \forall i\in N^r; \quad (10)$$

$$y_i^r\in\{0,1\} \ \text{ for } \forall r\in R,\ \forall i\in N^r; \quad (11)$$

$$z_{ij}^r\in\{0,1\} \ \text{ for } \forall r\in R,\ \forall (i,j)\in L^r. \quad (12)$$

The objective function Eq. (1) consists of three polynomials with different weights which is aimed at balancing the workload of two YCs, minimizing the total number of parking times, and minimizing the total travel distance respectively. Eq. (2) represents flow conservation of each time-space network. Eqs (3)–(4) guarantee that two YCs will not collide during loading process. As the two-layer time-space networks are of the same structure, one bay is represented by two corresponding nodes lying in two layers. Eq. (3) restricts the inflow and upper limit of every two-node group to ensure that each bay is served only by one YC at a time. Eq. (4) restricts the same content for each retrieving operation of each sequence to guarantee that two YCs can keep a safe distance. Eq. (5) implies that two YCs move in proper order and are not permitted to travel through one another. Eqs (6)–(7) defines the moving, parking and retrieving rules of YCs. Eq. (6) denotes that only the bay lying in the visiting path can be parked. Eq. (7) means that only when the YC parks can the containers of the parked bay be retrieved. Eq. (8) determines that only if one sequence is completed can the next sequence begin. Eq. (9) is the number constraint for stocked containers, i.e., for each bay, the total amount of containers retrieved from it is equal to the initial number of containers stacked in the bay when all sequences are completely finished. Eqs (10)–(12) are the non-negative integer constraint of the decision variables.

3. Two-Stage Hybrid Algorithm

As it is well known, the single YC loading scheduling problem is an NP-complete problem. Needless to say, the problem presented in this paper is also an NP-complete problem which makes simple exact algorithms lacking of practicability to deal with large-sized cases. Therefore, hybrid algorithms are required to solve the problem efficiently. Bian *et al.* (2016) proposed a hybrid optimization algorithm based on dynamic programming to solve the loading sequence problem. It can be divided into 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; at the dynamic programming phase, a dynamic programming with heuristic rules is applied to solve the loading sequence problem for the rest of the containers. During the second phase, heuristic rules are used to reduce the complexity.

In this paper, we proposed a new algorithm defined as 'the two-stage hybrid algorithm'. The algorithm consists of two phases, namely, the greedy algorithm and the dynamic programming. The former is used to obtain feasible solutions and the latter for the best solution. Besides, the absolute value function proposed in Section 2 is difficult to be dealt with. By solving the problem, although we did not change the form of the objective function, we added some heuristic rules when programming the proposed algorithm, i.e. keeping the two YCs working simultaneously as far as possible to maintain their workload balanced.

3.1. Definition of Node

In order to describe the proposed algorithm precisely, a new concept 'node' is introduced in this paper, which represents the general states of the yard and the YCs at a time. More specifically, a node imports two types of information of which one is the yard information (i.e. the total number of remaining containers in the yard, the number and the category of remaining containers in each bay), the other is the YCs information (i.e. the

loading sequence, current position, total travel distance, retrieving quantity and parking times of the two YCs).

A new node *J* will be generated after a node *I* going through a serial of operations, and *I*, *J* can be defined as a father node and a child node respectively. It is obvious that the number of remaining containers of a child node *J* is no more than that of a father node *I*. The proposed hybrid algorithm which is aimed at reducing the amount of remaining containers is divided into two stages. In the first stage, greedy algorithm is proposed to generate an initial feasible solution; in the second stage, dynamic programming is suggested to find the optimal solution. In other words, according to the scheduling rules, the father node is divided into several child nodes (i.e. one or more) which will be processed to get a solution. The solution corresponding with the child node of the least objective function value (hereafter called cost) is the initial feasible solution while all child nodes should be considered in order to get the optimal solution.

As previously mentioned, a feasible solution is obtained when there is no container left. Then the cost of the feasible solution will be calculated, and a feasible solution turns into an optimal solution if its cost is no more than that of other feasible solutions. The final purpose of the two-stage hybrid algorithm is to find an optimal solution.

3.2. Greedy Algorithm

In this section, a feasible solution is generated by the following steps:

- *Step 0*: build a node queue and join initial nodes to the tail of the queue;
- *Step 1*: set the head node as *I*, and go to *Step 2*;
- *Step 2*: if there is no container to be loaded in *I*, then set *I* as the feasible solution, and the algorithm is terminated; otherwise, go to *Step 3*.
- *Step 3*: according to scheduling rules, *I* is divided into several Child Nodes, recorded as *J* and so on; if there exists several child nodes, then sort the nodes in an ascending order based on different costs and go to *Step 4*.
- *Step 4*: join the child node with the least cost to the tail of the node queue, and go to *Step 1*.

Fig. 4 illustrates the process to obtain an initial feasible solution.

3.3. Dynamic Programming

In dynamic programming, the process of generating child nodes can be regarded as the process in which one *N*-dimensional problem is converted into $k(N-1)$ -demensional sub-problems, and the figure k is unpredictable. Thus, a serial of new child nodes can be obtained. The quantity of remaining containers decreases till no container is left. During the process, the dynamic equation can be formulated as follows:

$$cost\ (I) = \min\ (cost\ (I_1) + cost\ (K)\_1, cost\ (I_2) + cost\ (K)\_2, \ldots), \quad (13)$$

where: *I* represents the father node; J_1, J_2, … are child nodes generated by *I* after scheduling operations; *K* indicates the number of containers involved in the scheduling process (i.e. the father node contains *K* more containers than the child nodes); cost *K* represents the cost as a result of conversion from the father node to the child nodes.

During each stage of dynamic programming, several new states are generated. Thus, the number of states during computing process will explosively increase which will lead to a large amount of operations. Considering the complexity of the algorithm, a lopping operation to deal with the nodes is suggested, i.e., ignoring the child node whose cost is greater than the existing lowest cost in each stage.

Fig. 5 shows the framework of dynamic programming to obtain an optimal solution. To be more precise, the algorithm can be described as follows:

- *Step 0*: set the cost of the initial feasible solution as cost. Build a Node queue and join initial Nodes to the tail of the queue;
- *Step 1*: if the queue is empty, then the algorithm is terminated; otherwise, go to *Step 2*;
- *Step 2*: set the head Node as *I′*, and set the cost of *I′* as *cost′* and go to *Step 3*;

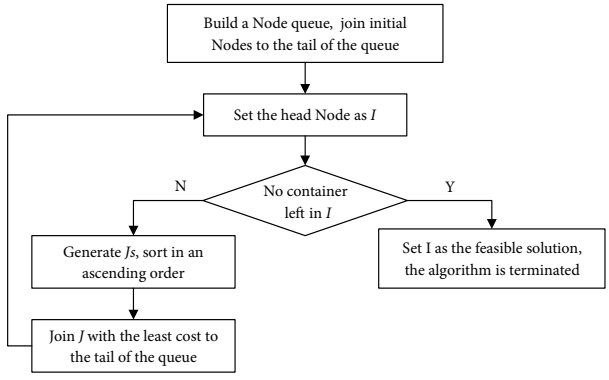


Fig. 4. The framework of greedy algorithm

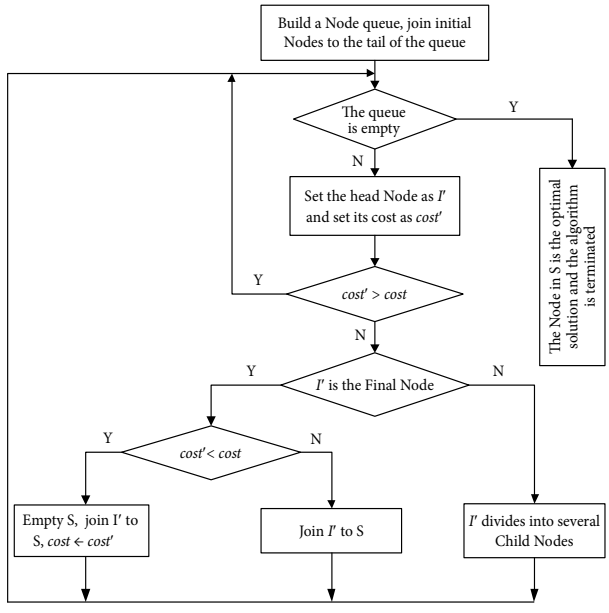


Fig. 5. The framework of dynamic programming

- *Step 3*: compare *cost′* with *cost*, if the former is larger than the latter, which means the optimal solution will not be obtained after *I′*, then ignore *I′* and go to *Step 1*; otherwise, go to *Step 4*;
- *Step 4*: if *cost′* equals *cost* and *I′* is a final node, then join *I′* to the set of optimal solutions *S* and go to *Step 1*; otherwise, go to *Step 5*;
- *Step 5*: if *cost′* is smaller than *cost* and *I′* is a final node, then empty *S* and join *I′* to the set and replace *cost* by *cost′* and go to *Step 1*; otherwise, go to *Step 6*;
- *Step 6*: if *cost′* is smaller than *cost* and *I′* is no longer a final node, then *I′* is divided into several child nodes according to scheduling rules and join all the child nodes to the tail of the queue and go to *Step 1*.

4. Numerical Experiments

4.1. Experiment Design

In this section, computational examples to demonstrate the performance of the algorithm developed in this paper are provided. The proposed algorithm has been implemented in *Microsoft Visual C++* and run on a personal computer, which has an *InterCore I5 CPU (3210M)* running at 2.50GHz and with 4.0GB memory RAM.

Sample problems are generated based on the following preconditions:

- the number of containers for each sample ranges from 100 to 500;
- for each sample problem, the containers are randomly classified into several groups, namely A, B, C, and so forth;
- the work schedule of the QC is generated by joining these container groups in a random sequence;
- containers required by the QC are allocated randomly in one or more blocks, each of which consists of 35 bays subjected to the constraint that only one group of container can be stacked in a bay.

Numerical experiments are organized by two sections: the first section is an actual case analysed with detailed results; the second section begins with a number of small-sized experiments followed by different large-sized tests designed to validate the effectiveness of the algorithm.

During actual operations of container terminal, YCs are always randomly scheduled, i.e., two YCs operate randomly under the premise of safety constraints without the consideration of balancing the workload and minimizing parking frequency. In this paper, such usual practice is defined as actual rules and also programmed to get actual solutions set as controlled trials.

Some preference settings are shown as follows: the travel distance between two adjacent bays is 7 meters, the distance between two YCs remains at least 12 meters for safety purpose at anytime, the moving speed of YCs is 5 meters per second, and the time required for a YC to load a container is 2 minutes. In order to conduct trial comparisons expediently, w_1, w_2, w_3 are set to 0.4, 0.4, 0.2 respectively.

4.2. An Actual Case

Actual data from DCT (Dalian Container Terminal, China) were collected and processed to build a case as shown in Tables 1 and 2. Assuming that YC1 and YC2 park at bay 45 and bay 72 respectively when the loading operation begins. The following shows the computational results solved by actual rules and the proposed algorithm respectively.

Table 3 illustrates the solution for two-YC scheduling problem according to actual rules, while Table 4 represents the solution by the two-stage hybrid algorithm. It is obvious that the make-span via the proposed algorithm is shorter than that of actual rules (206.653 vs. 226.770 mins). The performance of workload balance via the proposed algorithm is much better than that of actual rules (14 vs. 42 imbalances). The parking frequency decreases distinctly from 19 to 11 times. Besides, the total travel distance and of course the objective function value both see a dramatic decline comparing actual rules with two-stage hybrid algorithm. In general, by comparing figures of Table 3 with that of Table 4, optimization levels obtained by the proposed algorithm of the workload balance, parking frequency, travel distance, and make-span are 66.7, 42.1, 13.5, 9.7% respectively.

Table 3. The scheduling results of YCs by actual rules

	YC No	Work time [min]	Workload [TEU]	Parking frequency	Travel distance [m]	Objective function value (cost)
	1	220.607	110	8	224	
	2	226.770	68	11	399	
Total		226.770	178 (sum)/ 42 (difference)	19	623	149

Table 4. The scheduling results of YCs by two-stage hybrid algorithm

	YC No	Work time [min]	Workload [TEU]	Parking frequency	Travel distance [m]	Objective function value (cost)
	1	206.49	96	5	203	
	2	206.6533	82	6	336	
Total		206.6533	178(sum)/ 14 (difference)	11	539	117.8

Moreover, the proposed two-stage hybrid algorithm allows the scheduling process to be visual to the workers by outputting the details of the solution, as shown in Table 5. The time-space information can also be obtained during the computational process, which is illustrated in Table 6. The column 'space between YCs' explains that two YCs are under safety constraints during loading operation.

4.3. Various-Sized Experimental Tests

In this section, two types of certain-sized problems were developed featuring the amount of bays (i.e. small-sized: less than 10 bays, large-sized: 11 to 20 bays). For each problem of a given number of bays, 20 various experiments were conducted to obtain an average value of the indexes, of which 'travel distance' and 'make-span' are much more concerned during loading operations. Thus, the comparisons of these two indexes between the proposed two-stage hybrid algorithm and the actual rules are specially represented as follows.

Figs 6 and 7 show that for small-sized problems, differences of the travel distance and the make-span between two algorithms are both conspicuous. On average, the travel distance and the make-span obtained from the two-stage hybrid algorithm are 13.87, 10.22% shorter than that of the actual rules respectively.

Table 5. The scheduling details of two YCs

The scheduling details of YC 1						
Sequence	Start time [min]	End time [min]	Time consuming [min]	Action*	Park (Y/N)	Travel distance [m]
0	0.000	36.000	36.000	R: 45, 18, A	N	0
1	36.000	36.117	0.117	M: 45, 50	Y	35
2	36.117	60.117	24.000	R: 50, 12, C	N	0
3	60.117	60.234	0.117	M: 50, 45	Y	35
4	60.233	76.233	16.000	R: 45, 8, A	N	0
5	76.233	78.116	1.883	W: 45, 1.883	N	0
6	78.117	78.187	0.070	M: 45, 42	Y	21
7	78.187	90.117	11.930	W: 42, 11.930	N	0
8	90.117	114.117	24.000	R: 42, 12, B	N	0
9	114.117	114.304	0.187	M: 42, 50	Y	56
10	114.303	170.303	56.000	R: 50, 28, C	N	0
11	170.303	170.490	0.187	M: 50, 42	Y	56
12	170.490	206.490	36.000	R: 42, 18, B	N	0
Total			206.490		5	203
The scheduling details of YC 2						
Sequence	Start time [min]	End time [min]	Time consuming [min]	Action	Park (Y/N)	Travel distance [m]
0	0.000	36.000	36.000	R: 72, 18, A	N	0
1	36.000	36.187	0.187	M: 72, 80	Y	56
2	36.187	58.187	22.000	R: 80, 11, C	N	0
3	58.187	58.373	0.187	M: 80, 72	Y	56
4	58.373	60.117	1.744	W: 72, 1.743	N	0
5	60.117	90.117	30.000	R: 72, 15, A	N	0
6	90.117	90.163	0.046	M: 72, 70	Y	14
7	90.163	114.163	24.000	R: 70, 12, B	N	0
8	114.163	114.397	0.234	M: 70, 80	Y	70
9	114.397	130.397	16.000	R: 80, 8, C	N	0
10	130.397	132.303	1.906	W: 80, 1.907	N	0
11	132.303	132.420	0.117	M: 80, 85	Y	35
12	132.420	170.303	37.883	W: 85, 37.883	N	0
13	170.303	200.303	30.000	R: 85, 15, B	N	0
14	200.303	200.653	0.350	M: 85, 70	Y	105
15	200.653	206.653	6.000	R: 70, 3, B	N	0
Total			206.653		6	336

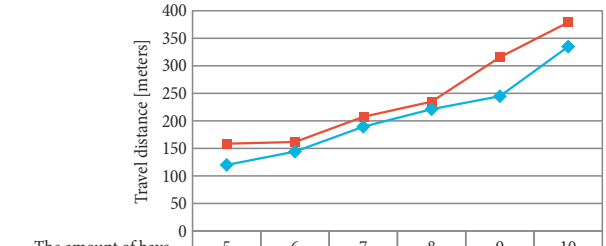
Notes: in column 'Action', 'R: 45, 18, A' represents 'Retrieve 18 containers of group A at bay 45'; 'M: 45, 50' represents 'Move from bay 45 to bay 50'; 'W: 45, 1.883' represents 'Stay at bay 45 for 1.883 min'.

Table 6. The time-space details of two YCs

Time [min]	Space of YC 1 (bay)	Space of YC 2 (bay)	Space between YCs (counted by bay)
0.000	45	72	27
36.000	45	72	27
36.117	50	77	27
36.187	50	80	30
58.187	50	80	30
58.373	50	72	22
60.117	50	72	22
60.233	45	72	27
76.233	45	72	27
78.117	45	72	27
78.187	42	72	30
90.117	42	72	30
90.163	42	70	28
114.117	42	70	28
114.163	44	70	26
114.303	50	76	26
114.397	50	80	30
130.397	50	80	30
132.303	50	80	30
132.420	50	85	35
170.303	50	85	35
170.490	42	85	43
200.303	42	85	43
200.653	42	70	28
206.490	42	70	28

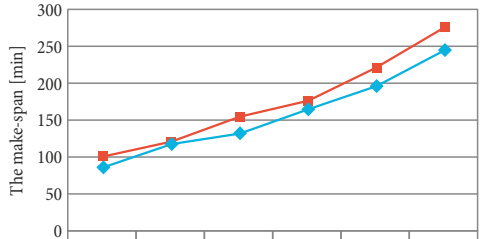
Similarly, the two-stage hybrid algorithm is able to achieve better solutions for large-sized problems compared with the actual rules, as shown in Figs 8 and 9. The results of travel distance and the make-span are 8.12, 8.10% better than the actual rules respectively.

In conclusion, the two-stage hybrid algorithm is appropriate to the structure of the solution space of the load-scheduling problem in this paper and a more promising solution method compared with the actual rules.



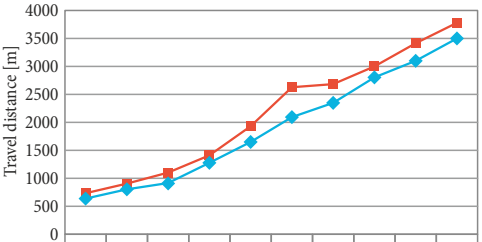
The amount of bays	5	6	7	8	9	10
Two-stage hybrid algorithm	120	144	190	220	245	335
Actual rules	157	162	208	235	315	379

Fig. 6. Comparison of travel distance between two-stage hybrid algorithm and actual rules for small-sized cases



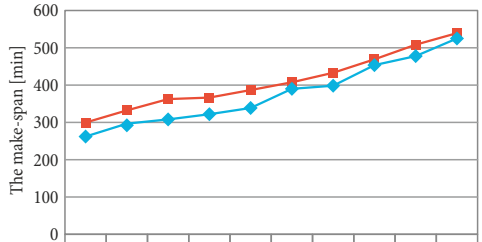
The amount of bays	5	6	7	8	9	10
Two-stage hybrid algorithm	85	118	132	164	196	244
Actual rules	100	120	154	176	220	276

Fig. 7. Comparison of make-span between two-stage hybrid algorithm and actual rules for small-sized cases



The amount of bays	11	12	13	14	15	16	17	18	19	20
Two-stage hybrid algorithm	632	798	911	1284	1631	2100	2340	2804	3100	3476
Actual rules	732	903	1102	1419	1934	2646	2700	2990	3420	3785

Fig. 8. Comparison of travel distance between two-stage hybrid algorithm and actual rules for large-sized cases



The amount of bays	11	12	13	14	15	16	17	18	19	20
Two-stage hybrid algorithm	264	298	311	325	341	391	401	455	479	523
Actual rules	299	335	364	370	386	411	435	471	510	541

Fig. 9. Comparison of make-span between two-stage hybrid algorithm and actual rules for large-sized cases

Conclusions

A load-scheduling problem of export containers was introduced for the case that multiple YCs and one QC are used in container terminals. The load-scheduling problem simultaneously determines three key factors: where to pick up containers, how many containers should be picked up and how the YC moves among bays during the loading operation. It was attempted to minimize the make-span of loading tasks by the means of balancing workload of YCs, reducing the parking frequency, and minimizing the travel distance. A mathematical model was developed for the load-scheduling problem. In order to evaluate the objective function, the container handling time at a bay, the travel distance between two

adjacent bays and the waiting time due to interference among YCs were considered.

A two-stage hybrid algorithm for optimal solutions was proposed to solve the problem. Usual operations obtained by actual rules were conducted as controlled trials. Both small and large-sized experiments showed that the two-stage hybrid algorithm outperforms the actual rules in their average make-span and average objective function value.

The load sequence of individual containers within a specific bay has not been treated in this paper, which is a topic for our further research.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China (No 71172108, No 71302044), Doctoral Program Foundation of Institutions of Higher Education of China (No 20122125110009), the Fundamental Research Funds for the Central Universities (No 3132013320, No 3132013076) and Postdoctoral Science Foundation of China (No 2013M530927).

References

Bian, Z.; Shao, Q.; Jin, Z. 2016. Optimization on the container loading sequence based on hybrid dynamic programming, *Transport* 31(4): 440–449. http://doi.org/10.3846/16484142.2014.994563

Chung, Y.-G.; Randhawa, S. U.; McDowell, E. D. 1988. A simulation analysis for a transtainer-based container handling facility, *Computers & Industrial Engineering* 14(2): 113–125. http://doi.org/10.1016/0360-8352(88)90020-4

Han, X. L. 2005. Routing problem of transfer crane at container terminals, *Journal of Shanghai Maritime University* 26(2): 39–41. (in Chinese).

Han, X.-L.; Ding, Y.-Z. 2008. Yard crane allocation and optimization in container terminal, *Navigation of China* 31(1): 6–14. (in Chinese).

He, J.; Mi, W.; Yan, W. 2007. Container yard crane scheduling based on hill-climbing algorithm, *Journal of Shanghai Maritime University* 28(4): 11–15. (in Chinese).

Kim, K. H.; Kim, K. Y. 1999. An optimal routing algorithm for a transfer crane in port container terminals, *Transportation Science* 33(1): 17–33. http://doi.org/10.1287/trsc.33.1.17

Kim, K. Y.; Kim, K. H. 2003. Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals, *Naval Research Logistics* 50(5): 498–514. http://doi.org/10.1002/nav.10076

Kim, K. Y.; Kim, K. H. 1997. A routing algorithm for a single transfer crane to load export containers onto a containership, *Computers & Industrial Engineering* 33(3–4): 673-676. http://doi.org/10.1016/S0360-8352(97)00219-2

Kim, K. H.; Lee, K. M.; Hwang, H. 2003. Sequencing delivery and receiving operations for yard cranes in port container terminals, *International Journal of Production Economics* 84(3): 283–292. http://doi.org/10.1016/S0925-5273(02)00466-8

Kim K. H.; Park, Y. M. 2004. A crane scheduling method for port container terminals, *European Journal of Operational Research* 156: 752–768. http://doi.org/10.1016/S0377-2217(03)00133-4

Lee, D.-H.; Cao, Z.; Meng, Q. 2007. Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm, *International Journal of Production Economics* 107(1): 115–124. http://doi.org/10.1016/j.ijpe.2006.08.003

Lee, D.-H.; Meng, Q.; Cao, Z. 2006. Scheduling two-transtainer systems for loading operation of containers using revised genetic algorithm, in *TRB 85th Annual Meeting Compendium of Papers CD-ROM*, 22–26 January 2006, Washington, DC, US, 1–13.

Narasimhan, A.; Palekar, U. S. 2002. Analysis and algorithms for the transtainer routing problem in container port operations, *Transportation Science* 36(1): 63–78. http://doi.org/10.1287/trsc.36.1.63.576

Ng, W. C. 2005. Crane scheduling in container yards with inter-crane interference, *European Journal of Operational Research* 164(1): 64–78. http://doi.org/10.1016/j.ejor.2003.11.025

Ng, W. C.; Mak, K. L. 2005a. An effective heuristic for scheduling a yard crane to handle jobs with different ready times, *Engineering Optimization* 37(8): 867–877. http://doi.org/10.1080/03052150500323849

Ng, W. C.; Mak, K. L. 2005b. Yard crane scheduling in port container terminals, *Applied Mathematical Modelling* 29(3): 263–267. http://doi.org/10.1016/j.apm.2004.09.009

Wie, Z.; Shen, J.; Xiao, R.; Zhang, Z.; Shi, D. 2007. Research on rubber tired gantry crane scheduling of port container terminal, *Engineering Sciences* 9(8): 47–51. (in Chinese).

Zhang, C.; Wan, Y. W.; Liu, J.; Linn, R. J. 2002. Dynamic crane deployment in container storage yards, *Transportation Research Part B: Methodological* 36(6): 537–555. http://doi.org/10.1016/S0191-2615(01)00017-0