Taylor & Francis
Taylor & Francis Group

TRANSPORT
ISSN 1648-4142 print / ISSN 1648-3480 online
2011 Volume 26(4): 383–393
doi:10.3846/16484142.2011.638022

A METHOD INTEGRATING SIMULATION AND REINFORCEMENT LEARNING FOR OPERATION SCHEDULING IN CONTAINER TERMINALS

Qingcheng Zeng[1], Zhongzhen Yang[2], Xiangpei Hu[3]

[1,2]College of Transportation Management, Dalian Maritime University, Dalian, China
[3]Institute of Systems Engineering, Dalian University of Technology, Dalian, China
E-mails: [1]zqcheng2000@hotmail.com (corresponding author); [2]yangzhongzhen@263.net; [3]drhxp@dlut.edu.cn

Submitted 10 July 2010; accepted 2 September 2011

Abstract. The objective of operation scheduling in container terminals is to determine a schedule that minimizes time for loading or unloading a given set of containers. This paper presents a method integrating reinforcement learning and simulation to optimize operation scheduling in container terminals. The introduced method uses a simulation model to construct the system environment while the Q-learning algorithm (reinforcement learning algorithm) is applied to learn optimal dispatching rules for different equipment (e.g. yard cranes, yard trailers). The optimal scheduling scheme is obtained by the interaction of the Q-learning algorithm and simulation environment. To evaluate the effectiveness of the proposed method, a lower bound is calculated considering the characteristics of the scheduling problem in container terminals. Finally, numerical experiments are provided to illustrate the validity of the proposed method.

Keywords: container terminals, scheduling, simulation, reinforcement learning.

1. Introduction

Along with the development of container transport, the throughput of container terminals rapidly increases. To meet the increasing container volume, container terminals have to construct new berths, expand equipment for processing loading and unloading operations or improve operation efficiency. Meanwhile, container terminals are facing challenges to providing better service so that vessel turnaround time can be shortened. Therefore, one of the most important issues relating to container terminals is improving operation efficiency.

For a major part of container terminals, there are mainly three types of equipment involved in loading and unloading operations, i.e. quay cranes, yard trailers and yard cranes. Upon ship arrival, quay cranes unload containers from or load containers onto the ship, yard trailers move containers from the quayside to the storage yard and vice versa. At the storage yard, yard cranes load and unload yard trailers.

Operation scheduling in terminals includes the features of multi-objectives, uncertainty and complexity, which has been proved to be a NP-hard problem. It is deemed unable to obtain optimal solutions to large-scale problems. Hence, heuristic algorithms are wildly used for obtaining near-optimal solutions efficiently. However, because of numerous constraints, it is difficult to evaluate a scheduling scheme in the process of heuristic algorithms. Meanwhile, although many constraints are considered in the scheduling model, it is too complex to analytically model all constraints. In addition, heuristic algorithms considered the set of operation tasks as having all required information at the initial time, which makes the algorithms to schedule the tasks in a static manner. In fact, information on operation scheduling in container terminals is uncertain in most cases.

To tackle complex constraints and stochastic factors, simulation is used for the scheduling problem in container terminals. Researchers developed simulation models for the problem of berth assignment, equipment deployment, storage optimization, traffic analysis etc. A simulation model can be used for evaluating the scheduling scheme, however, as a test and validation tool, it can only evaluate the given design rather than provide a more assistant decision making function.

The paper uses the Q-learning algorithm (reinforcement learning algorithm) for obtaining self-adaptability and dynamic scheduling rules for yard cranes and yard trailers and for integrating with simulation that is used for constructing the system environment. The Q-learning algorithm is applied to learn optimal dispatching rules for different equipment. The optimal scheduling scheme is obtained by the interaction between the Q-learning algorithm and simulation environment.

This paper is organized as follows. Section 2 briefly reviews previous works. The operation scheduling problem in container terminals is described in Section 3. Q-learning algorithms for scheduling yard cranes and yard trailers are designed in Section 4. The framework for the method integrating the Q-learning algorithm with simulation is developed in Section 5. The method for calculating a lower bound is designed in Section 6. Numerical experiments are used for testing the performance of the proposed method in Section 7. Conclusions are given in Section 8.

2. Literature Review

The issues related to container terminal operations have gained attention and have been extensively studied recently due to the increased importance of container transport. Studies on operation scheduling in container terminals can be divided into two types, namely mathematical optimization models and simulation models.

Owing to the complexity of container terminal operation, it is difficult to optimize the whole operation system with a single analytical model. Generally, the operation system in the container terminal is divided into several sub-processes each of which is respectively optimized. Researchers developed mathematical models for different sub-processes, e.g. a quay crane scheduling model (Daganzo 1989; Kim, Park 2004; Goodchild, Daganzo 2007; Lee *et al.* 2008), a yard crane allocation and scheduling model (Zhang *et al.* 2002; Linn *et al.* 2003; Kim *et al.* 2003, Ng 2005; Lee *et al.* 2007), storage optimization (Kim, Park 2003; Zhang *et al.* 2003) and a yard vehicle routing model (Liu, Ioannou 2002; Vis *et al.* 2005; Nishimura *et al.* 2005; Zeng *et al.* 2009) etc.

The operation efficiency of container terminals depends on the coordination of different sub-processes while the optimization models mentioned above cannot deal with the cooperation issues. To improve the coordination and efficiency of operation in container terminals, some researchers carried out studies on the cooperation of several activities. For example, Kozan and Preston (1999) established a model for optimizing loading and unloading orders and the storage strategy of containers in the yard. Bish (2003) provided models for determining storage location for each unloaded container, dispatch vehicles to containers and schedule loading and unloading operations on the cranes so as to minimize maximum time required for serving a given set of ships. Chen *et al.* (2007) developed an integrated model for optimizing the whole loading and unloading process. Lau and Zhao (2008) constructed an operation model for an automatic container terminal the objective of which was to simultaneously optimize AGV, working orders of quay cranes and yard trailers. These models and algorithms improved the coordination and integration of operation scheduling in container terminals. However, the problems of how to tackle with complex constraints and interrelation and how to improve computation efficiency have not been completely solved.

To solve scheduling models, mainly two methods are presently used. The first one searches for an optimal scheme in the solution space of the combinatorial optimization problem. Most of the above described studies belong to it, and heuristic algorithms are wildly used for obtaining near-optimal solutions. With an increase in the problem scale, computation efficiency greatly decreases. The second method deals with obtaining optimal scheduling rules given the initial and objective states. Reinforcement learning belongs to it. First, it observes changes in the environment from one state to another caused by the action of agents; then, it calculates the value function and finds the optimal scheduling rule employing the learning process of agents. Reinforcement learning can considerably reduce the calculation complex and obtain relatively rational scheduling rules. Thus, it has received more and more attention when dealing with the scheduling problem, e.g. Aydin and Öztemel (2000) proposed a dynamic scheduling system based on an agent and reinforcement learning was used for training the agents to obtain the optimal scheduling strategy. Wang and Usher (2004) applied the Q-learning algorithm to obtain an optimal assignment strategy for a single machine. Although reinforcement learning has been proved an efficient method to solve the scheduling problem, it has not been appropriately used in container terminals. This paper employs reinforcement learning to reduce the computation complexity of operation scheduling in container terminals.

The scheduling problem of container terminals involves numerous variables and constraints. When tackling with the complexion of the model and computation, especially considering uncertain and stochastic factors, analytic models often confront either the problem that the model is too simple or the problem that computation is too complex. Therefore, recently, the simulation has been wildly used for the scheduling problem of container terminals.

To simulate Kwai Chung container terminals, Shabayek and Yeung (2002) developed a simulation model using Witness software. Yun and Choi (1999) proposed a simulation model for the analysis of a container terminal system. The simulation model was developed using an object-oriented approach and applying SIMPLE++ object-oriented simulation software. Bielli *et al.* (2006) outlined a container terminal simulation model and gave component architecture that was implemented using Java.

A simulation model can be used for evaluating the scheduling scheme; however, as a test and validation tool, it can only evaluate the given design rather than provide a more assistant decision making function. Recently, the simulation optimization method has been proposed to overcome these limitations (Allaoui, Artiba 2004). Combining simulation analysis and the optimal decision-making mechanism, the simulation optimization method cannot only enhance intelligent decision-making of simulation but also build the complex system model easily, which is more difficult if employing traditional optimization methods. Zeng and Yang (2009) developed a simulation optimization model for optimizing the schedules of quay cranes, yard cranes and yard

trailers in container terminals. However, the main disadvantage of simulation optimization is long computation time. This paper focuses on integrating simulation with reinforcement learning that is used for reducing computation complexity, while simulation is used for tackling with complex constraints and obtaining the evaluation of each scheduling scheme.

3. Operation Scheduling in Container Terminals

Operations in container terminals fall into two groups, namely loading outbound containers and unloading inbound ones. For example, the process of loading outbound containers involves three stages: yard cranes pick up the desired containers from yard blocks and load them onto yard trailers first, then the yard trailers transport the containers to quay cranes, and finally the quay cranes load the containers onto the vessels.

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

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

For loading outbound containers, each container must undergo three handling operations: a transfer operation within the storage yard, a transfer operation of a container onto the ship and a transfer operation between quay cranes and yard cranes. There are three different sets of machines: quay cranes, yard cranes, and yard trailers. Therefore, a job can be defined as a complete loading process for a container. While comparing with the classical hybrid flow shop scheduling problem, operation scheduling in container terminals has several unique characteristics:

Job precedence constraints: for example, for loading, containers in the hold must precede containers on the deck of the same vessel, whereas for unloading, containers on the deck must be unloaded before containers in the hold.

Blocking: container terminals have no buffer between two successive machines; thus, blocking happens when the buffer is full. For example, when a yard trailer carries a container to a quay crane that is handling another container, it has to wait for the quay crane.

Setup times: in container terminals, there is empty movement when a crane or yard trailer moves between two jobs. For example, once a yard trailer carries an outbound container to a quay crane, it has to make an empty trip to the storage yard in order to processed next container. We denote it as *setup time*.

Based on the above analysis, the scheduling problem for loading or unloading operations in container terminals is treated as the HFSS problem. The objective is to assign each operation to a machine, sequence the assigned operations on each machine, and thus to minimize the makespan of loading or unloading operations.

4. Q-learning Algorithms for Operation Scheduling in Container Terminals

4.1. Q-learning Algorithms

Reinforcement learning is a kind of an unsupervised machine learning technique dealing with the problem of how an autonomous agent can learn to select proper actions through interacting with its system environment. Each time after an agent performs an action, the environment's response (as indicated by its new state) is used by the agent to reward or penalize its action. The objective is to develop a decision-making policy on selecting appropriate action rules for each agent. By reinforcement learning, optimal assigning rules for each agent can be obtained.

The Q-learning algorithm is one of the most wildly used reinforcement learning algorithms proposed by Watkins and Dayan (1992). The objective of this algorithm is to learn state-action pair value $Q(s, a)$ representing a long-term expected reward for each pair of the state and action (denoted by s and a respectively). $Q(s, a)$ can be denoted by the following equation:

$$Q(s_{t+1}, a) = (1-\alpha)Q(s_t, a) + \alpha(r + \gamma V^*), \quad (1)$$

where: $Q(s_{t+1}, a)$ is the expected value to execute action a at state s_t; r is the immediate reward for executing action a; α is the step-size parameter that influences the learning rate; γ is discount-rate parameter ($0 \leq \gamma \leq 1$) having an impact on the present value of future rewards; t is the stage of the action taken; V^* is the maximal value of Q under state s_{t+1}:

$$V^* = \max_i Q(s_{t+1}, a_i). \quad (2)$$

At each state, the probability of implementing a certain action can be calculated by the following equation:

$$p(a_i / s_t) = \frac{\{1 / Q(s_t, a_i)\}}{\sum_j \{1 / Q(s_t, a_j)\}}. \quad (3)$$

In the first iteration of the Q-learning algorithm, the probabilities of selecting all possible actions will be the same. However, along with the repeated iteration, the action with a smaller estimate of $Q(s, a)$ has a higher probability to be selected as the next action. Taking into account the used iterations, the optimal scheduling rule can be obtained.

A scheduling decision on container terminals can be divided into several inter-related stages. The decisions made at each stage depend on the current state and influence their successor states. The decisions reached at all stages form a dynamic sequence, the objective of which is the optimization of the whole scheduling process. Re-

inforcement learning exhibits the characteristics of being dynamic, multi-stage and real-time with the objective to obtain a scheduling strategy; thus, the expected accumulative rewards can be maximized at all states.

When using reinforcement learning for operation scheduling in container terminals, each piece of equipment can be regarded as an autonomous agent. When the system state or environment is changed, the agent can make a decision according to real-time state, namely determine the dispatching rule to select the following operation task. The existing studies indicate that the agent can select proper dispatching rules from a set of the given rules illustrating the feasibility and validity of reinforcement learning in the scheduling problem.

In this paper, we first design the Q-learning algorithms for yard cranes and yard trailers respectively to obtain optimal scheduling strategies for these two kinds of equipment. Then, we combine the Q-learning algorithm with simulation to develop an integrating scheduling model that includes all stages of the operation process.

4.2. The Q-learning Algorithm for Yard Crane Scheduling

In container terminals, yard storage is a place for the temporary storage of import and export containers to facilitate loading and unloading operations. In the storage yard, the cranes process the loading or unloading of yard trailers and the movement or rehandling of containers. The goal of yard crane scheduling is to decrease the waiting time of yard trailers by optimizing the operation sequence of yard cranes.

Let $w_i, i = 1, 2, \ldots, n$ denotes the time that the yard trailer arrives at the storage yard to wait for the yard crane to load or unload the container, $h_i, i = 1, 2, \ldots, n$ denotes the time needed for the yard crane to load or unload container i, $d_{ij}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, n$ denotes the time needed for the yard crane to move from storage location i to j, t_i denotes the time that the yard crane finishes the operation of container i. M is a sufficiently large constant. If the operation of container j is immediately precedes container i by the yard crane, $x_{ij} = 1$, and 0, otherwise.

For container i, waiting time in the storage yard can be denoted as $t_i - h_i - w_i$. Thus, the model for yard crane scheduling can be formulated as:

$$\text{Min} \sum_{i=1}^{n} (t_i - h_i - w_i) \tag{4}$$

$$\text{s. t. } t_i \geq w_i + h_i, \ i = 2, 3, \cdots, n; \tag{5}$$

$$t_j - t_i \geq d_{ij} + h_j - (1 - x_{ij})M;$$

$$i, j = 2, 3, \cdots, n \text{ and } i \neq j; \tag{6}$$

$$x_{ij} + x_{ji} = 1, \ i, j = 2, 3, \cdots, n \text{ and } i \neq j \tag{7}$$

$$x_{ij} = 0 \text{ or } 1, \ i, j = 2, 3, \cdots, n. \tag{8}$$

The objective function (4) is to minimize the total waiting time of yard trailers. Constraints (5) denote the relation of start, operation, and completion time of each task. Constraints (6) denote the relation of each operation task with its predecessor task. Constraints (7) ensure each operation task having at most one predecessor or successor task. Constraints (8) are binary constraints for decision variables.

The process of the Q-learning algorithm for the yard crane scheduling problem is:

Step 0: Initialize the value of Q. For all states s and actions a, let $Q(s,a) = 1$, $n = 1$.

Step 1: Obtain the current state s; if $n >$ N, the algorithm is the end, next, go to Step 2, otherwise.

Step 2: Select the action according to the current state. The probability of implementing a certain action is calculated by equation (3).

Step 3: Implement the selected action and obtain immediate rewards r and the next state. The objective of our model is to minimize the waiting time of yard trailers; therefore, r denotes time penalty for implementing a certain action, namely changes in the waiting time of the yard trailer. r can be calculated by equation (9) where n_j^1, n_j^0 are the numbers of yard trailers to wait for yard cranes at time t^1, t^0 in bay j, B is the number of the bay in a block.

$$r = \sum_{j=1}^{B} \sum_{t=1}^{n_j} \max(0, t^1 - d_{ij}) - \sum_{j=1}^{B} \sum_{t=1}^{n_j^0} \max(0, t^0 - d_{ij}). \tag{9}$$

Step 4: Update Q function according to equation (10):

$$Q(s_0, a) = (1 - \alpha) Q(s_0, a) + \alpha [r + \gamma \min_b Q(s_1, b)]. \tag{10}$$

Step 5: Update the system state, let $s_0 = s_1$, $n = n + 1$.

Step 6: If the stop criterion is reached, stop the algorithm and go to Step 1 otherwise.

To apply the Q-learning algorithm, a table showing states and policies should be designed. For yard crane scheduling, the state is defined by the waiting time of yard trailers and the expected mean service time (EMPT) of yard cranes. Three actions (rules) are used for assigning yard cranes to the yard trailer, namely first come first served (FCFS). Yard cranes make an unidirectional travel to select yard trailers in turn (UT) and the nearest yard trailer to serve first (NT) etc. State-action pairs are shown in Table 1.

The value of α influences learning efficiency and can be either a constant or a dynamic value changing along with the learning process. In this paper, we suppose α = 0.1. The value of γ is between 0 to 1; if it is close to zero, only an immediate penalty will be considered when selecting an action while the immediate penalty has small weight relative to a succeeding cumulative penalty if it is close to 1. This paper attempts to minimize the total penalty in the long run, so γ is set to be 0.9.

Numerical experiments are used for illustrating the validity of the Q-learning algorithm for yard cranes. Suppose that the block that the yard crane operated is composed of 40 bays. The arrival interval of yard trailers follows exponential distribution the mean value of which is 4 minutes. The operation efficiency of yard cranes follows normal distribution the expected value of

Table 1. States and policies of the Q-learning algorithm for yard crane scheduling

State	State criteria	FCFS	UT	NT
Dummy state	No waiting yard trailer	0	0	0
Dummy state	One waiting yard trailer	0	0	0
1	0≤AVT <m·EMPT	$Q(1, 1)$	$Q(1, 2)$	$Q(1, 3)$
2	m·EMPT≤AVT <2·m·EMPT	$Q(2, 1)$	$Q(2, 2)$	$Q(2, 3)$
3	2·m·EMPT≤AVT <3·m·EMPT	$Q(3, 1)$	$Q(3, 2)$	$Q(3, 3)$
4	3·m·EMPT≤AVT <4·m·EMPT	$Q(4, 1)$	$Q(4, 2)$	$Q(4, 3)$
5	4·m·EMPT≤AVT <5·m·EMPT	$Q(5, 1)$	$Q(5, 2)$	$Q(5, 3)$
6	5·m·EMPT≤AVT	$Q(6, 1)$	$Q(6, 2)$	$Q(6, 3)$

AVT: the average waiting time of yard trailers; m: multiple

which is 2 minutes/move. The movement speed of yard cranes is 7 seconds/bay. Stop criteria is 10000 iterations. The obtained results are shown as Fig. 1 indicating that we can obtain stable scheduling results, and the waiting time of yard trailers can reach convergence.

Furthermore, numerical tests are used for comparing the Q-learning algorithm and other three scheduling rules including FCFS, NT and UT. Table 2 shows the results of four methods for different arrival intervals of yard trailers. The obtained results disclose that FCFS is the worst rule among three rules (FCFS, NT, UT) while UT is the best one. When the arrival interval of yard trailers is short (e.g. 3 or 4 minutes), Q-learning is not the best method comparing to other three rules; however, when the arrival interval of yard trailers is long (e.g. 5 or 6 minutes), Q-learning is the best one. This indicates that with an increase in the arrival interval of yard trailers, the Q-learning algorithm becomes more efficient comparing to other three rules.

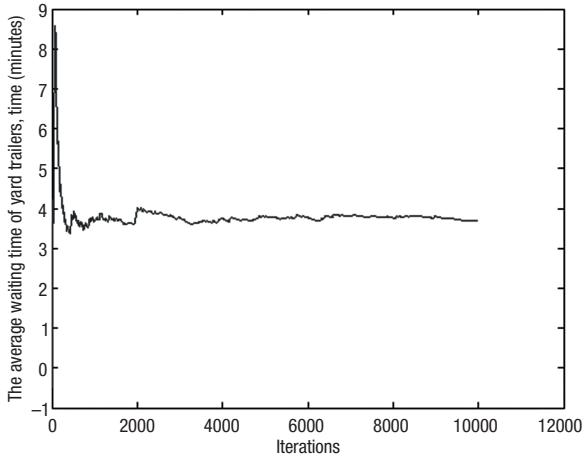

Fig. 1. The results of the Q-learning algorithm

Table 2. The results of different scheduling policies for different arrival intervals of yard trailers

Method	Arrival interval of yard trailers(minutes)			
	3	4	5	6
FCFS	8.986	4.9311	2.6803	1.5663
UT	7.6201	3.4631	2.1524	1.2997
NT	7.9564	3.5812	1.9589	1.3064
Q-learning	7.7239	3.7097	1.8012	1.2665

4.3. The Q-Learning Algorithm for Yard Trailer Scheduling

In container terminals, yard trailers transport a container between the quayside and the storage yard. Yard trailers are usually dispatched according to the operation order of quay cranes, the objective of which is to decrease the waiting time of quay cranes, thus improving loading or unloading efficiency.

Taking the unloading process as an example, the yard trailer transports an inbound container from the quayside to the storage yard first, and then returns to the quayside empty to transport next inbound container. Let $J=\{1, 2, \ldots, i, \ldots, n-1, n\}$ denotes the operation sequence of quay cranes, i denotes an inbound container, s denotes the operation efficiency of quay cranes, λ_i denotes the time required for the yard trailer to transport container i from the quayside to the storage yard, ST_i denotes starting time for container i that is the time when the quay crane unloads container i from the ship to the yard trailer, CT_i denotes completion time for container i that is the time when the yard trailer returns from the quayside after transporting container i to the storage yard and $V=\{v_1, v_2, \ldots, v_K\}$ denotes the set of yard trailers.

If container i is transported by yard trailer k, $x_{ik}=1$, and 0 otherwise. If the operation of container j is processed immediately after i by yard trailer k, $y_{ijk}=1$, and 0 otherwise. Thus, the model for yard trailer scheduling can be formulated as:

$$\min(CT_n - ST_1) \tag{11}$$

$$\text{s.t. } ST_i \geq 0 \quad i=2, 3, \cdots, n; \tag{12}$$

$$CT_i \geq ST_i + 2\lambda_i, \ i=1, 2, \cdots, n; \tag{13}$$

$$\sum_{k\in K} x_{ik} = 1, \ i=1, 2, \cdots, n; \tag{14}$$

$$\sum_{j\in N} y_{ijk} \leq 1, \forall k \in K, \forall i \in N; \tag{15}$$

Table 3. State policy for the Q-learning algorithm for yard trailer scheduling

State	State criteria	LW	LT	SCO
Dummy state	No waiting quay crane (QC)	0	0	0
Dummy state	One waiting QC	0	0	0
1	Number of waiting QCs = 1	$Q(1, 1)$	$Q(1, 2)$	$Q(1, 3)$
2	Number of waiting QCs = 2	$Q(2, 1)$	$Q(2, 2)$	$Q(2, 3)$
3	Number of waiting QCs = 3	$Q(3, 1)$	$Q(3, 2)$	$Q(3, 3)$
4	Number of waiting QCs = 4	$Q(4, 1)$	$Q(4, 2)$	$Q(4, 3)$
5	Number of waiting QCs = 5	$Q(5, 1)$	$Q(5, 2)$	$Q(5, 3)$
i	Number of waiting QCs = i	$Q(i, 1)$	$Q(i, 2)$	$Q(i, 3)$
n	Number of waiting QCs = n	$Q(n, 1)$	$Q(n, 2)$	$Q(n, 3)$

$$CT_i \le ST_j + H(1 - y_{ijk}); \tag{16}$$

$$ST_i + s \le ST_j + H(1 - y_{ijk}); \tag{17}$$

$$x_{ik}, y_{ijk} = 1 \text{ or } 0 \forall i, j \in N, k \in K. \tag{18}$$

The objective function (11) is to minimize the total unloading time. Constraints (12) ensure that all operation tasks begin after time zero. Constraints (13) denote the relation between the starting and completion time of each operation task. Constraints (14) ensure that each operation task is assigned only one yard trailer. Constraints (15) ensure that each operation task has at most one successor or operation task. Constraints (16)–(17) denote the relation between two adjacent operation tasks. Constraints (18) are simple binary constraints.

The process of the Q-learning algorithm for yard trailer scheduling is:

Step 0: Initialize the value of Q. For each state s and action a, let $Q(s,a) = 1$, $n = 1$.

Step 1: Obtain the current states; if $n > N$, the algorithm is the end, next, go to Step 2, otherwise.

Step 2: Select the action according to the current state. The probability of executing a certain action can be calculated by equation (3).

Step 3: Execute the selected action and obtain immediate rewards r and the next state. The objective of our model is to minimize the waiting time of quay cranes; therefore, r denotes a time penalty for executing a certain action, namely changes in the waiting time of quay cranes. r can be calculated by equation (19) where n', n are the numbers of quay cranes waiting for yard trailers at time t', t, D_i is the time that quay crane i can start the following operation task:

$$r = \sum_{i=1}^{n'} \max(0, t' - D_i) - \sum_{i=1}^{n} \max(0, t - D_i). \tag{19}$$

Step 4: Update Q according to equation (10).

Step 5: Update the system state, let $n = n + 1$.

Step 6: If the stop criterion is reached, stop the algorithm and go to Step 1, otherwise.

The state is defined according to the number of waiting quay cranes (Table 3) where n denotes the number of quay cranes. Three actions (dispatching rules) are used, namely assign yard trailers to the longest waiting (LW), assign yard trailers to the container with longest travel time (LT) and assign yard trailers to the fixed quay crane (SCO). State-action pairs for yard trailer scheduling are shown in Table 3.

Suppose, that 6 quay cranes are assigned to process an unloading operation, the processing times of quay cranes are generated from uniform distribution of $U(100, 150)$ seconds, the number of yard trailers is 30, the number of unloading containers is 400 and the travel time of yard trailers from the quayside to the yard storage follows the uniform distribution of $U(8, 11)$ seconds. The number of iterations is 10000. Let $\alpha = 0.1$, and $\gamma = 0.1$. The obtained results shown in Fig. 2 indicate that the algorithm can efficiently reach convergence.

Furthermore, numerical experiments are used for comparing the Q-learning algorithm with other three scheduling rules, e.g. LW, LT, and SCO. Table 4 shows the average quay crane waiting times of four methods and the selected probability of LW, LT and SCO rules in Q-learning. The received results indicate that SCO is the worst rule among three rules (LW, LT and SCO) while LW is the best one. When the number of yard trailers is 25, Q-learning is not the best method comparing to LW, LT and SCO while in other conditions

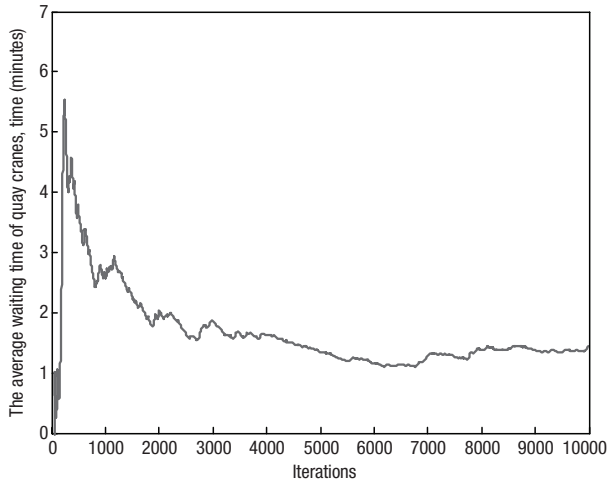

Fig. 2. The results of the Q-learning algorithm

Table 4. The average quay crane waiting time of different scheduling policies

Rules	Number of yard trailers			
	25	30	35	40
SCO	6.9873	3.8525	1.8374	1.3803
LW	2.7280	1.1837	0.5471	0.2524
LT	3.8391	1.3872	0.8735	0.3589
Q-learning	2.8290	0.9529	0.3201	0.2029
SCO (%)	5.14	3.01	3.47	1.35
LW(%)	80.26	86.42	81.96	89.83
LT (%)	14.58	10.57	14.57	8.82

(when the numbers of yard trailers are 30, 35, and 40 respectively), Q-learning is the best one. This indicates that with an increase in yard trailers, the Q-learning algorithm becomes more efficient comparing to LW, LT and SCO. With reference to the results of the Q-learning algorithm, LW appears as the rule selected as having the highest probability while SCO is the rule having the lowest probability.

5. The Method Integrating the Q-Learning Algorithm and Simulation

5.1. Integrating a Framework

The framework for the method integrating Q-learning and simulation (RLSO) is shown in Fig. 3. There are two kinds of agents, namely autonomous and global ones. Based on state *s* given by the simulation module, autonomous agents select action *a* with the probability of considering information on the 'state-action'. Then, the action is executed and reward value *r* is fed back to autonomous agents by the simulation module. Information on the 'state-action' is updated according to reward value *r*. After a certain learning period, a solution can be obtained and information is fed back to global agents that use the obtained data for updating information on the 'state-action', thus supervising the learning process of autonomous agents. Information on the 'state-action' includes state, action, *Q* value, the selected number of each 'state-action' pair etc.

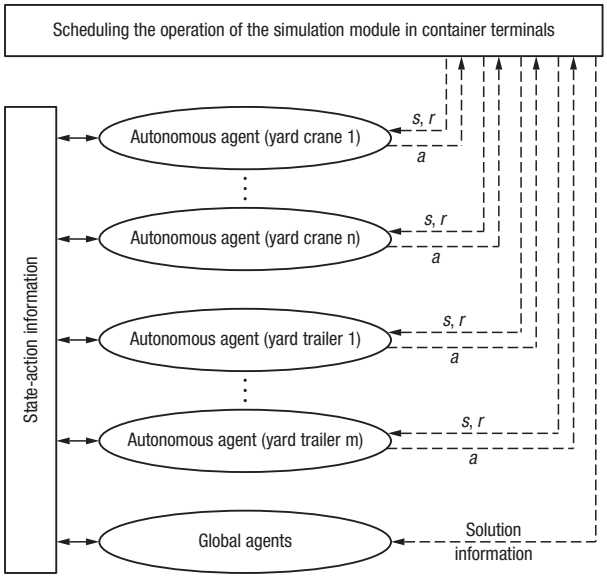

Fig. 3. A method integrating the Q-learning algorithm and simulation

5.2. Design of the State and Action

When dealing with the loading and unloading process of container terminals, yard cranes and yard trailers are dispatched according to the operation sequence of quay cranes pre-determined by the loading and unloading plan. Therefore, in real-time scheduling, we need not consider dispatching rules for quay cranes, but dispatching rules for yard trailers and yard cranes.

In our case, yard cranes and yard trailers are regarded as autonomous agents. These two kinds of agents are heterogeneous and have a different set of 'state-action' pairs. The design of the states and actions of agents are shown in Tables 1 and 3.

5.3. Calculating the Reward Function

When operating container terminals, reduction in the waiting time of quay cranes can improve loading and unloading efficiency. Therefore, the waiting time of quay cranes is regarded as a reward function. Let D denotes the waiting time of quay cranes at time t in a learning period. Autonomous agent i executes action $a_t(i)$ under state $s_t(i)$. Next, operation task J_t is processed. Following task J_t, the waiting time of quay cranes and the system state will change to D' and $s_t'(i)$ respectively. Thus, the reward function can be denoted as equation (20):

$$r_t(i) = D' - D. \tag{20}$$

This design of the reward function helps with decreasing the delay of quay cranes, and thus improves the efficiency of quay cranes and decrease the whole operation time.

5.4. Action Strategy for Global Agents

Let D_n denotes the quay crane waiting time obtained by the autonomous agents of the current learning period and D_{n-1} denotes the quay crane waiting time of the previous learning period. Then, the supervisor's information can be denoted as $\Delta = D_n - D_{n-1}$. Thus, the action strategy for global agents is as follows: when 'state-action' (s,a) is selected in such learning period, the Q function is updated according to equation (21):

$$Q(s,a) = Q(s,a) + \Delta. \tag{21}$$

For a pair of 'state-action', if $D_n < D_{n-1}$, $(\Delta < 0)$, the Q value of this pair of 'state-action' will decrease, which will increase the probability of selecting this action. On other hand, if $D_n > D_{n-1}$, $(\Delta > 0)$, the Q value of this pair of 'state-action' will increase, which will decrease the probability of selecting this action. By this means, global agents can supervise autonomous agents so that to select actions causing a reduction in quay crane waiting time.

6. Lower Bound

To determine the effectiveness of the developed method, we need to compare the makespan obtained by the proposed algorithms with the optimal one obtained by the solution to the HFSS problem in container terminals. However, finding the optimal makespan requires a solution to the mixed-integer program that can be rather time-consuming even for medium-sized problems. Therefore, an optimal solution is measured against a well-defined lower bound.

The problem of a lower bound is widely discussed within the scope of the HFSS problem. In this case, we derive the lower bound on the optimal makespan based on the method proposed by Stantos (1995) and the technique used by Jin *et al.* (2006), Zeng and Yang (2009).

Let $\mathrm{LS}(i, s)$ denotes the left-hand side sum of processing time from stage 1 to $s-1$ for job i, and $\mathrm{RS}(i, s)$ – the right-hand side sum of processing time from stage $s+1$ to S for job i. $\mathrm{LS}(i, s)$ and $\mathrm{RS}(i, s)$ are given in Eq.(22) and Eq.(23):

$$\mathrm{LS}(i, s)=\begin{cases}\sum_{l=1}^{s-1} p(i, l), & s>1;\\ 0, & s=1;\end{cases} \tag{22}$$

$$\mathrm{RS}(i, s)=\begin{cases}\sum_{l=s+1}^{S} p(i, l), & s<S;\\ 0, & s=S.\end{cases} \tag{23}$$

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

$$\mathrm{lb}_s=\frac{1}{m_s}\left[\sum_{i=1}^{m_s}\mathrm{LS}(i, s)+\sum_{i=1}^{n} p(i, s)+\sum_{i=1}^{m_s}\mathrm{RS}(i, s)\right],$$
$$s=1, 2, \ldots, S; \tag{24}$$

$$\mathrm{glb}=\max_{0\le s\le S}\left\{\mathrm{lb}_s\right\}. \tag{25}$$

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

For each container i, let $\mathrm{SET}_s^{\mathrm{L}}(i)$ denotes minimum time required setting up the container immediately preceding container i at stage s and w_{ijs} denotes setup time between container i and container j at stage s:

$$\mathrm{SET}_s^{\mathrm{L}}(i)=\min_j w_{ijs}. \tag{26}$$

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

$$\mathrm{LB}_s=\frac{1}{m_s}\left[\sum_{i=1}^{m_s}\mathrm{LS}(i, s)+\sum_{i=1}^{n} p(i, s)+\sum_{i=1}^{m_s}\mathrm{RS}(i, s)+\sum_{i=1}^{n-m_s}\mathrm{SET}_s^{L}(i)\right],$$
$$s=1, 2, 3; \tag{27}$$

$$\mathrm{LB}=\max_{1\le s\le 3}\left\{\mathrm{LB}_s\right\}. \tag{28}$$

7. Numerical Experiments

First, numerical experiments are used for indicating the validity of our method (RSLO). The details of the conducted experiments are as follows:

- The unloading process is considered.
- The processing time of quay cranes is generated in line with the uniform distribution of $U(100, 150)$ s, and the processing time of yard cranes follows the uniform distribution of $U(260, 320)$ s.
- The storage location in the yard for each container is selected randomly. The storage location determines processing time for yard trailers.
- Parameters for yard trailers are the same as in numerical experiments provided in Section 4.3.

According to the number of quay cranes, yard cranes, yard trailers and unloading containers, 10 scenarios are designed. The total operation and computation time of two methods is shown in Table 5. Also, the makespans obtained using RSLO are compared with the global lower bound. The compared index is a relative deviation (RD) calculated by Eq. (29), where $C_{\max}$ is the makespan obtained by the developed algorithms and LB is its lower bound:

$$\mathrm{RD}=\frac{C_{\max}-\mathrm{LB}}{\mathrm{LB}}\cdot 100. \tag{29}$$

Table 5 shows that the RLSO method can reach convergence in a relatively efficient time. The required time increases with an increase in the size of the problem; however, it is reasonable to be applied in realistic container terminal scheduling. In addition, RDs are between 0.52% and 5.20% for 10 scenarios, which indicates the validity of RSLO.

Table 5. The results of the RLSO method

Scenarios				Operation time (minutes)	Computation time (minutes)	RD (%)
Containers	QC	YC	YT			
10	1	2	5	23.1	<1	0.52
20	1	2	5	48.6	2.0	1.15
40	2	4	8	51.6	6.4	1.26
50	2	4	10	61.2	10.5	1.93
80	2	4	8	108.6	21.7	2.47
80	2	4	10	105.1	24.3	3.02
100	2	4	8	130.0	39.2	3.14
100	2	4	10	125.4	42.1	3.62
200	3	6	15	164.8	80.9	4.28
300	3	6	15	250.4	104.7	5.20

Furthermore, the makespans obtained by RSLO are compared with other rules designed in Section 4. The number of unloading containers is set to 300, and the numbers of yard cranes and yard trailers are 12 and 6 respectively. The obtained results are shown in Table 6.

Table 6 indicates that RLSO performs better than other 9 methods regarding RD. However, with an increase in the numbers of quay cranes, the difference between RLSO and other methods decreases. The first reason is that computation becomes more complex with an increase in quay cranes, which decreases the solution quality of RLSO. The other reason is that operation tasks dispatched to each yard crane and yard trailer in a unit time increase with an increase in quay cranes, which may decrease difference in varying rules.

Considering the operation practice of container terminals, FCFS and SCO are the most wildly used scheduling rules for yard cranes and yard trailers. These two rules are easy to implement, nevertheless, they cannot ensure to obtain an optimal scheduling scheme. In our case, we further compared RLSO with FCFS-SCO method, namely the scheduling rule for yard cranes is FCFS and that for the yard trailer is SCO. The results shown in Table 7 reveal that RSLO can improve the quality of the solution comparing to FCFS-SCO rule, which indicates that the 'state-action' strategy obtained by RSLO is better than a simple FCFS-SCO rule.

Finally, we compare RLSO with a Tabu search based algorithm designed by Chen *et al.* (2007). The results displayed in Table 8 indicate that the Tabu search algorithm designed by Chen *et al.* (2007) performs better than RLSO regarding computation time and RD. This is because when agents are trained employing the trail-and-error method, computation time increases greatly with an increased number of agents. On the other hand, the Q-learning algorithm is not aimed at obtaining the optimal schedule, thus the RD of RLSO is greater than that of Tabu search. However, in realistic scheduling, due to uncertain factors such as operation delay, mechanical breakdown etc., processing loading or unloading operations entirely by the pre-determined order obtained by the optimization algorithm is difficult.

Table 6. Comparing RLSO with different methods

Methods			RD for different number of quay cranes			
No	Yard cranes	Yard trailers	2	3	4	5
1	FCFS	SCO	15.47	14.45	12.28	11.70
2	FCFS	LW	14.28	13.70	11.57	10.48
3	FCFS	LT	13.30	11.84	10.60	9.72
4	UT	SCO	14.15	13.62	11.97	10.54
5	UT	LW	11.47	11.20	10.43	9.26
6	UT	LT	11.30	9.95	10.39	9.30
7	NT	SCO	12.75	11.37	10.96	9.45
8	NT	LW	10.29	9.70	9.25	8.40
9	NT	LT	10.43	9.85	9.16	8.51
10	RLSO		4.25	4.54	5.10	5.46

Table 7. The results after using RSLO to obtain dispatching rules

Scenarios					RD for different methods	
No	Containers	QC	YC	YT	FCFS-SCO	RSLO
1	40	2	6	8	2.82	1.25
2	40	2	6	10	3.14	1.47
3	50	2	6	8	2.75	1.68
4	50	2	6	10	3.60	2.04
5	80	2	6	8	3.81	2.41
6	80	2	6	10	4.19	3.05
7	100	2	6	10	3.92	2.94
8	100	2	6	12	4.58	3.42
9	200	4	10	15	4.76	3.76
10	200	4	10	18	6.14	4.69
11	400	4	12	18	11.47	5.93
12	400	4	14	20	14.79	6.40
13	500	4	12	18	14.90	6.95
14	500	4	14	20	15.28	7.22
15	600	4	12	18	16.17	7.38

Table 8. A comparison of RLSO with Tabu search

Scenarios					RSLO		Tabu search	
No	Containers	QC	YC	YT	Operation time (minutes)	RD (%)	Operation time (minutes)	RD (%)
1	20	1	2	5	2.0	1.15	<1	0.24
2	40	2	4	8	6.4	1.26	2.4	1.10
3	80	2	4	10	24.3	3.02	9.8	2.95
4	100	2	4	10	42.1	3.62	13.8	3.35
5	200	3	6	15	80.9	4.28	32.5	3.86
6	300	3	6	15	104.7	5.20	48.2	4.30
7	400	4	14	20	132.4	6.40	68.0	5.28
8	500	4	12	18	127.9	6.95	56.2	5.90
9	500	4	14	20	162.0	7.22	73.8	6.24
10	600	4	12	18	287.4	7.38	105.6	6.92

The optimal sequence cannot be adjusted according to the real-time state. The objective of the Q-learning algorithm is to obtain self-adaptability and other scheduling strategies that can be adjusted according to the system state. Therefore, the Q-learning algorithm has more manoeuvrability comparing to the optimization method.

8. Conclusions

Reinforcement learning is an important machine learning method wildly used in the control system. Recently, studies on reinforcement in the flow shop problem have gained more and more attention. The paper integrates the Q-learning algorithm with simulation to solve scheduling problems in container terminals. The obtained results indicate that the integrated method cannot only efficiently tackle with but also improves the self-adaptability of the operation scheduling problem in container terminals.

The method proposed in this paper can be extended in several directions. The first option is designing a reward function dispatching rules and action strategies more efficiently, so that to improve the solution quality and adaptability of the proposed method. The second one is studying the interaction among different types of agents, thus improving the coordination of the operation system used in container terminals.

Acknowledgement

This work is supported by the National Natural Science Foundation of China for distinguished young scholars (No. 70725004), the National Natural Science Foundation of China (Grant No. 70890080, 70890083, 71001012), the Ph.D. program Foundation of the Ministry of Education of China (No. 20092125120001) and Fundamental Research Funds for Central Universities (Grant No. 2011JC010).

References

Allaoui, H.; Artiba, A. 2004. Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints, *Computers and Industrial Engineering* 47(4): 431–450. doi:10.1016/j.cie.2004.09.002

Aydin, M. E.; Öztemel, E. 2000. Dynamic job-shop scheduling using reinforcement learning agents, *Robotics and Autonomous Systems* 33(2–3): 169–178. doi:10.1016/S0921-8890(00)00087-7

Bielli, M.; Boulmakoul, A.; Rida, M. 2006. Object oriented model for container terminal distributed simulation, *European Journal of Operational Research* 175(3): 1731–1751. doi:10.1016/j.ejor.2005.02.037

Bish, E K. 2003. A multiple-crane-constrained scheduling problem in a container terminal, *European Journal of Operational Research* 144(1): 83–107. doi:10.1016/S0377-2217(01)00382-4

Chen, L.; Bostel, N.; Dejax, P.; Cai, J.; Xi, L. 2007. A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal, *European Journal of Operational Research* 181(1): 40–58. doi:10.1016/j.ejor.2006.06.033

Daganzo, C. F. 1989. The crane scheduling problem, *Transportation Research Part B: Methodological* 23(3): 159–175. doi:10.1016/0191-2615(89)90001-5

Goodchild, A. V.; Daganzo, C. F. 2007. Crane double cycling in container ports: planning methods and evaluation, *Transportation Research Part B: Methodological* 41(8): 875–891. doi:10.1016/j.trb.2007.02.006

Jin, Z.; Yang, Z.; Ito, T. 2006. Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem, *International Journal of Production Economics* 100(2): 322–334. doi:10.1016/j.ijpe.2004.12.025

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. doi:10.1016/S0925-5273(02)00466-8

Kim, K. H.; Park, K. T. 2003. A note on a dynamic space-allocation method for outbound containers, *European Journal of Operational Research* 148(1): 92–101. doi:10.1016/S0377-2217(02)00333-8

Kim, K. H.; Park, Y.-M. 2004. A crane scheduling method for port container terminals, *European Journal of Operational Research* 156(3): 752–768. doi:10.1016/S0377-2217(03)00133-4

Kozan, E.; Preston, P. 1999. Genetic algorithms to schedule container transfers at multimodal terminals, *International Transactions in Operational Research* 6(3): 311–329. doi:10.1111/j.1475-3995.1999.tb00158.x

Lau, H. Y. K.; Zhao, Y. 2008. Integrated scheduling of handling equipment at automated container terminals, *International Journal of Production Economics* 112(2): 665–682. doi:10.1016/j.ijpe.2007.05.015

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. doi:10.1016/j.ijpe.2006.08.003

Lee, D.-H.; Wang, H. Q.; Miao, L. 2008. Quay crane scheduling with non-interference constraints in port container terminals, *Transportation Research Part E: Logistics and Transportation Review* 44(1): 124–135. doi:10.1016/j.tre.2006.08.001

Linn, R.; Liu, J.-Y.; Wan, Y.-W.; Zhang, C.; Murty, K. G. 2003. Rubber tired gantry crane deployment for container yard operation, *Computers and Industrial Engineering* 5(3): 429–442. doi:10.1016/S0360-8352(03)00046-9

Liu, C.-I.; Ioannou, P. A. 2002. A comparison of different AGV dispatching rules in an automated container terminal, in *The IEEE 5th International Conference on Intelligent Transportation Systems, 2002: Proceedings.* 3–6 September 2002, Singapore, 880–885. doi:10.1109/ITSC.2002.1041336

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

Nishimura, E.; Imai, A.; Papadimitriou, S. 2005. Yard trailer routing at a maritime container terminal, *Transportation Research Part E: Logistics and Transportation Review* 41(1): 53–76. doi:10.1016/j.tre.2003.12.002

Santos, D. L.; Hunsucker, J. L.; Deal, D. E. 1995. Global lower bounds for flow shops with multiple processors, *European Journal of Operational Research* 80(1): 112–120. doi:10.1016/0377-2217(93)E0326-S

Shabayek, A. A.; Yeung, W. W. 2002. A simulation model for the Kwai Chung container terminals in Hong Kong,

European Journal of Operational Research 140(1): 1–11. doi:10.1016/S0377-2217(01)00216-8

Vis, I. F. A.; De Koster, R. B. M.; Savelsbergh, M. W. P. 2005. Minimum vehicle fleet size under time-window constraints at a container terminal, *Transportation Science* 39(2): 249–260. doi:10.1287/trsc.1030.0063

Wang, Y.-C.; Usher, J. M. 2004. Learning policies for single machine job dispatching, *Robotics and Computer-Integrated Manufacturing* 20(6): 553–562. doi:10.1016/j.rcim.2004.07.003

Watkins, C. J. C. H.; Dayan, P. 1992. Q-learning, *Machine Learning* 8(3–4): 279–292. doi:10.1007/BF00992698

Yun, W. Y.; Choi, Y. S. 1999. A simulation model for container-terminal operation analysis using an object-oriented approach, *International Journal of Production Economics* 59(1–3): 221–230. doi:10.1016/S0925-5273(98)00213-8

Zeng, Q.; Yang, Z.; Lai, L. 2009. Models and algorithms for multi-crane oriented scheduling method in container terminals, *Transport Policy* 16(5): 271–278. doi:10.1016/j.tranpol.2009.08.006

Zeng, Q.; Yang, Z. 2009. Integrating simulation and optimization to schedule loading operations in container terminals, *Computers and Operations Research* 36(6): 1935–1944. doi:10.1016/j.cor.2008.06.010

Zhang, C.; Liu, J.; Wan, Y.-W.; Murty, K. G.; Linn, R. J. 2003. Storage space allocation in container terminals, *Transportation Research Part B: Methodological* 37(10): 883–903. doi:10.1016/S0191-2615(02)00089-9

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. doi:10.1016/S0191-2615(01)00017-0