VEHICLE DISPATCHING PROBLEM AT THE CONTAINER TERMINAL WITH TANDEM LIFT QUAY CRANES

A Dissertation

by

YAO XING

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2013

Major Subject: Civil Engineering

ABSTRACT

The most important issue at a container terminal is to minimize the ship's turnaround time which is determined by the productivities of quay cranes (QCs). The tandem lift quay cranes have 33% higher productivities than single lift QCs. However, the tandem lift operations bring new challenges to the vehicle dispatching at terminals and this has become a big issue in the application of tandem lift QCs. The vehicle dispatching at terminals is to enhance the QCs' productivities by coordinating the QCs' operation schedules and the vehicles' delivery schedules.

The static version of the problem can be formulated as an MILP model and it is a combinational optimization problem. When the type of QC is tandem lift, the problem becomes more complicated because it requires two vehicles side by side under the QC. Thus, the alignments of vehicles have to be considered by coordinating the delivery schedules between vehicles. On the other hand, because the containers are operated alone by the yard cranes, the vehicles could not be grouped and dispatched in pairs all the time.

This dissertation investigates the static and dynamic version of the problem and proposes heuristic methods to solve them. For the static version, Local Sequence Cut (LSC) Algorithm is proposed to tighten the search space by eliminating those feasible but undesirable delivery sequences. The time windows within which the containers should be delivered are estimated through solving sub-problems iteratively. Numerical experiments show the capability of the LSC algorithm to find competitive solutions in substantially reduced CPU time.

To deal with the dynamic and stochastic working environment at the terminal, the dissertation proposes an on-line dispatching rule to make real-time dispatching decisions without any information of future events. Compared with the longest idle vehicle rule, the proposed priority rule shortens the makespan by 18% and increases the QCs' average productivities by 15%. The sensitivity analysis stated that the superiority of the priority rule is more evident when the availability of vehicles is not sufficient compared with the frequency of releasing transportation requests.

ACKNOWLEDGEMENTS

I would like to specially thank my advisor and committee chair, Dr. Luca Quadrifoglio for his patient guidance and mentorship during the past five years. I would also like to thank my committee members, Dr. Zhang, Dr. Wang, and Dr. Wilhelm, for their guidance and suggestions that help me finish this dissertation.

Thanks to UTCM who provided financial support for my PhD study and research in Texas A&M University. I also appreciate Mr. Shi in Nansha Container Terminal, who answered me questions about terminal operations and provided me precious operation data for my research. I would also like to thank Dr. Alan Zhang in Moffatt and Nichol Engineers for his help and guidance for my research.

Thanks also go to all my friends and classmates at Texas A&M University, especially those in the division of transportation engineering, for their friendships and help.

Finally, I especially thank my deeply loved parents and my husband for their love, patience and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

1. INTRODUCTION

The first regular sea container service began in 1961 between and ports in the Caribbean, Central and South America. The year 2010 witnessed an increase of global containerized trade from around 30 million TEUs (twenty feet equivalent units) in 1990 to 150 million TEUs or more in 2010 (*76*). Not surprisingly, such dramatic increase imposes the heavy pressure on the operations at container terminals.

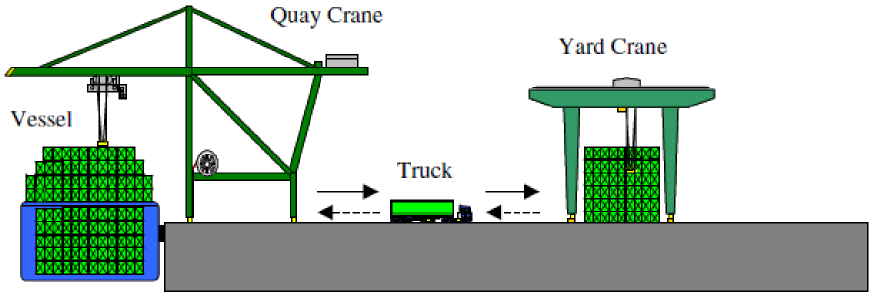


FIGURE 1 Container flow in the loading and unloading operations (*61*).

For a terminal, the most critical issue is to minimize the ship's turnaround time by accelerating operations and smoothing the container flow (FIGURE 1). There are two types of operations – loading and unloading – at a terminal. Upon a ship arrives at the terminal, it is berthed at the quay side according to the berth allocation decision and a certain number of quay cranes (QCs), dependent on the size of the ship and the number of containers, are assigned to load containers onto the ship and unload them off the dock and hold. When a QC is unloading a container from the ship, it hoists the container and lifts it from the ship then positions it on a vehicle for delivery. A fleet of vehicles with unit capacities, namely trucks, automated guide vehicles (AGVs), and manned or automatic saddle carriers (ASCs), transports containers from the quay side to the storage area.

The storage area (yard area) is an open zone used to store the import and export containers temporarily. It consists of myriad blocks, further differentiated into lanes, rows (bays) and stacks (tiers), dependent on the span and height of the yard crane (FIGURE 2). In each block, there are always one to two yard cranes (YCs) that store and retrieve containers to and from the blocks. When the vehicle delivers the container to the storage area, the yard crane stacks it on its destination storage position according to the pre-defined storage planning. The loading operation is conducted in the reverse order.

Apparently, the ship's turnaround time is directly determined by the working speeds of QCs. In order to further accelerate the operations, terminals in the Netherlands, Shanghai, Shenzhen, Dalian, Singapore, Dubai, and other areas either have already been equipped with or have considered investing in tandem lift QCs in recent years (*81, 83*). A tandem lift QC is designed to increase productivity by lifting two 40ft containers or four 20ft containers each time, while the conventional single trolley quay crane lifts only one 40ft or two 20ft containers each time (FIGURE 3). Tandem lift quay cranes are equipped with a specific twin spreader. This special design facilitates its adjustment to different container heights, as well as to side-to-side clearances of two containers when landing them onto adjacent vehicles. To make a full use of tandem lift QC's capacity in practice, two vehicles are required to arrive at a QC, not necessarily at the same time, However, they should get ready before a tandem lift, as these containers need to be lifted or released together by the QC (*54, 83*). Tandem lift QCs are designed to accelerate the unloading/loading operations to meet the demands of serving mega vessels in a short time. According to published data, the tandem lift QCs can provide 33% greater productivity than the traditional single trolley QCs (*64*).

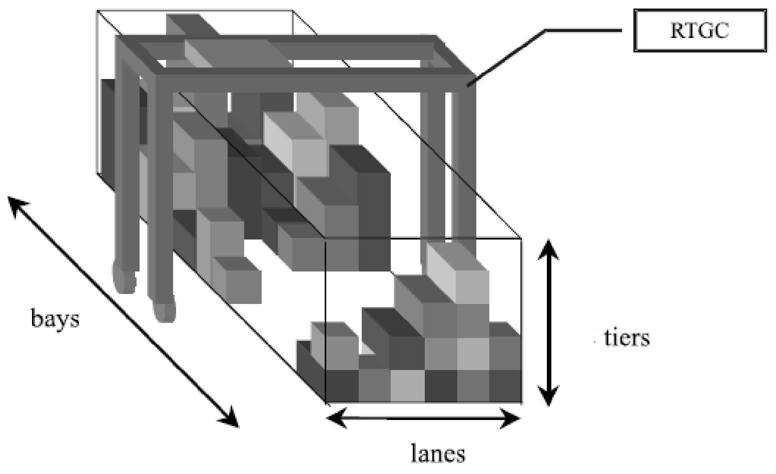
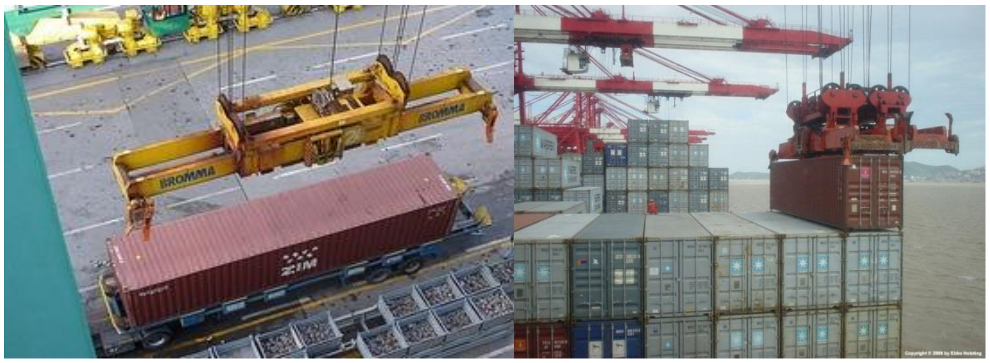


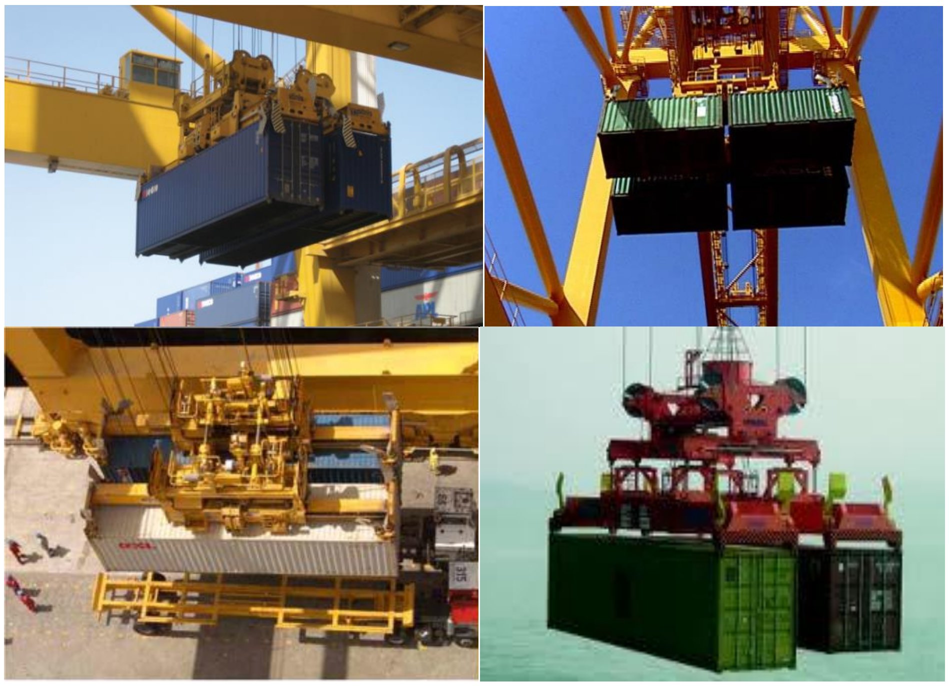
FIGURE 2 A storage block served by a yard crane (RTGC) (*86*).

However, tandem lift operation poses new challenges and difficulties for the vehicle dispatching problem at terminals. At the operational level, the vehicle dispatching problem concerns scheduling a fleet of vehicles to transport containers to enhance productivity of QCs and to smooth the container flows. Due to the lack of buffer area in most terminals, the quay crane unloads/loads a container directly on/from a vehicle. Thus, the QC cannot proceed to its next operation if there is no vehicle waiting under it. According to studies, the QC's actual productivity in operation only achieves around 40-60% of its technical maximum capacity and its waiting for a vehicle's availability is one reason for this (*51, 73*). It is well understood that the QCs' operation efficiency and the terminal throughput critically depend on the vehicle dispatching operation (*34, 28*), the main focus in this dissertation.

When the quay crane is a traditional single lift, the problem involves a combinational optimization that can be categorized in the Vehicle Routing Problem (VRP) (*7, 29*) or Parallel Machine Scheduling Problem. Conversely, when the quay cranes are tandem lifts, the problem becomes more complicated because of the alignment of vehicles dispatched to pick up or drop off containers at the QCs (*9*). To accommodate tandem lift operation, there are supposed to be two vehicles side by side waiting under a QC. Otherwise, the QC and the vehicle that arrives earlier must wait for other vehicle's arrival. Such idling not only delays the QC's operations However, also affects the productivity of vehicles. On the other hand, when the two vehicles leave the tandem lift QC and travel to the storage area, they separate and become independent again. Because the two containers operated in the same tandem lift may be stored in totally different storage blocks and the YC can only operate one 40ft or two 20ft containers each time. Thus, we could not simply bind two vehicles as a group and dispatch them together all the time.



(a) Conventional single-trolley quay crane



(b) Tandem lift QC

FIGURE 3 Different types of QC.

Actually, the transportation of containers between the QCs and YCs has become a big issue of the application of tandem lift QCs. Without an effective operation technique and efficient vehicle dispatching policy, the application of tandem lift QCs may not increase the productivity as expected. In turn, the lack of such a policy reduces the productivity of vehicles (*1, 64 and 82*). In such a case, in addition to the coordination between the QCs' operation schedules and the vehicles' delivery schedules, the delivery schedules of different vehicles have to be synchronized simultaneously.

Nonetheless, until now, there exists very little literature that investigates the special working pattern of tandem lift QCs and their influence on the vehicle dispatching problem at the container terminal. No researcher has yet solved the problem through employment of mathematical models or algorithms. Thus, the contribution of this study is as follows:

This study considers the working characteristics of tandem lift QCs in the modeling of and solution to the vehicle dispatching problem. In addition, both the model and the proposed method in this dissertation can be applied to the similar vehicle dispatching problems with precedence relationships.

For the static version of the problem, we formulated the problem as a mixed integer linear program with the objective of minimizing the makespan, the total time it takes to load and unload all containers. To solve such a combinatorial optimization problem, this dissertation proposes an innovative heuristic method to tighten the search space by eliminating those feasible but undesirable delivery sequences. This method is capable of finding competitive solutions within a significantly shorter CPU time.

For the dynamic version of the problem, this dissertation develops an on-line dispatching policy to make decisions without information of future events. The numerical experiments reveal that the proposed method performs better than the currently employed dispatching rule. Its superiority is more substantial when there are too few vehicle resources compared to transportation requests.

The rest of this dissertation is organized as follows: the second section reviews the related work and literature in the area of terminal operations and vehicle dispatching problems. The third section introduces the container terminal from which operation data for this dissertation was gathered. The third section is followed by the section of the problem statement and the mathematical model. The subsequent two sections present the algorithms for solving the static version of the problem and assess their performance through a series of numerical experiments. Then, the on-line dispatching rule is introduced and analyzed in the seventh section. The conclusion and discussion based on this study comprise the last section of this dissertation.

2. LITERATURE REVIEW

Vehicle dispatching problems frequently arise in logistic systems. Unfortunately, most published research does not apply to a container terminal due to the container terminal's specific operation characteristics. This, in turn, requires the development of algorithms that consider the special characteristics and constraints associated with the operations at terminals. In the following, we restrict literature review to the research related to the following topics: terminal logistics, transportation optimization in the terminal, and the optimization method of solving the vehicle dispatching problem. The review is not meant to be exhaustive, but rather deals with issues and problems similar to the one discussed in this dissertation.

2.1 Terminal Logistics

Operation research methods have become the main approaches to the optimization of operational issues in container terminals. Terminal logistics can be generally classified into the following categories.

Berth allocation: Berth allocation ideally begins an average of two to three weeks before the ship's arrival. Its main objectives are to determine the berthing times and positions of ships along the berth (*30, 41, 57 and 63*), and to minimize the mooring and process time of the shi (*21, 31 49, and 42*).

Crane assignment and split: Depending on the size of the ship, three to five cranes operate at one oversea vessel and one to two cranes for a feeder ship. The optimization of the crane assignment and scheduling could be formulated as an MIP model, with the objective of minimizing the sum of the delays and turnaround times of all ships (*4, 12, and 67*) and the maximization of the utilization of the cranes' capacities (*56*). The crane assignment was also discussed as part of an integrated optimization problem with berth allocation (*66, 56*).

Stowage planning: Stowage planning assigns positions within the ship to specific categories of containers and considers the container's attributes, such as the destination, weight, or container type. The shipping lines and the terminal operators using an off-line optimization method decide planning. The stowage planning heavily affects the loading and unloading sequences of container, a major factor for determining the working sequences and schedules of cranes and horizontal transporters (*23*). Its objectives focus on the minimization of the number of shifts during terminal operation (*1, 2, and 13*) and the maximization of the utilization of the ship's storage capacity (*17*).

Storage and stacking policies: Storage and stacking policies attract increasing attention due to the scarcity in land area and continuous growth in containerization transportation. These policies are used to decide the specific block and slot for each container stored in the storage area. The yard reshuffle plays the most important role in the construction of storage and stacking policies. Most related studies were dedicated to the investigation of the stack configurations and their influences on the number of yard reshuffles (*25, 35, 43, and 48*). Besides, the utilization of storage area (*37, 36, 38 and 75*); the turnaround time of the ship (*48, 68*); the travel distance between the quay and yard side (*40, 85*) are also considered when making decisions.

A more comprehensive and detailed review about the logistics and operation problem in container terminals was given in the work of Steenken *et al.* (*73*), Stahlbock *et al.* (*72*), Vis and de Koster (*78*), Günther and Kim *(23*).

2.2 Transportation Optimization at Terminals

The transportation in a terminal can be distinguished between horizontal and vertical (stacking) transport. The former one can be further divided into the quay side transport and the land side transport. The quay side transport refers to the containers' transportation between the quay crane and the yard crane, carried out by a fleet of trucks, trailers, AGVs, or straddle carriers. External trucks or trains outside the terminal carry out land side transport. Vertical transport refers to the stacking transport carried out by gantry cranes, straddle carriers (SC), rubber-tired gantry cranes (RTG), and rail-mounted gantry cranes (RMG; also called automated stacking cranes (ASC)). Because neither stacking transport nor land side horizontal transport is the concern of this dissertation, we only review related papers on the quay side transport in the terminal. As the connection between the operations in the quay side and the yard side, optimization of the vehicle dispatching process should be integrated with other activities.

Bish *et al.* (*6*) addressed the vehicle-scheduling-location problem of assigning a storage location to each import container and dispatching vehicles to the containers in order to minimize the total time for unloading a vessel. To solve such an NP-hard problem, an assignment problem based (APB) heuristic algorithm was presented. The effectiveness of the proposed heuristic was analyzed from both worst-case and computational points of view.

Bish (*4*) developed a mathematical method for the so-called multiple-crane-constrained vehicle scheduling and location problem (MVSL). The problem is to determine the storage location for import containers; dispatch vehicles to containers and schedule the loading and unloading operations of the cranes with the objective of minimizing the turnaround time of ships. He analyzed the complexity of the problem and proved that it was NP-hard. He proposed a heuristic algorithm and measured its effectiveness according to the worst-case performance ratio of the heuristic algorithm.

Kang *et al.* (*32*) presented mathematical models to optimize the number of the quay cranes and trucks with the objective of minimizing the operation cost. A Markovian-based decision model was proposed as a complement to the queue model to handle generally distributed service times and non-steady-state operations. However, their study only focused on the unloading operations at the container terminals.

Koo *et al.* (*44*) proposed a fleet management procedure for the transportation system in a terminal. The objective of the management is to simultaneously find the minimum fleet size and route for each vehicle while satisfying all requirements. The first phase was constructed to obtain a lower bound on the fleet size. Then a tabu search-based procedure was presented for the travel route for each vehicle, satisfying all the transportation requirements within the planning horizon.

Meersmans and Wagelmans (*55*) considered an integrated problem that involved the scheduling of different equipment at automated terminals. They presented a branch-

and-bound algorithm and a heuristic beam search algorithm to minimize the makespan of their schedules. However, they only considered loading operations in their work.

Murty *et al.* (*58, 59*) described and analyzed a variety of interrelated activities of daily operations at a container terminal and proposed an integrative decision support system (DSS) for decision making. The objective of the DSS is multi-folded and included the minimization of waiting times for external trucks, minimization of the congestion inside and outside the terminal, and so on. The proposed DSS was evaluated at a terminal in Hong Kong and revealed a reduction of 30% in a vessel's turnaround time, as well as a reduction of 35% in container handling costs.

Qiu and Hsu (*69*) presented a bi-directional path layout and an algorithm for routing AGVs without conflicts and to minimize the space requirement of the layout. The routing efficiency was analyzed in the terms of the distance and time AGVs need to complete all transportation jobs.

Saanen *et al.* (*71*) compared the productivity of the automated container terminal (ACT) where the AGVs or automated lift vehicles (ALVs) are used as the horizontal transporter. Their performances were compared using simulation and cost modeling.

Vis *et al.* (*79*) developed a network formulation and a minimum flow algorithm to determine the necessary number of AGVs required at a semi-automated container terminal. The algorithm proposed was a strongly polynomial time algorithm and usable in warehouses and manufacturing systems, as well.

2.3 Mathematical Algorithms

To solve the mathematical models of vehicle dispatching problems and evaluate different vehicle dispatching policies, the existing approaches can be generally divided into two categories: simulation models and mathematical solution; as well, the latter can be divided further into exact methods and heuristic methods.

2.3.1 Exact Methods

Due to the extremely high computational burden, there has been very little research attempted to ascertain exact solutions to the vehicle dispatching problem.

Langevin *et al.* (*45*) proposed a dynamic programming based method to solve instances with two vehicles.

Correa *et al.* (*11*) solved the dispatching and conflict-free routing of AGVs by combining constraint programming for scheduling and mixed integer programming for conflict-free routing.

2.3.2 Heuristic Algorithm

Although the exact method can guarantee the optimal solution, its unacceptable computation time is the main obstacle to its application. Thus, further efforts have been dedicated to propose an effective heuristic method that solves the problem quickly with tolerable errors.

Ng *et al.* (*62*) formulated the vehicle dispatching and scheduling problem in a terminal as an MILP and solved it using the genetic algorithm (GA) with greedy crossover technique. The numerical experiments revealed that the proposed GA method outperformed the GA methods that employed other crossover schemes. However, their models did not consider the availability of the cranes, and each job starts as long as the vehicle arrives.

Local search techniques are heuristics that start from an initial feasible solution and "move" locally in the neighborhood of the solution space. The main drawback is that the solution found might be a local ideal but potentially far from the global best. Due to the varied techniques for escaping the local optimum, the local search technique is also a classic and effective method to solve vehicle dispatching problems. Chen *et al.* (*10*) presented an integrated model to schedule the different equipment in a container terminal. They solved the problem using tabu search algorithm and developed certain mechanisms to assure the quality and efficiency of the algorithm.

Homayouni *et al.* (*27*) solved the integrated scheduling of quay crane and AGV by using simulated annealing algorithm. They investigated the effects of initial temperature and the number of trials on the algorithm and compared them with the results obtained from solving the MILP model using branch-and-bound method.

In addition to classic search algorithms, there are plenty of heuristic methods designed to deal with characteristics of the working environment and the operation requirement. Some of them were proposed as an on-line dispatching method for practical application.

Hartmann (*26*) proposed a general optimization model for scheduling jobs at the container terminal. The model could be applied to AGVs, straddle carriers, gantry cranes, and workers. He proposed a priority rule based heuristic method and developed a genetic algorithm for solution. However, they only considered the loading operation in the research, and all the AGVs return to a common point after delivering a container.

Bish *et al.* (*5*) developed an easily implementable heuristic algorithm based on the greedy and reversed greedy algorithm. They identified both the absolute and asymptotic worst-case performance ratios of these heuristics. The above three studies did not consider the job ready times in the scheduling decisions.

Kim and Bae (*39*) developed a mixed integer programming model for assigning optimal delivery tasks to vehicles and proposed a look-ahead vehicle dispatching method to minimize the total idle time of a quay crane resulting from the late arrivals of AGVs.

Briskorn *et al.* (*8*) proposed an AGV dispatching strategy based on inventory management. The simulation study revealed the inventory-based strategy outperformed the conventional due-date-based strategies. However, their model required buffer area at the quay side, and they did not consider the precedence relationship in containers' discharging process.

As opposed to the assumption of fixed AGV's capacity in almost all research, Grunow *et al.* (*19, 20*) proposed a flexible priority rule for dispatching multi-load AGVs. Their numerical experiments revealed that the developed pattern-based off-line heuristic outperformed conventional on-line dispatching rule used in flexible manufacturing system (FMS).

Lim *et al.* (*50*) proposed an auction-based assignment algorithm in the sense that the dispatching decisions were made through communication among related vehicles and machines for matching multiple tasks with multiple vehicles. Their method considered future events and the performance of the method was compared through a simulation study.

Egbelu and Tanchoco (*16*) investigated those popular AGV dispatching rules in the manufacturing system and classified them into two categories. Vehicle-initiated rules are applied when an idle vehicle appears, and work-center-initiated rules are applied when there is a delivery task available. Their study suggested the modified first come first served (MFCFS) rule outperformed other rules. They also concluded that distance based rules are sensitive to the guide-path layout and the location of pick-up and drop-off stations.

Egbelu (*15*) posited that the source driven rules are not suitable in the systems based on just-in-time principle. He proposed a priority dispatching rule based on the demand states of the load destinations. They suggested that AGVs should be first dispatched to delivery tasks bound for input buffers whose lengths are below a threshold value.

Kim *et al.* (*33*) suggested an AGV dispatching method in a manufacturing job shop environment. They proposed a procedure to dispatch AGVs for balancing workload among different machines, as well as the workload between the machines and the transporters.

Taghaboni-Dutta (*74*) proposed Vehicle Assignment by Load Utility Evidence (VALUE) method for the assignment problem. The objective of their method attempted to achieve a higher throughput of machines by using a value-added approach to determine the AGV assignment.

2.3.3 Simulation

Due to the numerous stochastic and uncertainty elements, for instance ship delays, equipment failure, human factors, and so on, discrete event simulation approach is always used to evaluate and compare the operation process and equipment installations. Simulation results provide valuable decision information and support guidance for the design of the terminal and the logistic process (*18*).

Bae *et al.* (*3*) compared the performances of AGV and automated lifting vehicles (ALV) in an automated container terminal through simulation experiments and concluded that ALVs reached the same productivity level as the AGVs, using far fewer vehicles.

Duinkerken and Ottjes (*14*) developed a simulation model to determine the number of AGVs, maximum AGV speed, crane capacity, and stack capacity.

Huo *et al.* (*28*) examined the influence of the quantity and the speed of the internal trucks on the handling time of ships, using Witness simulation software. These researchers indicated that the quantity of trucks should be configured dynamically, according to the different stages of handling and that increasing speed of trucks would not necessarily improve the handling efficiency of the cranes.

Lau *et al.* (*52*) discussed the relationship between the number of AGVs and the terminal's layout via simulations and concluded that the yard layout has inevitable effects on the terminal performance, as well as on the number of AGVs.

Liu and Ioannou (*53*) compared four different vehicle-initiated AGV dispatching rules in an ACT from aspects of the throughput of quay crane, average waiting rate of AGV in the quay crane's queue, and so on.

Park *et al.* (*65*) developed a simulation model to compare the performance of different transporter in an ACT, using ARENA package. Based on the simulation results, they concluded that automated stacking crane is more efficient than AGV.

Vis *et al.* (*80*) analyzed the minimum vehicle fleet size under time window constraints at a container terminal. In their study, there were buffer areas under the quay cranes and in the storage area. The transportation of containers must start in the time window, determined in advance.

Yang *et al.* (*82*) and Vis and Harika (*77*) examined and compared the effects of using AGVs and ALVs on unloading times of a ship through simulation studies.

Yun and Choi (*84*) developed an object-oriented approach to simulate the operation of a container terminal using SIMPLE++ simulation software.

3. NANSHA CONTAINER TERMINAL

The operation data used in this study is provided by Guangzhou Nansha Container Terminal Phase I (NCT) in Guangzhou, China. NCT is located in the southeast of Guangzhou City and at the estuary of Pearl River, the geographic and geometrical center of Pearl River Delta, as seen in FIGURE 4. The businesses at NCT include container loading/discharging, stacking, warehousing, manufacturing, processing, maintenance, and so on.

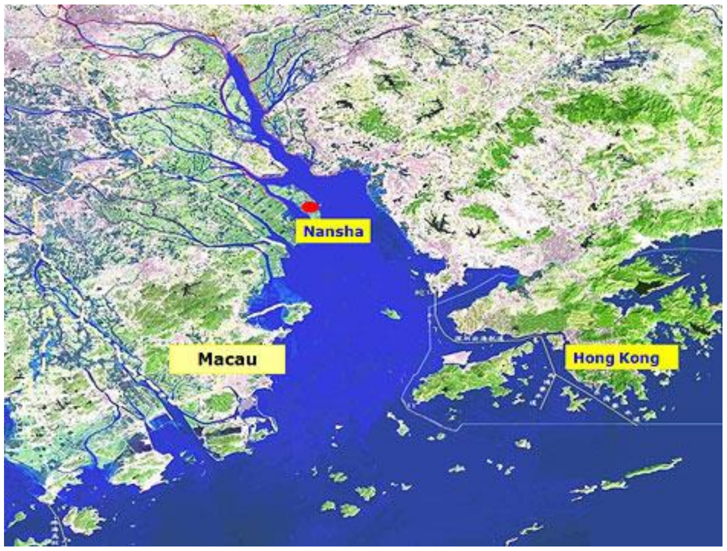


FIGURE 4 Geographic location of NCT.

3.1 Layout

The terminal is 1400m long and 1300m wide with a water depth of 15.5m. The layout of NCT is a typical parallel, side-loaded terminal layout (FIGURE 5) where the containers stacks are parallel with the berth. This type of layout is the most popular in Asian container terminals. All the QCs are equipped at the waterside of the terminal, and all the storage blocks are located at the other side. Rubber-tired gantry cranes (RTGCs) and rail-mounted gantry cranes (RMGCs) are always used as yard cranes in the storage area (*46, 47*). Vehicles can park at a side of a block where the container is picked up (put down onto) the vehicles. The path area in the quay side is a multi-lane loop layout where vehicles travel in a single loop. It is very convenient and simple for traffic control at the terminal. The loop is a fixed sequence of picking up and dropping off points at QCs and YCs.

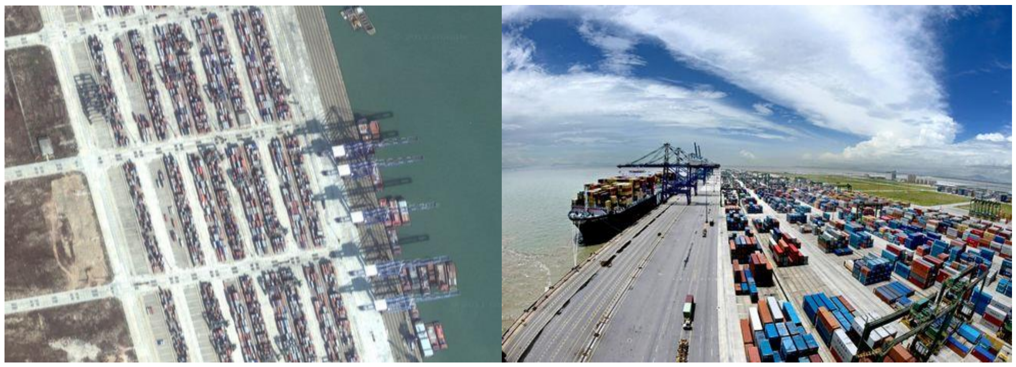


FIGURE 5 Satellite view and the layout of NCT.

3.2 Equipment

The quay side of NCT is equipped with sixteen quay cranes. Thirteen are conventional single trolley QCs, and three are tandem lift QCs. A few days prior to the arrival of a ship, the terminal will be informed of the status of the ship and containers. Then the terminal operators make the plan of quay crane allocation and generate working sequences for those QCs and the storage plan of import containers, according to the rules and regulations in the terminal. The generation of the working sequence has to consider many elements, like the weight, size, and content of the container, the contract between the shipper and the terminal, the stability of the ship, and so on. FIGURE 6 is part of a QC's working sequence, generated using the software in NCT. The content of the working sequence is explained in the following way:

SEQ – the number of the operation sequence;

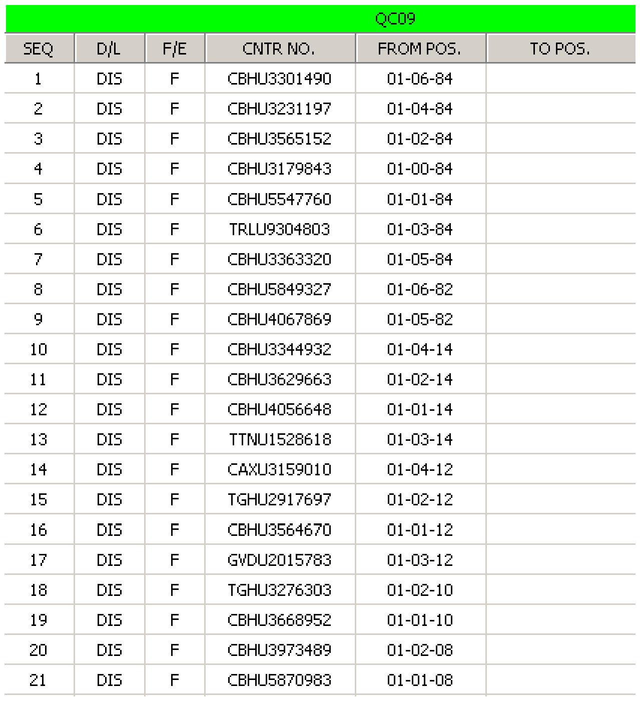
D/L – the type of operation, where D represents discharge (unload) and L represents load the container;

F/E – the weight of the container, where F represents that the container is full and E as empty;

CNTR No. – Each container is labeled with a unique code number. This code number is used for the operator to track the information and status of every container. During the operation process, the operator and the manager can access the information of a container by scanning its code number.

FROM POS. and TO POS. – the container's storage position on the ship. In the unloading process, the QC unloads the containers from the ship and the TO POS.

column is empty. During the loading process, the quay crane loads the containers onto the ship, and the FROM POS. column is empty.



QC09					
SEQ	D/L	F/E	CNTR NO.	FROM POS.	TO POS.
1	DIS	F	CBHU3301490	01-06-84	
2	DIS	F	CBHU3231197	01-04-84	
3	DIS	F	CBHU3565152	01-02-84	
4	DIS	F	CBHU3179843	01-00-84	
5	DIS	F	CBHU5547760	01-01-84	
6	DIS	F	TRLU9304803	01-03-84	
7	DIS	F	CBHU3363320	01-05-84	
8	DIS	F	CBHU5849327	01-06-82	
9	DIS	F	CBHU4067869	01-05-82	
10	DIS	F	CBHU3344932	01-04-14	
11	DIS	F	CBHU3629663	01-02-14	
12	DIS	F	CBHU4056648	01-01-14	
13	DIS	F	TTNU1528618	01-03-14	
14	DIS	F	CAXU3159010	01-04-12	
15	DIS	F	TGHU2917697	01-02-12	
16	DIS	F	CBHU3564670	01-01-12	
17	DIS	F	GVDU2015783	01-03-12	
18	DIS	F	TGHU3276303	01-02-10	
19	DIS	F	CBHU3668952	01-01-10	
20	DIS	F	CBHU3973489	01-02-08	
21	DIS	F	CBHU5870983	01-01-08	

FIGURE 6 Part of a quay crane's working sequence.

Unless encountering an unexpected event, the containers are loaded/unloaded following the QCs' working sequences, predetermined according to the stowage planning and other detailed information sent to the terminal, such as the historical average cycle time and travel time (*23, 60*). The cycle time of a QC is the time required to transfer a container between the ship and a vehicle. The cycle time depends, for example, on the design capabilities of the crane, the operation experience of the crane driver, the type of ship, and the position of the container in the ship. The cycle time of the tandem lift QC used in this study is based on the record of 10hrs' operations in NCT. According to the empirical distribution listed in TABLE 1, the time interval 80-90 seconds takes the highest percentage. The cycle time used in the mathematical model is supposed to be the QC's designed working speed, assuming there is no idling due to other external influences. Thus, the cycle time intervals shorter than 50 seconds or longer than 140 seconds are not included. These long cycle times might due to the mistake records or possible large disturbances, like breakdowns of equipment, wrong information on containers, or generally unexpected events and mistakes.

TABLE 1 Empirical Distribution of QCs' Cycle Times

Cycle time (sec)	Fraction
50-60	0.04
60-70	0.08
70-80	0.15
80-90	0.19
90-100	0.14
100-110	0.13
110-120	0.09
120-130	0.10
130-140	0.08

The yard cranes employed in the storage area of NCT are rubber-tired gantry crane (RTG, FIGURE 7). Among the seven lanes under each RTG, six lanes are used to stock the containers and the other one is used as the passenger lane for trucks passing.

The height of the containers stacked in the storage area is always fiver layers. There are total eighty blocks in the yard area for the containers' temporal storage and 46 RTGs for storing and retrieving containers. The RTGs can travel between the blocks according to the working loads among different blocks. There is always one or two RTGs stack and retrieve containers at each block.

FIGURE 7 RTGs at the storage blocks.

3.3 Container Flow

To facilitate understanding the container flow in the terminal and interrelated activities of different equipment, we precisely describe the unloading process as an example.

When a tandem lift QC starts unloading a pair of containers following its working sequence, two trucks are chosen from the pool of idle vehicles to pick up the two containers at the QC side. When these two vehicles have arrived there and stop under the QC side by side, the QC starts to position the containers on them; otherwise, the QC has to wait for them, delaying and disrupting the current unloading operation and the succeeding operations in that QC’s working sequence. In contrast, the two vehicles might arrive before the QC is ready to position the two containers onto them. Obviously this case does not generate any additional waiting time for the QC. However, it might affect other operations’ availabilities of vehicles. After this step in the process, the two vehicles travel to the two containers’ storage blocks and drop them off there. When the vehicle arrives at the storage block, the vehicle must stop in front of the block and wait for the signal. When the yard crane is available, the vehicle proceeds to the passenger lane and the YC lifts the container from the vehicle and stacks it to its destination position according to the command sent from the control center. Note that the storage blocks for the two containers in the same tandem lift might be different from each other. For the loading process, since it is conducted in the reverse order of unloading process, we do not elaborate here further.

4. PROBLEM STATEMENT

The problem central to this dissertation is to seek a way to dispatch a fixed number of vehicles to transport containers between the tandem lift QCs and YCs with the objective of minimizing the makespan or the total time it takes to finish loading and unloading all containers in the planning horizon.

For the purpose of modeling, the layout of NCT is simplified as a single loop layout with multi-lane paths without losing the main characteristics (FIGURE 8). The QCs are lined up in a row along the berth, while all storage blocks are located at the opposite side. The containers are transported by a fleet of trucks that travel in a counterclockwise direction. All the vehicles are identical and have the same unit capacity. Each vehicle can carry only one 40ft container or two 20ft containers at a time. In the remainder of this dissertation, two 20ft containers are treated as one 40ft container since they are always operated and transported together.

The cycle time (denoted by τ) of a tandem lift QC is defined as the duration necessary to transfer two containers from a ship (vehicle) to a vehicle (ship). In an unloading process, the cycle time starts from the moment the QC lifts containers from a deck or a hold, and ends at the moment the QC is ready to position them onto vehicles. In a loading process, it starts from the moment the QC lifts containers from vehicles, and ends at the moment the QC positions them onto the ship.

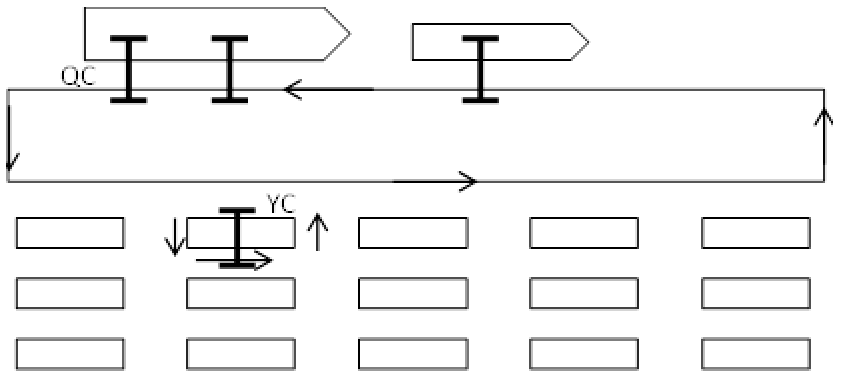


FIGURE 8 Single loop layout at the quay side.

4.1 Problem Description

The QCs equipped along the berth and the vehicles used for the transportation are numbered as $\{q_{1,}q_2,\cdots q_k,\cdots\}$ and $\{v_{1,}v_2,\cdots v_m,\cdots\}$, respectively. All the operations in the q_k 's working sequence are numbered as $\{o_k^1,o_k^2,\cdots,o_k^s,\cdots\}$ and each could be loading or unloading operation. The two containers operated by q_k as its s^{th} operation are referred to as $c_k^{s,1}$ and $c_k^{s,2}$. The number 1 and 2 in the superscript refer to the two containers in the same tandem lift. When a QC starts one operation, a transportation request r_k^s and four tasks are generated correspondingly. Take for instance, the following: when o_k^s is a loading operation, the transportation request r_k^s is to transport the two containers $c_k^{s,1}$ and $c_k^{s,2}$ from q_k to their destination blocks where the YCs stack them to their destination positions. And two vehicles are required to complete two pickup tasks ($p_k^{s,1}$ and $p_k^{s,2}$) of containers $c_k^{s,1}$ and $c_k^{s,2}$ at q_k and two drop-off tasks ($d_k^{s,1}$ and $d_k^{s,2}$) of them at the YCs. The relationship between the operations, the transportation requests and the tasks are summarized in TABLE 2.

TABLE 2 Relationship Between the Operations, Requests and Tasks

o_k^s	r_k^s	QC side task	YC side task
Unload $c_k^{s,1}$ and $c_k^{s,2}$ from the ship	Transport $c_k^{s,1}$ and $c_k^{s,2}$ from the QC to the YC	$p_k^{s,1}$ and $p_k^{s,2}$	$d_k^{s,1}$ and $d_k^{s,2}$
Load $c_k^{s,1}$ and $c_k^{s,2}$ onto the ship	Transport $c_k^{s,1}$ and $c_k^{s,2}$ from the YC to the QC	$d_k^{s,1}$ and $d_k^{s,2}$	$p_k^{s,1}$ and $p_k^{s,2}$

Given a number of vehicles and QCs with predetermined sequences of operations, vehicle dispatching problem is to dispatch vehicles to complete all the tasks with the objective of minimizing the makespan. As long as the assignments of vehicles are decided, the schedules of all tasks are decided at the same time. In addition to those constraints shared among those classical vehicle routing and scheduling problem, there are extra constraints to account for the working characteristics of tandem lift operations:

(1) Precedence Constraints: Since all the containers are operated following the pre-defined working sequences, their transportation could not violate their precedence relationships. In other words, the containers $c_k^{s,1}$ and $c_k^{s,2}$ cannot be picked up or dropped off by any vehicle before the containers $c_k^{s',1}$ or $c_k^{s',2}$ if $s > s'$. Consequently, the earliest starting time of QC side tasks, pick-up or

drop-off tasks whose locations are at the QCs are heavily dependent on the containers' delivery sequences.

(2) Simultaneous Constraint: Because of the tandem lift operation, the two containers in the same tandem lift should be picked up or dropped off by the vehicles at the same time. In other words, the starting times of their corresponding QC side tasks should be the same. Thus, the starting time of a QC side task $d_k^{s,1}$ or $p_k^{s,1}$ depends on not only the transportation of $c_k^{s,1}$, but also the delivery schedule of $c_k^{s,2}$.

(3) Capacity Constraint: Because of the vehicle's unit capacity, the two containers in the same lift cannot be transported by the same vehicle.

4.2 Model Formulation

Based on the above description, the problem can be formulated as an MILP model with the objective of minimizing the makespan. The parameters and the notations that will be used in the dissertation are presented as follows:

i, j, t	the index of tasks
m, n	the index of vehicles
k	the index of quay cranes
s	the index of the operations in a QC's working sequence
g	the index of the two containers operated in one tandem lift
o_k^s	the s^{th} operation in the working sequence of q_k

$c_k^{s,g}$	the container unloaded/loaded in the operation o_k^s
$p_k^{s,g}$	the pick-up task of container $c_k^{s,g}$
$d_k^{s,g}$	the drop-off task of container $c_k^{s,g}$
$y_k^{s,g}$	the quay side task of container $c_k^{s,g}$, i.e. the pick-up or drop-off task whose location is at the QC side. For unloading operation $y_k^{s,g} = p_k^{s,g}$ while for loading operation $y_k^{s,g} = d_k^{s,g}$
b	the dummy begin task
e	the dummy end task
C	the set of all containers
V	the set of all vehicles
Q	the set of all quay cranes
S_k	the set of all operations in the working sequence of q_k
P	the set of all pick-up tasks
D	the set of all drop-off tasks
T	the set of all tasks $T = P \cup D$
Y	the set of all quay side tasks
$\delta_{i,j}$	the travel time from task i to task j
τ	the tandem lift QC's working cycle time
σ	the time needed to position (lift) a container from a crane to (from) a vehicle
λ	the vehicles' average waiting time at a YC
M	a sufficient large positive number

The decision variables in the model include:

$x_{i,j}$	the binary variable which equals to 1 if task j is immediately completed after task i by the same vehicle;
α_i	the time when the vehicle arrives at the QC/YC where the task i is;
ε_i	the time when the task i starts;
π	makespan; the time when all tasks have been completed

With the notations, parameters and variables described above, the problem is formulated as the following MILP model.

$$\min\ \pi \tag{1}$$

Subject to:

$$\sum_{j\in P} x_{b,j} = |V| \tag{2}$$

$$\sum_{i\in D} x_{i,e} = |V| \tag{3}$$

$$\sum_{i\in T\cup b} x_{i,t} = 1\,,\ \forall t \in T \tag{4}$$

$$\sum_{j\in T\cup e} x_{t,j} = 1\,, \forall t \in T \tag{5}$$

$$x_{p_k^{s,g},d_k^{s,g}} = 1\,, \text{ where } k \in Q,\ s \in S_k,\ g = 1 \text{ or } 2 \tag{6}$$

$$\alpha_j \geq \varepsilon_i + \sigma + \delta_{i,j} + M\ (x_{i,j}\text{-}1)\,,\ \ \forall i \in T \cup b, \forall j \in T \tag{7}$$

$$\alpha_j \leq \varepsilon_i + \sigma + \delta_{i,j} + M \cdot (1 - x_{i,j})\,,\ \ \forall i \in T \cup b, \forall j \in T \tag{8}$$

$$\varepsilon_t \geq \alpha_t\ , t \in Y \tag{9}$$

$$\varepsilon_t \geq \alpha_t + \lambda\ , t \in T/Y \tag{10}$$

$$\varepsilon_{y_k^{s+1,g}} - \varepsilon_{y_k^{s,g}} > \tau,\ \text{ where } k \in Q, s, s+1 \in S_k,\ g = 1 \text{ or } 2 \tag{11}$$

$$\varepsilon_{y_k^{s,1}} = \varepsilon_{y_k^{s,2}}, \text{ where } k \in Q, s \in S_k \tag{12}$$

$$\pi \geq \varepsilon_t, \quad \forall t \in T \tag{13}$$

$$x_{i,j} \in \{0, 1\}, \quad \forall i, j \in T. \tag{14}$$

The objective is to minimize the time of completing all pick-up and drop-off tasks. Constraints (2) and (3) ensure that each vehicle delivers at least one container by assigning the dummy begin and end tasks. $|V|$ denotes the cardinal of the set of vehicles. Constraints (4) and (5) ensure that each container is delivered by one and only one vehicle. Constraint (6) ensures the vehicle that picks up a container at a QC (YC) must be the same one that drops it off at a crane. Constraints (7) and (8) define the arrival time when a vehicle executes a task. Constraint (9) ensures that a quay side task's starting time cannot be earlier than the vehicle's arrival time. The vehicle' average waiting time at the container's storage block is represented in constraint (10). The precedence relationship in the QCs' working sequences is guaranteed by constraint (11). Constraint (12) ensures that the pick-up or drop-off tasks of two paired containers start at the same time. Constraint (13) defines the calculation of makespan. Constraint (14) defines the binary variable.

5. LOCAL SEQUENCE CUT ALGORITHM

5.1 Motivation

The difficulties in solving the problem lie in the fact that when the number of containers increases, the volume of binary decision variables and the search region expands quickly and significantly. The exact method, like branch-and-bound method, cannot find the optimal, even a feasible solution within acceptable computation time. Thus, this dissertation proposes an algorithm to reduce the search region by eliminating those feasible but undesirable solutions. Then the solver searches for a good, or even the optimal solution, within a reduced feasible region.

The proposed algorithm is inspired by the concept of "logic cuts." The algorithm reduces the feasible region by eliminating some integer feasible solutions demonstrably not optimal, according to the logic considerations. These "cuts" can be indeed very effective. They may significantly shrink the feasible region and considerably quicken the reduction of the optimality gap throughout the iterations of the solver. As a result, they can be extremely beneficial in reducing the CPU time in the search for optimality. The concept of "logic cuts" has been successfully applied in solving the complex combinational problems. Quadrifoglio *et al.* (*70*) developed sets of logic cuts to solve a MIP formulation for the static scheduling problem of a mobility allowance shuttle transit (MAST) system. They took advantage of those cuts based on the reasonable assumptions of the passengers' behavior to develop sets of logic cuts and impose them into the MIP formulation. Indeed, they developed sets of logic cuts to speed up the search for optimality. The computation experiment revealed that the developed inequalities effectively removed inefficient solutions from the feasible region and reduced the computation time up to 90%. Guignard *et al.* (*22*) formulated an integrated timber harvest and transportation planning problem as 0-1 MIP model. They sped the searching process by using different tightening techniques, including the addition of logical inequalities, lifting of inequalities, and branch-and-bound branching priorities based on consideration of double-contraction.

The algorithm proposed in this dissertation is named Local Sequence Cut (LSC) Algorithm. If container $c_k^{s,g}$ is transported immediately after $c_{k'}^{s',g'}$ by the same vehicle, then $(c_{k'}^{s',g'}, c_k^{s,g})$ is defined as a delivery sequence of $c_k^{s,g}$. Thus, a solution to the problem is a combination of every container's delivery sequence. The entire feasible search region is the set of all containers' feasible delivery sequences satisfying all constraints. Therefore, the mission of LSC algorithm is to reduce the search region by eliminating those feasible but undesirable delivery sequences without losing the possibility of finding a good, even optimal, integer solution. Since the objective of the problem is to minimize the makespan, it is reasonable that there should be a *time window* for every QC side task's starting time. If a QC side tasks starts within its time window, it implies that the delivery of the container does not delay the operation of that container or postpone the current best makespan. Otherwise, it means that the container is transported too late or too early, and its current delivery sequence is defined as a *cut-off delivery sequence* and removed from the search space.

The upper and lower bounds of these time windows are estimated through constructing and solving upper and lower sub-problems iteratively until all time windows have been updated or the maximum number of iterations is reached. At the end of the algorithm, all the cut-off delivery sequences based on the final time windows are excluded from the search space of the original MILP model by setting those corresponding binary decision variables to 0. Consequently, the solver searches for the solution within the reduced feasible region and is supposed to find a good, even optimal solution much more quickly. The process and details of the algorithm are stated in the following.

5.2 Basic Scheme of Local Sequence Cut Algorithm

At first, we introduce the basic scheme of the proposed LSC algorithm. Denote the upper and lower bounds of a QC side task's starting time $\varepsilon_{y_k^{s,g}}$ as follows:

$\underline{\varepsilon}_{y_k^{s,g}}$, $k \in Q$, $s \in S_k$, $g = 1$ or 2	the earliest allowable starting time of the QC side task $y_k^{s,g}$; or the lower bound of $\varepsilon_{y_k^{s,g}}$ ($\underline{\varepsilon}_{y_k^{s,1}} = \underline{\varepsilon}_{y_k^{s,2}}$)
$\overline{\varepsilon}_{y_k^{s,g}}$, $k \in Q$, $s \in S_k$, $g = 1$ or 2	the latest allowable starting time of the QC side task $y_k^{s,g}$; or the upper bound of $\varepsilon_{y_k^{s,g}}$ ($\overline{\varepsilon}_{y_k^{s,1}} = \overline{\varepsilon}_{y_k^{s,2}}$)

5.2.1 Initial Solution

To start the iteration, the algorithm requires a set of initial lower and upper bounds for the QC side tasks. The initial solution can be obtained by any method; we present one simple approach here. We divide all vehicles into several groups as evenly as possible and each group serves one QC only. For instance, assume that there are three QCs and six vehicles, then all vehicles are divided into three groups. Then the two vehicles in each group only serve one QC from the beginning to the end. FIGURE 9 shows the initial assignments of vehicles and delivery sequences of containers. The containers transported sequentially by Vehicle 1 are $\{c_k^{1,1}, c_k^{2,1}, c_k^{3,1}, \cdots, c_k^{s,1}, \cdots, c_k^{|S_k|,1}\}$ and those by Vehicle 2 are $\{c_k^{1,2}, c_k^{2,2}, c_k^{3,2}, \cdots, c_k^{s,2}, \cdots, c_k^{|S_k|,2}\}$. As long the delivery sequences are determined, each QC side task's starting time is known at the same time. Assume that the first pair of containers $c_k^{1,1}$ and $c_k^{1,2}$ unloaded by q_k are picked up by vehicles at time μ . Because of the precedence relationship and the QC's cycle time, the succeeding containers $c_k^{s,g}$ operated by q_k cannot be picked up or dropped off by vehicles earlier than

$$\underline{\varepsilon}_{y_k^{s,g}} = \mu + (s-1)\cdot\tau \qquad (15)$$

The initial upper bounds are calculated according to the objective value of the initial solution, viewed as the current best makespan denoted by π^*. And the upper bounds of all QC side tasks' starting times are calculated as:

$$\bar{\varepsilon}_{y_k^{s,g}} = \pi^* - (|S_k| - s) \cdot \tau \tag{16}$$

The initial lower and upper bound calculated by equation (15) and (16) are quite loose and defined as the absolute lower and upper bounds. Because there is no QC side task could start earlier or later than its corresponding absolute lower or upper bound.

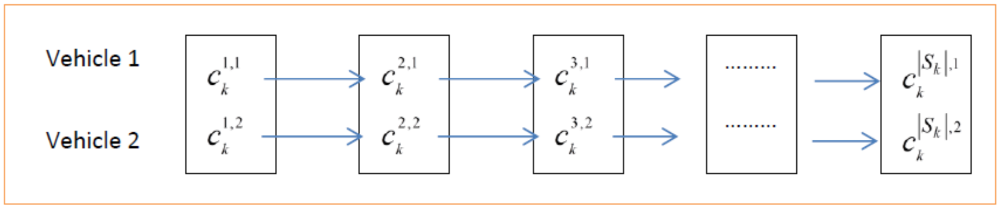


FIGURE 9 Initial delivery sequences of the two vehicles in a group.

5.2.2 Time Windows and Cut-off Delivery Sequences

One core mission of the algorithm is to determine whether or not a feasible delivery sequence of a container is a cut-off delivery sequence, dependent on the relationship between the QC side task's starting time and its time window. When container $c_k^{s,g}$ is transported following the sequence $(c_{k'}^{s',g'}, c_k^{s,g})$, if the QC side task's starting time is later or earlier than the upper or lower bound of its current time window, then $(c_{k'}^{s',g'}, c_k^{s,g})$ is defined as a cut-off delivery sequence. However, the starting times are unknown until the entire problem has been solved. Thus, the algorithm uses an easy-to-check and comparative variable – arrival time -- taking the place of starting time in deciding cut-off delivery sequences.

Because the traveling times and the operation times at the YC side are deterministic, the earliest and latest arrival time of a QC side task ($\bar{\alpha}_{y_k^{s,g}}$ and $\underline{\alpha}_{y_k^{s,g}}$) are easily estimated by the following equations (17) and (18). After finishing its current QC side task $y_{k'}^{s',g'}$, the vehicle needs some time to arrive at its next QC side task $y_k^{s,g}$. It includes the traveling times and necessary operation and waiting times at the YC side. Here, we define the sum of those times as setup time between the two tasks ($\eta_{y_{k'}^{s',g'},y_k^{s,g}}$). Now the earliest and latest arrival time $\bar{\alpha}_{y_k^{s,g}}$ and $\underline{\alpha}_{y_k^{s,g}}$ can be calculated by adding the setup time to the lower and upper bound of task $y_k^{s,g}$'s starting time as stated in equations (17) and (18).

$$\bar{\alpha}_{y_k^{s,g}} = \bar{\varepsilon}_{y_{k'}^{s',g'}} + \eta_{y_{k'}^{s',g'},y_k^{s,g}}, \tag{17}$$

$$\underline{\alpha}_{y_k^{s,g}} = \underline{\varepsilon}_{y_{k'}^{s',g'}} + \eta_{y_{k'}^{s',g'},y_k^{s,g}}. \tag{18}$$

The calculation of $\eta_{y_{k'}^{s',g'},y_k^{s,g}}$ is determined by what types the tasks $y_{k'}^{s',g'}$ and $y_k^{s,g}$ are. All the four possible cases are listed in TABLE 3. In case 1, both tasks $y_{k'}^{s',g'}$ and $y_k^{s,g}$ are pick-up tasks at the QC side, denoted by (P, P). The vehicle leaves $q_{k'}$ after picking up container $c_{k'}^{s',g'}$ and then travels to the YC where container $c_{k'}^{s',g'}$ is stacked. After that, this vehicle travels to the QC where container $c_k^{s,g}$ is unloaded and picks it up. This whole process, i.e., from QC to YC and then to QC, is denoted by Q-Y-Q in TABLE 3. One can easily see that $\eta_{y_{k'}^{s',g'},y_k^{s,g}}$ consists of two traveling times and two handle times as shown in the equation (19). For the case (P, D), the two adjacent QC side tasks are the pick-up and drop-off tasks, respectively. The vehicle first picks up the container $c_{k'}^{s',g'}$ at $q_{k'}$ and drops it off at its destination block at the YC side. Then this vehicle picks up another container $c_k^{s,g}$ at the YC side and drops it off at q_k. Other cases in TABLE 3 can be explained in a similar fashion.

TABLE 3 Calculation of Setup Time

Case (P, P): Q – Y – Q

$$\eta_{y_{k'}^{s',g'},y_k^{s,g}} = \sigma + \delta_{p_{k'}^{s',g'},d_{k'}^{s',g'}} + \lambda + \sigma + \delta_{d_{k'}^{s',g'},p_k^{s,g}} \tag{19}$$

Case (P, D): Q – Y – Y – Q

$$\eta_{y_{k'}^{s',g'},y_k^{s,g}} = \sigma + \delta_{p_{k'}^{s',g'},d_{k'}^{s',g'}} + \lambda + \sigma + \delta_{d_{k'}^{s',g'},p_k^{s,g}} + \sigma + \delta_{p_k^{s,g},d_k^{s,g}} + \lambda \tag{20}$$

Case (D, P): Q – Q

$$\eta_{y_{k'}^{s',g'},y_k^{s,g}} = \sigma + \delta_{d_{k'}^{s',g'},p_k^{s,g}} \tag{21}$$

Case (D, D): Q – Y – Q

$$\eta_{y_{k'}^{s',g'},y_k^{s,g}} = \sigma + \delta_{d_{k'}^{s',g'},p_k^{s,g}} + \lambda + \sigma + \delta_{p_k^{s,g},d_k^{s,g}} + \lambda \tag{22}$$

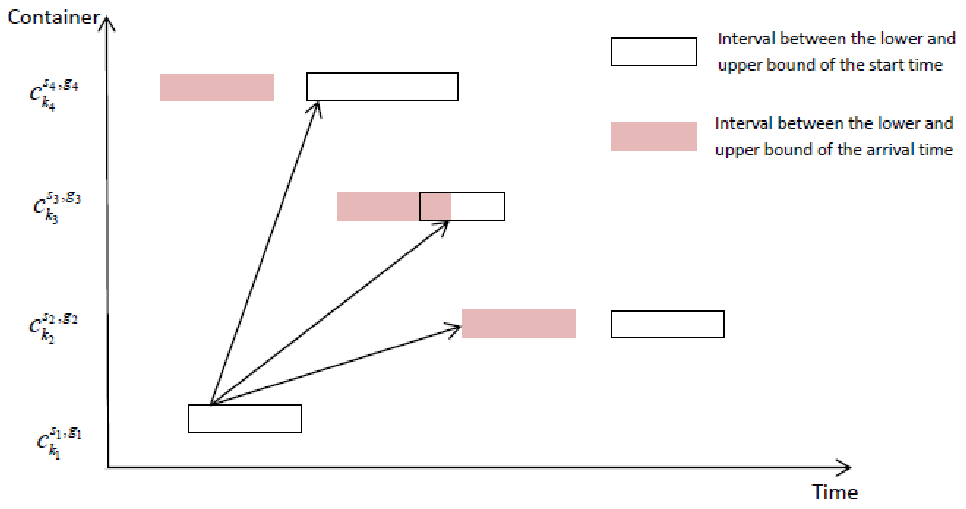
Note: P: pick-up task; D: drop-off task; Q: quay crane; Y: yard crane.

Now we are ready to present the criteria employed to determine the cut-off delivery sequences. Illustrated in FIGURE 10 as an example, now the criterion depends on the relationship between the earliest (latest) arrival time and the upper (lower) bound of the starting time. Assume that time windows of QC side tasks $y_{k_m}^{s_m,g_m}$'s starting times, where $m=1$ to 4, have been decided during earlier iteration of the proposed method and are $[\underline{\varepsilon}_{y_{k_m}^{s_m,g_m}}, \overline{\varepsilon}_{y_{k_m}^{s_m,g_m}}]$. The ranges of those time windows are signified by the solid red blocks in FIGURE 10. To determine whether or not container $c_{k_2}^{s_2,g_2}$, $c_{k_3}^{s_3,g_3}$ and $c_{k_4}^{s_4,g_4}$ are allowed to be transported immediately after container $c_{k_1}^{s_1,g_1}$ by the same vehicle without violating their current time windows, their earliest and latest possible arrival times $[\underline{\alpha}_{y_{k_m}^{s_m,g_m}}, \overline{\alpha}_{y_{k_m}^{s_m,g_m}}]$ are calculated according to the equations (17) to (22) and presented by the white blocks.

According to the criteria, the delivery sequence $(c_{k_1}^{s_1,g_1}, c_{k_4}^{s_4,g_4})$ is a cut-off delivery sequence, as the earliest arrival time is still later than the upper bound of the time window. The vehicle's late arrival delays the operation of container $c_{k_4}^{s_4,g_4}$ and all its succeeding operations and the current best makespan. At the same time, $c_{k_2}^{s_2,g_2}$ should not be transported immediately after $c_{k_1}^{s_1,g_1}$ by the same vehicle, either. Its latest arrival time $\overline{\alpha}_{y_{k_2}^{s_2,g_2}}$ is earlier than the time window's lower bound $\underline{\varepsilon}_{y_{k_2}^{s_2,g_2}}$. This case, of course, does not delay the operation of its succeeding operations. However, the vehicle's waiting time at the QC is a waste of vehicle resources and might affect other operations' availabilities of vehicles, postponing the current best makespan as a result. Thus, only $(c_{k_1}^{s_1,g_1}, c_{k_3}^{s_3,g_3})$ is not a cut-off delivery sequence because its arrival time block overlaps with the corresponding starting time block, i.e. $\left[\underline{\alpha}_{y_{k_3}^{s_3,g_3}}, \overline{\alpha}_{y_{k_3}^{s_3,g_3}}\right] \cap \left[\underline{\varepsilon}_{y_{k_3}^{s_3,g_3}}, \overline{\varepsilon}_{y_{k_3}^{s_3,g_3}}\right] \neq \phi$.

In summary, assuming that a vehicle is dispatched to transport the container $c_k^{s,g}$ immediately after delivering $c_{k'}^{s',g'}$, if the vehicle's arrival time at the q_k violates the criteria expressed in (23), then the delivery sequence $(c_{k'}^{s',g'}, c_k^{s,g})$ is a cut-off delivery sequence.

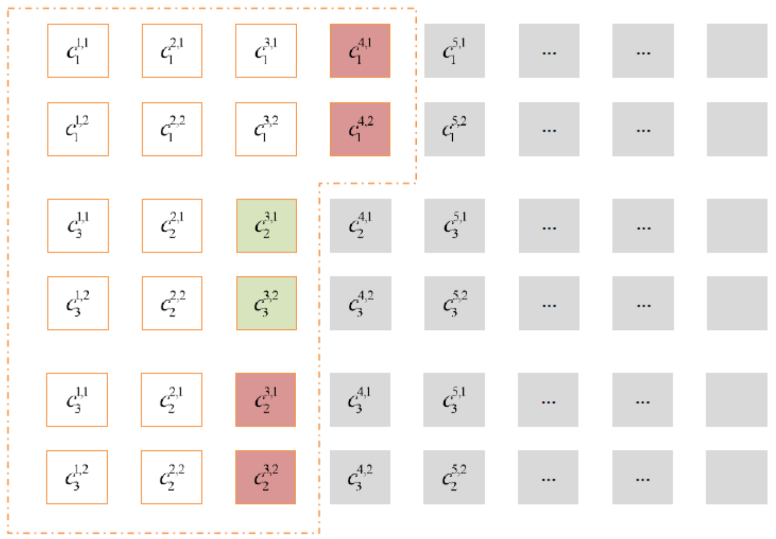
$$\left[\underline{\alpha}_{y_k^{s,g}}, \overline{\alpha}_{y_k^{s,g}}\right] \cap \left[\underline{\varepsilon}_{y_k^{s,g}}, \overline{\varepsilon}_{y_k^{s,g}}\right] \neq \phi \tag{23}$$



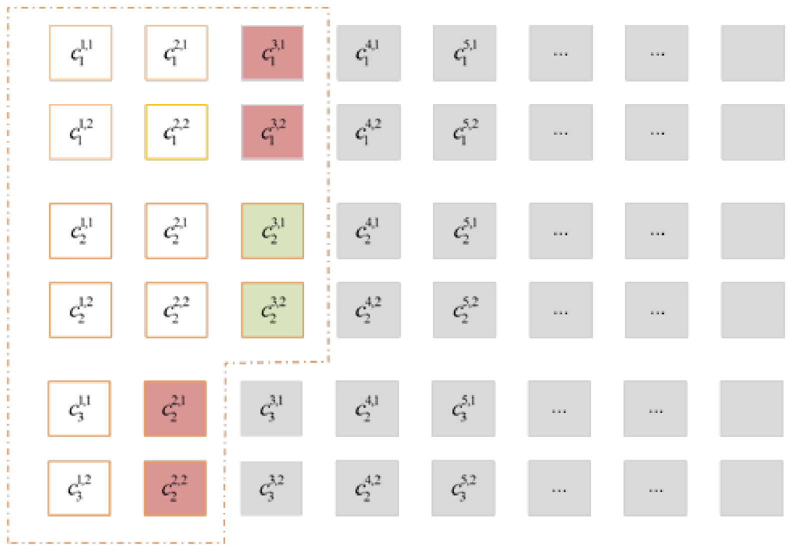
Container
$c_{k_4}^{s_4,g_4}$
$c_{k_3}^{s_3,g_3}$
$c_{k_2}^{s_2,g_2}$
$c_{k_1}^{s_1,g_1}$
Time
Interval between the lower and upper bound of the start time
Interval between the lower and upper bound of the arrival time

FIGURE 10 Determination of cut-off delivery sequences.

5.2.3 Sub-problems



(a) Upper bound sub-problem



(b) Lower bound sub-problem

FIGURE 11 Construction of sub-problems.

Now, the only issue left that critically entails the algorithm is the estimation of time windows. Since the container's delivery schedule is dependent on its delivery sequence, as long as the earliest and latest delivery sequence is determined, the lower and upper bound of the QC side task's starting time can be determined by constructing and solving sub-problems. In addition, the optimized results of sub-problems are also used to check whether or not the containers' earliest and latest delivery sequence affects the operations of containers and the current best makespan.

For the sake of illustration, we explain an upper bound sub-problem in detail. In FIGURE 11, this sub-problem is to estimate $\bar{\varepsilon}_{y_2^{3,1}}$ and $\bar{\varepsilon}_{y_2^{3,2}}$. Since these two containers are operated in the same tandem lift, there should be $\bar{\varepsilon}_{y_2^{3,1}} = \bar{\varepsilon}_{y_2^{3,2}}$. We define L_2^3 as the set of the latest delivery sequences of containers $c_2^{3,1}$ and $c_2^{3,2}$. According to the criteria in the determination of cut-off delivery sequences, if the container is transported later than its latest delivery sequence, the earliest arrival time will be later than the current time window's upper bound. As shown in FIGURE 11, if the container $c_2^{3,1}$ is transported immediately after container $c_1^{4,1}$ by the same vehicle, there will be $\underline{\alpha}_{y_2^{3,1}} \le \bar{\varepsilon}_{y_2^{3,1}}$. However, if $c_2^{3,1}$ is transported immediately after container $c_1^{5,1}$ then $\underline{\alpha}_{y_2^{3,1}} > \bar{\varepsilon}_{y_2^{3,1}}$ (a violation). Consequently, the container $c_1^{4,1}$ is added to the set L_2^3. Because the delivery schedule of $c_2^{3,1}$ is also dependent on the delivery of $c_2^{3,2}$, operated in the same tandem lift, the latest delivery sequences of $c_2^{3,2}$ are also added into the set L_2^3.

However, until now the latest delivery sequences of containers and are only defined according to the estimated time windows and the calculated arrival times, instead of the actual starting times. To check whether or not the two target containers' latest delivery sequences delay their operations or the current best makespan, the algorithm optimizes the delivery sequences of all related containers that are circled by the dashed frame, using an upper bound sub-problem. In addition to the constraints of the original MILP model, there are some extra constraints imposed on the MILP model of the sub-problem:

(1) The two target containers $c_2^{3,1}$ and $c_2^{3,2}$ must be transported following the latest delivery sequence in the set L_2^3 by adding constraint $\sum_{c_{\tilde{k}}^{s,\tilde{g}} \in L_2^3} \sum_{g=1}^{2} x_{d_{\tilde{k}}^{\tilde{s},\tilde{g}}, p_2^{3,g}} = 2$.

(2) The delivery sequences of containers represented by the dashed grey blocks (denoted by Φ_2^3) do not affect the delivery sequences or schedules of related containers, so the sub-problem does not optimize those containers' delivery sequences or schedules. Thus, they are transported following their delivery sequences of the initial solution.

(3) To reduce the computation burden of the sub-problem, all *temporary cut-off delivery sequences* of related containers (denoted as the set Θ_2^3) based on current time windows are eliminated from the search space by setting the corresponding binary decision variables to zero. They are defined as temporary cut-off sequences because they are only valid in this specific sub-

problem. When constructing the next sub-problem, all the temporary cut-off sequences need to be re-checked according to the new time windows.

After adding the extra constraints listed in TABLE 4, the algorithm constructs the upper sub-problem for $\bar{\varepsilon}_{y_2^{3,1}}$ and $\bar{\varepsilon}_{y_2^{3,2}}$ then solves it using the branch-and-bound method. Assume that the optimized delivery sequences of the target containers are $(c_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1},\ c_2^{3,1})$ and $(c_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2},\ c_2^{3,2})$ and the starting time of $y_2^{3,1}$ and $y_2^{3,2}$ is $\varepsilon^*_{y_2^{3,1}}$. If $\varepsilon_{y_2^{3,2}} = \varepsilon_{y_2^{3,1}} = \varepsilon^*_{y_2^{3,1}} \le \bar{\varepsilon}_{y_2^{3,1}}$; this implies that their latest delivery sequences do not violate the current upper bound of the time window or postpone the current best makespan. Then the optimized starting time $\varepsilon^*_{y_2^{3,1}}$ is used as the new upper bound ($\bar{\varepsilon}_{y_2^{3,1}} = \varepsilon^*_{y_2^{3,1}}$). In addition, all the containers operated before $c_2^{3,1}$ and $c_2^{3,2}$ update their upper bound at the same time. Due to the precedence constraints and the QC's cycle time, they could not be picked up or dropped off by any vehicle later than $\bar{\varepsilon}_{y_2^{3,1}} - (3-s)\cdot\tau$ at the QC side.

If instead, $\varepsilon_{y_2^{3,2}} = \varepsilon_{y_2^{3,1}} = \varepsilon^*_{y_2^{3,1}} > \bar{\varepsilon}_{y_2^{3,1}}$, it states that even following the best of the latest delivery sequences, the deliveries of the two target containers still delay the operations of container $c_2^{3,1}$ and $c_2^{3,2}$ so much that the current best makespan is postponed. Then those two delivery sequences $(c_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1},\ c_2^{3,1})$ and $(c_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2},\ c_2^{3,2})$ are added to the set of *permanent cut-off delivery sequences* (denoted as Ω_2^s). Logically, since it is too late to transport the container $c_2^{3,1}$ or $c_2^{3,2}$ immediately after the container $c_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}$ and $c_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}$, it is reasonable to forbid them or the containers operated before them by q_2 to be transported immediately after the containers that are operated after $c_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}$ or $c_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}$. Furthermore, all these corresponding delivery sequences are defined as the permanent cut-off delivery sequence, as well. Distinct from those temporary cut-off sequences introduced above, the permanent cut-off sequences are permanently removed from the search spaces of following sub-problems from this point on. Note that this is a sequential heuristic approach and we are not guaranteeing that the overall optimal solution of the problem cannot include any one permanent cut-off delivery sequence.

Lower bound sub-problems are constructed and solved in a similar fashion; indeed, the only difference proves to be whether or not the earliest delivery sequences of target containers affect other QCs' operations excessively, such that the current best makespan is postponed. Refer to FIGURE 11 (b): when container $c_2^{3,1}$ is transported immediately after $c_1^{2,2}$ the latest arrival time is earlier than the lower bound of its current time window, i.e. $\bar{\alpha}_{y_2^{3,1}} < \underline{\varepsilon}_{y_2^{3,1}}$. Meanwhile, if $c_2^{3,1}$ is transported immediately after $c_1^{3,2}$ by the same vehicle, the vehicle's arrival time does not violate the current lower bound of the starting time, i.e. $\bar{\alpha}_{y_2^{3,1}} \geq \underline{\varepsilon}_{y_2^{3,1}}$. Then the container $c_1^{3,2}$ is added into the set of the earliest delivery sequence F_2^3. After determining the earliest delivery sequences of $c_2^{3,1}$ and $c_2^{3,2}$, the lower bound sub-problem is constructed through addition of extra constraints (28)-(31) to the original MILP model.

If the optimized results of the lower bound sub-problem reflect that the target containers' earliest delivery sequences do not delay other QCs' operations, the optimized starting time is employed to update the lower bound of the target containers' QC side tasks' starting time. At the same time, the containers operated after these target containers update their lower bounds of starting times. Otherwise, if the results state that the target containers are transported too early and that other QCs' operations are delayed so much that the current best makespan is postponed as a result, their earliest delivery sequences are defined as permanent cut-off delivery sequences. The containers operated after these target containers cannot be transported earlier than those earliest delivery sequences.

TABLE 4 Extra Constraints Imposed to a Sub-Problem

Extra constraints added to the upper bound sub-problem of $\bar{\varepsilon}_{y_k^{s,g}}$	
$\sum_{c_k^{\hat s,\hat g} \in L_k^s} \sum_{g=1}^{2} x_{d_{\hat k}^{\hat s,\hat g}, p_k^{s,g}} = 2$	(24)
$x_{d_{\hat k}^{\hat s,\hat g}, p_{k'}^{s',g'}} = 0 \ , \ \left(c_{\hat k}^{\hat s,\hat g} \ , \ \mathrm{c}_{k'}^{s',g'}\right) \in \Theta_k^s$	(25)
$x_{d_{\hat k}^{\hat s,\hat g}, p_{k'}^{s',g'}} = 1 \ , \ \left(c_{\hat k}^{\hat s,\hat g} \ , \ \mathrm{c}_{k'}^{s',g'}\right) \in \Phi_k^s$	(26)
$\pi >= \ \varepsilon_{y_k^{s,g}} , \ k \in Q, s \in \lvert S_k \rvert$	(27)
Extra constraints added to the lower bound sub-problem of $\underline{\varepsilon}_{y_k^{s,g}}$	
$\sum_{c_k^{\hat s,\hat g} \in F_k^s} \sum_{g=1}^{2} x_{d_k^{\hat s \bar k,\hat g}, p_k^{s,g}} = 2$	(28)
$x_{d_{\hat k}^{\hat s,\hat g}, p_{k'}^{s',g'}} = 0 \ , \ \left(c_{\hat k}^{\hat s,\hat g} \ , \ \mathrm{c}_{k'}^{s',g'}\right) \in \Theta_k^s$	(29)
$x_{d_{\hat k}^{\hat s,\hat g}, p_{k'}^{s',g'}} = 1 \ , \ \left(c_{\hat k}^{\hat s,\hat g} \ , \ \mathrm{c}_{k'}^{s',g'}\right) \in \Phi_k^s$	(30)
$\pi >= \ \varepsilon_{y_k^{s,g}} , \ k \in Q, s \in \lvert S_k \rvert$	(31)

5.2.4 Analysis of the Sub-problem Result

Due to time constraints, not every sub-problem can be solved to optimality. Therefore, it is important to analyze the results and extract as much information as possible to update the time windows and the permanent cut-off delivery sequences. This procedure is referred to as the analysis of the sub-problem result. The first goal of the process is to determine whether or not the solution is eligible for further consideration. If it is, the analysis is to determine whether or not the target containers $c_k^{s,1}$ and $c_k^{s,2}$ are allowed to be delivered following the current optimized delivery sequences without worsening the current optimal makespan. The time window or the set of permanent cut-off delivery sequences are updated accordingly; otherwise, the results are discarded directly. The complete process is presented in TABLE 5 and TABLE 6

Upper Bound Sub-problem

The core mission of the upper bound sub-problem is to check whether or not the target container $c_k^{s,1}$ and $c_k^{s,2}$'s latest delivery sequences are late, delay the operations of q_k , and postpone current optimal makespan.

Assume that the optimal solution to the sub-problem is $\varepsilon_{y_k^{s,g}}*$. The results obtained within the time limit is $\varepsilon_{y_k^{s,g}}$ and the current delivery sequences are $(c_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}, c_k^{s,1})$ and $(c_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}, c_k^{s,2})$. If the sub-problem has been solved to optimal within the time limit, i.e. $\varepsilon_{y_k^{s,g}} = \varepsilon_{y_k^{s,g}}*$, the algorithm updates the time window or the set of permanent cur-off delivery sequences following the rules introduced in the last part. If the sub-problem is not solved to the optimal within the time limit, it is possible that $\varepsilon_{y_k^{s,g}} > \varepsilon_{y_k^{s,g}}*$. Therefore, even if $\varepsilon_{y_k^{s,g}} > \bar{\varepsilon}_{y_k^{s,g}}$ holds, we cannot conclude that the containers $c_k^{s,1}$ and $c_k^{s,2}$ are transported too late following their latest delivery sequences. To avoid the exclusion of potential good solutions due to the possibly inaccurate estimation of actual starting time and time windows, an alternative criterion is utilized in making the decision. Note that $\pi * -(|S_k| - s) \cdot \tau$ is the absolute upper bound for any QC side task $y_k^{s,g}$ because of the precedence relationship. For the solution to the upper bound sub-problem, there are three cases:

(1) $\varepsilon_{y_k^{s,g}} > \pi * -(|S_k| - s) \cdot \tau$

$\varepsilon_{y_k^{s,g}}$ has exceeded its upper bound so much that the makespan is necessarily delayed. The current upper bound $\bar{\varepsilon}_{y_k^{s,g}}$ shall not be updated based on this optimization result. However, whether or not the current delivery sequences $(c_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}, c_k^{s,1})$ and $(c_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}, c_k^{s,2})$ should be cut forever from the search space as permanent cut-off sequences still requires more discussion.

(a) If the starting time of $y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}$ or $y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}$ is also later than its absolute upper bound ($\varepsilon_{y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}} > \pi * -(|S_{\tilde{k}_1}| - \tilde{s}_1) \cdot \tau$ or $\varepsilon_{y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}} > \pi * -(|S_{\tilde{k}_2}| - \tilde{s}_2) \cdot \tau$), it is highly possible that the size of the sub-problem is too large to be solved to optimal within the short time limit. Hence, the results are viewed as ineligible for any update.

(b) If both $y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}$ and $y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}$ start earlier than their absolute upper bounds ($\varepsilon_{y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}} \le \pi^* - (\left|S_{\tilde{k}_1}\right| - \tilde{s}_1)\cdot\tau$ and $\varepsilon_{y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}} \le \pi^* - (\left|S_{\tilde{k}_2}\right| - \tilde{s}_2)\cdot\tau$), the upper bound sub-problem is viewed as solved to near-optimal. The solution may be very close, or even the same as the optimal solution. In such a case, we determine that it is too late to transport the containers $c_k^{s,1}$ and $c_k^{s,2}$ in the current delivery sequences and their succeeding operations cannot be finished before their absolute upper bounds. Then the algorithm updates the set of permanent cur-off delivery sequences accordingly.

(2) $\varepsilon_{y_k^{s,g}} \le \pi^* - (|S_k| - s)\cdot\tau$

$\varepsilon_{y_k^{s,g}}$ does not exceed its absolute upper bound and the makespan is not necessarily delayed. Thus the current delivery sequences are not defined as the permanent cut-off sequences. However, whether or not the optimized starting time can be used to update $\bar{\varepsilon}_{y_k^{s,g}}$ still necessitates further discussion.

(a) If both $y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}$ and $y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}$ start later than their absolute upper bounds, it is highly possible that the size of the sub-problem is too large to be solved to optimal within the time limit. Hence, the results are viewed as ineligible to update the time window.

(b) If at most only one task, $y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}$ or $y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}$, starts later than its absolute upper bound, it implies that the current best delivery sequences of the containers $c_k^{s,1}$ and $c_k^{s,2}$ do not necessarily postpone the current best makespan $\pi*$; thus, the algorithm updates the upper bound using the optimized starting time, i.e. $\bar{\varepsilon}_{y_k^{s,g}} = \varepsilon_{y_k^{s,g}}$. At the same time, all the containers operated before the target containers by q_k should check, and update if necessary, the upper bound of their QC side tasks' starting times according to the precedence relationship using equation (32).

$$\bar{\varepsilon}_{y_k^{s',g}} = \min(\bar{\varepsilon}_{y_k^{s',g}}, \bar{\varepsilon}_{y_k^{s'+1,g}} - \tau), s' < s \qquad (32)$$

(c) If $\pi < \pi*$ and all containers' delivery sequences have been optimized in the sub-problem, π is the makespan of the entire problem. Thus, if the makespan π obtained from the optimization result is better than the current best solution $\pi*$, then $\pi* = \pi$.

TABLE 5 Analysis of an Upper Bound Sub-problem

Analysis of the upper bound sup-problem of $\bar{\varepsilon}_{y_k^{s,g}}$

```
if $\varepsilon_{y_k^{s,g}} \le \pi* - (|S_k| - s) \cdot \tau$ then
    if $\varepsilon_{y_{\tilde{k}_1}^{\tilde{s}1,\tilde{g}1}} > \pi* - (|S_{\tilde{k}_1}| - \tilde{s}_1) \cdot \tau$ and $\varepsilon_{y_{\tilde{k}_2}^{\tilde{s}2,\tilde{g}2}} > \pi* - (|S_{\tilde{k}_2}| - \tilde{s}_2) \cdot \tau$ then
        go to the next sub-problem;
    else
        $\bar{\varepsilon}_{y_k^{s,g}} = \varepsilon_{y_k^{s,g}}$
```

TABLE 5 Continued

```
        $\bar{\varepsilon}_{y_k^{s',g}} = \min(\bar{\varepsilon}_{y_k^{s',g}}, \bar{\varepsilon}_{y_k^{s'+1,g}} - \tau), s' < s$
        if $\pi < \pi^*$ and all containers' delivery sequences have been optimized in the sub-problem then
            $\pi^* = \pi$;
            $\bar{\varepsilon}_{y_{k'}^{s',g'}} = \min(\bar{\varepsilon}_{y_{k'}^{s',g'}}, \bar{\varepsilon}_{y_{k'}^{s'+1,g'}} - \tau)$ where $k' \in Q,\ s', s'+1 \in S_{k'}$;
        end if
    end if
end if
if $\varepsilon_{y_k^{s,g}} > \pi^* - (|S_k| - s) \cdot \tau$ then
    if $\varepsilon_{y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}} \le \pi^* - (|S_{\tilde{k}_1}| - \tilde{s}_1) \cdot \tau$ and $\varepsilon_{y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}} \le \pi^* - (|S_{\tilde{k}_2}| - \tilde{s}_2) \cdot \tau$ then
        remove $(c_{\tilde{k}_1}^{\tilde{s}',\tilde{g}}, c_k^{s',g})$ from the search space as permanent cut-off delivery sequences, where $\tilde{s}' \ge \tilde{s}_1$ and $s' \le s$;
        remove $(c_{\tilde{k}_2}^{\tilde{s}',\tilde{g}}, c_k^{s',g})$ from the search space as permanent cut-off delivery sequences, where $\tilde{s}' \ge \tilde{s}_2$ and $s' \le s$;
    else
        go to the next sub-problem;
    end if
end if
```

Lower Bound Sub-problem

For a lower bound sub-problem, the optimization result is to decide whether or not the containers $c_k^{s,1}$ and $c_k^{s,2}$ are transported too early such that the other QCs' operations are delayed due to lack of accessibility to the vehicles. Now, assume that the lower bound sub-problem of $\underline{\varepsilon}_{y_k^{s,g}}$ has been solved within a short time limit. The possible situations raised are:

(1) $\varepsilon_{y_k^{s,g}} > \overline{\varepsilon}_{y_k^{s,g}}$

According to the principle in constructing the lower bound sub-problem, this scenario should be very rare. It is highly possible that the size of the sub-problem is too large to be solved to optimal in the time limit. Therefore, the results are ignored without any further analysis. Here we use $\overline{\varepsilon}_{y_k^{s,g}}$ instead of $\pi^* - (|S_k| - s)\cdot\tau$ to avoid the situation that might result in the mistake in seeking cut-off delivery sequences.

(2) $\varepsilon_{y_k^{s,g}} \le \overline{\varepsilon}_{y_k^{s,g}}$

(a) If both $y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}$ and $y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}$ start later than their absolute upper bounds ($\varepsilon_{y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}} > \pi^* - (|S_{\tilde{k}_1}| - \tilde{s}_1)\cdot\tau$ and $\varepsilon_{y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}} > \pi^* - (|S_{\tilde{k}_2}| - \tilde{s}_2)\cdot\tau$), the implication is that containers $c_k^{s,g}$ are transported too early and that other QCs' operations are delayed so much that the current optimal makespan is postponed as a result. Thus, the algorithm updates the set of permanent cut-off delivery sequences according to the results.

(b) the implication is that the current delivery sequences do not affect the operations of $q_{\tilde{k}_1}$ or $q_{\tilde{k}_2}$. Hence, the results are eligible to update the lower bound, i.e., $\underline{\varepsilon}_{y_k^{s,g}} = \varepsilon_{y_k^{s,g}}$. At the same time, all the containers operated after the target container check and update their lower bounds according to the equation (33)

$$\underline{\varepsilon}_{y_k^{s',g}} = \max(\underline{\varepsilon}_{y_k^{s'-1,g}} + \tau, \underline{\varepsilon}_{y_k^{s',g}}) \text{ where } s' > s \tag{33}$$

TABLE 6 Analysis of a Lower Bound Sub-problem

```
Analysis of the lower bound sub-problem of $\underline{\varepsilon}_{y_k^{s,g}}$
if $\varepsilon_{y_k^{s,g}} \le \bar{\varepsilon}_{y_k^{s,g}}$ then
    if $\varepsilon_{y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}} \le \bar{\varepsilon}_{y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}}$ or $\varepsilon_{y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}} \le \bar{\varepsilon}_{y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}}$ then
        $\underline{\varepsilon}_{y_k^{s,g}} = \varepsilon_{y_k^{s,g}}$;
        $\underline{\varepsilon}_{y_k^{s',g}} = \max(\underline{\varepsilon}_{y_k^{s'-1,g}} + \tau, \underline{\varepsilon}_{y_k^{s',g}})$ where $s' > s$;
    end if
    if $\varepsilon_{y_{\tilde{k}_1}^{\tilde{s}_1,\tilde{g}_1}} > \pi^* - (|S_{\tilde{k}_1}| - \tilde{s}_1)\cdot\tau$ and $\varepsilon_{y_{\tilde{k}_2}^{\tilde{s}_2,\tilde{g}_2}} > \pi^* - (|S_{\tilde{k}_2}| - \tilde{s}_2)\cdot\tau$ then
        remove $(c_{\tilde{k}_1}^{\tilde{s}',\tilde{g}}, c_k^{s',1})$ from the search space as permanent cut-off delivery sequences,
        where $\tilde{s}' \le \tilde{s}_1$ and $s' \ge s$;
        remove $(c_{\tilde{k}_2}^{\tilde{s}',\tilde{g}}, c_k^{s',2})$ from the search space as permanent cut-off delivery sequences,
        where $\tilde{s}' \le \tilde{s}_2$ and $s' \ge s$;
    end if
else
        go to the next sub-problem;
end if
```

5.2.5 Process of Basic Scheme

Combining the parts introduced above, the basic scheme of the proposed LSC algorithm is illustrated in TABLE 7.

TABLE 7 Process of the Basic Scheme for Local Sequence Cut Algorithm

The Basic Scheme for Local Sequence Cut Algorithm

```
for each transport request $r_k^s$ do
    seek the temporary cut-off delivery sequences that violate the current time bounds
    and determine the set $L_k^s$, $\Theta_k^s$, and $\Phi_k^s$;
    construct the upper bound sub-problem of $\overline{\varepsilon}_{y_k^{s,g}}$;
    solve the sub-problem;
    analyze the results of the upper bound sub-problem;
end for
for each transport request $r_k^s$ do
    seek the temporary cut-off delivery sequences that violates the current time bounds
    and determine the set $F_k^s$, $\Theta_k^s$, and $\Phi_k^s$;
    construct the lower bound sub-problem of $\underline{\varepsilon}_{y_k^{s,g}}$;
    solve the sub-problem;
    analyze the results of the lower bound sub-problem;
end for
introduce all cut-off sequences into the original MILP model and set all corresponding
decision variables $x$ to be 0;
solve the original MILP model.
```

5.3 Enhanced Scheme

As the problem size increases, even solving the sub-problems becomes time-consuming. In addition to the basic scheme, it is necessary to develop several enhanced schemes for improving the performance of the algorithm.

5.3.1 Treatment at Early Stages

During the first iterations, the time windows are usually too loose to generate enough cut-off delivery sequences, and there are always too many containers that can be delivered before or after the target containers without violating the time windows. As a result, the size of the sub-problem is always too large for the exact method. Thus, it is reasonable to regulate that the delivery sequence $(c_{\tilde{k}}^{\tilde{s},\tilde{g}}, c_k^{s,g})$ to be a cut-off sequence if

$$abs\ (\tilde{s} - s) > floor(\ |\mathrm{V}| \ / \ |Q|\ ) \tag{34}$$

5.3.2 Size Limit on Sub-problems

If the size of a sub-problem is too large for the MILP solver, the solver may not find any feasible integer solution or only finds a bad feasible solution whose result is not eligible for any update. To avoid either of the cases, the size of the sub-problem should be examined before its construction. For each sub-problem in the basic scheme, a threshold is imposed on the number of the decision variables that have been decided according to the set Θ_k^s and Φ_k^s. If the minimum threshold cannot be met, the algorithm directly skips to the next sub-problem.

One should note that the upper and lower bounds of starting times have different effects on searching solution. A relatively higher upper bound does not exclude a desirable delivery sequence as a cut-off sequence while a relatively higher lower bound may remove “good” sequences in the succeeding searching processes. To guarantee a higher quality of the solution, a stricter limit is imposed on the problem size for the lower bound sub-problems than that for the upper bound sub-problems. Moreover, we point out here that because the computation speed of the LSC algorithm is greatly determined by the updates of time windows, the computation might be accelerated along with the increasing vehicles.

5.4 Overall Process of LSC Algorithm

After adding the enhanced schemes, the overall process of LSC algorithm is stated in TABLE 8.

TABLE 8 Entire Process of Local Sequence Cut Algorithm

```
input initial upper and lower bounds;
do while not all the $\overline{\varepsilon}_{y_k^{s,g}}, k \in Q, s \in |S_k|$ have been updated or the maximum number of iteration has not been reached;
        for each transport request $r_k^s$ do
```

TABLE 8 Continued

```
            determine the set $L_k^s, \Theta_k^s$ and $\Phi_k^s$;
            if the sub-problem size is smaller than the threshold then
              construct the upper bound sub-problem of $\overline{\varepsilon}_{y_k^{s,g}}$;
              solve sub-problem within a time limit;
              if the solver cannot find any feasible solution then
                go to the next sub-problem;
              else
                analyze the results of the upper bound sub-problem;
              end if
            end if
          end for
          for each transport request $r_k^s$ do
            determine the set $F_k^s, \Theta_k^s$ and $\Phi_k^s$;
            if the sub-problem size is smaller than the threshold then
              construct the lower bound sub-problem of $\underline{\varepsilon}_{y_k^{s,g}}$;
              solve the sub-problem within a time limit;
              if the solver cannot find any feasible solution then
                go to the next sub-problem;
              else
                analyze the results of the lower bound sub-problem;
              end if
            end if
          end for
end do while
introduce all cut-off sequences into the original MILP model and set all corresponding decision variables $X$ to be 0;
solve the MILP model within a time limit;
if $\pi < \pi^*$ then
```

TABLE 8 Continued

```
    $\pi^* = \pi$
end if
```

5.5 Numerical Experiments

In this section, we evaluate the performance of LSC algorithm against the branch-and-bound method realized by CPLEX. All the runs are performed using CPLEX 12.1 with default setting and C++ in a 3.30 GHz CPU with 4.0 GB RAM. The QC's cycle time uses the mode value in the empirical distribution listed in TABLE 1. The handle time at the QC/YC includes the necessary time that a QC/YC lifts/retrieves a container onto/from a vehicle, as well as the extra travel time because of the vehicle's deceleration as it approaches and leaves a QC/YC (*39*). The average waiting time of vehicles at the YC side uses the historical value in the NCT, also consistent with the value in the study of Lee and Kim (*46, 47*). It includes the waiting time of the pass permit before the vehicle enters the passenger lane and the possible waiting for availability of the YC.

5.5.1 LSC v.s. Optimality

To illustrate the LSC method and test its performance, this section presents a series of numerical experiments. To allow for statistically meaningful results, for each problem size 10 cases were executed. The results are presented herein.

We compare the performance of the LSC method against the MILP model, directly solving by the ILOG-CPLEX in terms of objective value and the computation time. To ensure that the CPLEX finds the optimal solution, we first compared the two methods in twenty small size instances, listed in TABLE 9. The scale of a case is represented by the numbers of the case number. For example, 24-9-1 implies that there are in total twenty-four containers, 48 pick-up and drop-off tasks, and nine vehicles in the model; and the last number 1 denotes the first case of the same problem size. All the working sequences of QCs and the storage blocks of containers are randomly generated by the program. The second and third columns present the objective values found by the CPLEX and the proposed LSC method, respectively.

The results listed in TABLE 9 indicate that the LSC method is able to find the optimal solution in most cases. For the cases with twenty-four containers and 9 vehicles, both methods solve the MILP model very quickly. The problem is solved to optimal via the LSC method in 9 out of 10 cases. Even in the only exception, i.e., case 1, the solution gap is marginal compared to the optimal one. When there are 36 containers and 12 vehicles in the model, the number of binary decision variables increases significantly from 595 to 1326. Thus, the CPLEX needs up to 1.5 hours to solve the problem to optimal. However, the proposed LSC method solves the problem within 15 minutes and finds the optimal solution within such a short time in 6 out 10 cases. The largest gap between the optimal solutions is only about 3.4% and the average gap is smaller than 1%.

Next, the proposed LSC algorithm is computationally tested in a large test bed, where the instances are considered rigorously, using exact methods. In TABLE 10, the computational times of the CPLEX and LSC method are illustrated in the columns with the name T_CPLEX and T_LSC, respectively. We allow the CPLEX to work for a time limit up to 10 hours. The best integer solutions found by the CPLEX within the time limit are used as the benchmark for assessing the LSC algorithm. The objective of the solution and computational time of the LSC method are revealed in the columns named LSC and T_LSC, respectively.

TABLE 9 LSC v.s. Optimality (Small Size Problems)

Case	CPLEX	LSC	Gap (%)
24-9-1	988	998	1.012
24-9-2	1049	1049	0.000
24-9-3	1006	1006	0.000
24-9-4	984	984	0.000
24-9-5	1088	1088	0.000
24-9-6	770	770	0.000
24-9-7	925	925	0.000
24-9-8	1102	1102	0.000
24-9-9	1018	1018	0.000
24-9-10	923	923	0.000
average			**0.101**
36-12-1	1191	1191	0.000
36-12-2	1020	1044	2.353
36-12-3	1182	1182	0.000
36-12-4	1096	1108	1.095
36-12-5	1103	1103	0.000
36-12-6	1121	1128	0.624
36-12-7	958	958	0.000
36-12-8	1127	1127	0.000

TABLE 9 Continued

Case	CPLEX	LSC	Gap (%)
36-12-9	1098	1098	0.000
36-12-10	1108	1145	3.339
average			**0.741**

Note: CPLEX – the optimal solution found by CPLEX-OLOG;
LSC – the best solution found by LSC method;
Gap – (LSC – CPLEX) / CPLEX * 100%

In 16 out of 40 cases, the LSC method finds a better solution than the CPLEX. The objective values obtained from the LSC method is 11.111% lower in the best case and 6.397% higher in the worst case, compared with the CPLEX. The average gap between the two methods does not exceed 4%. However, the computation time consumed by the LSC method is only 1/5 to 1/4 of the CPLEX.

TABLE 10 LSC v.s. Optimality (Large Size Problems)

Case No.	Best Integer	Best Node	T_CPLEX	LSC	T_LSC	Gap
72-15-1	2268	1354	10hr	2016	100min	-11.111
72-15-2	2045	1357	10hr	2031	100min	-0.685
72-15-3	1980	1326	10hr	1882	100min	-4.949
72-15-4	2050	1288	10hr	2040	100min	-0.488
72-15-5	2124	1314	10hr	2076	100min	-2.260
72-15-6	2074	1307	10hr	2114	100min	1.929
72-15-7	2065	1186	10hr	2052	100min	-0.630
72-15-8	2347	1357	10hr	2314	100min	-1.406
72-15-9	2154	1308	10hr	1940	100min	-9.935
72-15-10	2225	1337	10hr	2078	100min	-6.607
average						**-3.614**
72-18-1	1676	1250	10hr	1536	90min	-8.353
72-18-2	1625	1349	10hr	1685	90min	3.692

TABLE 10 Continued

Case No.	Best Integer	Best Node	T_CPLEX	LSC	T_LSC	Gap
72-18-3	1783	1390	10hr	1772	90min	-0.617
72-18-4	1715	1492	10hr	1683	90min	-1.866
72-18-5	1726	1436	10hr	1754	90min	1.622
72-18-6	1628	1434	10hr	1662	90min	2.088
72-18-7	1850	1409	10hr	1852	90min	0.108
72-18-8	1562	1504	10hr	1596	90min	2.177
72-18-9	1794	1450	10hr	1802	90min	0.446
72-18-10	1779	1376	10hr	1779	90min	1.124
average						**0.042**
72-21-1	1490	1410	10hr	1524	40min	2.282
72-21-2	1569	1555	10hr	1577	40min	0.510
72-21-3	1613	1498	10hr	1627	40min	0.868
72-21-4	1488	1474	10hr	1489	40min	0.067
72-21-5	1445	1417	10hr	1445	40min	0.000
72-21-6	1476	1449	10hr	1516	40min	2.710
72-21-7	1574	1556	10hr	1600	40min	1.652
72-21-8	1424	1417	10hr	1424	40min	0.000
72-21-9	1509	1315	10hr	1533	40min	2.916
72-21-10	1499	1487	10hr	1517	40min	1.201
average						**1.221**
90-21-1	1932	1635	10hr	1932	2hrs	0.000
90-21-2	2006	1608	10hr	1946	2hrs	-2.991
90-21-3	1901	1782	10hr	1973	2hrs	3.787
90-21-4	1780	1711	10hr	1816	2hrs	2.022
90-21-5	2037	1605	10hr	1915	2hrs	-5.989
90-21-6	2059	1571	10hr	1954	2hrs	-5.100
90-21-7	1696	1692	10hr	1728	2hrs	1.887
90-21-8	2079	1629	10hr	2212	2hrs	6.397
90-21-9	2062	1268	10hr	2036	2hrs	-1.261
90-21-10	1913	1599	10hr	1987	2hrs	2.265
average						**-0.139**

Note: Best Node – the best non-integer solution found by the CPELX

Another important observation is that for the problems that consist of the same number of containers, the more vehicles there are, the faster the problem is solved. The main reason is because more vehicles lead to the relatively lower upper bounds of starting times. Since the construction of a sub-problem is dependent on the range of time windows, the lower upper bound is undoubtedly helpful in estimating the starting times more precisely and controlling the size of sub-problem effectively. Consequently, they accelerate the solution of sub-problems and shorten the total CPU time of the algorithm.

5.5.2 Time Windows and Cut-off Delivery Sequences

At the end of the algorithm, the CPLEX is allowed to solve the original MILP model within only 20 minutes. Its solution is comparable to the one found by the CPLEX alone running 10 hours. The success is attributed to the introduction of those cut-off delivery sequences defined by the algorithm. According to the numerical experiments, at the end of the algorithm, the binary decision variables of the original MILP model have been halved before the CPLEX starts to solve it. To state how the cut-off delivery sequences and the time bounds are updated through iteration, we take one case 72-18-4 as an instance.

FIGURE 12 plots the number of cut-off delivery sequences eliminated from the search space of the sub-problems. They are comprised of those temporary cut-off delivery sequences according to the current time windows and the permanent cut-off delivery sequences decided during earlier iteration. Here, we only record the sub-problems whose sizes satisfy the pre-defined threshold, as only these would proceed to be solved by the CPLEX. There are a total of 151 sub-problems constructed and solved during iteration. We notice that the number of cut-off delivery sequences always keeps increasing when the algorithm is solving upper bound sub-problems. When the algorithm starts to construct and solve lower bound sub-problems, the number of cut-off delivery sequences reaches relative stability or increases very slowly. This implies that the update of starting times' lower bounds is lesser than upper bounds. That is because (1), in constructing a lower bound sub-problem, the target containers are forced to be transported following their earliest delivery sequences. As a result, the optimized starting time of the corresponding QC side task is the same as its current lower bound; and (2), to avoid excluding potential "good" delivery sequences from the search spaces, the criteria in updating time windows or the set of permanent delivery sequences is stricter than analysis of the results of upper bound sub-problems. FIGURE 12 implies that the upper bound sub-problems are more critical to the update of time windows and the set of permanent cut-off delivery sequences.

To present how the time windows are updated through iteration, FIGURE 13 records the time window of all containers operated by the same QC. Because the two containers operated in the same tandem lift have the same time window for their QC side tasks, each bar in FIGURE 13 represents the time window of those two QC side tasks' starting time. The lines above and below the bars are the upper and lower bound of the time windows.

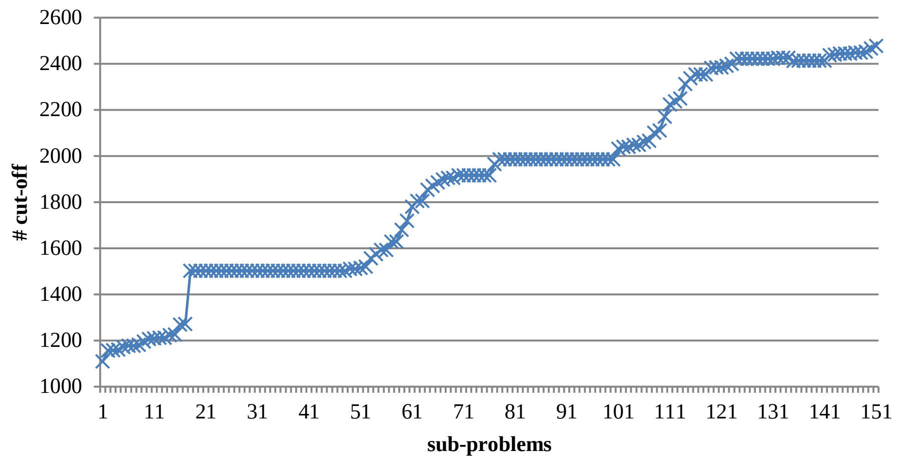
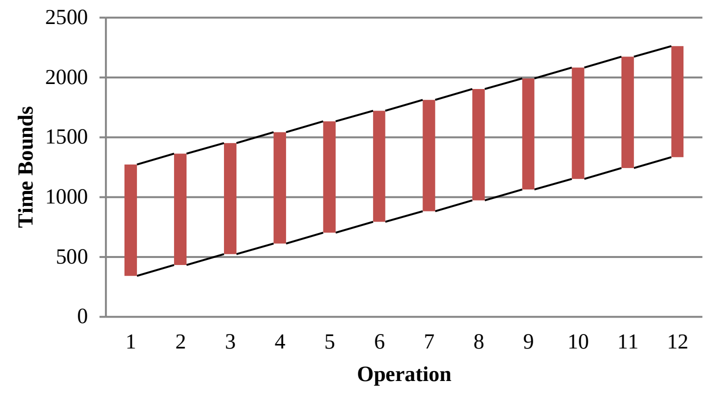
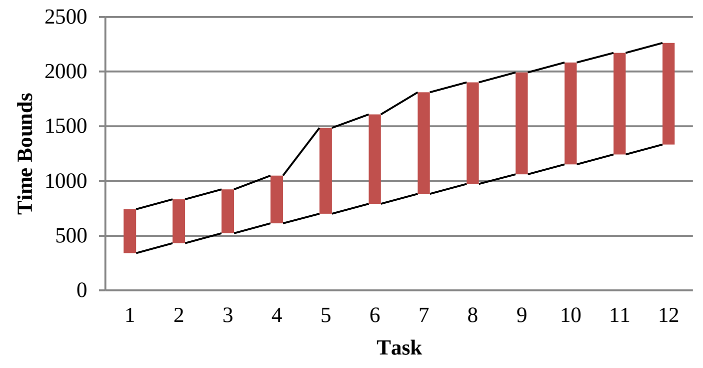


FIGURE 12 Number of cut-off delivery sequences through iteration.

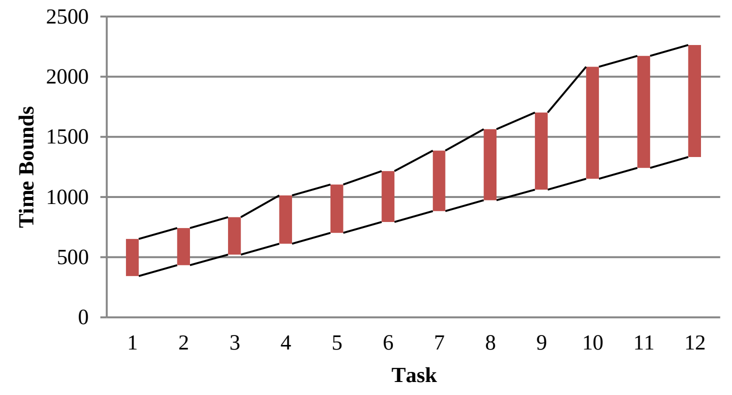
FIGURE 13(a) is the original time windows calculated using the initial solution and the QC's cycle time. The initial time windows are very loose and they share the same ranges. During the first iteration, only the first few time windows are updated, while others remain the same, because most sub-problems are not solved by the CPLEX due to their problem sizes. When more time windows are updated, more sub-problems satisfy the pre-defined threshold and are solved by the CPLEX. FIGURE 13(c) – (d) show very clearly that the time windows are narrowed greatly through the next two iterations. When the third iteration ends, all starting times have been updated. The green triangles plotted in FIGURE 13(d) represent the starting times in the final solution, all visibly positioned within the final time windows. It proves that the effectiveness of LSC algorithm in estimating the time windows.



(a) The initial time windows



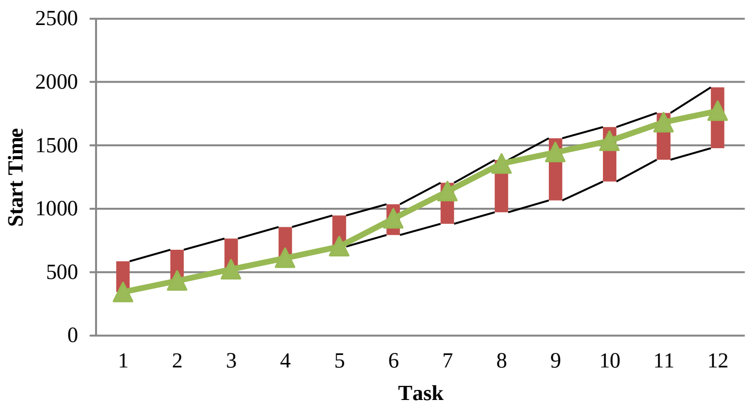
(b) After the first iteration



(c) After the second iteration

FIGURE 13 Update of time bounds through iterations.

FIGURE 13 Continued



(d) After the third iteration

6. MODIFIED LSC ALGORITHM

The numerical experiments in the last section have shown that the proposed LSC algorithm is capable of reducing the search space effectively by cutting off the delivery sequences based on the estimation of time windows. However, the original algorithm is still not efficient enough to handle the problems of larger size. The reasons can be summarized as follows:

(1) According to the analysis in previous section, the updates of upper bounds play a key role in the algorithm. It determines the quality and the speed of the algorithm significantly. Unfortunately, the upper bounds of many starting times cannot be updated at the beginning stage. Such a disadvantage is more crucial when the operations increase. The algorithm needs more iteration to update the time windows of all QC side tasks' starting times.

(2) Contrary to those sub-problems not constructed or solved due to their large sizes, there are some other sub-problems are constructed and solved repeatedly through iteration without contributing any new information to the algorithm. The set of the latest or earliest delivery sequences are the same as those in the last iteration; the optimized starting time of sub-problem is the same as the current upper or lower bound. In such cases, these sub-problems are defined as *meaningless sub-problems*, because the algorithm does not benefit from their solutions but consume CPU time only.

To solve the above two problems and further accelerate computation, this section is dedicated to propose the modified LSC algorithm. The solution to the first problem is to propose a heuristic method as an alternative to the CPLEX to solve the sub-problems, especially when the sub-problem size does not satisfy the pre-defined threshold. In addition, its computation time should be more competitive than the CPLEX and not affected significantly by the problem size as the CPLEX. The solution to the second problem is to develop a set of selection mechanisms to prevent the algorithm from constructing and solving those meaningless sub-problems. In summary, the employment of the heuristic method is to accelerate the update of the time windows, especially the upper bounds, of the starting times. The introduction of the selection mechanism is to detect the meaningless sub-problems and prevent wasting CPU time on them. Thus, the modified LSC method is expected to quicken the searching process significantly compared to the original algorithm, without affecting the solution quality. The following parts introduce the modified LSC method in more details.

6.1 Priority Based Heuristic Method

To provide a more computationally efficient method to solve the sub-problem, a heuristic method is proposed and used to solve the upper bound sub-problems without any size limit. If the result is eligible to update the upper bound of the starting time, it will not be solved by the CPLEX again even if its size satisfies the pre-defined threshold. To avoid the elimination of good solutions, the result from the heuristic method would not be used to define the permanent cut-off sequences, even if it is eligible for that. Based on the similar consideration, the heuristic method is not used to solve the lower bound sub-problems. In addition, to accelerate the computation further, as long as one time window is updated, the algorithm solves the entire problem via the heuristic method. If the objective is better than the current best makespan, it is used as the current best makespan, and every starting time's upper bound is checked and updated if necessary according to the equation (16).

The heuristic method only considers the active transportation requests in making the decisions. An active request refers to the one whose predecessor requests have all been completed, and we can predict when the container is ready to be picked up or dropped off at the QC side. For example, if q_k 's $(s-1)^{th}$ operation is a loading (unloading) operation, then the request r_k^s is released and becomes active when q_k finishes lifting (positioning) the containers from (onto) the vehicles. Thus, at any moment, there are at most $|Q|$ active requests, and each of them needs two vehicles to transport the containers. An assignment of a transport request r_k^s to two vehicles v_m and v_n is denoted as $ass(r_k^s, v_m, v_n)$ here and after.

The mission of the algorithm is to determine which one is the best among all possible assignments. To deduce the assignment that benefits the most to minimize the makespan, a priority-based heuristic method is developed to measure the priorities of the assignments by comparing their attributes. Then the vehicles are dispatched according to the assignment with the highest priority. The consideration of the priority measurement includes the minimization of the QC's idle time and maximization of the vehicles' utilization.

6.1.1 Creation of Assignments

An active request r_k^s needs two vehicles to transport the target containers $c_k^{s,1}$ and $c_k^{s,2}$. The principles in constructing sub-problems are still applied in creating the assignments. If $\left(c_{k_1}^{s_1,g_1}, c_{k_2}^{s_2,g_2}\right)$ is a temporary cut-off delivery sequence according to the current time windows or a permanent cut-off delivery sequence according to the previous iteration, then the vehicle that is delivering container $c_{k_1}^{s_1,g_1}$ cannot be dispatched to transport container $c_{k_2}^{s_2,g_2}$ consecutively. At the same time, the two target containers $c_k^{s,1}$ and $c_k^{s,2}$ must be transported following one of their latest or earliest delivery sequences. All satisfying assignments of the request r_k^s are added to the set A_k^s. For the purpose of priority measurement, the following attributes of each assignment are calculated and recorded:

(1) QC idle time — denotes the QC's waiting time due to the late arrival of vehicles. When the vehicles arrive later than the time when a QC is ready to lift/position containers from/onto vehicles, the QC has to wait for them. It is obviously that the minimization of a QC's idle time is helpful to shorten the makespan and increase the productivity of the QC.

(2) Arrival time difference — denotes the difference between the two vehicles' arrival times at a QC. In such a case, both the QC and the vehicle, which

arrives earlier, have to wait for the other vehicle. It not only results in the QC's idle time but also affects the utilization of vehicles.

(3) Empty travel time — denotes the travel time of the vehicle when it is empty. After dropping off a container at the YC or QC side, the time an empty vehicle takes to travel to pick up the next container is defined as its empty travel time.

(4) Arrival time difference — denotes the difference between the two vehicles' arrival times at a QC. In such a case, both the QC and the vehicle, which arrives earlier, have to wait for the other vehicle. It not only results in the QC's idle time but also affects the utilization of vehicles.

(5) Free time — denotes the moment when the vehicle finishes dropping off its current container at the QC or YC side and becomes free to respond another transportation request.

(6) Remaining requests — denotes the number of succeeding requests after r_k^s according to q_k 's working sequence. The more remaining requests an active request has, the more its delay would affect that QC's succeeding operations and the final makespa.

6.1.2 Priority Measurement

After the construction of assignments, the priority measurement is conducted in two steps. In the first step, the comparison is among the assignments with the same active request and only the best ones are kept for the further consideration. The QC idle time and arrival time difference are chosen for the priority measurement in the first step. They are the two attributes that affect the makespan directly and reflect the characteristics of the tandem lift QC. The assignments with the shortest QC idle time and the smallest arrival time difference are favored over others. The priority measurement in the first step is illustrated in TABLE 11.

TABLE 11 Priority Measurement in the First Step

```
if $|A_k^s| > 1$
    keep the assignment with the shortest QC idle time then delete others;
end if
if $|A_k^s| > 1$
    keep the assignment with the shortest arrival time difference then delete others;
end if
```

After the comparison and filtration in the first step, only those best assignments are preserved for the measurement in the second step. However, the same vehicle may appear in the best assignments of different requests. To make the final decision, all assignments of different active requests are compared and set priorities in the second step. The choice of attributes and the order in which the attributes are compared mainly focus on the maximization of the vehicles' utilization and the degree to which the attributes affects the makespan. If more than one assignment has the equally highest priority at the end of the second step, the algorithm picks one randomly from them. The priority measurement in the second step is presented in TABLE 12.

TABLE 12 Priority Measurement in the Second Step

```
if $\sum_{k \in Q} |A_k^s| > 1$ then
    keep the assignments that have the shortest empty travel time and delete others;
end if
if $\sum_{k \in Q} |A_k^s| > 1$ then
    keep the assignments that have the most remaining requests and delete others;
end if
if $\sum_{k \in Q} |A_k^s| > 1$ then
    keep the assignments that have the shortest QC idle time and delete others;
end if
if $\sum_{k \in Q} |A_k^s| > 1$ then
    keep the assignments that have the earliest free time and delete others;
end if
if $\sum_{k \in Q} |A_k^s| > 1$ then
    keep the assignments that have the smallest arrival time difference and delete others;
end if
if $\sum_{k \in Q} |A_k^s| > 1$ then
    keep the assignments that have the smallest early arrival time and delete others;
end if
if $\sum_{k \in Q} |A_k^s| > 1$ then
    pick up one assignment randomly;
end if
if $\sum_{k \in Q} |A_k^s| = 1$ then
    dispatch vehicles according to the assignment with the highest priority;
end if
```

Assume that the assignment $ass\left(r_k^s, v_m, v_n\right)$ has the highest priority and the vehicles have been dispatched accordingly; the next request r_k^{s+1} is released and becomes active. Then the heuristic method repeats the process until the two target containers, whose starting times' upper bound is the concern of the sub-problem, have been transported. If the result is eligible for the update of the upper bound, the sub-problem will not be solved by the CPLEX again. Otherwise, the sub-problem is sent to the CPLEX if its size satisfies the pre-defined threshold.

6.2 Selection Mechanism

To avoid the waste of CPU time on the meaningless sub-problems, a selection mechanism is proposed in the modified algorithm. The mechanism is triggered before the construction of a sub-problem. In the n^{th} iteration, the upper bound sub-problem or lower bound sub-problem will be viewed as a meaningless sub-problem when all the following criteria are met:

(1) In the $(n-1)^{th}$ iteration, the upper (lower) bound sub-problem has been solved by the CPLEX and its result was eligible for the update of upper (lower) bound of the time window.

(2) In the $(n-1)^{th}$ iteration, the optimized starting time was the same as the upper (lower) bound before the construction of the sub-problem.

(3) In the n^{th} iteration, the set of the latest (earliest) delivery sequences of the target containers are the same as those in the $(n-1)^{th}$ iteration.

If an upper bound or lower bound sub-problem is detected to be a meaningless sub-problem, the algorithm skips to the next sub-problem directly without constructing or solving it at all. The overall process of the modified algorithm is illustrated in TABLE 13.

TABLE 13 Process of Modified LSC Algorithm

```
for each transport request $r_k^s$ do
    if the upper bound sub-problem of $\overline{\varepsilon}_{y_k^{s,g}}$ is not a meaningless sub-problem then
        determine the set $L_k^s$, $\Theta_k^s$ and $\Phi_k^s$;
        construct the upper bound sub-problem;
        solve the sub-problem using priority-based heuristic method;
        analyze the results of the upper bound sub-problem;
        if the result is not eligible to update the upper bound and the sub-problem satisfies
          the pre-defined threshold then
            solve the sub-problem using the CPLEX;
            analyze the result of the upper bound sub-problem;
        end if
    end if
end for
solve the entire problem using priority-based heuristic method based on the current
upper and lower bounds;
if $\pi < \pi^*$ then
    $\pi^* = \pi$
end if
for each transport request $r_k^s$ do
    if the upper bound sub-problem of $\underline{\varepsilon}_{y_k^{s,g}}$ is not a meaningless sub-problem then
```

TABLE 13 Continued

```
            determine the set $F_k^s, \Theta_k^s$ and $\Phi_k^s$;
            construct the lower bound sub-problem;
            solve the sub-problem;
            analyze the result of the lower bound sub-problem;
        end if
end for
solve the problem using priority-based heuristic method based on the current upper and lower bounds;
if $\pi < \pi^*$ then
    $\pi^* = \pi$
end if
add all cut-off sequences into the original MILP model and set all corresponding decision variables $x$ to be 0;
solve the original MILP model using the CPLEX
```

6.3 Numerical Experiment Results

To test the effectiveness of the proposed heuristic method and the selection mechanism, a series of numerical experiments are conducted. The solution and the computation time of the modified algorithm are compared with the original algorithm introduced in the last section and the branch-and-bound method realized using the CPLEX.

TABLE 14 Numerical Experiment Results of Different Methods

	CPLEX	LSC	Mod LSC			Gap (%)	
	Obj	Obj	Obj	time	Cut (%)	CPLEX	LSC
72-15-1	2268	2016	2028	30min	29.137	-10.582	0.595
72-15-2	2045	2031	2103	30min	38.050	2.836	3.545
72-15-3	1980	1882	1954	30min	58.275	-1.313	3.826
72-15-4	2050	2040	2014	30min	37.984	-1.756	-1.275
72-15-5	2124	2076	2052	30min	37.802	-3.390	-1.156
72-15-6	2074	2114	2068	30min	38.355	-0.289	-2.176
72-15-7	2065	2052	1958	30min	31.442	-5.182	-4.581
72-15-8	2347	2314	2291	30min	32.584	-2.386	-0.994
72-15-9	2154	1940	2010	30min	54.561	-6.685	3.608
72-15-10	2225	2078	2113	30min	38.545	-5.034	1.684
average					**39.673**	**-2.578**	**0.276**
72-18-1	1676	1536	1664	30min	53.514	-0.716	8.333
72-18-2	1625	1685	1596	30min	56.618	-1.785	-5.282
72-18-3	1783	1772	1772	30min	47.553	-0.617	0.000
72-18-4	1715	1683	1759	30min	50.029	2.566	4.516
72-18-5	1726	1754	1730	30min	50.181	0.232	-1.368
72-18-6	1628	1662	1652	30min	59.379	1.474	-0.602
72-18-7	1850	1852	1871	30min	61.969	1.135	1.026
72-18-8	1562	1596	1619	30min	49.133	3.649	1.441
72-18-9	1794	1802	1832	30min	59.208	2.118	1.665
72-18-10	1779	1779	1801	30min	49.533	1.237	0.111
average					**53.712**	**0.929**	**0.984**
72-21-1	1490	1524	1490	25min	64.114	0.000	-2.231
72-21-2	1569	1577	1570	25min	52.543	0.064	-0.044
72-21-3	1613	1627	1627	25min	56.370	0.868	0.000
72-21-4	1488	1489	1488	25min	55.151	0.000	-0.067
72-21-5	1445	1445	1445	25min	39.935	0.000	0.000
72-21-6	1476	1516	1476	25min	52.657	0.000	-2.639
72-21-7	1574	1600	1574	25min	53.133	0.000	-1.625
72-21-8	1424	1424	1449	25min	46.655	1.756	1.756
72-21-9	1509	1533	1533	25min	41.725	1.590	-1.288
72-21-10	1499	1517	1507	25min	59.874	0.534	-0.659
average					**52.216**	**0.481**	**-0.726**
90-21-1	1932	1932	1904	40min	49.776	-1.449	-1.449
90-21-2	2006	1946	2026	40min	43.665	0.997	4.111
90-21-3	1901	1973	1958	40min	44.100	2.998	-0.760

TABLE 14 Continued

	CPLEX	LSC	Mod LSC			Gap (%)	
	Obj	Obj	Obj	time	Cut (%)	CPLEX	LSC
90-21-4	1780	1816	1832	40min	47.128	2.921	0.881
90-21-5	2037	1915	2033	40min	57.441	-0.196	6.162
90-21-6	2059	1954	1978	40min	52.803	-3.934	1.228
90-21-7	1696	1728	1706	40min	65.929	0.590	-1.273
90-21-8	2079	2212	2073	40min	60.002	-0.289	-6.284
90-21-9	2062	2036	1994	40min	62.068	-3.298	-2.063
90-21-10	1943	1987	1983	40min	55.321	2.059	-0.201
average					**53.823**	**2.059**	**0.061**

Note: Cut (%) – percentage of decision variables that have been set to be 0 according to the cut-off delivery sequences defined by the algorithm
Gap/CPLEX (%) – (Mod LSC – CPLEX)/CPLEX*100%
Gap/LSC (%) – (Mod LSC – LSC)/LSC*100%

The results listed in TABLE 14 show clearly that the proposed heuristic method and the selection mechanism are effective in saving CPU time greatly, as expected. The computation time consumed by the modified LSC algorithm is only 1/4 to 1/3 of the original algorithm. More important is that when the decision variables increase greatly, from around 5250 to around 8200, the computation time of the modified algorithm does not increase as significantly as the original. It proves that the modified algorithm has more potential and is more appealing in handling larger size problems. To assess the effectiveness of the algorithm in tightening the search space, we recorded the number of binary decision variables that have been set to be 0 according to the cut-off delivery sequences at the end of the algorithm. The results listed in TABLE 14 indicate that the search space of the original MILP model is halved. Thus, the CPLEX finds the equally good or even better solution in 20 minutes compared with the solution found by the CPLEX alone running 10 hours.

Compared to the best integer solution found by the CPLEX alone, the objective value obtained from the modified LSC method is 3.649% higher in the worst case and 10.582% lower in the best case. Compared with original one, the modified algorithm is 8.333% higher in the worst case and 6.284% lower in the best case. On average, the gap between the modified LSC algorithm, the branch-and-cut method, and the original algorithms is within 3% and 1%, respectively. In approximately half of those forty cases, the modified LSC method renders a better solution than other two methods in a much shorter CPU time.

When the problem size keeps extending, the CPLEX alone could not find a feasible solution, even running up to 10 hours. Thus, the absolute lower bound (LB) is chosen for examining the performance of the modified LSC algorithm in the next 10 cases. The LB is calculated assuming that there is no idling and delay during the QCs' operations. Thus, the optimal solution is impossibly lower than the LB provided here. The total computation time consumed by the modified LSC method is 30 minutes, and 10 minutes are consumed to solve the sub-problems iteratively. According to the results listed in TABLE 15, the number of decision variables is reduced by about 45% because of the introduction of cut-off delivery sequences. With a much lower computation load, the CPLEX finds a good solution in only 20 minutes. And its average gap between the LB does not exceed 15.4%.

TABLE 15 Modified LSC Algorithm v.s. LB

	Mod. LSC	time	# bin	Cut (%)	LB	Gap (%)
108-24-1	2055	30min	11765	45.261	1878	9.425
108-24-2	2168	30min	11765	47.488	1838	17.954
108-24-3	2128	30min	11767	46.239	1898	12.148
108-24-4	2092	30min	11763	39.369	1530	36.732
108-24-5	2350	30min	11766	38.484	1902	23.554
108-24-6	2238	30min	11766	48.683	1910	17.173
108-24-7	2183	30min	11765	41.156	1823	19.748
108-24-8	1896	30min	11765	48.729	1872	1.282
108-24-9	1992	30min	11764	44.713	1866	6.752
108-24-10	2044	30min	11766	49.074	1872	9.188
average				**44.920**		**15.393**

Note: # bin—number of binary decision variables
Cut (%) – percentage of decision variables that have been set to be 0 according to the cut-off delivery sequences defined by the algorithm

6.4 Sensitivity Analysis

To assess the robustness of the algorithm, sensitivity analysis is conducted to see whether or not the search process is greatly affected by the important parameters: initial solution and the time limit. The initial solution determines the initial upper and lower bounds of all starting times and therefore determines the construction of sub-problems during iteration. The time limit in solving sub-problems might affect the optimization of sub-problems and thus the update of time windows and the set of permanent cut-off delivery sequences. The number of containers and vehicles in the following experiments are set to be 108 and 21, respectively.

6.4.1 Initial Solution

As described, we choose a very simple method to generate the initial solution to start the iteration. However, if the algorithm is sensitive to the initial solution, we have to re-consider the method of obtaining the initial solution or even the design of the algorithm. To test the sensitivity of the algorithm to the initial solution, we increase and decrease the initial solution by 10% and 20% to determine whether the final solution would be significantly affected by the changes. Each objective value listed in TABLE 16 is the average of 10 cases.

TABLE 16 Algorithm's Sensitivity to the Initial Solution

Initial Solution	-20%	-10%	0%	10%	20%	-20%	-10%
Objective	2451.1	2444.9	2434.2	2490.3	2488.5	2451.1	2444.9

The computation results presented in TABLE 16 show that the changes in the initial solution do not affect the objective value. When the objective value of the initial solution decreases and increases by 10% and 20%, the objective value only changes around 1% and 2.5%. Additionally, the total CPU time is not affected. This implies that the algorithm is robust and not sensitive to the initial solution. The iteration can start with an initial solution obtained from any simple and straightforward method.

6.4.2 Time Limit for Solving Sub-problems

The longer the CPLEX is allowed to solve a sub-problem, the more possible that the sub-problem is solved to optimal, and the more accurate the upper and lower bounds could be. However, the price for that accuracy is the possible increase in the CPU time, especially when the problem size expands. To analyze to what extent the time limit affects the algorithm, we re-calculate those 10 cases, varying the time limit in solving sub-problems from 40 to 100 seconds.

TABLE 17 Algorithm's Sensitivity to the Time Limit in Solving Sub-problems

Time Limit (sec)	40.00	50.00	60.00	70.00	80.00	90.00	100.00
Objective	2485.8	2463.1	2451.0	2450.1	2434.2	2440.8	2440.3

The results listed in TABLE 17 reveal that when the time limit increases from 40 to 100 seconds, there is only a minor decrease in the objective values. The gap between the lowest and the highest averages of objective values is only 2.12%. Additionally, the computation time also remains the same when the time limit varies from the 50 to 90 seconds. The reason that the algorithm is not sensitive to the time limit in solving sub-problems can be summarized as follows: (1), because of the introduction of cut-off delivery sequences, most sub-problems can be solved to optimal within a short time limit. Thus, the change of the time limit does not affect the best solution and the computation time of most sub-problems, and (2), even if the sub-problem is not solved to optimal due to the reduction of the time limit, the analysis of sub-problem results prevent those “bad” results from being used to update the time windows or the sets of permanent cut-off delivery sequences.

In sum, the sensitivity analysis proves that the algorithm is robust. Its computation time and objective value are not affected by the initial solution or the time limit in solving sub-problems. In addition, the comparison between the original and modified LSC algorithms has shown that the application of heuristic method as the sub-problem solver quickens the searching process greatly. Thus, it is reasonable to believe that the proposed algorithm would be improved further if another more effective heuristic method is employed to solve the sub-problems and the original MILP model.

7. ON-LINE DISPATCHING POLICY

In addition to closely interrelated activities, the high degree of complexity of the terminal operations is due to the terminal's dynamic working environments. They include weather conditions; wrong information of containers and external vehicles; the mistakes during operations; the reshuffle operation in the storage area; and so on. As a result, the uncertainties in the system become a huge challenge to the terminal's daily operations.

In such a stochastic working environment, decisions have to be made without complete knowledge of future events, opposite to the static problem described in the last two sections. Compared with the off-line dispatching rule, the on-line dispatching method is more appropriate in a highly dynamic working environment where only limited information about future events is available. Thanks to the employment of advanced localization system and communication technologies, such as Differential Global Positioning System (DGPS) and Radio Frequency Identification (RFID), the supervisors can monitor the locations and status of different equipment and containers and communicate with operators and drivers effectively (*24*). Thus, the fleet of vehicles can be dispatched following an on-line dispatching rule, using the real-time process and travel time.

The on-line dispatching methods are typically one-to-many assignments and all these approaches generally are classified into two categories by Egbelu and Tanchoco (*16*). The first category is a request initiated dispatching rule that determines an appropriate vehicle from a set of idle vehicles to match the transportation request. The second category is vehicle-initiated dispatching rule that assigns one request from all available requests to the single vehicle. The on-line dispatching rule currently employed in NCT is the Longest Idle Vehicle (LIV) rule, according to the classification and definitions in the work of Egbelu and Tanchoco (*16*). Assume that there are two dummy depots at the single loop layout. The vehicles queue at depot A after dropping off containers at the storage blocks and at depot B after dropping off containers at the quay cranes (FIGURE 14). Once there is a newly released transportation request, the first two vehicles in the queue at depot A or B, which are also the vehicles that remained idle the longest among all the idle vehicles, are dispatched to pick up the containers from the QC side or the YC side. The main advantage of the rule is its easy application in the dynamic working environment. However, there are two main shortcomings: One is that the rule does not make full use of the vehicle resources. Every time a vehicle drops off a container at the QC or YC side, it returns to depot A or B, directly waiting for the next task. Thus, the vehicles will not be dispatched to pick up the containers on their way back to the depot. The second is that the rule does not take into consideration the simultaneous arrival of the two vehicles. For example, during the loading process, although the two vehicles that are dispatched to pick up the two containers may leave the depot almost at the same time, it is still highly possible that they arrive at the QC side at different times.

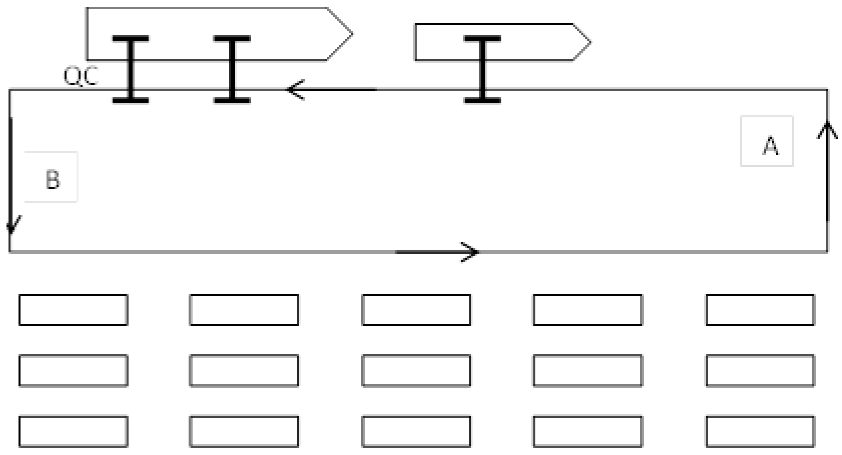


FIGURE 14 Current dispatching rule.

To make full use of the vehicles' capacity and make account of the requirement of the tandem lift operation, this section proposes a heuristic method as the on-line dispatching rule. It is a priority-based heuristic method with the objective of minimizing the makespan. The priority rule is not only easily implemented, but also flexible for practical application. The choice of attributes and the order in which they are compared can be adjusted according to the demands of the working environment. Series of numerical experiments are conducted to compare the performance of the proposed priority dispatching rule against the LIV rule under different degrees of stochasticity. The sensitivities of the approach to the QC's cycle time and the vehicles' average traveling speed are assessed in the sensitivity analysis.

7.1 Priority On-line Dispatching Rule

The proposed approach is also a request-initiated dispatching rule. It is similar to the heuristic method introduced in the modified LSC algorithm. The differences between the on-line and the off-line approaches are:

(1) The on-line dispatching rule only considers the first available transportation requests, instead of all available requests in making dispatching decisions because the QC's cycle time is dynamic, and we cannot predict when the two containers are available to be picked up or dropped off by vehicles at the QC side.

(2) In constructing the assignment, the on-line rule only assigns those vehicles idle when the requested is released. Because the vehicles' traveling time and their waiting time at the YC are not deterministic any more, the vehicles' arrival times cannot be calculated using the setup times as the off-line approach. If there is no vehicle available at the time when the first request is released, the first two available vehicles are dispatched to transport those two containers.

(3) Since only one request is considered every time, the on-line dispatching rule does not need to compare the assignments with different requests. Consequently, the priority measurement is completed in one step and the number of remaining requests is not necessary anymore.

After some initial experiments, we decide to measure the assignments' attributes following the order which is presented in TABLE 18.

TABLE 18 Priority Dispatching Rule

```
do while not all the transportation requests have been assigned;
for the newly released transportation request $r_k^s$ do
    if there is no idle vehicle or there is only one idle vehicle then
        dispatch the first two available vehicles to respond the request directly;
    else then
        create the set of feasible assignments $|A_k^s|$ by assigning the request to any two idle
         vehicles;
        if $|A_k^s| > 1$ then
            keep the assignments with the smallest QC idle time and delete others;
        end if
        if $|A_k^s| > 1$ then
            keep the assignments with the smallest arrival time difference and delete others;
        end if
        if $|A_k^s| > 1$ then
            keep the assignments with the smallest empty travel time and delete others;
        end if
        if $|A_k^s| > 1$ then
            keep the assignments with the smallest early arrival time and delete others;
        end if
        if $|A_k^s| > 1$ then
            keep the assignments with the earliest free time and delete others;
        end if
        if $|A_k^s| > 1$ then
            pick up an assignment randomly as the dispatching decision;
        end if
        if $|A_k^s| = 1$ then
```

TABLE 18 Continued

```
            dispatch the vehicles to the request according to it;
        **end if**
    **end if**
**end do while**
```

7.2 Numerical Experiment

7.2.1 Design of the Experiment

The numerical experiments in this section are designed to test the performance of the proposed priority rule and the LIV rule in the dynamic working environment. The performance measurements include the makespan and the QC's average productivity. The QC's average productivity is the average number of containers unloaded/loaded by each QC in one hour.

7.2.2 Experimental Scenarios

According to the empirical distribution in TABLE 1, the mode, the most frequent value of the QC's cycle time, is 90 seconds. Considering the traffic control problem in the paths, the fleet size is always 8 to 12 vehicles per QC. Thus, throughout the experiments, the number of QCs; containers and vehicles are set to be 3, 960, and 30, respectively.

To make account of the dynamic working environment, the experimental scenarios are distinguished by the different degrees of stochasticity. In simulating the dynamic working environment, the QC's cycle time, the vehicles' waiting times at the YC side and the vehicles' traveling time between the QC and YC side are simulated as random values according to the empirical data. The different degrees of stochasticity are simulated in the following four levels:

Deterministic: The cycle times are set to be the mean values of each time interval in TABLE 1 and the fraction of each time interval remains. For example, the mean value of the time interval [50, 60] is 55, and its frequency is 0.04. The vehicles' waiting times and travel times are set to the mean values used in the last two sections.

Low: To simulate the scenario with smaller variance in the process and traveling times, the empirical distributions are compressed such that the largest deviations from the mode value are reduced to half of the original value. For example, in TABLE 1, the cycle time varies from 50 to 140 seconds and the highest frequent value is 90 seconds. The smallest and largest values are away from mode value by 40 and 50 seconds. Thus, the distribution of the cycle times used in the Low scenario is compressed to [70,120]. The faction of each time interval remains the same, and the randomly generated cycle times in each interval follow the uniform distribution. To take account of the variance in the vehicles' waiting times and traveling times, their values are allowed to be up to 10% higher or lower than the mean value. Moreover, the randomly generated times within above range follow the normal distribution.

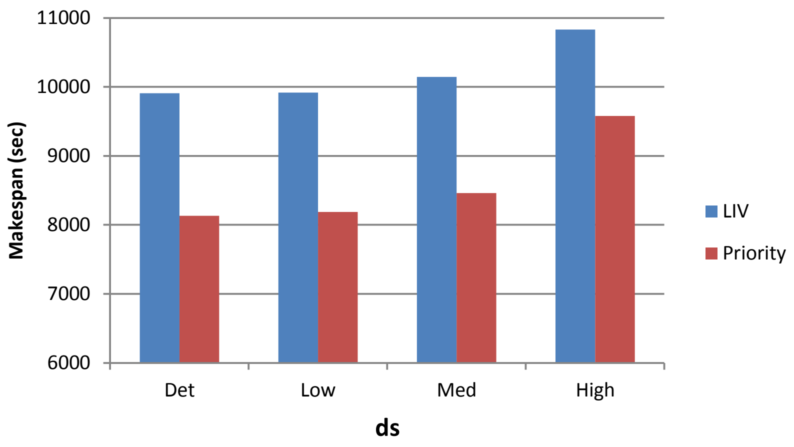
Medium: The cycle time is randomly generated following the empirical distribution in TABLE 1. The highest and lowest values of the vehicles' waiting and traveling times cannot exceed the 30% of the mean values.

High: The distribution in the High scenario is designed to account for the large disturbance caused by the unexpected events, like the bad weather, wrong operation, and so on. The empirical distribution is expanded such that the deviations are doubled. Because there is no cycle time smaller than 20 seconds, the cycle time accounted in the High scenario is from 30 to 190 seconds. The randomly generated waiting and traveling times are allowed to be 50% lower or higher than the mean values.

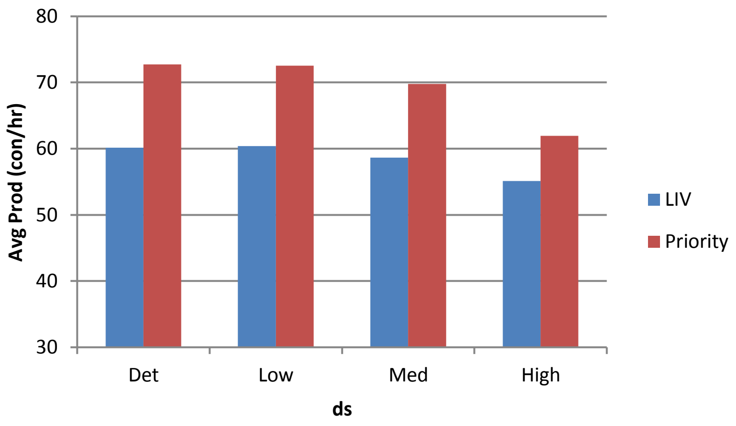
Throughout the experiments, each scenario is repeated thirty times and the LIV rule and the priority rule are tested in all those thirty cases. The QCs and the blocks where the containers are picked up and dropped off are generated by the program randomly for each case. The performances of the two rules are measured in the terms of the makespan and the QCs' average productivity.

7.2.3 Numerical Experiments Results

FIGURE 15 shows the system performance when the vehicles are dispatched following two different rules. Generally, the effect of the increasing degree of stochasticity is identical for the two rules. It impairs the performance of the LIV rule and the priority rule. However, the performance reduction is not so significant, except for the scenario of high degree of stochasticity. Compared to their own performances in the Deterministic scenario, the performance reductions are only about 4% and 2% when the vehicles are dispatched following the priority rule and LIV rule. Under the highly stochastic working environment, the makespan is increased by 17.76% and the QCs' average productivity is decreased by 14.85% when utilizing the priority rule; to account of the large disturbances in the highly stochastic working environment, more long cycle times are included during the simulation.



(a) Makespan



(b) QCs' average productivity

FIGURE 15 Performance under different degrees of stochasticity.

No matter what the scenario is, the priority rule clearly demonstrates the LIV rule. According to the results of numerical experiments, the priority rule shortens the makespan by about 19% and increases the QCs' average productivity by about 17% in the scenarios of Deterministic, Low and Medium. In the scenario of high degree of stochasticity, the 11.58% shorter makespan and 12.38% higher QC productivity are observed when the priority rule is employed.

7.3 Sensitivity Analysis

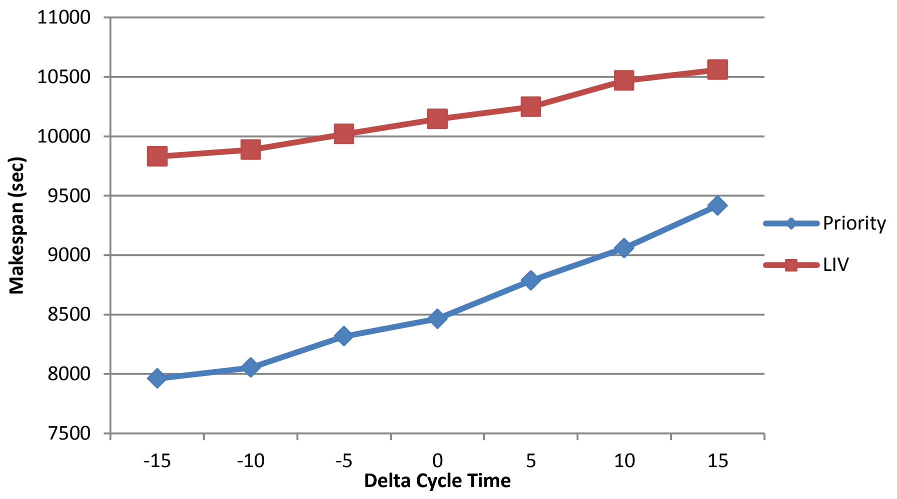
It is clear that the speed of loading and unloading operations is influenced by many factors, namely the cycle times of QCs, the availability of vehicles, and the working speed of yard cranes. To assess how the system performance respond to those system input, this section conducts sensitivity analysis in the scenario of a medium degree of stochasticity.

7.3.1 Cycle Time

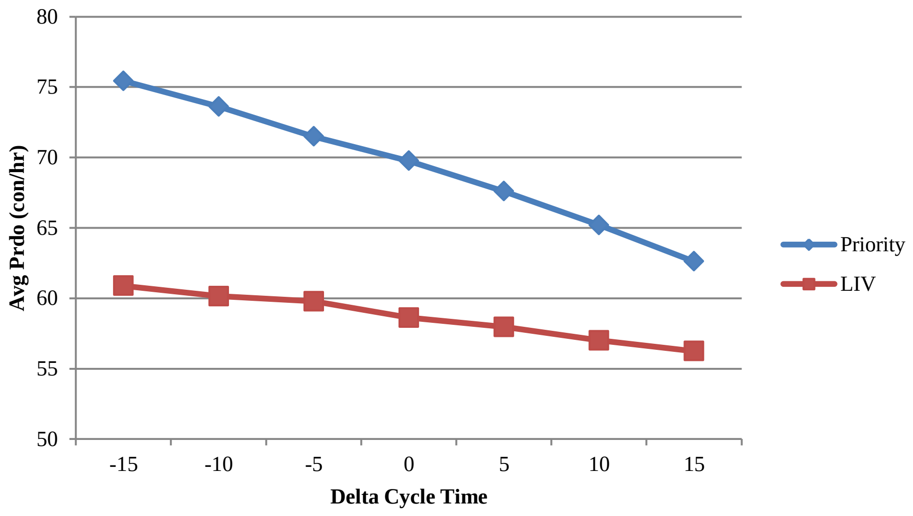
The cycle time of a QC depends on the technique design of the crane and the experience of the operator. To test the effects of varying cycle times, we decrease and increase the cycle time (τ) with 5, 10 and 15 seconds, respectively ($\Delta\tau = \pm 5, \pm 10, \pm 15$). It should be noted that it only changes the value of the cycle time listed in TABLE 1 but each interval's fraction remains the same. For example, when $\Delta\tau = -5$, the first interval of the cycle time becomes 45-55 seconds and its fraction is still 0.04. The effect of the change in the cycle time, which is also the frequency of generating transportation requests on the makespan and the QC's productivity, is investigated in this part.

FIGURE 16 shows the changes in the makespan and the QCs' average productivity with respect to the cycle time. Compared with the LIV rule, the priority rule is more sensitive to the changes in the QC's cycle time. When the $\Delta\tau$ changes from -15 to 15 seconds, the makespan increases by 7.42% and the QCs' average productivity reduces by 7.61% with the application of the LIV rule. At the same time, when the vehicles are dispatched following the priority rule, the increase in the makespan and the decrease in the average productivity are up to 16.99% and 18.28%, respectively.

It is obviously that the application of the priority rule is capable of shortening the makespan and enhancing the QCs' productivity. However, along with the increasing cycle time, the gap between the two rules' performances gets smaller. When $\Delta\tau = -15$, the makespan is 19.01% shorter and the QCs' average productivity is 23.90% higher utilizing the priority rule. When $\Delta\tau$ reaches 15 seconds, the improvements in the makespan and the QCs' productivity are reduced to 10.82% and 11.32%. That is because when the QC is capable of unloading and loading containers at a high speed, the performance of the system is greatly determined by the availability of the vehicles. However, when the QC's design working speed slows down, the dispatching policy becomes less important. The working capacity of the QC instead of the availability of vehicles becomes the bottleneck of the system performance in that case. It proves again that the higher a QC's designed working capacity is, the more important the vehicles' dispatching policy is to the QC's actual working speed and productivity.



(a) Makespan v.s. Cycle Time



(b) Avg Productivity v.s. Cycle Time

FIGURE 16 Performances of dispatching rules against the varied cycle times.

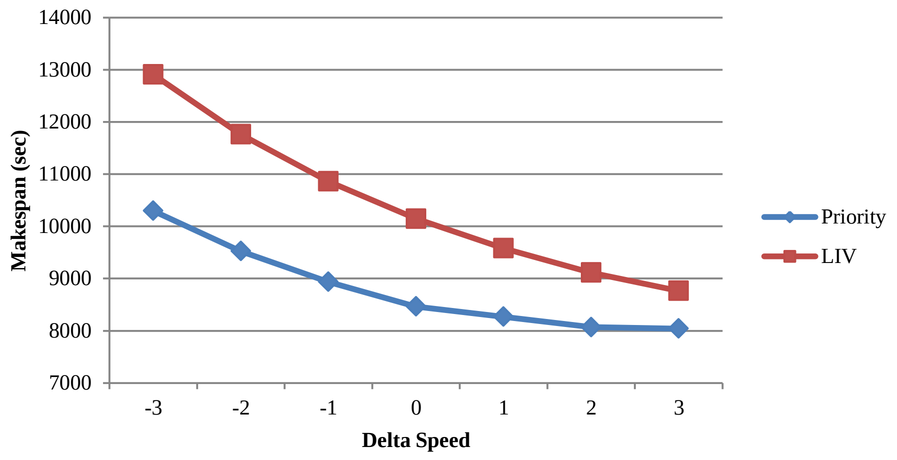
7.3.2 Vehicle Speed

The optimal fleet size is dependent on the frequency of the transportation request generation and the time it takes vehicles to the transport containers between the QC and YC side. The former one is decided by the QCs' cycle time, which is discussed in the last part, and the latter one is greatly affected by the vehicles' traveling speed and the dispatching policy. In this section, we investigate that to what extent the vehicles' traveling speed affects the system performance by increasing and decreasing the vehicles' average traveling speed by 1, 2 and 3 m/s.

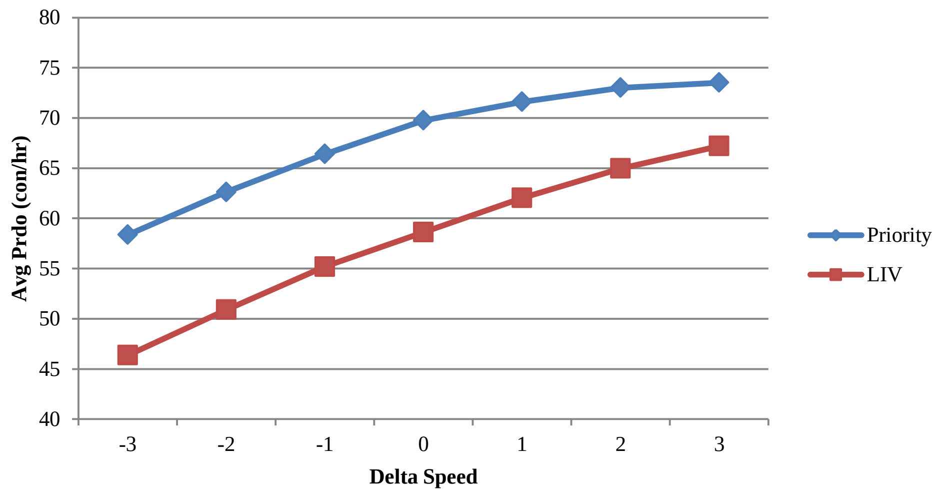
FIGURE 17 presents the makespan and the QCs' average productivity with different vehicle speeds and dispatching rules. The increasing vehicle speed is helpful to enhance the QC's productivity but such improvement becomes less significant when the speed increases. When the vehicles travel slower, they always arrive later than the time that the QC is ready to transfer the container onto/from them. As a result, in most cases, the starting times of QC side tasks or the QCs' actual working speed are more dependent on their availabilities of vehicles. When the vehicles travel faster, the QCs' productivity are increased due to the improved availabilities of vehicles. However, when the vehicles travel faster, they arrive at the QC earlier and the vehicles may spend more time waiting under the QC. Consequently, the increase in the vehicles' waiting time offset the reduced travel time. In those cases, the QCs' productivity cease increasing unless there are improvements of other elements.

In contrast to the result in the last section, the LIV rule is more sensitive to the change of the vehicle speed. When the vehicles' average speed varies from 7 to 13 m/s, the increase in the makespan achieves 44.90% and the reduction in the QCs' average productivity is up to 32.10% under the LIV rule. However, those changes are only 21.89% and 25.97% with the application of the priority rule. That is because that when utilizing the LIV rule, the task's availability of the vehicle is greatly dependent on the time when the vehicle returns the depot A or B after dropping off the container. Thus, the faster a vehicle travels, the earlier it returns to the depot and becomes available for the next delivery. Consequently, the vehicles' speed becomes the distinct element in determining the performance of the LIV rule. On the contrary, when the vehicles are dispatched following the priority rule, it is not necessary that the first available vehicle will be dispatched to the newly released transportation request. The dispatching decision is made according to the assignments' priorities which are determined by multiple attributes. As a result, the priority rule is less sensitive to the vehicles' speed than the LIV rule.

Being similar to the experiments in the previous section, the gap between the the two rules becomes less significant with the increasing vehicles' speed. When the vehicles speed increases from 7 to 13m/s, the improvement of the makespan is reduced from 20.23% to 8.25%, and the improvement of the QCs' average productivity decreases from 25.85% to 9.41%. Because when the vehicles travel faster between the QC and YC, the availability of vehicles is necessarily improved no matter which rule is used. Thus, a similar conclusion can be derived from the sensitivity analysis results in this section. The vehicle dispatching rule is more important when the vehicle speed is relatively slow and the superiority of the priority rule is more substantial in these cases.



(a) Makespan



(b) QC productivity

FIGURE 17 Performances of dispatching policies against the varied vehicle speeds.

8. CONCLUSION AND FUTURE RESEARCH

This dissertation investigates the vehicle dispatching problem in a container terminal equipped with tandem lift QCs. The subject remains a relatively new in the terminal operation and lacks systematic research.

The static version of the vehicle dispatching problem is mathematically formulated as a mixed integer linear program model. To solve such an NP-hard problem, a heuristic algorithm is proposed to reduce the feasible search region by eliminating those feasible but undesirable delivery sequences. The determination of cut-off delivery sequences are defined according to the lower and upper bounds of the starting times, estimated through constructing and solving sub-problems iteratively. The numerical experiments results prove that, compared with the branch-and-bound method, realized using the CPLEX, the proposed LSC algorithm is capable of finding a competitive, even better solution and saving the CPU time up to 80%. In addition, both the computation time and the objective value of the solution are not sensitive to the initial solution or the time limit in solving sub-problems. In addition, the module-design of the proposed LSC algorithm provides the flexibility in choice of the method to solve sub-problems and the original MILP model.

For the terminal's daily operation, this dissertation develops a less sophisticated, priority-based on-line dispatching method to make the dispatching decision without information of future events. The proposed priority rule and the LIV rule are compared through series of numerical experiments simulating the dynamic QCs' cycle times, vehicles' traveling times, and the vehicles' waiting times at the YCs. The results reflect that the priority rule performs better in shortening the makespan and enhancing the QCs' average productivity under different degrees of stochasticity. The superiority of the priority rule is more substantial when the QCs' cycle times are shorter and/or the vehicles' speed is slow. In other words, the role of the vehicle dispatching rule is more important when the availability of vehicles is not sufficient compared with the frequency of releasing transportation requests, and the priority rule is very flexible in comparing the priorities of assignments.

Future research on the vehicle dispatching problem in the container terminal should focus on the integration of different activities and the investigation of the influences of new equipment. Simultaneously, it still needs a more efficient algorithm to solve large-scale vehicle dispatching problems to accommodate the terminal's daily operation.

REFERENCES

1. Avriel, M., M. Penn, and N. Shpirer. Container Ship Stowage Problem: Complexity and Connection to the Coloring of Circle Graphs. *Discrete Applied Mathematics*, Vol. 103, No. 1-3, 2000, pp. 271–279.
2. Avriel, M., M. Penn, N. Shpirer, and S. Witteboon. Stowage Planning for Container Ships to Reduce the Number of Shifts. *Annals of Operations Research*, Vol. 76, 1998, pp. 55-71.
3. Bae, H.Y., R. Cheo, T. Park, and K. R. Ryu. Comparison of Operations of AGVs and ALVs in a Container Terminal. *Journal of Intelligent Manufacturing*, Vol. 26, 2009, pp. 117-143.
4. Bish, E. K. A Multiple-crane-constrained Scheduling Problem in a Container Terminal. *European Journal of Operational Research*, Vol. 144, 2003, pp. 83-107.
5. Bish, E. K., F. Y. Chen, Y. T. Leong, B. L. Nelson, J. W. Cheong Ng, and D. Simchi-Levi. Dispatching Vehicles in a Mega Container Terminal. *OR Spectrum*, Vol. 26, 2005, pp. 491-506.
6. Bish, E. K., T. Y. Leong, C. L. Lil, J. W. C. Ng, and D. S. Levi. Analysis of a New Vehicle Scheduling and Location Problem. *Naval Research*, Vol. 48, No. 5, 2001, pp. 363-385.
7. Bodian, L., and B. Golden. Classification in Vehicle Routing and Scheduling. *Networks*, Vol. 11, No. 2, 1981, pp. 97-108.
8. Briskorn, D., A. Drexl, and S. Hartmann. Inventory-based Dispatching of Automated Guided Vehicles on Container Terminals. *OR Spectrum*, Vol. 28, No. 4, pp. 611-630.
9. Chao, S. L., and Y. J. Lin. Evaluating Advanced Quay Cranes in Container Terminals. *Transportation Research Part E*, Vol. 47, 2011, pp. 432-445.
10. Chen. L., N. Bostel, P. Dejax, J. Cai, and L. Xi. A Tabu Search Algorithm for the Integrated Scheduling Problem of Container Handling Systems in a Maritime

Terminal. *European Journal of Operational Research*, Vol. 181, 2007, pp. 40-58.

11. Correa, A. I., A. Langevin, and L. M. Rousseau. Dispatching and Conflict-free Routing of Automated Guided Vehicle: A Hybrid Approach Combing Constraint Programming and Mixed Integer Programming. *Lecture Notes in Computer Science*, Vol. 3011, 2004, pp. 370-379.
12. Daganzo, C. F. The Crane Scheduling Problem. *Transportation Research Part B*, Vol. 23, 1989, pp. 159–175.
13. Dubrovsky, O., G. Levitin, and M. Penn. A Genetic Algorithm with a Compact Solution Encoding for the Container Ship Stowage Problem. *Journal of Heuristics*, Vol. 8, 2002, pp. 585–599.
14. Duinkerken, M. B., and J. A. Ottjes. A Simulation Model for Automated Container Terminals. *Proceedings of the Business and Industry Simulation Symposium (ASTC 1999)*, Washington, D.C., 1999.
15. Egbelu, P. J. Pull versus Push Strategy for Automated Guided Vehicle Load Movement in a Batch Manufacturing System. *Journal of Manufacturing Systems*, Vol. 6 1987, pp. 209–221.
16. Egbelu, P. J., and J. M. A. Tanchoco. Characterization of Automatic Guided Vehicle Dispatching Rules. *International Journal of Production Research*, Vol. 22, No.3, 1984, pp. 359-374.
17. Fan, L., M. Y. H. Low, H. S. Ying, H. W. Jing, Z. Min, and W. C. Aye. Stowage Planning of Large Containership with Tradeoff between Crane Workload Balance and Ship Stability. *Proceedings of the International Multi Conference of Engineerings and Computer Scientists 2010*. Vol. III, March 17-19, 2010, Hong Kong.
18. Gibson, R. R., B. C. Carpenter, and S. P. Seeburger. A Flexible Port Traffic Planning Model. *Proceeding of the 1992 Winter Simulation Conference*, 1992, pp. 1296-1306.
19. Grunow, M. G., H. O. Gunther, and M. Lehmann. Strategies for Dispatching AGVs at Automated Seaport Container Terminals. *OR Spectrum*, Vol. 28, 2006, pp. 587-

610.

20. Grunow, M., H. O. Gunthr, and M. Lehmann. Dispatching Multi-load AGVs in Highly Automated Seaport Container Terminals. *Container Terminals and Automated Transport Systems Part I,* 2005, pp. 231-255.
21. Guan, Y., and R. K. Cheung. The Berth Allocation Problem: Models and Solution Methods. *OR Spectrum*, Vol. 26, 2004, pp. 75-92.
22. Guignard, M., C, Ryu, and K. Spielberg, Model Tightening for Integrated Timber Harvest and Transportation Planning. *European Journal of Operational Research*, Vol. 111, 1998, pp. 448-460.
23. Günther, H. O., and K. H. Kim. Container Terminals and Terminal Operations. *OR Spectrum*, Vol. 28, 2006, pp. 437-445.
24. Guo, X., S. Y. Huang, W. J. Hsu, and M. Y. H. Low. Yard Crane Dispatching Based on Real Time Data Driven Simulation for Container Terminals. *Proceeding of the 2008 Winter Simulation Conference*, 2008, pp. 2648-2655.
25. Han, Y., L. H. Lee, E. P. Chew, and K. C. Tan. A Yard Storage Strategy for Minimizing Traffic Congestion in a Marine Container Transshipment Hub. *OR Spectrum*, Vol. 30, No. 4, 2008, pp. 697–720.
26. Hartmann, S. A. General Framework for Scheduling Equipment and Manpower at Container Terminals. *OR Spectrum*, Vol. 26, No. 1, 2004, pp. 51-74.
27. Homayouni, S. M., T. S. Hong, N. Ismail, and A. M. K. Ariffin, Optimization of Integrated Scheduling of Quay Cranes and Automated Guided Vehicles Using Simulated Annealing algorithm. *Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management*, Kuala Lumpur, Malaysia, Jan, 22-24, 2011, pp. 550-555.
28. Huo, J. Z., L. Zhang, and C. Peng. Simulation Studies of Truck Configuration in a Container Terminal. *Proceedings of the 6th world congress on intelligent control and automation*. Dalian, China, June 21-23, 2006. pp. 6217-6221.
29. Ichoua, S., M. Gendreau, and J. Y. Potvin. Diversion Issues in Real-time Vehicle Dispatching. *Transportation Science*, Vol. 34, 2000, pp. 426-438.

30. Imai, A., E. Nishimura, and S. Papadimitriou, The Dynamic Berth Allocation Problem for a Container Port. *Transportation Research Part B*, Vol. 35, 2001, pp. 401–417.
31. Imai, A., K. Nagaiwa, and C. W. Tat. Efficient Planning of Berth Allocation for Container Terminals in Asia. *Journal of Advanced Transportation*, Vol. 31, 1997, pp. 75-94.
32. Kang, S., K., J. C. Medina, and Y. Ouyang. Optimal Operations of Transportation Fleet for Unloading Activities at Container Ports. *Transportation Research Part B*, Vol. 42, 2008, pp. 970-984.
33. Kim, C. W., M. A. Tanchoco, and P. H. Koo. AGV Dispatching Based on Workload Balancing. *International Journal of Production Research*, Vol. 37, No. 17, 1999, pp. 4053-4066.
34. Kim, K. H. 2009. Harbor Logistics in Busan, Korea. *The Asia-Pacific Weeks Berlin 2009*.
35. Kim, K. H. Evaluation of the Number of Rehandles in Container Yards. *Computers & Industrial Engineering*, Vol. 32, 1997, pp. 701-711.
36. Kim, K. H., and H. B. Kim. Segregating Space Allocation Models for Container Inventories in Port Container Terminals. *International Journal of Production Economics*, Vol. 59, 1999, pp. 415–423.
37. Kim, K. H., and H. B. Kim. The Optimal Determination of the Space Requirement and the Number of Transfer Cranes for Import Containers. *Computers & Industrial Engineering*, Vol. 35, 1998, pp. 427–430.
38. Kim, K. H., and H. B. Kim. The Optimal Sizing of the Storage Space and Handling Facilities for Import Containers. *Transportation Research Part B*, Vol. 36, 2002, pp. 821-835.
39. Kim, K. H., and J. W. Bae. A Look-ahead Dispatching Method for Automated Guided Vehicles in Automated Port Container Terminals. *Transportation Science*, Vol. 38, No. 2, 2004, pp. 224-234.
40. Kim, K. H., and J. W. Bae. Re-marshaling Export Containers in Port Container

Terminals. *Computers & Industrial Engineering*, Vol. 35, 1998, pp. 655–658.

41. Kim, K. H., and K. C. Moon. Berth Scheduling by Simulated Annealing. *Transportation Research Part B*, Vol. 37, 2003, pp. 541–560.
42. Kim, K. H., and K. T. Park. A Crane Scheduling Method for Port Container Terminals. *European Journal of Operational Research*, Vol. 156, 2004, pp. 752–768.
43. Kim, K. H., and K. T. Park. A Note on a Dynamic Space-allocation Method for Outbound Containers. *European Journal of Operational Research*, Vol. 148, No. 1, 2003, pp. 92–101.
44. Koo, P. H., W. S. Lee, and D. W. Jang. Fleet Sizing and Vehicle Routing for Container Transportation in a Static Environment. *OR Spectrum*, Vol. 26, No. 2, 2004, pp. 193-209.
45. Langevin, A., D. Lauzon, and D. Riopel, Dispatching, Routing and Scheduling of Two Automated Guided Vehicles in a Flexible Manufacturing System. *International Journal of Flexible Manufacturing System*, Vol. 8, 1996, pp. 246-262.
46. Lee, B. K., and K. H. Kim. Comparison and Evaluation of Various Cycle-time Models for Yard Cranes in Container Terminals. *International Journal of Production Economics*, Vol. 126, 2010, pp. 350-360.
47. Lee, B. K., and K. H. Kim. Optimizing the Yard Layout in Container Terminal. *OR Spectrum*, Vol. 35, No. 2, pp. 363-398.
48. Lee, L. H., E. P. Chew, K. C. Tan, and Y. Han. An Optimization Model for Storage Yard Management in Transshipment Hub. *OR Spectrum*, Vol. 28, 2006, pp. 539-561.
49. Li, C. L., X. Cai, and C. Y. Lee. Scheduling with Multiple-job-on-one-Processor Pattern. *IIE Transactions*, Vol. 30, No. 5, 1998, pp. 433-445.
50. Lim, J. K., K. H. Kim, K. Yoshimoto, J. H. Lee, and T. A. Takahashi, Dispatching Method for Automated Guided Vehicles by Using a Bidding Concept. *OR Spectrum*, Vol. 25, 2003, pp. 25-44.
51. Lind, D., J. K. Hsieh, and M. A. Jordan. Tandem-40 Dockside Container Cranes and

Their Impact on Terminals. *ASCE Ports 2007 Conference*, San Diego, CA.

52. Liu, C. I., H. Jula, K. Vukadinovic, and P. Ioannou. Automated Guided Vehicle System for Two Container Yard Layouts. *Transportation Research Part C,* Vol. 12, 2004, pp. 349-368.
53. Liu, C., I., and P. A. A. Ioannou. Comparison of Different AGV Dispatching Rules in an Automated Container Terminal. *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pp. 880-885.
54. McCarthy, P. W., M. A. Jordan, and L. Wright. Dual-hoist, tandem 40 crane considerations. Port Technology International, http://www.porttechnology.org/technical_papers/dual_hoist_tandem_40_crane_considerations/
55. Meersmans, P. J. M., and A. P. M. Wagelmans. 2001. Effective Algorithms for Integrated Scheduling of Handling Equipment at Automated Container Terminals. *ERIM report series research in management*, ERS-2001-36-LIS, 2001.
56. Meisel, F., and C. Bierwirth. Integration of Berth Allocation and Crane Assignment to Improve the Resource Utilization at a Seaport Container Terminal. *Operation Research Proceedings,* 2005, pp. 105-110.
57. Moon, K. C. A Mathematical Model and a Heuristic Algorithm for Berth Planning. *PhD thesis*, Pusan National University, Pusan, Korean, 2000.
58. Murty, K. G., J. Liu, Y. Wan, and R. Linn. A Decision Support System for Operations in a Container Terminal. *Decision Support System,* Vol. 39, 2003, pp. 309-332.
59. Murty, K. G., J. Liu, Y. Wan, C. Zhang, M. Tsang, and R. Linn. DSS (decision support systems) for Operations in a Container Shipping Terminal. In *Proceedings of the First Gulf Conference on Decision Support Systems*, Kuwait, November 6–8, 2000, pp. 189–208.
60. Ng, W. C. Crane Scheduling in Container Yards with Inter-crane Interference. *European Journal of Operational Research,* Vol. 164, 2005, pp. 64–78.
61. Ng, W. C., and K. L. Mak. Yard Crane Scheduling in Port Container Terminals.

Applied Mathematical modeling, Vol. 29, 2005, pp. 263-276.

62. Ng, W. C., K. L. Mak, and Y. X. Zhang. Scheduling Trucks in Container Terminals Using a Genetic Algorithm. *Engineering Optimization*, Vol. 39, No. 1, 2007, pp. 33-47.
63. Nishimura, E., A. Imai, and S. Papadimitriou. Berth Allocation Planning in the Public Berth System by Genetic Algorithms. *European Journal of Operational Research*, Vol. 131, 2001, pp. 282–292.
64. Nye, L. W. Advanced Technology in Terminal Design. Moffatt & Nichol. http://aapa.files.cms-plus.com/SeminarPresentations/2009Seminars/2009LatinAmerican/09LATEXEC_ Nye_ Larry.pdf.
65. Park, N., H. R. Choi, H. K. Kwon, S. W. Lee, and S. H. Lee. A Study on the Efficiency of Transportation Equipment at Automated Container Terminals. *Proceedings of the 2007 WSEAS International Conference on Computer Engineering and Applications*, Gold Coast, Australia, January 17-17, 2007, pp. 360-365.
66. Park, Y. M., and K. H. Kim. A Scheduling Method for Berth and Quay Cranes. *OR Spectrum*, Vol. 25, 2003, pp. 1–23.
67. Peterkofsky, R. I., and C. F. Daganzo. A Branch and Bound Solution Method for the Crane Scheduling Problem. *Transportation Research Part B*, Vol. 24, 1990, pp. 159–172.
68. Preston, P., and E. Kozan. An Approach to Determine Storage Locations of Containers at Seaport Terminals. *Computers & Operations Research*, Vol. 28, 2001, pp. 983–995.
69. Qiu, L., and W. J. A. Hsu. A Bi-directional Path Layout for Conflict-free Routing of AGVs. *International Journal of Production Research*, Vol. 39, 2001, pp. 2177-2195.
70. Quadrifoglio, L., M. M. Dessouky, and F. Ordonez, Mobility Allowance Shuttle Transit (MAST) Services: MIP Formulation and Strengthening with Logic

Constraints. *European Journal of Operational Research*, Vol. 185, 2008, pp. 481-494.

71. Saanen, Y., J. van Meel, and A. Verbraeck, 2003. The design and assessment of next generation automated container terminals. Technical Report, TBA Nederland/Delft University of Technology.
72. Stahlbock, R., and S. Voß, Operation Research at Container Terminal: a Literature Update. *OR Spectrum*, Vol. 30, 2008, pp. 1-52.
73. Steenken, D., S. Voß, and R. Stahlbock. Container Terminal Operation and Operations Research – a Classification and Literature Review. *OR Spectrum*, Vol. 26, 2004, pp. 3-49.
74. Taghaboni-Dutte, F., A Value-added Approach for Automated Guided Vehicle Task Assignment. *Journal of Manufacturing Systems*, Vol. 16, No. 1, 1997, pp. 24-34.
75. Taleb-Ibrahimi, M., and B. de Castilho, Storage Space vs. Handling Work in Container Terminals. *Transportation Research Part B*, Vol. 27B, No. 1, 1993, pp. 13-32.
76. United Nations Conference on Trade and Development (UNCTAD) 2011, Statistics of Worldwide Maritime Transport http://archive.unctad.org/Templates/Page.asp?intItemID=5803&lang=1.
77. Vis, I. F. A., and I. Harika. Comparison of vehicle types at an automated container terminal. *OR Spectrum*, Vol. 26, 2004, pp. 117-143.
78. Vis, I. F. A., and R. de Koster. Transshipment of Containers at a Container Terminal: An Overview. *European Journal of Operation Research*, Vol. 147, 2003, pp. 1-16.
79. Vis, I. F. A., R. de Koster, K. J. Roodbergen, and L. W. P. Peeters. Determination of the Number of Automated Guided Vehicles Required at a Semi-automated Container Terminal. *Journal of the Operational Research Society*, Vol. 52, 2001, pp. 409-417.
80. Vis, I. F. A., R. M. B. de Koster, and M. W. P. Savelsbergh. Minimum Vehicle Fleet Size under Time-window Constraints at a Container Terminal. *Transportation*

Science, Vol. 39, 2005, pp. 249-260.

81. World Maritime News. The Netherlands: APM Terminals Orders Quay Cranes from Cargotec. http://worldmaritimenews.com/archives/57559/. Accessed June 12, 2012.
82. Yang, C. H., Y. S. Choi, and T. Y. Ha, Simulation-based Performance Evaluation of Transport Vehicles at Automated Container Terminals. *OR Spectrum*, Vol. 26, 2004, pp. 149-170.
83. Yavary, M., A. Jacob, M. Richter, and L. Nye. Master Planning of a Semi-Automated Container Terminal. *Ports 2010: Building on the Past, Respecting the Future.* pp. 1254-1264.
84. Yun, W. Y., and Y. S. Choi. A Simulation Model for Container-terminal Operation Analysis Using an Object-oriented Approach. *International Journal of Production Economics*, Vol. 59, 1999, pp. 221-230.
85. Zhang, C., J. Liu, Y. Wan, and K. G. Murty, and R. J. Linn. Storage Space Allocation in Container Terminals. *Transportation Research Part B*, Vol. 37, No. 10, 2003, pp. 883-903.
86. Zhang, C., Y. Wan, J. Liu, and R. J. Linn. Dynamic Crane Deployment in Container Storage Yards. *Transportation Research Part B,* Vol. 36, 2002, pp. 537-555.