Computers & Industrial Engineering 94 (2016) 32–44

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Modelling of integrated vehicle scheduling and container storage problems in unloading process at an automated container terminal

Jiabin Luo [a,*], Yue Wu [b], André Bergsten Mendes [c]

[a] Faculty of Engineering and Computing, Coventry University, Priory Street, Coventry CV1 5FB, UK
[b] Southampton Business School, University of Southampton, Southampton SO17 1BJ, UK
[c] Department of Naval Architecture and Ocean Engineering, University of Sao Paulo, Av. Prof. Mello Moraes, 2231, Sao Paulo 05508-030, Brazil

ARTICLE INFO

Article history:
Received 14 June 2014
Received in revised form 30 November 2015
Accepted 18 January 2016
Available online 23 January 2016

Keywords:
Integer programming
Automated container terminal
Vehicle scheduling
Container storage
Container unloading

ABSTRACT

Effectively scheduling vehicles and allocating storage locations for containers are two important problems in container terminal operations. Early research efforts, however, are devoted to study them separately. This paper investigates the integration of the two problems focusing on the unloading process in an automated container terminal, where all or part of the equipment are built in automation. We formulate the integrated problem as a mixed-integer programming (MIP) model to minimise ship's berth time. We determine the detailed schedules for all vehicles to be used during the unloading process and the storage location to be assigned for all containers. A series of experiments are carried out for small-sized problems by using commercial software. A genetic algorithm (GA) is designed for solving large-sized problems. The solutions from the GA for the small-sized problems are compared with the optimal solutions obtained from the commercial software to verify the effectiveness of the GA. The computational results show that the model and solution methods proposed in this paper are efficient in solving the integrated unloading problem for the automated container terminal.

1. Introduction

Containers are large steel boxes with standardised sizes, designed for easily handling and transporting of cargos. Container trade is the fastest-growing freight segment which had an average annual increase of 6.1% in tonnage from 2005 to 2013 (UNCTAD, 2014). Container terminals, performing as the interfaces between seaside and landside, have been playing an important role in global trading. Container terminals are highly capitalised, and the competitions, particularly for those geographically closed terminals, are very intense. Therefore, improving the efficiency of container terminals becomes a vital challenge for all port managers.

Typically, there are two major operation processes in container terminals: unloading process and loading process. During the unloading process, containers (i.e. import containers) are transported from ships to storage yard, before being loaded onto external trucks and/or trains for onward delivery. During the loading process, after being received from external trucks and/or trains, containers (i.e. export containers) are allocated to the storage yard for temporary storing, and then loaded onto the ships. The flow of containers in the unloading and loading processes through a terminal is shown in Fig. 1. This paper will focus on the container handling in the unloading process.

In recent years, there has been a tremendous growing in the investment of automated equipment, i.e. automated vehicles and automated cranes, in container terminals, in order to satisfy the increasing container traffic flows and also reduce labour costs; such container terminals that use automated equipment are called automated container terminals, for example, ECT Rotterdam, CTA Hamburg, PPT Singapore, etc. Among all automated vehicles, the most commonly used is automated guided vehicle (AGV). AGV is a mobile robot that can move on a road-type network that incorporates electric wires or transponders in the ground to control its position. The popularity of using automated vehicles in container terminals is expected to continue since internal transportation in non-automated terminals have been proved to be inefficient and costly (Vis, 2006). Fig. 2 shows an air view of the automated container terminal in Hamburg.

Except AGVs, there are other types of container handling equipment involved in the terminal operations. This paper considers an automated container terminal involving quay cranes (QCs), AGVs and yard cranes (YCs) for container handling. QCs are located along the quayside for unloading containers from the ship to the AGVs; AGVs travel between the quayside and yard side for delivering

* Corresponding author.
E-mail addresses: jiabin.luo@coventry.ac.uk (J. Luo), y.wu@soton.ac.uk (Y. Wu), andbergs@usp.br (A.B. Mendes).

http://dx.doi.org/10.1016/j.cie.2016.01.010
0360-8352/

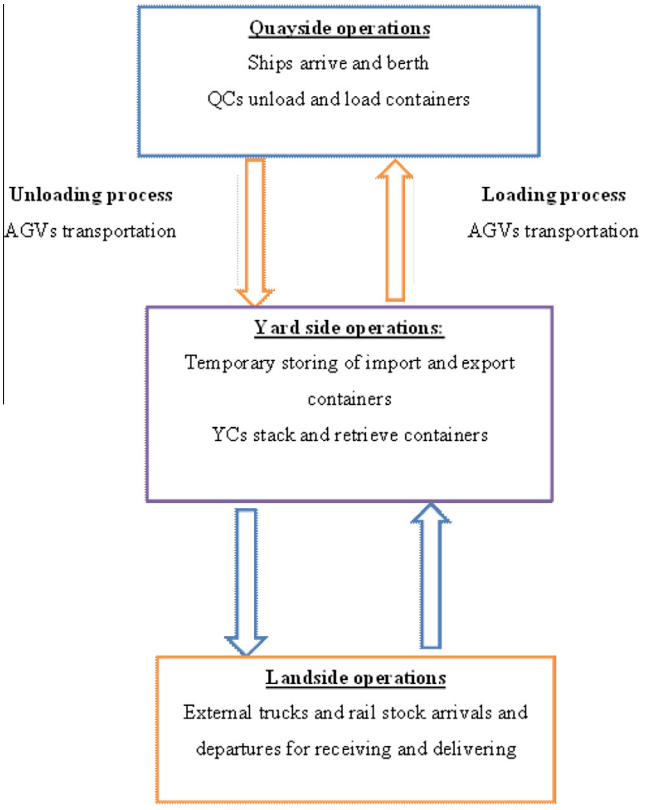

Fig. 1. Unloading and loading processes in a container terminal.

containers; YCs are used to move and stack containers within the yard. The slot in which a container is stored in the yard is called the yard location, and each container has to be assigned to a yard location, which is specified by a series of number (i.e. bay-row-tier).

Scheduling of vehicles has become one of the major planning issues for container terminals as inefficient vehicle schedules will cause delay in container-handling processes and thus affect the productivity of container terminals. In addition, container storage spaces are very limited due to the ever-increasing number of container flows through container terminals. Therefore, both scheduling vehicles and allocating containers are very critical in container terminal operations. Specifically, the vehicle scheduling problem determines delivery sequence and time to handle the containers for each vehicle; container storage problem determines the yard storage location for each container. Significant research has been devoted to the vehicle scheduling and container storage problems separately. The two problems are, however, highly interrelated for several reasons: (1) AGVs act an important role as the link between the quayside and yard side, and they interface the two problems; (2) container storage locations in the yard determine the YCs' schedules, which in turn affect the release time of each container from AGVs; (3) AGVs' schedule specify the time when each container is delivered to a yard location, i.e. where this container will be stored in. Therefore, it is important to address the two problems simultaneously. This paper focuses on the integration of the two problems during the container unloading operation, aiming to minimise the ship's berth time, which is one of the most important factors to evaluate the efficiency of container terminal operations.

The main contribution of this work is that we provide an integrated modelling approach to address the two critical problems, i.e. AGV/YC scheduling and container storage, which has not been considered in the literature. We also develop a novel-designed specialised method based on the genetic algorithm. This paper is organised as follows: Section 2 gives a brief review of previous studies on the AGV scheduling and container storage problems. The problem is described and formulated as a mixed-integer programming model in Section 3. Section 4 proposes a heuristic method, genetic algorithm (GA), which will be used for solving the large-sized problems. Section 5 gives the computational results for both small-sized and large-sized problems. Section 6 concludes this paper and suggests future works.

2. Literature review

Over the past decades, there have been emerging researches devoted on various aspects related to container terminal operations. The first comprehensive classification and review of the literature in the field of container terminals was given by Steenken, Voß, and Stahlbock (2004), followed by an updated paper by Stahlbock and Voß (2008). More recently, Carlo, Vis, and Roodbergen (2014b) presented an in-depth overview of studies on transport operations and analysed the container handling equipment used. A formal classification and overview of container

Fig. 2. Air view of a typical container terminal, Hamburg. source: www.maritimejournal.com

storage yard operations were provided by Carlo, Vis, and Roodbergen (2014a). In the following part of this section, we will focus on the previous studies about AGV scheduling and container storage problems in container terminals.

The use of AGVs has grown enormously since they were introduced in practice from 1955. Many AGV scheduling approaches were proposed in the literature. For example, Durrant-Whyte (1996) was among the first to investigate AGV application in container terminals. Chen et al. (1998) developed a greedy algorithm for dispatching AGVs; in their work, AGVs were assigned to a single QC during the handling process. Another study on the AGV scheduling was performed by Kim and Bae (1999), and the aim was to minimise the delays of QCs. In the follow-up work, Kim and Bae (2004) presented a look-ahead dispatching method to assign optimal delivery tasks to AGVs to minimise the delays of the QCs and the total travel time of AGVs. They proposed a mixed integer programming model, which was solved by a heuristic method. Briskorn, Drexl, and Hartmann (2007) formulated the AGV scheduling problem based on a rough analogy to inventory management, which was solved by an exact algorithm. More recently, Kim, Choe, and Ryu (2013) proposed a multi-criteria dispatching strategy for AGVs in an automated container terminal to minimise the delay of QCs and the empty travel by AGVs. A multi-objective evolutionary algorithm was developed to get a set of Pareto optimal solutions. The majority of the previous studies on the AGV scheduling assume that each AGV can only carry one container at a time. Grunow, Günther, and Lehmann (2004) proposed a novel heuristic dispatching algorithm for a fleet of multi-load (carrying more than one container) AGVs to minimise the total lateness of AGVs. Klerides and Hadjiconstantinou (2011) also considered the multi-load situation, and a mathematical optimisation model was proposed under a rolling horizon approach. Apart from the deterministic models that were addressed above, Angeloudis and Bell (2010) provided a flexible dispatching algorithm for real-time control of AGVs under various conditions of uncertainty, such as the uncertain waiting time at the quay cranes for AGVs.

The majority of the existing literature only considered the optimisation of AGVs without the consideration of other equipment, such as QCs and YCs, which could have huge impacts on the AGV scheduling. Meersmans and Wagelmans (2001) made the first attempt to integrate the scheduling of AGVs, QCs and YCs in an automated container terminal. A branch and bound algorithm and a beam search algorithm were developed to minimise the total makespan of the schedule, and the near-optimal solutions was obtained in a reasonable time in order. Lau and Zhao (2008) studied the scheduling of AGVs and YCs in an integrated way and presented a MIP model; the objective was to minimise the total travel time of AGVs/YCs and the delays of QC operations. Two heuristic methods based on GA were developed to obtain a near-optimal solution for the integrated scheduling problem. Xin, Negenborn, and Lodewijks (2014) proposed a method for scheduling QCs, AGVs and YCs in order to improve the handling capacity of automated container terminals in an energy-efficient way; they applied the simulation approach to show how energy-efficient scheduling of the equipment was achieved by the proposed model. However, none of the above work considered the integration of scheduling container handling equipment and container storage problem simultaneously.

There exist some works on the integration of vehicle scheduling and container storage problems in the literature, but not in automated container terminals, for example, the integrated yard truck and container storage problems in Han, Lee, Chew, and Tan (2008), Lee, Cao, and Shi (2008) and Lee, Cao, Shi, and Chen (2009). Han et al. (2008) studied the yard storage problem in the loading process for export containers, deciding how to reserve/assign yard storage blocks to each vessel. Traffic congestions for yard trucks were also considered in order to balance the workload among yard blocks. Given the generated yard storage plan, the objective was to determine the minimum total number of yard cranes to deploy. Their work was to assign yard storage spaces for the destination vessel of export containers without considering how each container is allocated in the yard, while our work gives integrated decisions on the schedules of vehicles, cranes and also on yard storage locations for each import container, and the objective of our work is to minimise the berth time of the ship. Lee et al. (2008) investigated the yard truck scheduling and storage allocation problem for import containers to reduce traffic congestions and total waiting time of trucks. Our work not only gives the decisions on the vehicle scheduling and storage allocation, but also determines the schedule of yard cranes, which was not considered in their paper. Lee et al. (2009) further extended the work of Lee et al. (2008). Their work, however, still focused on the traffic issues of yard trucks, which were used to move containers in the unloading and loading processes. The objective was to minimise the total delay and total travel time of yard trucks. The schedules of yard cranes in their work were known in advance. However, in our paper, the decisions on yard crane schedules are decided with other operations, such as container storage locations and vehicle schedules, which will result in a better decision as a whole. In summary, all of the above works focused on the reduction of the traffic during container handling process, while our paper aims to minimise the berth time of the ship, which is one of the most important factors to measure the efficiency of container terminals.

From the perspective of container storage problems, Bruzzone and Signorile (1998) combined simulation and genetic algorithm to determine the storage spaces of containers. Kim, Park, and Ryu (2000) proposed a dynamic programming model to determine the storage location for export containers with the consideration of containers' weight. The objective was to minimise the total number of rehandles (caused by retrieving the containers that stored underneath) during the loading operation. Chen, Fu, Lim, and Rodrigues (2003) developed a genetic algorithm for the general yard allocation problem to minimise the yard space to be occupied. Zhang, Liu, Wan, Murty, and Linn (2003) formulated the storage space allocation problem using a rolling-horizon approach and decomposed the problem into two levels: the first level was to determine the total number of containers associated with each block in the yard, and the second level was to determine the number of containers associated with each vessel. Bazzazi, Safaei, and Javadian (2009) extended the above problem by considering reefer and empty containers, and the objective was to minimise the total handling time, including storage and retrieve time of containers in the yard; an efficient GA was presented for obtaining feasible solutions for practical cases. Woo and Kim (2011) developed a method for allocating storage space for export containers using space-reservation strategies, in which adjacent stacks/slots were reserved for containers with the same attributes (i.e. same size, same weight group and same destination). Jiang, Lee, Chew, Han, and Tan (2012) studied two space-sharing approaches for container storage in the yard to improve the yard utilisation. Spaces were dynamically reserved for containers to balance the workload and ease the traffic congestions. Chen and Lu (2012) addressed the storage allocation problem for export containers. A MIP model was developed and a hybrid sequence stacking algorithm was proposed. Ndiaye, Yassine, and Diarrassouba (2014) developed a branch-and-cut algorithm to minimise the total vehicle travel for the container storage problems in order to increase the productivity of port. Simulation was used to demonstrate the effectiveness of the algorithm. Sriphrabu, Sethanan, and Theerakulpisut (2014) studied storage allocation and export container reshuffle problems to minimise the number of container lifting; heuristic approaches were developed and the results were compared with some practical rules, such as first-in-first-stored.

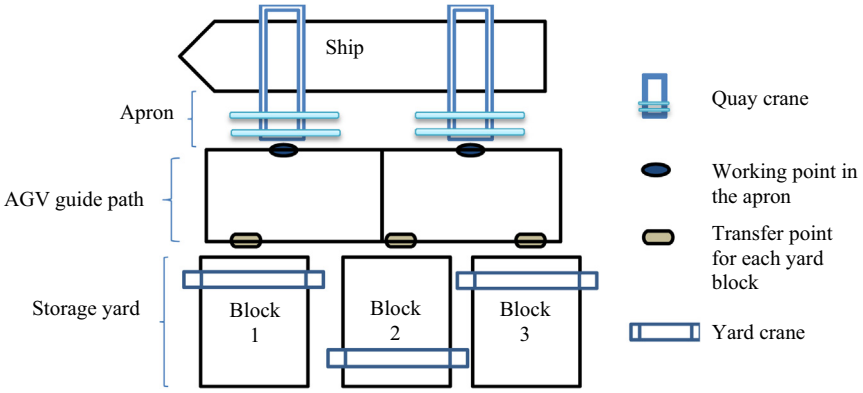

Fig. 3. The layout of an automated container terminal.

The integration problems that have been studied in the literature are either the integration of vehicle scheduling and yard crane scheduling, or the integration of vehicle scheduling and container storage. To our best knowledge, integration of vehicles, yard cranes and storage locations for import containers has hardly addressed in the literature.

3. Problem description

We consider an automated container terminal, which consists of a berthing area at the quayside, an AGV's travelling area and a storage yard. The berthing area is equipped with quay cranes (QCs) for unloading containers; the storage yard is used for temporarily storing of import containers before further delivery by trains or trucks; AGVs are used to move containers from the berthing area to the storage yard. Fig. 3 illustrates a layout of a typical automated container terminal.

The places where AGVs transfer containers to YCs are referred as transfer points (see Fig. 3). The transfer points are located in front of each block, where the YCs pick up the containers from the AGVs. Apron in Fig. 3 represents the area at the quayside where containers are loaded from QCs onto AGVs. Working points in Fig. 3 are the area where QCs drop off containers onto AGVs.

Before a ship arrives at the terminal, a berth area is allocated according to the tonnage, estimated arrival time and berth time (Lee et al., 2009). In addition, a number of QCs have also been decided to work on this ship in advance. The sequence of handling containers by QCs (i.e. the unloading sequences of the containers), are also known (Grunow et al., 2004). At the beginning of the unloading process, AGVs are at the quayside ready for handling containers from QCs. After a QC picks up a container from the ship and puts it onto an AGV, the AGV travels along the guided paths from a working point at the front of the QC to a transfer point at the front of a block, waiting for a YC to move the container to an assigned yard location. Then, the AGV with empty load is free to return to the quayside to handle the next container. If there are no QCs available to put a container onto an AGV, the AGV has to wait at the working point in the apron. Similarly, when an AGV arrives at the transfer point in the storage yard and there are no YCs available to pick up the container, the AGV needs to wait until the container is collected by a YC. Travelling times of AGVs between any QCs and any transfer point of the yard block are known. In this study, we consider YCs as rubber tyred gantry cranes (RTGC), which move on rubber tyred wheels spanning over a block of space, and also can move from a block to another, which is also the case in Linn, Liu, Wan, Zhang, and Murty (2003). We assume that the time needs for QCs dropping off containers onto AGVs and YCs picking up containers from AGVs are negligible.

When more than two QCs are working for a ship, either non-pooling or pooling policy will be adopted in the literature. Fig. 4 describes the non-pooling and pooling policies. The arrows in Fig. 4 indicate the assignment of AGVs to QCs. In the non-pooling policy, each AGV can only serve a QC. For example, in Fig. 4(a), AGV1 can only serve QC1. By contrast, in the pooling policy, each AGV can serve any QCs. For example, in Fig. 4(b), AGV1 can serve both QC1 and QC2. Although non-pooling is easy to implement because each AGV serves only one QC, we adopt the pooling policy in this paper since it is commonly used in the automated container terminals. In addition, congestion among AGVs on the path is not considered. Investigating the interference of vehicles involves more complex scheduling and control of detailed movements of vehicles, which is beyond the scope of this paper.

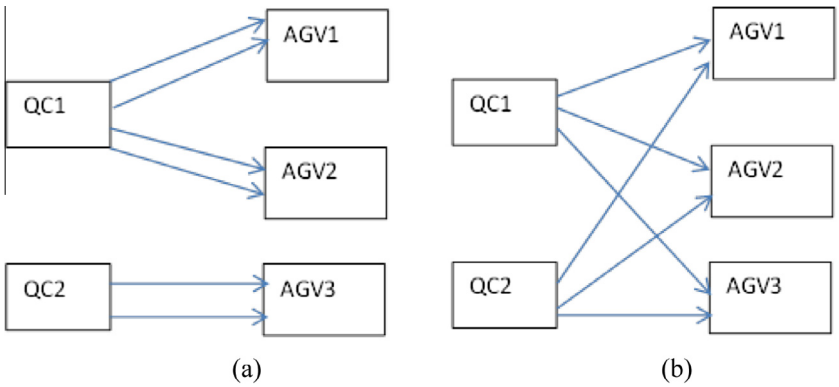

Fig. 4. (a) Non-pooling strategy and (b) pooling strategy in container unloading process.

The container storage yard is used to temporarily store containers until they are picked up by external vehicles/trains (or loaded onto the ship if they are export containers). A yard is normally divided into rectangular regions called blocks. A block consists of a number of stacks. In each stack, containers are stored one on top of each other at different levels (A stack could normally accommodate up to 6 containers). Therefore, each yard location can be represented by three indexes: block, stack and level. Moving and stacking containers in the yard are performed by YCs. In this paper, YC travelling times from any transfer point of blocks to any yard locations are known as they are depend on where the yard location is. The interference (conflict) among YCs and the interference (conflict) among QCs are also not considered, because the interferences are difficult to anticipate without scheduling and controlling detailed movements of the cranes. We also assume that number of import containers, and number of AGVs, QCs and YCs are all known. In addition, QCs, YCs and AGVs can only handle one container at a time.

The objective is to minimise the unloading time element of the berth time of the ship. The main operational decisions of the problem are to determine: (1) the schedules of AGVs to deliver containers; (2) the schedules of YCs to move containers; and (3) the assignment of yard locations to the containers.

The following notations will be used in modelling the integrated AGV scheduling and storage allocation problem in the unloading process.

Index, sets and parameters

D	set of import containers
K	set of QCs
P	set of container storage locations
B	set of blocks
V	set of AGVs
C	set of YCs
v	total number of AGVs available
c	total number of YCs available
$k,\ l$	index for QCs
$b,\ a$	index for blocks
n	index for yard stack
$(i,k),\ (j,l)$	index for containers; container (i,k) means the ith container to be handled by QC k
N_k	total number of containers to be handled by QC k. Therefore, container (N_k,k) means the last container (i.e. the N_kth) handled by QC k
(n,b)	stack n in block b
$Q_{(n,b)}$	maximum number of available slots/locations in stack n of block b
(q,n,b)	yard location, i.e. the qth level in stack n of block b, where $1 \leqslant q \leqslant Q_{(n,b)}$
$h_{(i,k)}$	QC's handling time for container (i,k)
$t_{(k,b)}$	AGV's travelling time between the working point for QC k and the transfer point in block b
$w_{(a,b)}$	YC's travelling time between the transfer point in block a and the transfer point in block b
$\varphi_{(q,n,b)}$	YC's travelling time between the transfer point of block b and level q in stack n of block b
M	a very large positive number
(S,I)	a dummy starting job (a job is defined as a container to be unloaded)
(F,I)	a dummy ending job
O_S	job set including all real jobs (i.e. all the containers to be unloaded) plus the dummy starting job: $O_S = D \cup (S,I)$
O_F	job set including all real jobs plus the dummy ending job, $O_F = D \cup (F,I)$
O	job set including all real jobs, the dummy starting job and ending job, $O = \{(S,I),(F,I)\} \cup D$

Decision variables

$u_{(i,k)}$	the time when QC k starts to handle container (i,k) from the ship
$d_{(i,k)}$	the time when a YC starts to handle container (i,k) (i.e. the time a YC picks up container (i,k) from an AGV.)

$$x_{(i,k)}^{(j,l)} = \begin{cases} 1, & \text{if the AGV, which just handles container } (i,k), \\ & \text{is scheduled to handle container } (j,l) \\ 0, & \text{otherwise. } \forall (i,k) \in O_S,\ \forall (j,l) \in O_F \end{cases}$$

$$z_{(i,k)}^{(q,n,b)} = \begin{cases} 1, & \text{if container } (i,k) \text{ is stored in location } (q,n,b) \\ 0, & \text{otherwise. } \forall (i,k) \in D, \forall (q,n,b) \in P \end{cases}$$

An intermediate decision variable $y_{(i,k)}^{b}$ is introduced:

$$y_{(i,k)}^{b} = \sum_{q=1}^{Q(n,b)} \sum_{n \in N^+} z_{(i,k)}^{(q,n,b)}, \quad \forall (i,k) \in D, \forall b \in B, \forall (q,n,b) \in P$$

$$y_{(i,k)}^{b} = \begin{cases} 1, & \text{if container } (i,k) \text{ is located in block } b \\ 0, & \text{otherwise. } \forall (i,k) \in D, \forall b \in B \end{cases}$$

$$\sigma_{(i,k)}^{(j,l)} = \begin{cases} 1, & \text{if the YC, which just handles container } (i,k), \\ & \text{is scheduled to handle container } (j,l) \\ 0, & \text{otherwise. } \forall (i,k) \in O_S,\ \forall (j,l) \in O_F \end{cases}$$

The mathematical programming model for this integrated problem can be formulated as follow:

$$\textbf{Min}: \ \max_{k} \left(u_{(N_k,k)} + h_{(N_k,k)}\right)$$

$$\textbf{Subject to}: \ \sum_{(j,l) \in O_F} x_{(i,k)}^{(j,l)} = 1,\ \forall (i,k) \in D \tag{1}$$

$$\sum_{(i,k) \in O_S} x_{(i,k)}^{(j,l)} = 1,\ \forall (j,l) \in D \tag{2}$$

$$\sum_{(j,l) \in D} x_{(S,I)}^{(j,l)} \leqslant v \tag{3}$$

$$\sum_{(i,k) \in D} x_{(i,k)}^{(F,I)} \leqslant v \tag{4}$$

$$\sum_{(q,n,b) \in P} z_{(i,k)}^{(q,n,b)} = 1,\ \forall (i,k) \in D \tag{5}$$

$$\sum_{(i,k) \in D} z_{(i,k)}^{(q,n,b)} \leqslant 1,\ \forall (q,n,b) \in P \tag{6}$$

$$\sum_{q=1}^{Q(n,b)} \sum_{n \in N^+} z_{(i,k)}^{(q,n,b)} = y_{(i,k)}^{b},\ \forall (i,k) \in D,\ \forall b \in B,\ \forall (q,n,b) \in P \tag{7}$$

$$\sum_{(i,k) \in D} z_{(i,k)}^{(q,n,b)} \leqslant \sum_{(i,k) \in D} z_{(i,k)}^{(q-1,n,b)},\ \forall (q,n,b) \in P,\ 1 \leqslant q \leqslant Q_{(n,b)} \tag{8}$$

$$\sum_{(j,l) \in O_F} \sigma_{(i,k)}^{(j,l)} = 1,\ \forall (i,k) \in D \tag{9}$$

$$\sum_{(i,k) \in O_S} \sigma_{(i,k)}^{(j,l)} = 1,\ \forall (j,l) \in D \tag{10}$$

$$\sum_{(j,l) \in D} \sigma_{(S,I)}^{(j,l)} \leqslant c \tag{11}$$

$$\sum_{(i,k) \in D} \sigma_{(i,k)}^{(F,I)} \leqslant c \tag{12}$$

$$u_{(i,k)} + h_{(i,k)} + \sum_{b \in B} t_{(k,b)} y_{(i,k)}^{b} \leqslant d_{(i,k)},\ \forall (i,k) \in D \tag{13}$$

$$d_{(i,k)} + \sum_{b \in B} t_{(l,b)} y_{(i,k)}^{b} \leqslant u_{(j,l)} + h_{(j,l)} + M * \left(1 - x_{(i,k)}^{(j,l)}\right), \quad \forall (i,k) \in O_S,\ (j,l) \in O_F \tag{14}$$

$$d_{(i,k)} + \sum_{(q,n,b) \in P} \varphi_{(q,n,b)} * z_{(i,k)}^{(q,n,b)} + \sum_{a,b} w_{(a,b)} * y_{(i,k)}^{b} * y_{(j,l)}^{a} \leqslant d_{(j,l)} + M * \left(1 - \sigma_{(i,k)}^{(j,l)}\right),\ \forall (i,k) \in O_S,\ (j,l) \in O_F,\ a,\ b \in B \tag{15}$$

$$u_{(i+1,k)} - u_{(i,k)} \geqslant h_{(i,k)},\ \forall (i+1,k), \quad (i,k) \in D,\ i = 1,2,\ldots N_k - 1 \tag{16}$$

$$x_{(i,k)}^{(j,l)},\ y_{(i,k)}^{b},\ z_{(i,k)}^{(p,n,b)},\ \sigma_{(i,k)}^{(j,l)} \in \{0,1\},\ \forall (i,k),\ (j,l) \in O, \quad \forall (q,n,b) \in P,\ \forall b \in B \tag{17}$$

$$u_{(i,k)},\ d_{(i,k)} \geqslant 0,\ \forall (i,k) \in D,\ i = 1,2,\ldots,N_k,\ \forall k \in K \tag{18}$$

The objective of the model above is to minimise the makespan of the unloading time of the QC's operations, i.e. the time when the last container has been unloaded from the ship. Constraint (1) ensures that each container (i,k) in D has a successor container (j,l) in O_F and both of them are delivered by the same AGV. Constraint (2) represents that every container (j,l) in D has a predecessor container (i,k) in O_S, and both of them are delivered by the same AGV. Constraints (3) and (4) ensure that the total number of AGVs to be used does not exceed the maximum number of the available AGVs. Constraint (5) means that each import container (i,k) will be assigned to one of the available locations (q,n,b) in the yard. Constraint (6) ensures that each location (q,n,b) in the yard can hold at most one container (some locations may not store any container). Constraint (7) means that if a container is assigned to block b, it can only be assigned to an available yard location in level p, stack n in block b, and the maximum level in a stack do not exceed its limitation. This constraint gives the relationship between the two decision variables $y^b_{(i,k)}$ and $z^{(q,n,b)}_{(i,k)}$. Constraint (8) ensures containers that are assigned in the same stack will be placed in order, i.e. containers will be placed in the first level, then the second level and so on within the same stack, and do not exceed stacks' maximum capacity. Constraint (9) implies that for any container (i,k) in D there is only a container (j,l) succeeding it handled by the same YC. Constraint (10) implies that for any container (j,l) in D, there is a predecessor container (i,k) handled by the same YC. Constraints (11) and (12) ensure that the total number of YCs deployed for handling containers does not exceed the maximum number of the available YCs. Constraint (13) means that a YC can only start to handle container (i,k) after an AGV delivers it to the transfer point in the assigned block. Constraint (14) means that an AGV can only start to handle container (j,l) at the working point after the AGV delivers container (i,k) to the yard and returns back to the working point, where container (j,l) is waiting. Constraint (15) ensures that a YC can only start to handle container (j,l), after the YC places container (i,k) in its assigned yard location and travels back to the transfer point of the block, where container (j,l) is waiting. Constraint (16) ensures that QC k can start to handle container $(i+1,k)$ only if it finishes handling the container (i,k). Constraints (17) and (18) are binary and non-negative restrictions.

When the constraint (15) is converted into a linear function (see below), the above model becomes a mixed integer programing model.

$$d_{(i,k)} + \sum_{(q,n,b)\in P} \varphi_{(q,n,b)} * z^{(q,n,b)}_{(i,k)} + w_{(a,b)} \leqslant d_{(j,l)} + M * \left(3 - \sigma^{(j,l)}_{(i,k)} - y^b_{(i,k)} - y^a_{(j,l)}\right), \quad \forall (i,k) \in O_S,\ (j,l) \in O_F,\ a,\ b \in B \tag{19}$$

The above integrated AGV scheduling and container storage problem is an NP-hard problem, which is difficult to be solved by using optimisation software. It is particularly true for large-scale problems that normally happen in practical unloading problems. Therefore, we develop a genetic algorithm (GA) in the following section.

4. Genetic algorithm

Genetic algorithm (GA) is a well-known heuristic approach for finding solutions to optimisation problems (Holland, 1975; Goldberg, 1989). The GA procedure adopted in this research is shown in Fig. 5.

4.1. Chromosome representation and initialisation

The initial step of the GA is to design the initial chromosome/-solution representations, which play an important role in the performance of the GA (Goldberg, 1989). By considering the decisions, represented by $x^{(j,l)}_{(i,k)}$, $y^b_{(i,k)}$, $z^{(q,n,b)}_{(i,k)}$ and $\sigma^{(j,l)}_{(i,k)}$, we use a matrix to represent solutions of the proposed problem. The matrix in Fig. 6 includes containers (Column 1), dispatched AGV (Column 2), assigned YC (Column 3), and assigned stack (Column 4). Each row in matrix Ψ is referred as the chromosome representation for each container under QC k.

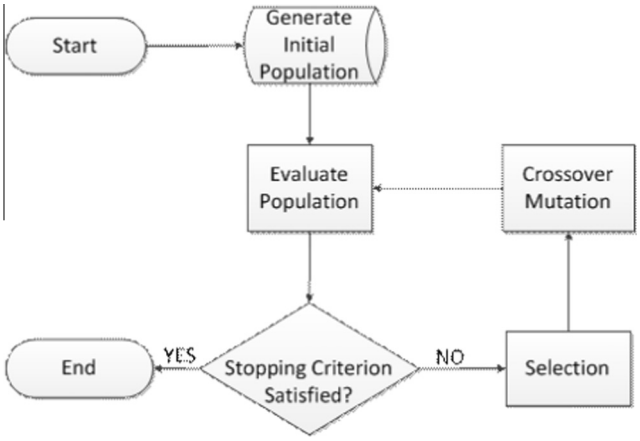

Fig. 5. Flow chart of the genetic algorithm.

Fig. 6 can be illustrated as follows. It gives a solution for the problem with 10 containers, handled by 2 QCs, 3 AGVs and 3 YCs. There are 5 stacks that are selected. Assuming stacks 1 and 2 are located in block 1, stacks 3 and 4 in block 2 and stack 5 in block 3, container (2, 2) will be handled by QC2, then by AGV3, and finally handled by YC1 to be stored in stack 1 (in block 1). Similarly, container (3, 2) will be handled by QC2 first, and then transported by AGV1, finally collected by YC2 to be stored in stack 2 (in block 1). For the containers that have been assigned to the same AGV (or YC), the handling sequences can be obtained according to their generated chromosomes. According to Fig. 6, we can obtain a delivery sequence of AGV1, which is container (1, 1), (5, 1), (3, 2) and (5, 2). We therefore can obtain the sequences for all AGVS and YCs by using the same method, which ensures that each container has one succeeding container and one preceding container delivered by the same AGV (or YC). In addition, the locations of each container can also be obtained from the chromosomes. For example, in Fig. 6, container (3, 1), (4, 1), (3, 2) are all assigned to a stack, and one possible allocation is that container (3, 1) locates in the first available level in stack 2, container (4, 1) locates in the second available level and container (3, 2) is on the top.

Let $|D|$ denote the total number of import containers to be unloaded from the ship to the yard, and v the total number of AGVs to be used and c the total number of YCs to be employed. The initial population is constructed by the following steps:

(1) Calculate the travelling times of containers from the working points to all available yard locations, which consists of the time that a AGV travels from the working point to the transfer point, and the time that a YC travels from the transfer point to yard location. The yard locations with shortest travelling times will be selected to hold all containers. As a result, the number of chosen yard locations equals to the number of import containers. Constraints (5) and (6) are satisfied.

(2) According to the selected yard locations in (1), we decide a new set of stacks (denoted by P_s, labelled from 1 to $|P_s|$) whose slots have been selected in (1). We randomly assign these stacks to each container. It is noted that a stack can

QC1			
Container	Dispatched AGV	Assigned YC	Assigned stack
(1, 1)	1	1	1
(2, 1)	2	2	3
(3, 1)	2	2	2
(4, 1)	3	3	2
(5, 1)	1	1	5
QC2			
Container	Dispatched AGV	Assigned YC	Assigned stack
(1, 2)	2	3	3
(2, 2)	3	1	1
(3, 2)	1	2	2
(4, 2)	3	1	4
(5, 2)	1	3	4

Fig. 6. Chromosome representation example for 10 containers handled by two QCs.

be assigned to more than a container since maybe there are several available locations within a stack. A possible sequence of storing containers within one stack can be obtained, based on the method in (1) and (2). Thus constraints (7) and (8) are satisfied.

(3) Randomly choose an AGV from 1 to v (constraints (3) and (4)), i.e. assign a AGV to deliver a container (column 2 of chromosome in Fig. 6) so that constraints (1) and (2) are satisfied.

(4) Randomly choose a YC from 1 to c (constraints (11) and (12)), i.e. assign a YC to handle a container (column 3 of chromosome in Fig. 6) so that constraints (9) and (10) are satisfied.

(5) Chromosomes can be generated, respectively by steps 1–4 until the population size *Pop* reaches a given number to ensure the initial search space is large enough to start with.

(6) Evaluate each matrix Ψ in the initial population by calculating the values of $u_{(i,k)}$ and $d_{(i,k)}$ according to constraints (13)–(18). The objective function value can be obtained by the minimisation of $\max_k(u_{(N_k,k)} + h_{(N_k,k)})$, where N_k the last container is handled by QC k.

We here give a simple example to explain the calculations in (6). It is assumed container (1, 1) and (2, 1) are two consecutive containers handled by QC1, AGV1 and YC1, respectively. The time QC1 starts to handle container (1, 1) is 0 ($u_{(1,1)} = 0$) because no container is handled before the first container (1, 1). $d_{(1,1)}$, which represents the time when YC1 picks it up, equals to the time that container (1, 1) is handled by QC1 ($u_{(1,1)} + h_{(1,1)}$) plus the AGV travelling time from the working point in the quay side to the transfer point in the yard. $u_{(2,1)}$ represents the time when QC1 is able to pick up container (2, 1), which means QC1 has to finish the unloading process on its predecessor container (container (1, 1)), and at the same time, AGV1 is ready at the working point to work on container (2,1) after delivering container (1, 1) to the transfer point in the yard. Therefore, $u_{(2,1)}$ equals to the maximum value of the finishing time of QC1 on container (1, 1) and the time that AGV1 returns to the working point after delivering container (1, 1) to the yard. Similarly, $d_{(2,1)}$, which represents YC1 picks up container (2, 1) from AGV1, equals to the maximum value of the time that container (2, 1) arrives at the transfer point by AGV1, and the time that YC1 returns to the transfer point after handling container (1, 1). Therefore, all values of $u_{(i,k)}$ and $d_{(i,k)}$ can be calculated.

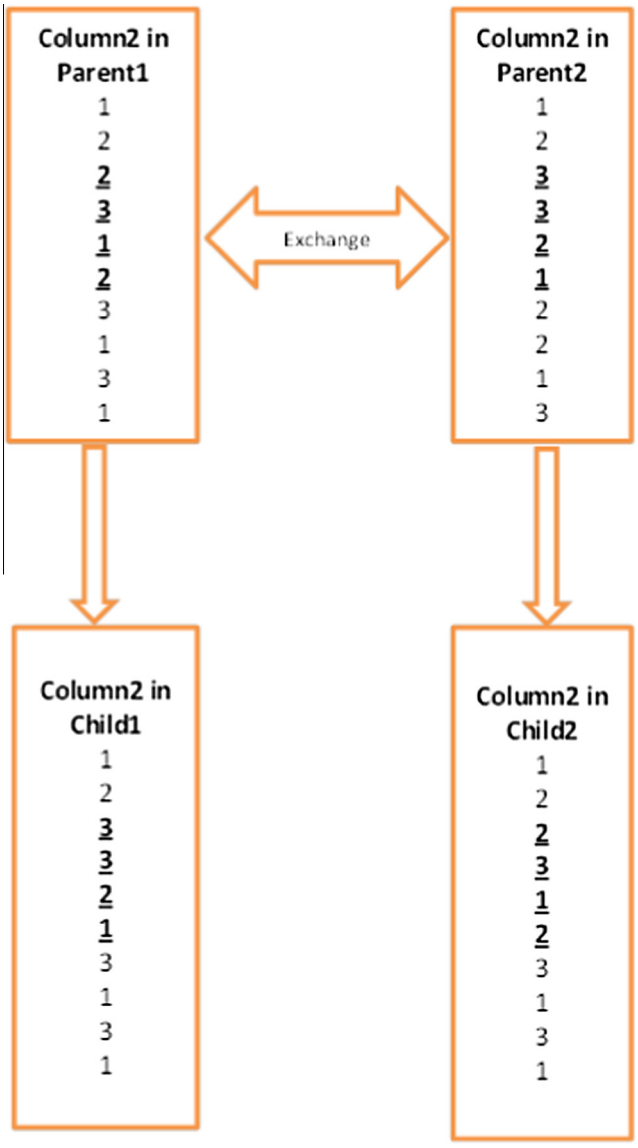

Fig. 7. An illustration of two-point crossover for an example of 10 containers.

4.2. Genetic operators design

In order to efficiently explore the solution space, and at the same time, maintain the feasibility of the newly generated offspring, the following crossover and mutation operations are proposed.

(1) *Two-point crossover:* the two-point crossover approach is proven to encourage the exploration of the global search space, rather than causing early convergence in the search (e.g. as a local optimal solution), thus making the search of optimal solutions more efficient (Spears & De Jong, 1991). This approach is proposed for the second and third columns of the chromosome matrix Ψ, because each container can be handled by any AGV and any YC. The new offspring is produced by randomly choosing two points along the length of chromosomes of parents and then exchanging the genes between the two points (as highlighted in Fig. 7). The cross-

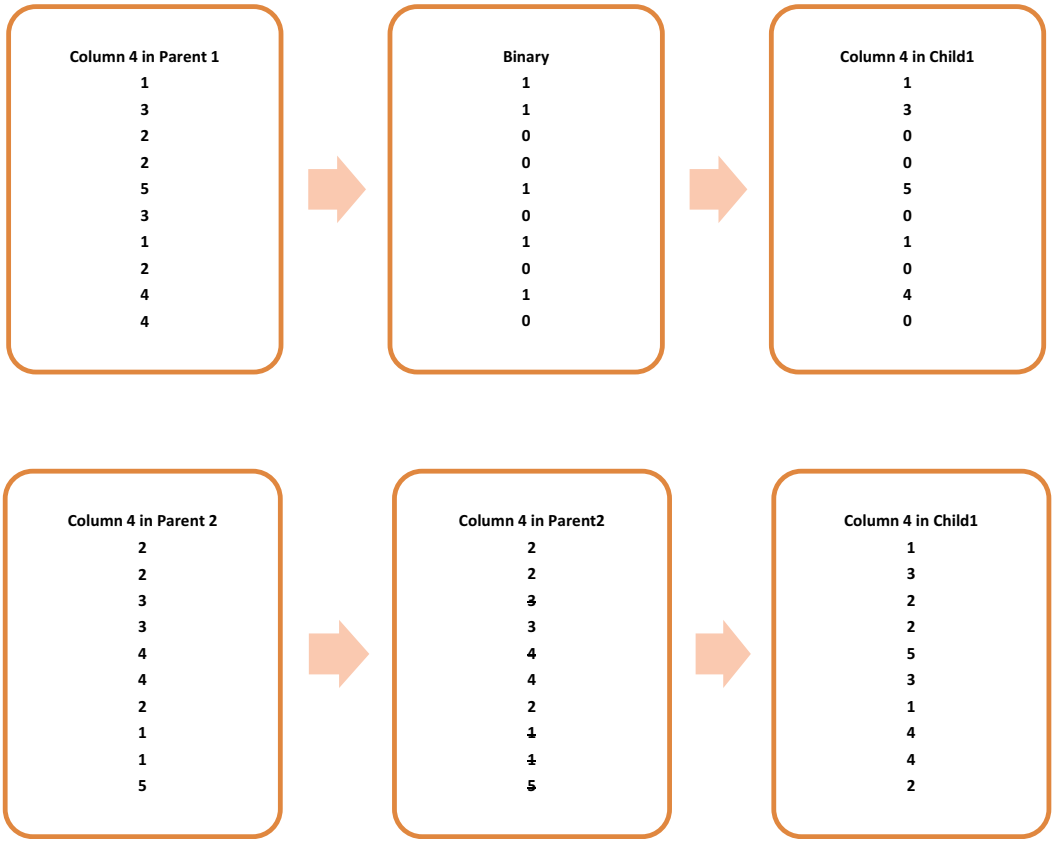

Fig. 8. An illustration of uniform order-based crossover for an example of 10 locations in 5 stacks.

over operation can be applied to the third column of both patents in the same way. The two-point crossover guarantees that the generated children will remain feasible if parents are feasible.

(2) *Uniform order-based crossover:* The two-point crossover operator has been adopted for the second and third columns of the chromosome matrix Ψ; however, such an approach cannot apply to the fourth column of the chromosome due to constraints (5), which requires one yard location is exactly assigned to only one container, otherwise redundant and missing genes in the generated children will appear. In this paper, we construct the uniform order-based crossover operator for the fourth column of the chromosome matrix Ψ in a similar way to the algorithm proposed in Cheng and Gen (1997). Fig. 8 shows how the uniform order-based crossover operator is used to create the fourth column of a child. There are 10 containers to be assigned to 5 selected stacks, numbered from 1 to 5, where there are 2 available locations in stack 1, 3 available locations in stack 2, 2 available locations in stack 3, 2 available locations in stack 4 and 1 available location in stack 5. This crossover operator generates a template binary string (the middle top one in Fig. 8) with the same length as the fourth column of the chromosome (10 rows in this example because there are 10 containers) based on the uniformly distributed "1"s and "0"s. The template string is then mapped to one of the selected parents, in which the genes that have the same positions with "1"s in the template string i.e. position 1, 2, 5, 7, 9, are given to a child, and the remaining empty genes of this child are filled from another parent with unused genes, i.e. position 3, 4, 6, 8, 10.

(3) *Swap mutation*: In addition to the crossover operators, mutation operation is adopted to maintain the diversity of the population in the successive generations and to maximise the exploitation of the solution space. To achieve the mutation operation, we specify a mutation probability P_m. As mutation happens rarely, P_m is set to be a very small number, for example, 0.1 suggested by Yang, Wang, and Li (2012). For each chromosome Ψ in the population, we generate a random value uniformly distributed between 0 and 1, and compare this value with P_m. If the value is less than P_m, we perform the swap mutation on that individual, otherwise, there is no mutation operation, i.e. the individual remains the same. This mutation operation is carried out by choosing two positions (two rows) of that individual at random and then swapping the genes on these positions (see Fig. 9).

4.3. Offspring acceptance strategy

To achieve evolution, chromosomes are randomly paired to produce offspring for the next generation. We propose a semi-greedy strategy to accept offspring, which was first introduced by Hart and Shogan (1987). In this strategy, an offspring is accepted as a new generation only if its fitness is better than the average fitness of its own parents, which ensures the next generation carries better genes than their parents. It can also reduce the computation time and guarantee a monotonous convergence, which means the best objective function value (OFV) in any generation is no worse than that of previous generations and it will evolves towards an optimal solution.

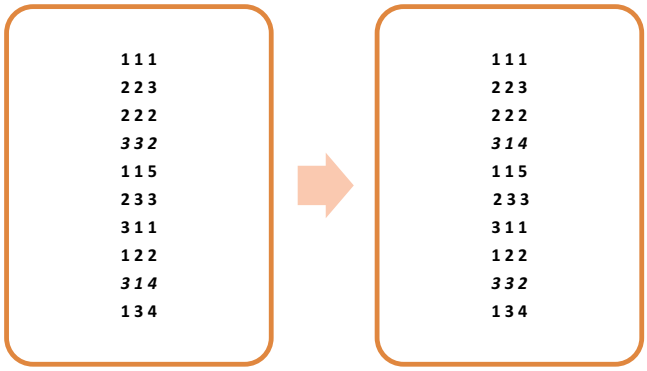

Fig. 9. An illustration of swap mutation for an example of 10 containers.

4.4. Parent selection strategy

Effective parent selection strategy (e.g. how to choose parents from chromosomes in the current population) is vital to improve evolution speed. In general, it is better to choose the best solutions in the current generation to create offspring so that the average OFV of the offspring is better than the average OFV of their parents.

In the context of the AGV scheduling and storage allocation problem, the chromosomes with short berth times (i.e. OFVs) are likely to be selected as parents for creating offspring. The most common method is called the roulette wheel sampling, in which each chromosome is assigned a slice of a circular roulette wheel and the size of the slice is proportional to the chromosome's fitness. The fitness in this paper is represented by 1/OFV, which means the chromosome with shorter berth time (better fitness) will be assigned a bigger slice and likely to be chosen as parents. The times that the wheel spins equal to the population-size. On each spin, the chromosome under the wheel's marker is selected to be in the pool of parents for the next generation. Here, as the objective is to minimise the berth time, we choose the ones with smaller OFVs as parents for the next generation.

This parent selection strategy would always select a population of parents with smaller OFVs. To ensure that the best solution in the current generation always survives to the next generation, we use elitism strategy, which means the best individual is always kept in the population. Such a parent selection strategy will accelerate the entire evaluation procedure and search fast for an approximately optimal solution.

4.5. Stopping criterion

In order to balance the computational time as well as evolving towards an optimal solution, two criteria are used as stopping rules: (1) the maximum number of evolving generations allowed for GA, which is a common criterion adopted by many GA-based optimisation problems (Bazzazi et al., 2009; Huang, Liang, & Yang, 2009; Kozan & Preston, 1999) and (2) the standard deviation of the fitness values of chromosomes (σ_T) in the current generation is below a small value (Tavakkoli-Moghaddam & Safaei, 2006). This parameter (σ_T) implies the diversity of the current generation in terms of the OFVs. The decreasing σ_T is equivalent to the decreasing diversity. If σ_T decreases below a small arbitrary constant ε, then the algorithm is stopped. The standard deviation of the fitness values of chromosomes in the generation T is calculated as $\sigma_T = \sqrt[2]{\left[\left(\frac{1}{Pop}\right)\sum_{n=1}^{Pop}\left(F_T^n - \overline{F_T}\right)^2\right]}$, where F_T^n is the fitness of the n*th* chromosome in the generation T, and $\overline{F_T}$ is the average fitness of all chromosomes in generation T, which can be calculated as $\overline{F_T} = \left(\frac{1}{Pop}\right)\sum_{n=1}^{Pop} F_T^n$. The algorithm stops when one of the two rules is satisfied.

5. Computational results

In this section, we first introduce the initial settings. The small-sized problems are solved by a commercial software, AIMMS 3.11, which uses branch and bound (B&B) algorithm in its solver (CPLEX 11.2). As the problem size increases (i.e. the number of containers and the number of equipment increase), it is difficult to obtain optimal solutions. Therefore, we adopt the GA proposed in Section 4 to obtain near optimal solutions for large-sized problems. We also provide the comparison results between the GA and AIMMS for small-sized problems to verify the effectiveness of the GA.

5.1. Initial settings

We consider the following experimental and parameter settings:

(i) All experiments are based on the layout as illustrated in Fig. 3.
(ii) The number of containers varies from 5 to 200, where 5–20 are considered as small-sized problems and 21–200 are considered as large-sized problems, which is based on the classification on Lee, Chew, Tan, and Wang (2010). We also consider different number of AGVs from 3 to 10; the number of YCs varies from 2 to 5; and the number of blocks varies from 2 to 8.
(iii) The uniform distribution was assumed for all the travelling times (Lau & Zhao, 2008), because containers are evenly distributed and located on the ship and in the yard. The travelling times (in seconds) of YCs from the transfer point in front of each block to each available location are generated from a uniform distribution $U(60, 140)$ s; the handling times of QCs follows a uniform distribution $U(30, 180)$ s.
(iv) AGV's travelling times from the working points near QCs to the transfer points at the front of blocks are known according to the terminal's layout. QCs do not move along the ship because it takes a very long time to move, which is very unproductive and should be avoided. The above values of the AGV's travelling times are generated from uniform distribution $U(20, 120)$ s; YC's travelling times between blocks are also known. Here we assume the travelling times between transfer points of any two adjacent blocks are 40 s, so the travelling times between any two blocks can be calculated similarly.

The following GA parameters are set up: crossover rate $P_c = 0.8$, mutation rate $P_m = 0.02$, Population size $Pop = 100$, and Maximum generation $M_g = 50$.

Our proposed GA is implemented using MATLAB (version 7.11). All experiments are run on a machine with Intel® Core™ i3 CPU M370@2.40 GHz and 4 GB RAM with the Windows 7 operating system. Results obtained from the GA for small-sized problems are compared with numerical optimal solutions obtained from B&B in terms of OFV and the computation time. With the settings above, we present the following evaluation results based on our proposed approaches.

5.2. Results for small-sized problems

Ten small-sized experiments are considered with the number of containers varies from 5 to 20. To reduce possible bias generated

Table 1
Results of computational experiments in small sizes.

No.	Containers	AGVs/QCs/YCs	B&B (MIP)		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
1	5	2/2/2	12.22	386	4.12	386	0
2	6	2/2/2	13.54	406	2.84	406	0
3	7	2/2/2	8.67	426	3.15	426	0
4	8	3/2/3	14.58	560	1.24	563	0.53
5	9	3/2/3	10.09	792	2.22	798	0.76
6	10	2/2/3	534.65	776	1.35	788	1.55
7	10	2/2/2	489.31	813	1.26	833	2.4
8	15	3/2/2	13045.26	976	4.11	1009	3.38
9	20	3/2/3	/	/	10.06	1208	/
10	20	4/2/3	/	/	17.51	873	/

Table 2
Results of large –sized problems.

No.	Containers	AGVs/QCs/YCs	Computation time (s)	OFV(s)
11	30	4/3/2	24.05	2379
12	30	4/3/3	17.62	1686
13	30	4/3/4	49.69	1483
14	40	3/3/3	49.57	2594
15	40	4/3/3	51.02	2443
16	40	5/3/3	40.01	2035
17	50	4/3/4	56.81	2903
18	50	4/2/4	90.38	3129
19	50	5/2/4	68.97	2796
20	80	6/3/3	137.29	5396
21	80	6/3/4	53.61	4463
22	80	7/3/5	106.57	3696
23	100	5/3/4	136.54	6828
24	100	6/3/4	282.67	6249
25	100	7/3/4	267.97	5996
26	100	7/3/5	421.01	5038
27	150	8/3/3	639.02	11344
28	150	8/3/4	282.27	9324
29	150	8/3/5	424.25	8363
30	200	8/3/5	503.01	11654
31	200	9/3/5	406.44	11490
32	200	10/3/5	761.10	11319

by the randomness of GA in a single experiment, each case is run 20 times by GA and the average results are recorded. For all the runs, we adopt the same setting of parameters to obtain more reliable results, and average values of OFVs (in seconds) and computation times (in seconds) are computed as the final results.

Table 1 shows that for the small-sized problems, our proposed GA can obtain approximately optimal solutions, compared with the optimal solution from the B&B, but in a faster speed, ranging from 1.24 to 17.51 s, while the B&B from 12.12 to 13045.26 s. The solutions from the GA are near to the optimal solutions provided by the B&B with the maximum gap of 3.38% in the 10 cases. The GA can obtain the optimal solution for the first three cases and the average gap is 1.72% for the first eight cases. However, we observe that the exact algorithm (B&B) cannot solve larger-sized problems within acceptable time duration. The B&B do not provide any results for the problems with more than 15 containers, and the computation time of B&B grows exponentially as the problem size increases.

5.3. Results for large-sized problems

Due to the difficulty in obtaining the exact solution for large instances, GA is used to solve them, and the results are indicated in Table 2. From our experiments, we observe that (1) our proposed GA performs stably to provide near optimal solutions for large-sized problems, for example, the number of containers reaches

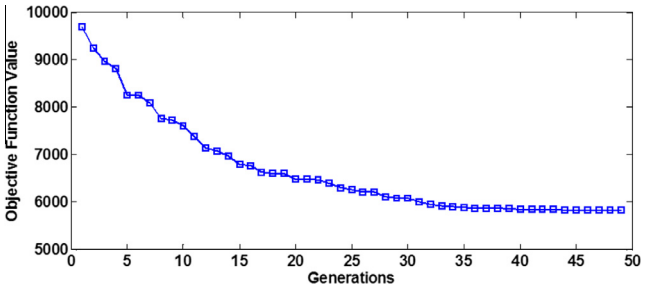

Fig. 10. Typical convergence of GA for the case with 100 containers, five AGVs, three QCs and four YCs.

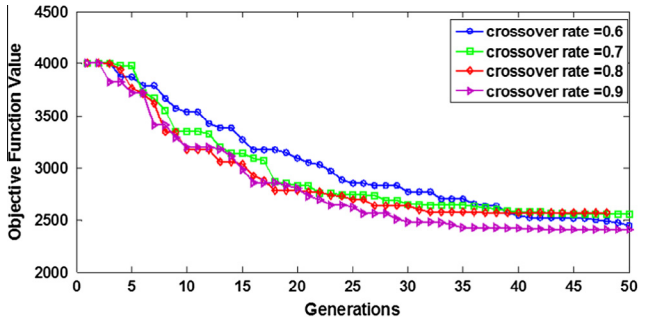

Fig. 11. Performance comparison of different crossover rates for an example under $Pop = 100$ and $P_m = 0.02$.

200; for all the cases in Table 2, the computational times are within minutes; (2) The OFVs increase with the problem size (number of containers) as expected, which means it takes more time to unload more containers; and (3) the trend of performances of the number of AGVs/QCs/YCs is similar: when increasing the AGV/QC/YC numbers, the OFV reduces (see case 15 and case 16 in Table 2). The effects of the number of cranes (QCs and YCs) are more significant than the effect of the number of AGVs on the OFVs (see case 18 and case 19 for the effects of QCs numbers and case 11, 12 and 13 for the effects of YCs numbers).

However, in practice, it is important to choose the appropriate number of vehicles and cranes; otherwise, it will cause the traffic congestions or conflicts during the container handling process, which influences the container terminal's efficiency. In the perspective of computational time, it will take longer time to get the solution with the increased number of equipment.

Fig. 10 shows the convergence of the GA for a case of 100 containers, 5 AGVs, 3 QCs and 4 YCs. It shows that GA converges steadily and fast to a fixed value, which demonstrates that our proposed GA is able to provide the good quality solutions. Short

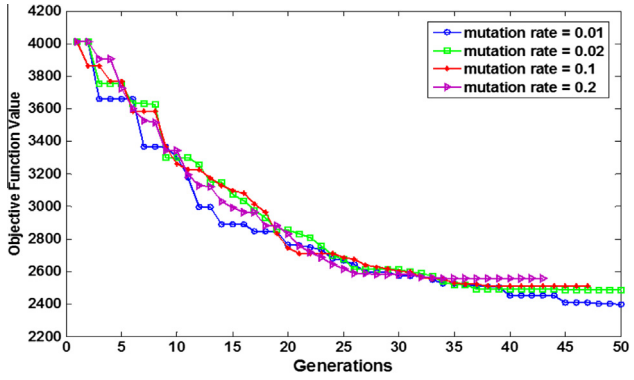

Fig. 12. Performance comparison of different mutation rates for an example under $Pop = 100$ and $P_c = 0.9$.

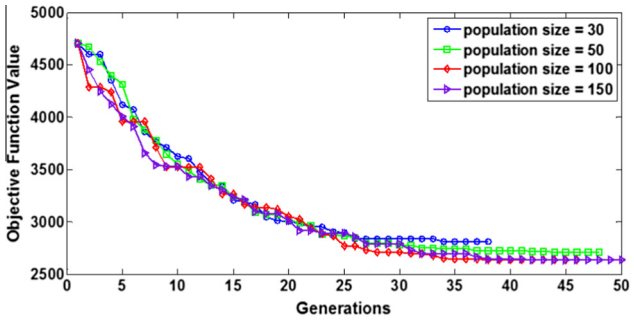

Fig. 13. Performance comparison of different population size for an example under $P_c = 0.9$ and $P_m = 0.01$.

computational time is important for real-world operations at the container terminals, because when the ship arrives at the terminal, the handling operations of containers should start as soon as possible.

5.4. Experiments on GA parameters

Now we look at the experiments on evaluating the impact of GA parameters, which would show the performance of the proposed GA with different initial parameters, and provide a better parameter setting for the GA. A combination of several different values is taken in order to find the values with best convergence performance curve (i.e. faster convergence, and better OFV).

For the problem with 50 containers, 5 AGVs, 3 QCs and 5 YCs, GA parameter settings take the following values:

Crossover rate $P_c = \{0.6, 0.7, 0.8, 0.9\}$;
Mutation rate $P_m = \{0.01, 0.02, 0.1, 0.2\}$;
Population size $Pop = \{30, 50, 100, 150\}$;
$M_g = 50$.

Figs. 11–13 show the performance comparisons for different parameter settings. According to the convergence curves, we observe that the setting with crossover rate P_c = 0.9, mutation rate P_m = 0.01 and population size Pop = 100 outperforms others for this particular problem. Specifically, in Fig. 11, the curve with $P_c = 0.9$ converges to a smaller value of OFV; in Fig. 12, the curve with $P_m = 0.01$ also converges to a smaller value of OFV, and in Fig. 13, the curve with $Pop = 100$ and $Pop = 150$ converge to the same value of OFV; however, the one with $Pop = 100$ stops earlier, which means a shorter evolving time. These figures also show that for all the experiments, the OFVs are not improved after 50 generations. So the maximum generations of 50 will be sufficient to obtain near-optimal solutions.

In order to test the stability of the proposed GA and analyse the possibility of random effect, the GA is run for 10 times on the case of 60 containers, 5 AGVs, 3 QCs and 4 YCs with exactly the same parameter settings. Fig. 14 shows the box plot of results. Box plot is able to show the range of OFVs in each generation. Each box represents the OFVs in one generation. The central mark is the median (50% percentile), the bottom and top of the box are the 25th and

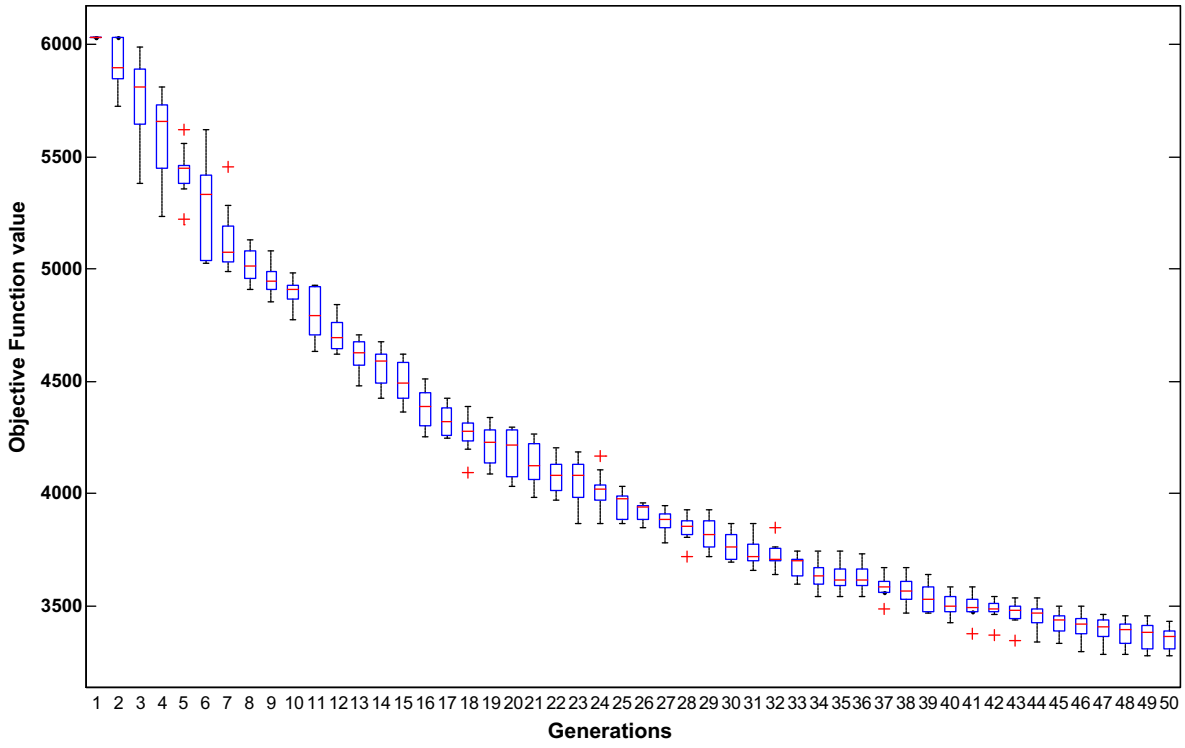

Fig. 14. GA performance in 10 runs with $P_c = 0.9$, $P_m = 0.01$ and $Pop = 100$ for the case with 60 containers, five AGVs, three QCs and four YCs.

75th percentiles (the lower and upper quartiles, respectively) and the whiskers are the most extreme data points (1.5 times more than upper quartile or 1.5 times less than lower quartile). The ends of the box plot in each generation are the maximum and minimum values excluding extreme values. Fig. 14 further demonstrates that our proposed GA performs in a stable manner.

6. Conclusions

In this paper, we provide a novel idea for improving the efficiency of an automated container terminal by integrating the vehicle scheduling and container storage problems. The objective is to minimise the unloading time from the ship and therefore to increase the productivity of the container terminal. A MIP model is formulated, which is solved optimally by commercial software in small sizes. However, it is difficult to obtain optimal solutions when the problem size increases, and the GA is developed for the large-sized problems.

A number of experiments are conducted to assess the efficiency of the integrated modelling approach and the solution quality of the proposed GA. The integrated solutions (i.e. how to dispatch AGVs/YCs and assign locations for containers), can be obtained simultaneously by solving the model. The computational results also demonstrate that the proposed GA is able to provide good solutions for all cases examined in this paper. Comparing the results from the exact algorithm (B&B), the GA can provide good solutions in a shorter time; and the gaps in terms of OFVs are very small. Therefore, the proposed methods in this paper have the potential to handle the problems in practical container terminals.

From theoretic point of view, this work provides a modelling technique for a broader integration (comparing with previous studies in the literature) with considering vehicle scheduling (AGVs), yard crane scheduling (YCs) and container storage allocation problems all together. Since all these problems are correlated with each other in practice, it is important to consider them simultaneously. The decisions obtained from the model provide the schedules of vehicles, yard cranes and container storage locations.

This study can be extended in the following ways for future research. For example, container loading process could be considered simultaneously with the unloading process, i.e. dual-cycle operations, which have currently been adopted by a few advanced container terminals. Apart from achieving the minimum berth times, other objectives, especially environmental related ones, can be included as well since environmental concerns are becoming more and more critical for container terminals. In addition, developing other exact algorithms and heuristic approaches, such as tabu search (TS) and simulated annealing (SA), for the MIP model developed in this paper is also an interesting area to look at.

References

Angeloudis, P., & Bell, M. G. H. (2010). An uncertainty-aware AGV assignment algorithm for automated container terminals. *Transportation Research Part E: Logistics and Transportation Review, 46*(3), 354–366.

Bazzazi, M., Safaei, N., & Javadian, N. (2009). A genetic algorithm to solve the storage space allocation problem in a container terminal. *Computers & Industrial Engineering, 56*(1), 44–52. http://dx.doi.org/10.1016/j.cie.2008.03.012.

Briskorn, D., Drexl, A., & Hartmann, S. (2007). Inventory-based dispatching of automated guided vehicles on container terminals. In K. Kim & H. O. Guenther (Eds.), *Container terminals and cargo systems* (pp. 195–214). Berlin Heidelberg: Springer.

Bruzzone, A., & Signorile, R. (1998). Simulation and genetic algorithms for ship planning and shipyard layout. *Simulation, 71*(2), 74–83.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014a). Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research, 235*(2), 412–430.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014b). Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research, 236*, 1–13.

Chen, Y., Leong, Y., Ng, J., Demir, E., Nelson, B., & Simchi-Levi, D. (1998). Dispatching automated guided vehicles in a mega container terminal. *Paper presented at INFORMS, May 1998*. Montreal.

Chen, P., Fu, Z., Lim, A., & Rodrigues, B. (2003). The general yard allocation problem. In *Genetic and evolutionary computation-GECCO 2003. Lecture Notes in Computer Science* (Vol. 2724). Berlin Heidelberg: Springer (pp. 1986–1997).

Cheng, R., & Gen, M. (1997). Parallel machine scheduling problems using memetic algorithms. *Computers & Industrial Engineering, 33*(3–4), 761–764.

Chen, L., & Lu, Z. (2012). The storage location assignment problem for outbound containers in a maritime terminal. *International Journal of Production Economics, 135*(1), 73–80. http://dx.doi.org/10.1016/j.ijpe.2010.09.019.

Durrant-Whyte, H. F. (1996). An autonomous guided vehicle for cargo handling applications. *International Journal of Robotics Research, 15*(5), 407–440.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Grunow, M., Günther, H. O., & Lehmann, M. (2004). Dispatching multi-load AGVs in highly automated seaport container terminals. *OR Spectrum, 26*(2), 211–235.

Han, Y., Lee, L. H., Chew, E. P., & Tan, K. C. (2008). A yard storage strategy for minimizing traffic congestion in a marine container transshipment hub. *OR Spectrum, 30*(4), 697–720.

Hart, J. P., & Shogan, A. W. (1987). Semi-greedy heuristics: An empirical study. *Operations Research Letters, 6*(3), 107–114.

Holland, J. H. (1975). *Adaptation in natural and artificial systems* (Vol. 53). Ann Arbor, MI: University of Michigan press.

Huang, Y., Liang, C., & Yang, Y. (2009). The optimum route problem by genetic algorithm for loading/unloading of yard crane. *Computers & Industrial Engineering, 56*(3), 993–1001.

Jiang, X., Lee, L. H., Chew, E. P., Han, Y., & Tan, K. C. (2012). A container yard storage strategy for improving land utilization and operation efficiency in a transshipment hub port. *European Journal of Operational Research, 221*(1), 64–73.

Kim, K. H., & Bae, J. W. (1999). A dispatching method for automated guided vehicles to minimize delays of containership operations. *Management Science and Financial Engineering, 5*(1), 1–25.

Kim, K. H., & Bae, J. W. (2004). A look-ahead dispatching method for automated guided vehicles in automated port container terminals. *Transportation Science, 38*(2), 224–234.

Kim, J., Choe, R., & Ryu, K. R. (2013). Multi-objective optimization of dispatching strategies for situation-adaptive AGV operation in an automated container terminal. In *Proceedings of the 2013 research in adaptive and convergent systems* (pp. 1–6). ACM.

Kim, K. H., Park, Y. M., & Ryu, K.-R. (2000). Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research, 124*(1), 89–101.

Klerides, E., & Hadjiconstantinou, E. (2011). Modelling and solution approaches to the multi-load AGV dispatching problem in container terminals. *Maritime Economics & Logistics, 13*(4), 371–386.

Kozan, E., & Preston, P. (1999). Genetic algorithms to schedule container transfers at multimodal terminals. *International Transactions in Operational Research, 6*(3), 311–329. http://dx.doi.org/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.

Lee, D. H., Cao, J. X., & Shi, Q. (2008). Integrated model for truck scheduling and storage allocation problem at container terminals. In *Transportation research board 87th annual meeting*. Washington, D.C, USA: IEEE.

Lee, D. H., Cao, J. X., Shi, Q., & Chen, J. H. (2009). A heuristic algorithm for yard truck scheduling and storage allocation problems. *Transportation Research Part E: Logistics and Transportation Review, 45*(5), 810–820.

Lee, L. H., Chew, E. P., Tan, K. C., & Wang, Y. (2010). Vehicle dispatching algorithms for container transshipment hubs. *OR Spectrum, 32*(3), 663–685.

Linn, R., Liu, J., Wan, Y., Zhang, C., & Murty, K. G. (2003). Rubber tired gantry crane deployment for container yard operation. *Computers & Industrial Engineering, 45*(3), 429–442.

Meersmans, P. J. M., & Wagelmans, A. P. M. (2001). *Effective algorithms for integrated scheduling of handling equipment at automated container terminals*. Erasmus Research Institute of Management, Erasmus Universiteit.

Ndiaye, N. F., Yassine, A., & Diarrassouba, I. (2014). A branch-and-cut algorithm to solve the container storage problem. In *ICONS 2014, The ninth international conference on systems* (pp. 226–233).

Spears, W. M., & De Jong, K. A. (1991). An analysis of multi-point crossover. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 301–315). San Mateo, CA: Morgan Kaufman.

Sriphrabu, P., Sethanan, K., & Theerakulpisut, S. (2014). A genetic algorithm for minimizing relocations at container yard in container terminal. *Advanced materials research* (Vol. 931, pp. 1683–1688). Trans Tech Publ.

Stahlbock, R., & Voß, S. (2008). Operations research at container terminals: A literature update. *OR Spectrum, 30*(1), 1–52.

Steenken, D., Voß, S., & Stahlbock, R. (2004). Container terminal operation and operations research – A classification and literature review. *OR Spectrum, 26*(1), 3–49.

Tavakkoli-Moghaddam, R., & Safaei, N. (2006). An evolutionary algorithm for a single-item resource-constrained aggregate production planning problem. In *IEEE conference on evolutionary computation (CEC)* (pp. 2851–2858). Vancouver, B.C., Canada: IEEE.

UNCTAD (2014). United nations conference on trade and development. Review of maritime transport. <http://unctad.org/en/PublicationsLibrary/rmt2014_en.pdf> Accessed 20.11.14.

Vis, I. F. (2006). Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research, 170*(3), 677–709.

Woo, Y. J., & Kim, K. H. (2011). Estimating the space requirement for outbound container inventories in port container terminals. *International Journal of Production Economics, 133*(1), 293–301. http://dx.doi.org/10.1016/j.ijpe.2010.04.032.

Xin, J., Negenborn, R. R., & Lodewijks, G. (2014). Energy-aware control for automated container terminals using integrated flow shop scheduling and optimal control. *Transportation Research Part C: Emerging Technologies, 44*, 214–230.

Yang, C., Wang, X., & Li, Z. (2012). An optimization approach for coupling problem of berth allocation and quay crane assignment in container terminal. *Computers & Industrial Engineering, 63*(1), 243–253.

Zhang, C., Liu, J., Wan, Y., Murty, K. G., & Linn, R. J. (2003). Storage space allocation in container terminals. *Transportation Research Part B: Methodological, 37*(10), 883–903.