DISPATCHING AND CONFLICT-FREE ROUTING OF VEHICLES IN NEW CONCEPTUAL AUTOMATED CONTAINER TERMINALS

XU YANHUA

NATIONAL UNIVERSITY OF SINGAPORE

2015

DISPATCHING AND CONFLICT-FREE ROUTING OF VEHICLES IN NEW CONCEPTUAL AUTOMATED CONTAINER TERMINALS

XU YANHUA

(B.Eng., Shanghai Jiao Tong University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL & SYSTEMS ENGINEERING

NATIONALUNIVERSITY OF SINGAPORE

2015

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in thesis.

This thesis has also not been submitted for any degree in any university previously.

XU YANHUA

14 Aug 2015

Acknowledgements

My PhD study at National University of Singapore is an invaluable experience in my life.

I would like to express my deepest appreciation to my supervisors, A/Prof. Chew Ek Peng and A/Prof. Lee Loo Hay. They offer me consistent encouragement and guidance throughout the whole course of my research. Without their invaluable instructions, this thesis would not be completed.

I would like to give thanks to my friends who offer valuable suggestions on my research. Thank Mu Aoran, Wang Qiang, Li Juxin, Fu Yinhui, Chen Liqing, Ma Sicong, Wang Yuan, Jiang Xinjia and Zhou Chenhao. I also would like to thank my peers who pursue Ph.D degree together. They are Ong Xianrong, Tang Baiwan, Xiao Yan, Lei Zong and Chao Ankuo.

Special thanks to my best friends in China: Huang Zhigang, Lin Zhenfeng, Deng Wanxi, Li Bojie, Li Nan, Huang Shanshan and Yang Xiuli.

Last, but not the least, I would like to thank my family for their continuous support and understanding, especially my brother-in-law Zhang Yi.

Table of Contents

Summary

This dissertation addresses the vehicle dispatching and conflict-free routing problems in two automated container terminal concepts. One is called Frame Bridge based Automated Container Terminal (FB-ACT) and the other is called Goods Retrieval and Inventory Distribution based Automated Container Terminal (GRID-ACT). As the size of realistic problem is large which requires very high computational and memory requirement, we develop a tree structure algorithm which is analogous to filtered beam search algorithm as the overall framework to solve the dispatching and routing problems. Based on the structure of the problems, we propose different novel solution methods and embed them into the analogous filtered beam search algorithm. In the FB-ACT system, the first stage method estimates the results by a heuristic method which determines the task sequence based on pseudo delays. The second stage method obtains the results by solving the MIP model whose scale is greatly reduced when the decisions of vehicle dispatching problem are determined. In the GRID-ACT system, the first stage method obtains the results by the sequential path generation method. The second stage method implements a modified column generation algorithm which uses historical information to improve the results. As we observe some operational limitations that impede on the performance of the FB-ACT, we propose a hybrid FB-ACT system which uses some good ideas from the GRID-ACT to be implemented in the transportation system at the quayside.

List of Tables

List of Figures

List of Abbreviations

ACT	Automated Container Terminal
AGV	Automated Guided Vehicle
AS/RS	Automated Storage/Retrieval System
CGA	Column Generation Algorithm
CGA-FS	Column Generation Algorithm with Further Search
FB-ACT	Frame Bridge based Automated Container Terminal
FBS	Filtered Beam Search
FT	Frame Trolley
GR-ACT	Grid Rail system Automated Container Terminal
GRID-ACT	Goods Retrieval and Inventory Distribution based Automated Container Terminal
GT	Ground Trolley
HFG-ACT	Hybrid of Frame Bridge and GRID systems based Automated Container Terminal
HSPG	Heuristic Sequential Path Generation
LMCS	Linear Motor Conveyance System
MIP	Mixed Integer Programming
QC	Quay Crane
RMGC	Rail Mounted Gantry Crane
RTGC	Rubber Tired Gantry Crane
SPG	Sequential Path Generation
TEU	Twenty-feet Equivalent Unit
TP	Transfer Platform
TU	Transfer Unit
YC	Yard Crane

List of Notations

Q	the set of QCs
R	the number of horizontal rails (berth rails).
F	the set of FTs
F_i	the set of FTs that belonging to Rail i, $F_1 \cup ... \cup F_r = F$
N_r	the number of FTs in the Rail r, $N_r = \mid F_r \mid$, specially $N_0 = 0$
P	the set of TPs
P_i	the set of TPs that belonging to Rail i, $P_1 \cup ... \cup P_r = P$.
L	the set of loading jobs
D	the set of discharging jobs
H	the set of all the jobs, $H = L \cup D$
QL	the set of subtasks handled by QC
TP	the set of subtasks handled by TP
G	the set of all subtasks, $G = QL \cup TP$
$B_{(i,\alpha,h)}$	the yard block that subtask (i,α,k) belongs to, $(i,\alpha,k) \in TP$
(i,α)	job index. The job (i,α) refers to the i th job in the sequence list of QC α.
$(i,\alpha,1)$	subtask index. The subtask of the first stage of job (i,α).
$(i,\alpha,2)$	subtask index. The subtask of the second stage of job (i,α).
$p_{(i,\alpha,1)}$	the processing time of subtask $(i,\alpha,1)$
$p_{(i,\alpha,2)}$	the processing time of subtask $(i,\alpha,2)$
$t^{ft}_{(i,\alpha,k)\ j(\beta,l)}$	the FT traveling time from location of subtask (i,α,k) to location of subtask (j,β,l)
$t^{tp}_{(i,\alpha,k)(j,\beta,l)}$	the TP traveling time from location of subtask (i,α,k) to location of subtask (j,β,l)
$l_{(i,\alpha,k)}$	the location of subtask (i,α,k).
$X^m_{(i,\alpha)}$	$X^m_{(i,\alpha)}$ =1 if the container of job (i,α) is carried by FT m; otherwise 0
$Z_{(i,\alpha,k)(j,\beta,l)}$	$Z_{(i,\alpha,k)(j,\beta,l)}$ =1 if the starting time of subtask (j,β,l) is greater

	than or equal to the finishing time of subtask (i,α,k); otherwise 0
$T_{(i,\alpha,1)}$	the starting time of the first subtask of job (i,α)
$T_{(i,\alpha,2)}$	the starting time of the second subtask of job (i,α)
K	the set of all TUs
N	the set of all cells
T	the set of all times
J	the set of all container jobs
W	the set of all guide-path segments
$L(n)$	the set of all feasible routes for TU n
$\lambda(n)$	the set of all feasible routes generated in the route generating problem for vehicle n
$G(i)$	the set of all cells in the physical layout that are within the safety distance to Cell i but excluding cell i
$F(w)$	the set of guide-path segments that are adjacent to w $(w \in W)$ and including itself, that is $w \in F(w)$
$A(i)$	the set of all cells in the physical layout that are adjacent to Cell i
p_τ	processing time of subtask τ
$c_{l,n}$	the total penalty cost of route l $(l \in L(n))$, where it equals to the total time that the route accomplishes its related subtasks
$A_{(i,t),l,n}$	$\begin{cases} 1, \text{if TU } n, \text{following route } l \text{ is at cell } i \text{ at time } t \\ 0, \text{otherwise} \end{cases}$
$B^i_{l,n}$	$\begin{cases} 1, \text{if TU } n, \text{following route } l \text{ covers job } i \\ 0, \text{otherwise} \end{cases}$
$f^t_{w^-,l.n}$	$\begin{cases} 1, \text{if TU } n, \text{following route } l \text{ enters segment } w \text{ at} \\ \quad \text{an upwards direction at time } t \\ 0, \text{otherwise} \end{cases}$
$f^t_{w^+,l.n}$	$\begin{cases} 1, \text{if TU } n, \text{following route } l \text{ enters segment } w \text{ at} \\ \quad \text{a downwards direction at time } t \\ 0, \text{otherwise} \end{cases}$

$X_{l,n}$	$\begin{cases}1, \text{if TU } n \text{ follows route } l \\ 0, \text{otherwise}\end{cases}$
$g^{t}_{\tau,lm}$	$\begin{cases}1, \text{if TU } m, \text{following route } l \text{ starts subtask } \tau \text{ at time } t \\ 0, \text{otherwise}\end{cases}$
p_{α}	processing time of subtask α.
$t^{TU}_{\alpha\beta}$	the traveling time for a TU from the location of subtask α to the location of subtask β using the shortest path ignoring vehicle collisions
YC	the set of subtasks handled at the storage yard transfer points.
$C(\alpha)$	the set of subtasks that compete for the same storage yard transfer point, where $\alpha \in YC$.
$Q(\alpha)$	the set of subtasks that cannot be performed until subtask α is accomplished according to the QC list.
SP	the set of pairs of successive subtasks belonging to the same vehicle
T_{α}	the starting time of subtask α
$z_{\alpha\beta}$	$\begin{cases}1, \text{if subtask } \alpha \text{ starts earlier than subtask } \beta \\ 0, \text{otherwise}\end{cases}$

CHAPTER 1 Introduction

In 1955, former trucking company owner Malcom McLean and engineer Keith Tantlinger proposed to put the cargoes in steel containers as transportation units, which initiated the modern freight transportation. This mode not only provides protections against weather and pilferage, but also significantly improves the efficiency of cargo transportation and reduces the logistic cost. The unnecessary packing and unpacking in the transfer stations is prevented. The volume of cargoes is greatly increased by stacking the containers, which can carry as many as 14,000 TEUs.

According to the statistics provided by Containerization International, annual container traffic has increased by approximately five fold in the past two decades from 28.7 million TEUs in 1990 to 135.4 million TEUs in 2014. Container traffic measures the flow of containers from land to sea transport modes, and vice versa, in twenty-foot equivalent units (TEUs). This growth trend is expected to continue. Facing with the continuously increasing container traffic, port operators insist on developing sustainable solutions to enhance the effectiveness and efficiency of container transportation between vessels and storage yard.

In this introduction section, we will introduce the background of container terminals, followed by some commercial implemented automated container terminals, and then several new conceptual terminals are presented.

1.1 Background of container terminals

To complete the whole procedure of delivering the cargoes, the containers usually need to be transported via different transportation modes like vessels,

trains and trucks. The container terminals serve as the interface between these transportation modes. A container terminal is usually composed of three parts: quayside, storage yard and landside. Figure 1.1 describes the layout and working flow of a container terminal.

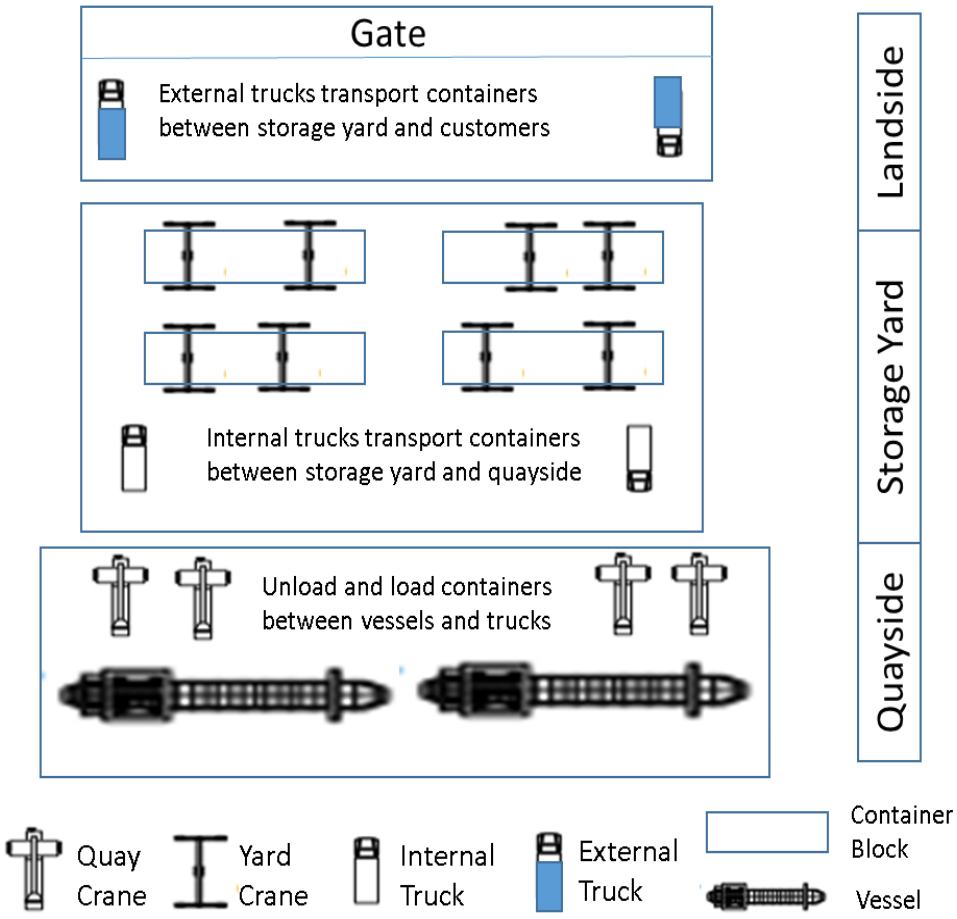


Figure 1.1 Working flow of a typical container terminal

According to the work flow, the container activities can be categorized into three types: import, export and transshipment activities. For export activities, the containers are brought in by trains or trucks in landside and temporarily stored in the storage yard. When the containers are ready to be loaded, they are retrieved from the storage place and transported by internal transporters to the designated vessels. The equipment that transfers the containers between

internal transporters and storage yard is called yard crane (YC). The equipment that transfers the containers between internal transporters and vessels is called quay crane (QC). The processes for import activities are performed similarly but in the reverse order. For transshipment activities, the containers are unloaded from a vessel, stored in the storage yard, and will be ultimately loaded onto the designated vessels. Figure 1.2 describes the flow of these three activities in the container terminal.

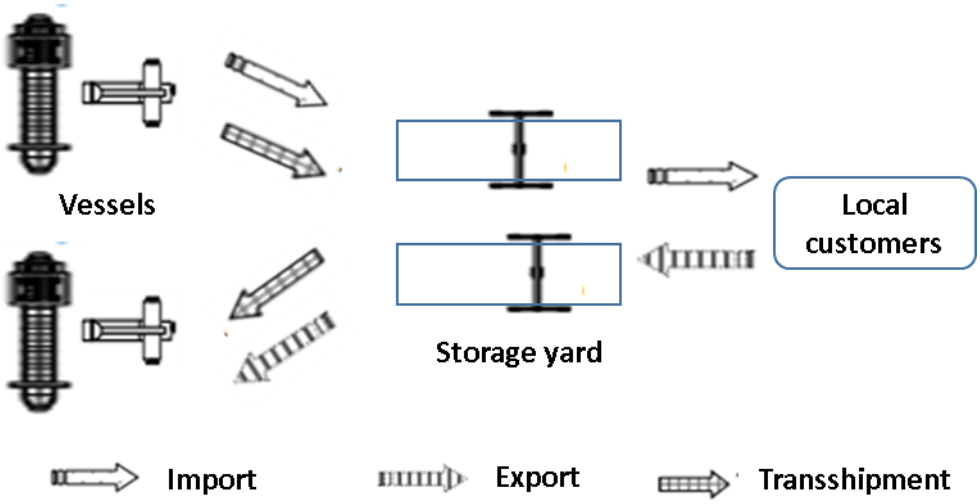


Figure 1.2 Import, export and transshipment

Readers can refer to review papers by Steenken et al. (2004) and Stahlbock et al. (2008) for more detailed information on the operations in the container terminals.

1.2 Automated Container Terminal

Automation is the use of various control systems for operating equipment such as machinery, and processes in factories. Automation is proved to bring great benefits in manufacturing industry. It can considerably reduce the manpower cost. It can also reduce the carbon dioxide release by using electricity instead of diesel. The productivity is improved because the involved procedures are

performed smoothly by the control systems, which can reduce delay that happens during the handshakes among the equipment. It also helps to reduce unnecessary loss brought about from improper operations by manpower.

The terminal operators over the world have been increasingly pressurized to provide better and faster service to vessel operators. More 16,000-19,000 TEU ships will be introduced in the trade. According to industry estimates (see Chan and Huat (2002)), the typical operating cost for a Post Panamax vessel per day, can easily come to US$ 30,000 (see Table 1.1 for details).

Table 1.1 Operating cost for a typical Post Panamax vessel

	US$/day
Vessel Depreciation Cost (25 years life span)	10,000
Fuel Cost (18 knots cruising speed)	10,000
Wages, Maintenance and Insurance	10,000

Considering the high operating cost, it is imperative for vessel operators to maximize the yields and the number of voyages made by each vessel. A major challenge in port management is thus to reduce the turnaround time of the container ships, especially for those with large container volume.

On the other hand, environmental protection is a great concern worldwide and is believed to become more and more important. The CO2 emissions have become an issue and consequently there is increasing pressure on governments and industries to come forward with initiatives to it. The equipment in a typical port is powered by diesel, which produces high carbon dioxide

emissions. Table 1.2 shows the CO2 emissions from container terminals of Rotterdam Port. (see Geerlings et al. (2011)).

Table 1.2 CO2 emissions from container terminals of Rotterdam Port

Container Terminal	Total CO2 Emissions (kton CO2/year)	CO2 Emission per TEU (kg CO2/TEU)
Delta	71.30	16.73
Home	15.01	15.01
Hanno	1.20	24.00
APM	35.95	16.34
RST	10.76	9.35
Uniport	6.53	17.18
Total/Average	140.75	17.29

Furthermore, the potential risk of strike cannot be neglected. It is reported that the West Coast container ports in US faced a loss of ten billion dollars due to the strike in early 2015. A strike also happened in the ports of Vancouver, which brought about eight hundred million dollars of loss per week.

Therefore, introducing automation into the terminals becomes a substantial solution to these challenges. During the past two decades, port operators are insisting on improving the productivity by implementing automated equipment. The most successful commercial application is the automated guided vehicles based automated container terminals (AGV-ACT). Another commercially implemented case is the automatic straddle carrier based automated container terminals (ASC-ACT).

The first automated container terminal was implemented by Europe Container Terminal (ECT) in the 1990s at the Delta Dedicated North Terminal. Automated stacking cranes (ASCs) and automated guided vehicles (AGVs) were used. The AGVs were used to transport containers between the quayside and storage yard, whereas, the ASCs were used to stack the containers in the storage yard. Later, Container Terminal Altenwerder (CTA) also introduced ASCs and AGVs to the terminal in Hamburg. The layout of the AGV-ACT is showed in Figure 1.3. (Kim et al. (2004))

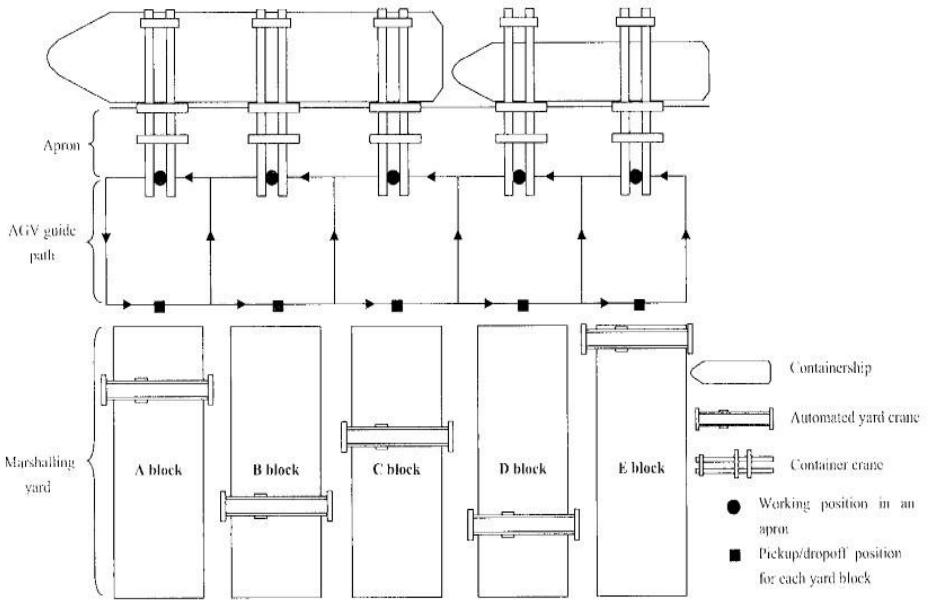


Figure 1.3 Layout of AGV-ACT

In the AGV-ACT system, AGVs take the place of trucks to transport the containers between quayside and storage yard. The AGVs are controlled by integrated information systems and move along the guide paths fixed on the ground. Rail mounted gantry cranes are implemented in the yard which are automated. For an unloading operation, a container picked up by a QC is put on an AGV which delivers the container to the storage yard. Different from the typical container terminal, the AGVs will not move into the storage yard.

They will stop at the end of yard blocks and wait for YCs to pick up the containers from them. After that, the AGVs are free and move to handle their next jobs. In the storage yard, a YC picks up the container from the AGV and will stack it onto a designated slot. The loading operation is performed similarly but in the reverse order.

Automatic straddle carrier can both transport and lift/release containers. It allows for decoupling the work flow of transport and crane activities by using buffers at the respective interfaces. Moreover, the straddle carriers are able to stack four containers. They can not only be used as transporters but also as stack cranes in the storage yards. In this way, yard cranes like RTGCs or RMGCs are no longer needed. The automated straddle carrier system began to be used in Brisbane port in 2005. They have also been implemented in Maersk APM Shipping Container Terminal Port in Portsmouth Virginia. A study on the efficiency of the transportation equipment (AGV and automatic straddle carrier) was conducted by Park et al (2007).

In addition to these commercially implemented automated container terminals, several new conceptual terminals are also proposed. They are Linear Motor Conveyance system based ACT (LMCS-ACT), Grid Rail system based ACT (GR-ACT), and Automated Storage/Retrieval system based ACT (AS/RS-ACT). LMC system has been constructed and successfully tested in Eurokai Container Terminal, Hamburg by P. A. Ioannou (2000). The vehicles are replaced with shuttles that are moving on the linear motor conveyance system. The shuttles can be considered as vehicles moving on a fixed path. GR-ACT is proposed by Sea-Land and August-Design. Overhead rail is utilized in the storage yard for loading and unloading containers. It uses linear induction

motors, located on overhead shuttles that move along a monorail above the terminal. The containers are stacked beneath the monorail. The concept of the overhead grid rail system was used to design, simulate and evaluate a GR based ACT system by Ioannou et al. (2004). The AS/RS system in AS/RS-ACT has three major hardware components: storage racks, storage and retrieval machines (SRM), and a shuttle. Rack structure is used to store the containers. A typical AS/RS structure module consists of single-deep stored unit loads and two parallel long narrow racks and an aisle between them. The SRM moves along the guide rails installed in the aisle. A shuttle is mounted on the SRM for pick-up and delivery in storage cells and P/D stands. Khoshnevis et al. (2006) proposed a simulation model which is developed for quantitative comparison of AS/RS systems. A comparison of the productivity among these ACTs can be found in the paper by Liu et al (2002).

Compared to the ACTs discussed above, another two types of ACTs appear to be of interest. One type is called the Frame Bridge based ACT (FB-ACT) and the other one is called the Goods Retrieval and Inventory Distribution based ACT (GRID-ACT). These two ACTs are constructively changed, which bring exclusive benefits from the design. More details of these two ACTs are discussed in Sections 1.4 and 1.5, respectively.

1.3 Frame Bridge Based Automated Container Terminal

Shanghai Zhenhua Heavy Industries Co. Ltd. recently designed a real-size prototype automated container terminal that utilizes frame bridges, rail-mounted frame trolleys and ground trolleys to transport containers between quay side and yard side. Figure 1.4 shows the layout of this ACT (Zhen et al. (2012))

Figure 1.4 illustrates that the newly designed ACT is composed of three major parts: the quayside operation area, transportation area and storage area. Rail bridges are built to transport containers, which can be categorized into two types. One is laid parallel to the berth and interfaced with quay cranes, denoted as berth rails. The other is laid parallel to the yard block and interfaced with yard cranes, denoted as yard rails. The berth rails are constructed above the ground, and the yard rails are laid at ground level. These two parts of rail bridges cross each other perpendicularly. The transfer platform (TP) that sits on the berth rail provides an interface between the berth rail and the yard rail.

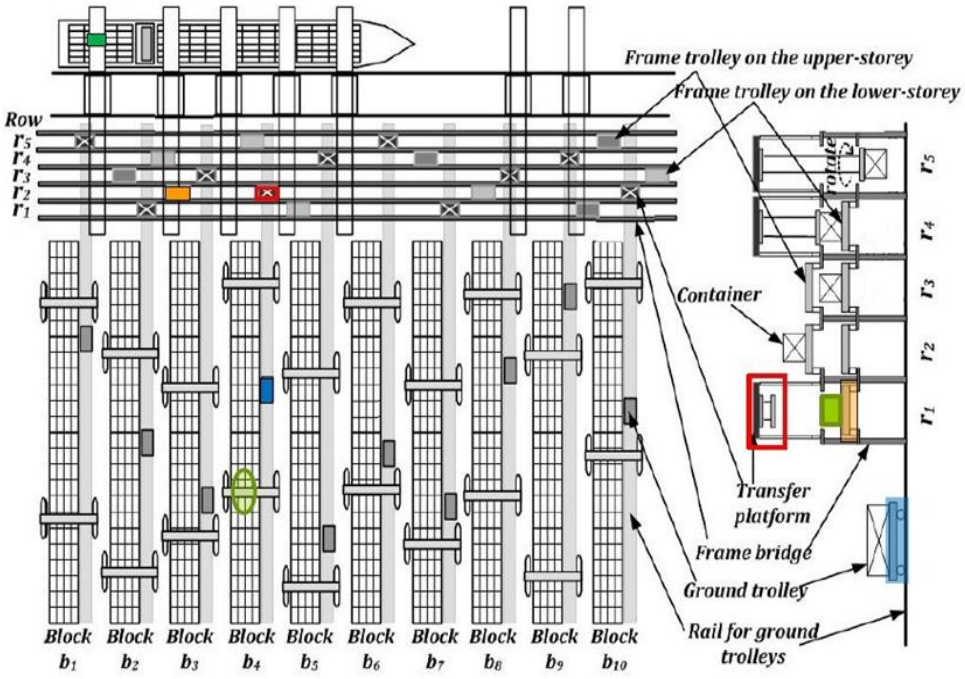


Figure 1.4 Layout of the new design ACT

The trolleys mounted on the berth rails are called frame trolleys (FTs). They are used to transport containers between QCs (quay cranes) and TPs. The trolleys mounted on the yard rails are called ground trolleys (GTs). They are used to transport containers between yard cranes and TPs. Because these two

parts of the rails are perpendicular to each other, the TP is used to rotate the containers 90 degrees during the handover of containers between these two types of trolleys. Because the terminal is covered by frame bridge rails, it is called Frame Bridge-based ACT (FB-ACT). This new ACT system has been proposed on the eastern side of Caofeidian Port (in Tianjin, China). In addition, SSA Marine is considering installing this ACT system at Long Beach in the coming years.

Compared to other container terminal designs, the advantages of FB-ACT can be summarized as follows:

1) The trolleys mounted on the rails can move at a high speed. They can reach speeds of up to 14 mph, whereas an AGV can travel up to 5 mph when fully loaded.

2) The productivity of yards is increased. In the AGV-ACT system, the YC needs to travel a longer distance on average to pick up/store containers. This is because the handover of containers between the yard crane and AGV occurs at the end of the block, whereas in the FB-ACT, the handover of containers can occur in the block and the GT speed is considerably faster than the YC.

3) This system is a flexible design because the capacity can be increased by adding an additional layer of rails below or on top of the original rails in the future when it is needed.

4) This system is green and requires less labor. FTs, GTs and TPs are powered by electricity instead of diesel.

However, the operation of the FB-ACT can be challenging. The handshakes among devices are one of the major issues to address because more equipment is required in this system. For a container handled by the QC, the handshake is between the QC and FT. For a container handled by the TP, the handshake is among the FT, GT and TP. Moreover, the system requires adequate traffic control, especially for FTs. Because the trolleys are mounted on the rails, they cannot cross over each other. The conflict of the trolleys can greatly affect the performance of this system. The research related to this type of configuration is limited. In our study, we are concerned with the problem of managing the involved resources and considering the vehicle collisions as well, to accomplish the container transportations between vessels and storage yard.

1.4 Goods Retrieval and Inventory Distribution based Automated Container Terminal

As the land becomes the scarcest resource in the cities, high land utilization becomes a significant benefit, which tends to be captured by the future generation ports. In addition, less handshakes among equipment is preferable when considering higher efficiency. It is quite challenging to maintain high efficiency under the circumstance of many handshakes. A new concept of ACT called Goods Retrieval and Inventory Distribution (GRID) based ACT proposed by BEC industries LLC, is a promising solution for the future generation ports. According to the study by Brian et al., it can save approximately 47% of the land for a layout with the scale of 31,200 TEUs.

The GRID system is a new concept to optimize land utility and improve productivity in a container terminal. A 3D model of the port implementing the GRID system is shown in Figure 1.5.

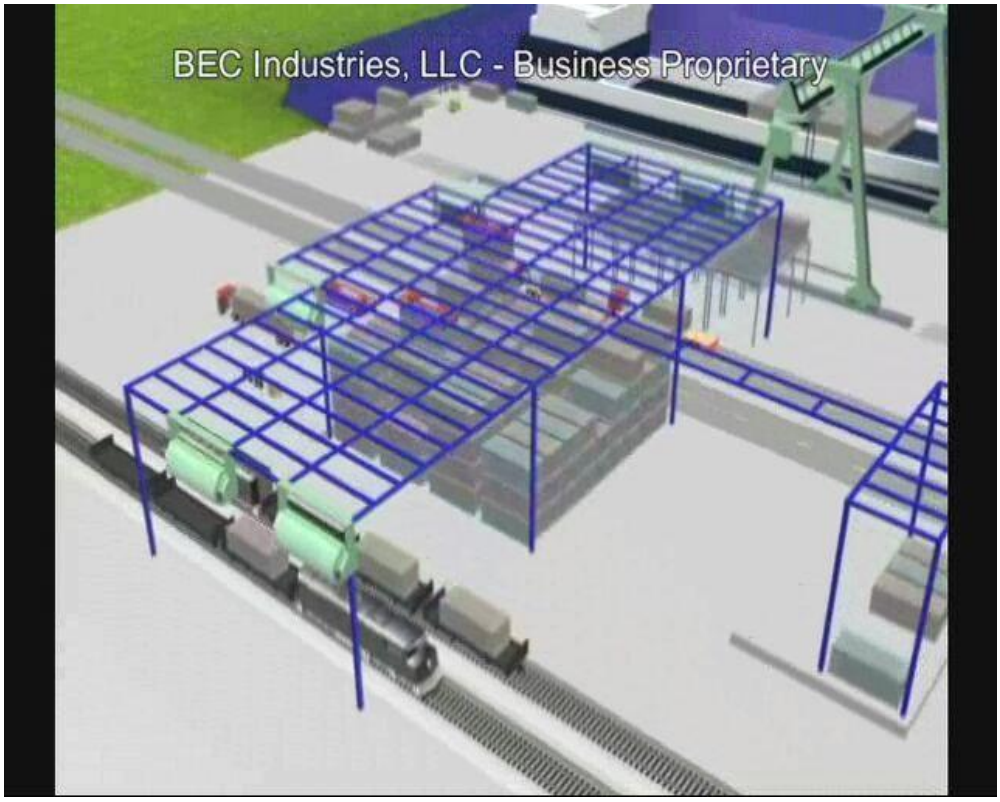


Figure 1.5 3D model of GRID system

It mainly consists of three components: transfer units (TUs), overhead rails and transfer tables. The TUs move along the rails which are bi-directional. It also can make a turn at the intersection of a horizontal rail and a vertical rail by switching the pair of wheels installed on the TU. Each TU has two pairs of wheels that are respectively responsible for vertical movement and horizontal movement. When a TU moves horizontally, the corresponding pair of wheels will be embedded in the horizontal rail and the other pair of wheels will be inactive. When the TU changes the direction from horizontal move to vertical move, the pair of wheels corresponding to horizontal move will be released from the horizontal rail and become inactive, while the other pair of wheels corresponding to vertical move will be embedded in the vertical rail. Once the procedure of the switch is accomplished, the TU can start to move vertically. The procedure is similar when the TU changes the direction from vertical

move to horizontal move. TUs are used to transfer the containers between storage yard and vessels. Transfer tables are built on the area at the quayside. They are used as buffers to decouple the operations of TUs and quay cranes. The flow of activities involving a container from vessel to storage yard can be described as follows: A quay crane picks up the container from the vessel and then puts it down on the transfer table. An empty TU arrives at the transfer table and picks up the container. The TU carrying the container moves along the rails to approach the specific slot where the container will be temporarily stored. Once the TU releases the container on the yard storage, it will move to handle other containers assigned to it. The procedure of moving containers from yard storage to vessels is similar in a reverse manner. In our study, we only focus on the operations of the inbound containers from vessels and outbound containers to vessels while the inbound containers from landside or outbound containers to landside is not in the scope of the study. Considering the scope of our study, the layout structure of the GRID system is simplified and it is shown in Figure 1.6.

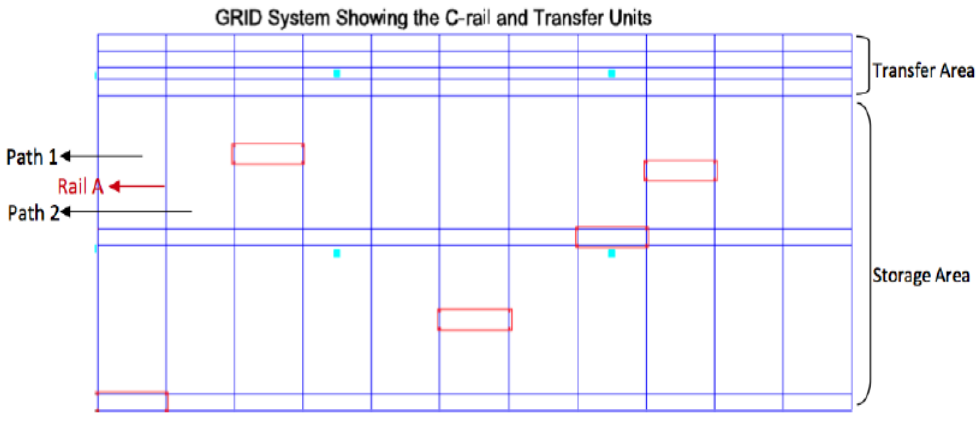


Figure 1.6 GRID system structure

The overhead rails cover two parts: transfer area and storage area. The storage area is where containers are stored. The transfer area is the place where the buffers are built. The operations on the buffers are shown in Figure 1.7. Half of the buffer is in the transfer area and the other half is under the quay crane. It is a two-layer buffer where the upper level is dedicated to inbound (discharging) containers and the lower level is dedicated to outbound (loading) containers. Each layer of the buffer is constructed by a belt conveyer. For an inbound container, the quay crane picks it up from the vessel and then puts it down on the buffer. Once the container is stable on the belt conveyer, the belt conveyer starts to convey the container to yard-side. The belt conveyer will be stopped when the container is under the location where a TU can approach to handle it. It is similar for the outbound container. When a TU carrying an outbound container arrives at the location above the buffer, it starts to put down the container on the belt conveyer at the lower layer. Once the container is

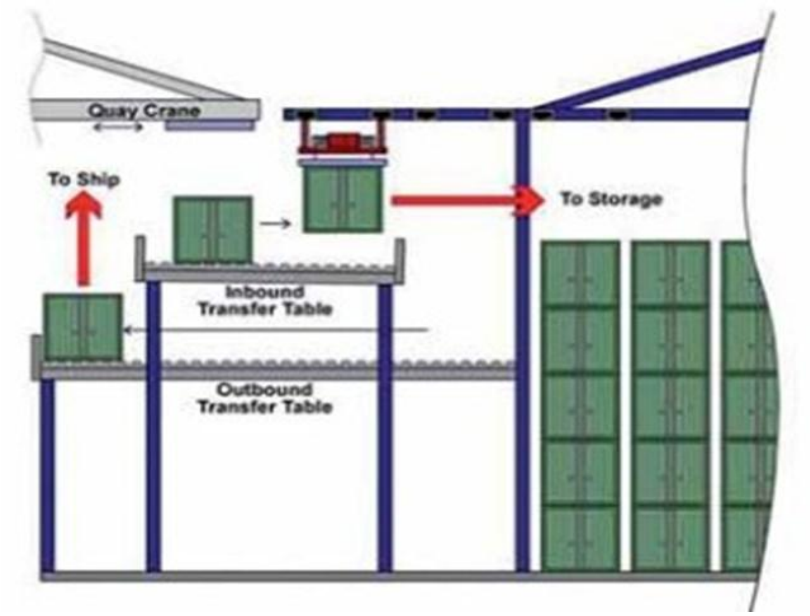


Figure 1.7 Operations on the buffers

released on the conveyer, the conveyer starts to move the container to the quay-side so that a quay crane can handle it. The conveyer will also be stopped once the container arrives at the end of the buffer. However, quay cranes and TUs may still be delayed if the buffer is full. In our study, considering the fact that the QC is always the bottleneck in a container terminal system, we assume the capacity of the buffer on the transfer table is only one.

Because the investment of the rails is high, the horizontal rails are not built on every row in the storage area and the contiguous vertical rails share one track in the middle of these two rails. However, two contiguous horizontal rails do not share the same track, which means the movement of TUs on the contiguous horizontal rails will not impact each other. This can improve the efficiency of the system by reducing the delay time or the detour of TUs.

According to the operations in the GRID system, we can find that the assignment of containers to TUs and the route of TUs to complete their jobs are the crucial issues that affect the productivity of the system. A good schedule can maintain high productivity of quay cranes, and reduce the waiting time as well as detour of TUs during the routing procedure. Therefore, the motivation of our study is to find a promising scheduling for the GRID system so that the terminal can approach high efficiency under the circumstance of high land utilization.

1.5 Contribution of the thesis

The contributions of this thesis can be listed as follows:

(1) One new conceptual ACT called the FB-ACT is studied. To the best of our knowledge, our study is the first work to provide the details of the operations of the FB-ACT system. We emphasize on the management of resources in order to reduce the delay that happens during the handshakes and avoid vehicle collisions which significantly affects the efficiency of the system. A mathematical programming model is developed to solve the problem. Because the model cannot be solved when the scale of the problem is large, we develop a tree structure algorithm. The motivation of the algorithm is to decompose the problem to the sub-problems with small scale which can be solved efficiently. By the decomposition, the variables of the mathematical programming model for the conflict-free routing problem are greatly reduced. Our algorithm can obtain much more promising solutions compared with a heuristic algorithm using randomly dispatching rule and First Come First Serve (FCFS) rule.

(2) Another new conceptual ACT called the GRID-ACT is studied. In the GRID-ACT, the main issues are the vehicle dispatching and conflict-free routing problems. Given the intrinsic difficulty of this problem, the tree structure algorithm is also adopted to solve the complicated problem and a column generation based algorithm is developed to solve the sophisticated conflict-free routing problem. Because of the complexity of the conflict-free routing problem, the typical column generation algorithm cannot solve the problem well. Thus, we will continue the search by using the information from the solution obtained by the typical column generation algorithm. New columns can be generated by adding new constraints to the master model. The motivation of the new constraints is to prevent the

TUs from running in the congested area, so that we can obtain a promising and feasible solution for the conflict-free routing problem. Our algorithm can obtain satisfying solutions when compared to the algorithm with heuristic rules.

(3) By analyzing the performance of FB-ACT and GRID-ACT, we proposed a new design of the container terminal. The main drawback of FB-ACT is the relatively large waiting time because of the prevention of vehicle collisions. The vehicles in GRID-ACT can flexibly select paths to prevent vehicle collisions, which can reduce the waiting time. However, because of the slow speed of the vehicles and the long distance of the paths, GRID-ACT has to spend a relatively large time on traveling. This newly designed terminal is a hybrid of FB-ACT and GRID-ACT, which appears to offset the drawbacks of these two kinds of terminals. The problem about the vehicle dispatching and conflict-free routing is also solved by the tree structure algorithm. The subtask sequence is critical when we solve the routing problem. We propose two methods to determine the subtask sequence. One is based on the least delay time when neglecting vehicle collisions. The other one is based on the estimated starting time of the subtasks calculated by the column generation algorithm. The first method is much faster than the second one, which is used in the screening procedure, while the second method is used to determine the decisions involved with the routing problem. This newly designed terminal outperforms FB-ACT as shown by a comparison study.

1.6 Organization of the thesis

This thesis consists of six chapters, which are organized as follows.

Chapter 2 reviews the related studies on the vehicle dispatching problem and conflict-free routing problem, as well as the literatures that solve these two problems simultaneously.

Chapter 3 proposes the study on the FB-ACT. A mathematical model is developed to solve the vehicle dispatching and conflict-free routing simultaneously. By decomposing the problem, the conflict-free routing problem can be solved optimally by a mixed integer programming.

Chapter 4 describes the research on GRID-ACT. The problem is modeled as a set partitioning problem. A column generation algorithm with a further search procedure algorithm is proposed to obtain an integer solution.

Chapter 5 develops a new design of the terminal, which is a hybrid of FB-ACT and GRID-ACT. We develop two methods for solving the routing problem. A comparison study is conducted, which shows that the new design can be more efficient than FB-ACT.

Finally, in Chapter 6, we consolidate the findings from the previous chapters and future research directions are also discussed.

CHAPTER 2 Literature Review

There are numerous research works in the area of container terminal operations. However, the published works related to our topic is not much. Thus this literature review will also contain the areas having large parallels with our topic. The major works are scheduling problems of quay cranes and yard cranes, vehicle dispatching and routing problem in ACT system, automated warehouse. Literature reviews on port operations can be found in Vis and de Koster (2003), Steenken et al. (2004) and Robert Stahlbock and Stefan (2008). After that, we will also conduct a reviews related to the major algorithms applied in our works.

2.1 Quay Crane Scheduling and Yard Crane Scheduling

Quay cranes and yard cranes are the critical resources in container terminals. A well scheduling of them can significantly improve the efficient of the system. The ideas of the works in this area are helpful to develop algorithms for our topic. Thus, the research works of quay crane scheduling and yard crane scheduling are discussed.

2.1.1 Quay Crane Scheduling

The quay crane scheduling problem aims to solve the allocation of quay cranes to the containers with non-interference. It is called Quay Crane Scheduling with Non-Interference constraints Problem (QCSNIP). Kim et al. (2004) proposed a mixed integer programming model which considers various constraints related to the operation of QCs. The cross over among QCs is avoided by constraints in the MIP model. A branch and bound method and a heuristic algorithm called greedy randomized adaptive search procedure

(GRASP) are proposed to solve the problem. Lee et al. (2008) propose a more concise MIP model for the QCSNIP. They proved the QCSNIP is NP-complete. A genetic algorithm is proposed to obtain its near optimal solutions. Bierwirth et al. (2009) present a revised optimization model for the scheduling of quay cranes and propose a heuristic solution procedure. The heuristic takes advantage of efficient criteria for branching and bounding the search, which is applied for searching a subset of above average quality schedules. Ng et al. (2006) propose a heuristic algorithm that decomposes the difficult multi-crane scheduling problem into easier sub-problems by partitioning the ship into a set of non-overlapping zones. The sub-problems for each possible partition are solved optimally by a simple rule. An effective algorithm is developed to find tight lower bounds. The results show that the heuristic can indeed find effective solutions with 4.8% above their lower bounds. Moccia et al. (2006) formulate this problem as a VRP with side constraints including precedence relations between vertices. The objective is to minimize the weighted sum of the completion time of a single vessel and the idle times of cranes which originate from interferences between cranes since cranes roll on the same rails and a minimum safety distance must be maintained between them. Marcello et al. (2007) decompose the quay crane scheduling problem into a routing problem and a scheduling problem. The routing problem is solved by a Tabu search heuristic, while a local search technique the minimizing the longest path length in a disjunctive graph, is used to generate the solution of the scheduling problem. Cononaco et al. (2008) present a queuing network model to solve this problem with the objective of minimizing the turnaround time of the vessels.

2.1.2 Integration of berth and quay crane scheduling problem

The integration of berth and quay crane scheduling problem is an important research topic in this area because it can help achieve a more significant improvement on the efficiency of the ports than the scheduling of berths and quay cranes separately. Kim et al. (2005) propose an integer programming model by considering various practical constraints. A two-phase solution procedure is developed for solving the mathematical model. The first phase aims to solve the berth allocation problem which determines the berthing position and time of each vessel and the number of cranes assigned to each vessel at each time segment. The second phase concentrates on solving the quay crane scheduling problem based on the solution found from the first phase. Imai et al. (2008) propose a heuristic algorithm by employing a genetic algorithm. The fitness value of a chromosome is found by crane transfer scheduling across berths, which is determined by a maximum flow problem-based algorithm based on the berth allocation problem solution defined by the chromosome. Liang et al. (2009) solve the problem of determining the berthing position and time of each ship as well as the number of quay cranes assigned to each ship, with the objective of minimizing the sum of the handling time, waiting time and the delay time for every ship. They find an approximate solution by combining genetic algorithm with heuristic. Chang et al. (2010) develop a hybrid parallel genetic algorithm (PGA), which combined parallel genetic algorithm and heuristic algorithm. The PGA is used to attain the sub-optimal solution for the BAP and QCAP. The heuristic algorithm is aimed at generating feasible solutions for population initialization, which can reduce the solution dimension. Lee et al (2010) propose a genetic algorithm to

obtain near optimal solution. The chromosome represents a sequence of container ships where a berth allocation can be constructed based on this sequence. An approximation algorithm is developed to solve the quay crane scheduling problem after the BAP is solved. Vacca et al. (2013) propose a model which is solved via column generation. An exact branch and price algorithm with several accelerating techniques is implemented to obtain optimal integer solutions to the problem. This is the first exact branch and price algorithm for the integrated planning of berth allocation and quay crane assignment.

2.1.3 Yard Crane Scheduling

The yard cranes are the equipment in the storage yard which can significantly affect the productivity of the container ports. Thus, yard crane scheduling is widely studied. Some research works solve the problem with single yard crane. Kim and Kim (1999) consider the dispatching of a single yard crane by formulating it as a transportation problem. The visiting route is determined with a dynamic programming procedure. Ng and Mak (2005) study the scheduling of a single yard crane for a given set of loading and unloading containers with different ready times. A branch and bound algorithm is implemented to solve the problem where efficient heuristics are proposed to find the lower bounds and upper bounds. Ng et al. (2005) study the problem of scheduling a yard crane with the objective of minimizing the sum of job waiting times. A heuristic algorithm based on a sequential sequence-building approach is proposed to solve the problem. The results show that the proposed heuristic can find effective solutions for the problem.

More research works are conducted with the consideration of multiple yard cranes. One of the critical constraints of this multiple yard crane scheduling problem is to prevent the yard crane interference. Ng. W. C. (2005) develops a dynamic programming based heuristic to solve the scheduling problem. It is a two phase algorithm where the first phase is to simplify and decompose the multiple yard crane scheduling problem into m independent single yard crane scheduling problems by partitioning the yard zone. Each sub-problem is solved by a heuristic rule based on the smallest job completion time. The second phase is to employ a job reassignment procedure to improve the schedule obtained in the first phase. The results show that the proposed algorithm can find solutions which are on average 7.3% above their lower bounds. Jung and Kim (2006) study the problem of scheduling multiple yard cranes to serve multiple quay cranes, where the adjacent yard cranes working in the same block have interference with each other. The algorithms based on GA and SA approaches are proposed to schedule the travelling route of the yard cranes and number of containers to pick up in each yard bay. Li et al. (2009) develop an efficient model for yard crane scheduling by taking into account realistic operational constraints such as inter-crane interference, fixed yard crane separation distances and simultaneous container storage/retrievals. They propose heuristics and rolling-horizon algorithm to solve the problem quickly with yielding near optimal solutions. Javanshir and Ganji (2010) propose a genetic algorithm to solve the yard crane scheduling problem with non-interference constraints. Chang et al. (2011) propose a novel dynamic rolling-horizon decision strategy for the yard crane scheduling. An integer programming model is built at the beginning to minimize the total task

delaying at blocks. A heuristic algorithm along with a simulation model is then applied. He et al. (2010) proposed a hybrid algorithm, which employs heuristic rules considering the workload of the yard crane and the number of yard cranes in the block, and then a parallel genetic algorithm is employed. Computational results show that the proposed method can solve the problem efficiently.

2.2 Vehicle Planning Problem

The vehicle planning problem aims to solve the problems like determining the vehicle to deal with certain containers considering a set of constraints. These constraints include: avoiding of vehicle collisions and deadlocks, and satisfying the time-window constraints. In our review, we will focus on the vehicle dispatching problem, vehicle routing problem and the integration of these two problems.

2.2.1 Vehicle Dispatching Problem

The vehicle dispatching is concerned with the problem that determines which vehicle transports which container to achieve certain goals. In the port area, the vehicles are the trucks, AGVs or straddle carriers. It is also an important topic in the research area of the port operations. Zhang et al. (2005) present three MIP models for the vehicle dispatching problem in a container terminal to determine the starting times of jobs as well as the work sequence of vehicles. The models only consider the unloading phase of a vessel in one berth and the vehicles are assumed to be dedicated to a certain quay crane. Two of these models can obtain a lower bound for the optimal value, while the complicated model can be solved by a greedy algorithm, which is capable of solving large scale problems. Kim et al. (2004) propose a heuristic algorithm

where two main steps of feasibility checking and delaying events are repeated. During the searching procedure, only the most imminent tasks are considered in the dispatching. Bise et al. (2005) develop easily implementable heuristic algorithms and identify both the absolute and asymptotic worst-case performance ratios of these heuristics. The greedy algorithm is based on the way that assigns container jobs to the vehicle with earliest arrival time. A refined greedy algorithm is proposed by considering a simple look-ahead rule when the algorithm is applied to the multiple crane model. Lee et al. (2010) solves the vehicle dispatching problem while considering the quay crane and yard crane capacity. A heuristic algorithm that combines the genetic algorithm and minimum cost flow (MCF) network model is proposed to tackle the problem. The ready time for jobs is used as the representation of the chromosome, while the MCF model is then used to decode the chromosome to determine the job sequence for the vehicles.

As the AGVs are implemented in container terminals in these two decades, the vehicle dispatching concerning AGVs becomes a critical part of the research in the port operations. Kim et al. (1999) formulate a mixed integer linear programming model for dispatching AGVs with the objective of minimizing the delays of the vessels and the traveling time of the AGVs. Liu et al. (2002) study the performance of four different heuristic AGV dispatching rules by simulation method. The four different vehicle-initiated AGV dispatching rules are: longest travel distance rule; shortest travel distance rule; random rule; minimum yard crane queue size rule. The results show that the minimum yard crane queue size rule can achieve the best throughput performance. Grunow et al. (2006) propose a simulation study of AGV dispatching strategies in a

seaport container terminal. A typical on-line dispatching strategy commonly adopted from flexible manufacturing systems is compared with a pattern-based offline heuristic proposed in the paper. Results of the simulation study reveal that the pattern-based heuristic clearly outperforms the on-line strategy. Briskorn et al. (2006) propose a formulation to avoid estimates of driving times, completion times, due times and tardiness, based on a rough analogy to inventory management and is solved using an exact algorithm. The results obtained from the simulation method show that the inventory-based model leads to better productivity on the terminal than the due-time-based formulation.

2.2.2 Vehicle Routing Problem

The vehicle routing problem (VRP) is the problem of designing optimal delivery or collection routes from one or several depots to a number of geographically scattered cities or customers, subject to side constraints. It is firstly proposed by George Dantzig and John Ramser in 1959. (The truck dispatch problem). In the container terminal, the transporters like trucks, straddle carriers or AGVs also face the VRP when they transfer the containers between the storage yard and vessels. The critical issue of the VRP in the container terminal is to avoid the collisions among the vehicles. Kim et al. (1991) propose an algorithm based on Dijkstra's shortest-path method. A concept of time window graph is introduced in which the node set represents the free time windows and the arc set represents the reachability between the free time windows. By using the Dijkstra's shortest-path method, a conflict-free route for each vehicle is generated sequentially. Kim et al. (1993) present a conservative myopic strategy to coordinate the movements of vehicles in a

bidirectional AGV system. Oboth et al. (1999) address design and operational control issues for an AGV based material handling system. An effective route-generation technique based on a bi-directional network that provides conflict-free routes for multiple AGVs with varying speeds is presented. This technique generates the routes sequentially when considering the demand selection policies, demand assignment policies and idle AGV positioning policies. Qiu et al. (2001) present a bi-directional path layout and an algorithm for routing AGVs. Based on the path topology and the routing algorithm, provably sufficient and necessary conditions are obtained to achieve the conflict-free routes and shortest possible time. Jeon et al. (2011) determine the shortest-time routes inclusive of the expected waiting times instead of the simple shortest-distance routes. They propose a method for estimating the waiting time for each vehicle that results from the interferences among vehicles during travelling. The estimation of the waiting times is achieved by using the Q-learning technique and by constructing the shortest-time routing matrix for each given set of positions of quay cranes. Möhring et al. (2005) present an algorithm which avoids collisions, deadlocks and livelocks for the problem of routing AGVs. A shortest path with time-windows is first determined by real-time computation, and a following readjustment is implemented to these time-windows. The results of comparing to a static routing approach show that the algorithm has an explicit advantage. Zeng et al. (2008) present a mathematical model for general container routing in mesh yard layouts. A simple routing algorithm based on the model is proposed to choose suitable vehicle speed such that the vehicles using the same junction will arrive at different points in time to prevent conflicts. Numerical results

show that the routing scheme has good performance and the conflicts are prevented. Maza et al. (2001) propose a robust predictive method of routing without conflicts, consisting of adding a layer of real time control. Maza et al. (2005) propose an approach which combines the optimized pre-planning algorithm and the real-time routing algorithm. The algorithm consists of two stages: the first stage is the control stage, where a pre-planning method is used to establish the fastest conflict-free routes for AVGS; the second stage is used to avoid conflicts in a real-time manner when it is needed. Singh et al. (2002) propose a multi-agent approach to the operational control of AGVs by integration of path generation, enumerating time-windows, searching interruptions, adjusting waiting time and making decisions on the selection of routes. It presents an efficient algorithm and rules for finding a conflict-free shortest-time path for AGVs by using loop formation in a flow path network to deal with the parking of idle vehicles without obstructing the path of moveable AGVs.

2.2.3 Integration of Vehicle Dispatching Problem and Conflict-free Routing Problem

The integration of vehicle dispatching problem and conflict-free routing problem is a complicated problem. This problem can be solved to optimality only when the number of vehicles is very small. Desaulniers et al. (2003) model the problem by a set partitioning formulation. The model is then solved to optimality by a column generation method, where a branch-and-cut exploration tree is applied. Due to the complexity of the problem, it can only be solved up to four vehicles within controllable time. Corréa et al. (2004) propose an approach which combines constraint programming for vehicle

dispatching and mixed integer programming for routing without conflict. These two methods are iteratively executed until an optimal solution is found. The approach also limits the number of AGVs to six. Most of the research works implement heuristic algorithms to solve the problem under a realistic environments. Ghasemzadeh et al. (2009) present an integrated algorithm for scheduling and routing of AGVs in mesh-like systems. The scheduling algorithm aims to achieve the goals including: prediction and prevention of conflicts; arbitrary choice for AGVs to traverse the shortest path from source to destination; effect of priority policies to the scheduling result; no theoretical limitation on the number of participating AGVs. The routing algorithm aims to reduce the average number of conflicts which is closely related to the scheduling algorithm. Nishi et al. (2011) decompose the problem into two levels: the upper level master problem of task assignment and scheduling; and the lower level routing subproblem. The master problem is solved by using Lagrangian relaxation and a lower bound is obtained. Two types of cuts are proposed to exclude previous feasible solutions before solving the master problem again. One of the cuts is the capacity constraint, while the other one is for restricting the feasible region of the Lagrangian relaxation problem for the upper level master problem. Nishi et al. (2012) present a Petri net decomposition approach to solve the problem in dynamic environments. Static problems for finding near optimal dispatching and conflict-free routing are solved first. The entire Petri net is decomposed into task and AGV subnets, which is solved by the penalty function method. A deadlock avoidance method is embedded to ensure the feasibility and quality of the solution.

2.3 Automated Warehouse

An automated warehouse is a facility where all or some of the tasks related to storing, retrieving, and moving inventory are carried out by automated systems. Automatic storage and retrieval (ASRS) is used in warehouses where robots store materials, selecting the best location on the basis of available space and inventory rotation needs, and retrieve those materials when they are needed. The robots in this system are called automated guided vehicles (AGVs). The routing of AGVs in this system is widely studied. Because of the complicated of this problem and the computing time restriction, various heuristic algorithms are proposed to solve this problem.

Amato et al. (2005) develop a control algorithm for the management of an automated warehouse system. A model is built up by using the colored time Petri nets framework, to optimize the operations of the cranes and the operations of the shuttle, respectively. The proposed architecture and control algorithms are applied to a real plant. Liu (2011) proposes a new multi-objective mathematical model to the integrated scheduling problem on the basis of a typical warehouse layout. A heuristic algorithm based on genetic algorithm is proposed to solve the problem. Vangeri and Hebbal (2014) present the picking route optimization of automated warehouse which is solved by modified genetic algorithm. The initial nodes are generated in a random manner for finding the complete route, and then applied the input parameters to modified genetic algorithm by which obtained the optimum route with low distance and time. Li et al. (2011) study the path optimization of automated warehouse. A mathematical model of stacker operation is built to minimize the length of operation path and operation time. The model is

solved by using the ant colony optimization method. Jiang et al. (2010) establish the corresponding simplified model of the warehouse and picking path model. Using the graph theory algorithm, it propose for the stacker picking path optimization of the existing warehouse. Wang et al. (2008) introduce the theory and methods of modeling automated warehouse with colored time Petri net. A colored token represent a job piece, RGV or crane and the color of each token was the residual sequence of places for a job piece visiting. He et al. (2007) propose a modular and computerized model and characterized control flow of the resources, and token colors were defined as the routes of storage/retrieval operations Lis et al. build a mathematical model to analysis of the optimization for AGVs in automated warehouse. A framework of a dispatching approach with genetic algorithm is proposed to solve the scheduling problem of AGV in automated warehouse. The coding, selection and mutation is discussed considering the characters of the problem. Deng et al. (2013) established a mathematical model of scheduling for the storage and retrieval path, which takes the shortest path as the optimization goal. An immune selection combined with discrete particle swarm optimization is proposed to optimize the path, to avoid falling into local optimum prematurely, and to find the optimal solution easier.

2.4 Filtered Beam Search

Filtered beam search algorithm is proposed by Ow et al. (1988). It is a kind of tree structure based algorithm. Evaluation functions are used to assess the performance of the nodes in the tree. A local evaluation which is fast but less accurate will first be applied to screen the nodes with poor performance. After the filtering procedure, a global evaluation which is more accurate but

expensive in computation effort will be applied to select the beam nodes. Filtered beam search algorithm can produce high quality solutions with a controllable computation effort. The algorithm is widely used to solve combinatorial optimization problems in various areas.

Ow et al. (1989) solve the problem of scheduling a given set of jobs on a single machine to minimize total early and tardy costs. The priority search using a priority function is applied as the evaluation function for screening. The probe search using cost function is the evaluation function to select the beam nodes. De and Lee (1990) solve the problem for scheduling jobs in a flexible manufacturing system (FMS). The algorithm uses a frame-based knowledge representation scheme and a problem-solving strategy based on filtered beam search. Nair et al. (1995) solve the product line design problem using the beam search approach. The solutions are closer to the optimal, have smaller standard deviation over replicates, and take less computation time. Also optimal solutions are obtained more often and a number of "good" product lines are identified explicitly. Sabuncuoglu and Bayiz (1999) develop a beam search based scheduling algorithm for the job shop problem. The makespan and mean tardiness are used as the performance measures. The results indicate that the beam search technique is a very competitive and promising tool to obtain good solutions efficiently. Kim et al. (2004) apply a beam search algorithm to solve the load-sequencing problem in container terminals with the objective of maximizing the operational efficiency of transfer cranes and quay cranes while satisfying various constraints on stacking containers onto vessels. Wang and Lim (2007) solve the berth allocation optimization problem by transforming it into a multiple stage

decision making procedure. A new multiple stage search method is proposed based on the beam search algorithm.

2.5 Column Generation Algorithm

Column generation is an efficient algorithm for solving large linear programs. Many linear programs are too large to consider all the variables explicitly. Since most of the variables will be non-basic with a value of zero in the optimal solution, only a subset of variables needs to be considered in theory when solving the problem. The algorithm generates only the variables which have the potential to improve the objective function. The algorithm was initially proposed by Gilmore and Gomory (1961). It is fruitfully applied in various areas, especially in the areas like routing and scheduling.

This paragraph will present some works that are using the algorithms based on column generation to solve the vehicle routing problem. Desrosiers et al. (1984) use a column generation approach in which the columns are generated by a shortest-path-with time windows algorithm. Agarwal et al. (1989) propose a computationally viable algorithm based on column generation algorithm where implementation strategies based on theoretical as well as empirical results are developed. Taillard E. D. (1999) presents a heuristic column generation method for solving vehicle routing problems with a heterogeneous fleet of vehicles. The column generation is based on the adaptive memory procedure which uses an embedded taboo search. Desaulniers et al. (2002) propose accelerating strategies implemented in conjunction with column generation to solve the vehicle routing and crew scheduling problems. The techniques are embedded in the five phases of the solution process: pre-processor, subproblem, master problem, branch-and-

bound, and post-optimizer. Choi and Tcha (2007) apply a column generation based approach for a vehicle routing problem with a heterogeneous fleet of vehicles having various capacities, fixed costs and variable costs. A couple of dynamic programming schemes developed for the classical vehicle routing problem are emulated with some modifications to efficiently generate feasible columns. Baldacci et al. (2008) present a new exact algorithm for the Capacitated Vehicle Routing Problem based on column generation with additional cuts which correspond to capacity and clique inequalities. The new columns are generated with the reduced costs that are smaller than the gap between an upper bound and the lower bound. Mourgaya and Vanderbeck (2007) present a truncated column generation procedure followed by a rounding heuristic to construct the approximate solutions.

This paragraph will present the works that implement column generation algorithm to solve the scheduling problem. Appelgren et al. (1969) apply column generation algorithm for a ship scheduling problem. Problems with about 40 ships and 50 cargoes are solved in about 2.5 min. Some integer programming experiments have been made in order to resolve the fractional cases. Desrochers et al. (1989) propose a column generation approach to solve the transit crew scheduling problem which has to create minimal cost bus driver schedules respecting both the collective agreement with labor unions and the bus schedule. Chen and Powell (1999) implement column generation algorithm to solve a class of problems of scheduling n jobs on m identical, uniform, or unrelated parallel machines with an objective of minimizing an additive criterion. Ribeiro and Soumis (1994) present a formulation to the multiple-depot vehicle scheduling problem as a set partitioning problem with

side constraints. The continuous relaxation is amenable to be solved by column generation. Van et al. (2000) discuss how the column generation can be applied to alleviate the difficulties associated with the size of time-indexed formulations. Bard and Purnomo (2005) present a methodology for scheduling nurses in which several conflicting factors guide the decision process. The methodology is a column generation approach that combines integer programming and heuristics. A double swapping heuristic is used to generate the columns.

Wilhelm Wilbert E (2001) and Vanderbeck (2005) conduct a review of using column generation in integer programming problems.

CHAPTER 3 Vehicle Dispatching and Conflict-free Routing Problems Under FB-ACT

3.1 Problem Definition

This thesis considers the problem that determines the assignment of FTs to deliver containers under the cooperation of related resources, such as QCs, TPs and GTs, to ensure efficiency of the operation of the FB-ACT system. The goal of our study is to minimize the makespan of a given number of container jobs.

Compared to existing terminals, more equipment is used in the FB-ACT, which requires more handshakes among different types of devices. In the AGV-based ACT, two types of handshakes exist: between QC and AGV and between YC and AGV. However, there are more handshakes in FB-ACT. The handshakes can be categorized into four types: between the QC and FT, FT and TP, TP and GT and GT and YC. Therefore, the operations of the FB-ACT can be challenging because of the handshakes between different devices. Another challenge is the conflict among FTs on the same track. Delays will occur during the process of handoffs and the process of avoiding FT conflict. An effective schedule is necessary to significantly reduce the delay. To solve the problem, we must determine the assignment and sequence of containers in each resource, such as the QC, FT, TP and GT. We also need to prioritize FTs when there is an FT conflict.

The following assumptions are introduced for the formulations of the problem:

(1) Each QC contains its own job sequence list.

(2) A TP is dedicated to several neighboring yards.

(3) Only one GT is running on each yard rail, i.e., there is no conflict among GTs.

(4) The yard location of each container job is known.

(5) The travelling speed of empty and loaded FTs and GTs is the same.

The elements in our problem can be summarized as:

(1) The assignment of containers to FTs;

(2) The job sequence of resources, including FTs, TPs and GTs.

(3) The priority of FTs when there are FT conflicts.

Unlike FTs, for which we need to determine the assignment of containers as well as the sequence, we only need to determine the job sequence in TPs and GTs. When a container is assigned to an FT, we will know the TP and GT to which this container will be assigned. The reason is that we know at which yard location the container is going to be stored or retrieved. Each yard block is served by only one GT; therefore, the assignment of the container to a GT is known. Additionally, each yard is served by a dedicated TP on each berth rail. Thus, when a container is assigned to an FT, the assignment of the container to a TP is known. Therefore, we only need to determine the sequence of the containers in each GT and TP.

We introduce the following notations:

Notations

Parameters

Q	the set of QCs;
R	the number of horizontal rails (berth rails);
F	the set of FTs;
F_i	the set of FTs that belonging to Rail i, $F_1 \cup ... \cup F_r = F$;

N_r	the number of FTs in the Rail r, $N_r = F_r$, specially $N_0 = 0$.
P	the set of TPs.
P_i	the set of TPs that belonging to Rail i, $P_1 \cup ... \cup P_r = P$;
B	the number of Block (several contiguous yard blocks form a Block, and each yard block belongs to only one Block);
L	the set of loading jobs;
D	the set of discharging jobs;
H	the set of all the jobs, $H = L \cup D$;
QL	the set of subtasks handled by QC;
TP	the set of subtasks handled by TP;
G	the set of all subtasks, $G = QL \cup TP$;
Q_α	the number of jobs of QC α
$B_{(i,\alpha,h)}$	the yard block that subtask (i,α,k) belongs to, $(i,\alpha,k) \in TP$;
(i,α)	job index. The job (i,α) refers to the i th job in the sequence list of QC α;
$(i,\alpha,1)$	subtask index. The subtask of the first stage of job (i,α);
$(i,\alpha,2)$	subtask index. The subtask of the second stage of job (i,α). If job (i,α) is a loading job, then $(i,\alpha,1)$ is a subtask handled by TP, and $(i,\alpha,2)$ is a subtask handled by QC; if job (i,α) is a

discharging job, then $(i,\alpha,1)$ is a subtask handled by QC, and $(i,\alpha,2)$ is a subtask that handled by TP.

$p_{(i,\alpha,1)}$ the processing time of subtask $(i,\alpha,1)$.

$p_{(i,\alpha,2)}$ the processing time of subtask $(i,\alpha,2)$.

$t^{ft}_{(i,\alpha,k)\ j(\beta,l)}$ the FT traveling time from location of subtask (i,α,k) to location of subtask (j,β,l).

$t^{tp}_{(i,\alpha,k)(j,\beta,l)}$ the TP traveling time from location of subtask (i,α,k) to location of subtask (j,β,l).

$l_{(i,\alpha,k)}$ the location of subtask (i,α,k).

Decision Variables:

$X^{m}_{(i,\alpha)}=1$ if the container of job (i,α) is carried by FT m; otherwise 0.

$Z_{(i,\alpha,k)(j,\beta,l)}=1$ if the starting time of subtask (j,β,l) is greater than or equal to the finishing time of subtask (i,α,k); otherwise 0.

$T_{(i,\alpha,1)}$ the starting time of the first subtask of job (i,α);

$T_{(i,\alpha,2)}$ the starting time of the second subtask of job (i,α).

3.1.1 Activity for discharging and loading jobs

The container jobs can be categorized into discharging jobs and loading jobs. We need to analyze the flow time of these two jobs so that they can be formulated into a mathematical model. The entire process of loading a container and discharging a container in the FB-ACT is described below.

Suppose a discharging job (i,α) is assigned to an FT k. QC α picks up the container from the vessel. Then, the spreader of QC α grasping the container moves to the rear of the QC and waits for FT k. When FT k arrives at the point where containers can be picked up or dropped off by QCs, the container is placed on FT k. At this time, QC α is available to conduct the next job. FT k, carrying this container, moves to next location where the container will be transferred to a GT. During the movement of FT k, it may be delayed if another FT is processing a container on the route of FT k. When FT k arrives at the location to transfer the container, the process of transferring the container can be started if both the TP and GT are ready. Once the TP picks up the container from FT k, it rotates the container 90 degrees so that the container can be placed on the GT. After the container is placed on the GT, the TP and FT are available for their next jobs. The GT will carry the container to the corresponding storage location. The yard crane will pick up the container and put it at its storage location. At this time, the discharging job for this container is completed. The operations of a loading job are similar.

The activity flow of discharging jobs and loading jobs is shown in Figures 3.1 and 3.2, respectively.

3.1.2 Activity flow for discharging jobs (Figure 3.1)

The procedure of a discharging job (i,α) is described in the activity flow in Figure 3.1.

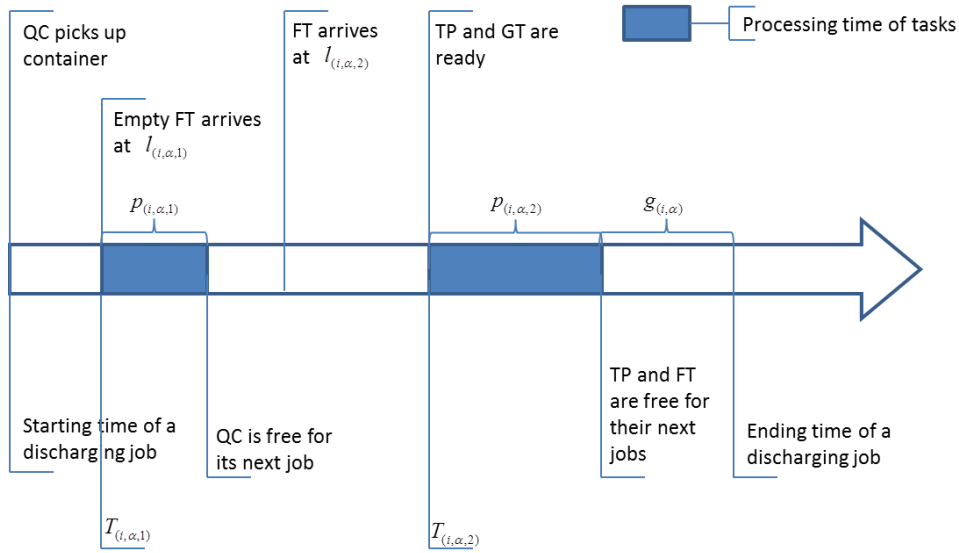


Figure 3.1 Activity flow for a discharging job

There are two activities in Figure 3.1. The first is related to the transfer of a container between the FT and QC, and the second is related to the transfer of a container between the FT and GT. They will block the rail during the transfer operations. Other FTs on the same rail cannot pass through the blocked location until those operations are completed. We define these two activities as two subtasks. Each job contains two subtasks.

$T_{(i,\alpha,1)}$ denotes the starting time of the first subtask of a container job (i,α). For discharging jobs, the starting time is the moment that QC α starts to release the container to an FT. The end of this subtask is the moment that the container is placed on the FT. The first shaded area in Figure 3.1 is the processing time of subtask $(i,\alpha,1)$. At time $T = T_{(i,\alpha,1)} + p_{(i,\alpha,1)}$, the QC is free and can move to serve the next container. $T_{(i,\alpha,2)}$ is the starting time of the second subtask of the container job. For discharging jobs, the starting time is the moment that the FT and the related TP and GT are ready for the transfer of the container. The discharge job ends at the moment the container is placed on the GT. The second shaded area in Figure. 2 is the processing time of subtask

$(i,\alpha,2)$. At time $T = T_{(i,\alpha,2)} + p_{(i,\alpha,2)}$, the FT and TP are free and can move to their next jobs. At time $T = T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + g_{(i,\alpha)}$, the container job is complete and the GT is free to serve its next job. The processing time in which the GT with the container moves to the specified slot on the yard and the YC transfers the container from the GT to the slot is denoted by $g_{(i,\alpha)}$.

3.1.3 Activity flow for loading jobs (Figure 3.2)

The following activity flow describes the procedure of a loading job (i,α).

Similarly, for loading jobs, $T_{(i,\alpha,1)}$ is the starting time of the first subtask. It is the time that TP starts transferring the container from the GT to the FT (denoted as FT k). At time $T = T_{(i,\alpha,1)} + p_{(i,\alpha,1)}$, the container is placed on the FT. The GT and TP are free for their next jobs. $T_{(i,\alpha,2)}$ is the starting time of the second subtask and is the time that QC begins to pick up the container from the FT. At time $T = T_{(i,\alpha,2)} + p_{(i,\alpha,2)}$, the container is picked up, and the FT is free for its next job. At time $T = T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + q_{(i,\alpha)}$, the loading job is completed and the QC is free for its next job. The processing time in which the QC moves to the specified slot on the vessel and then places the container in the slot is denoted by $q_{(i,\alpha)}$.

Therefore, the completion time of a loading container job is $C_{(i,\alpha)} = T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + q_{(i,\alpha)}$. The completion time of a discharging container job is $C_{(i,\alpha)} = T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + g_{(i,\alpha)}$.

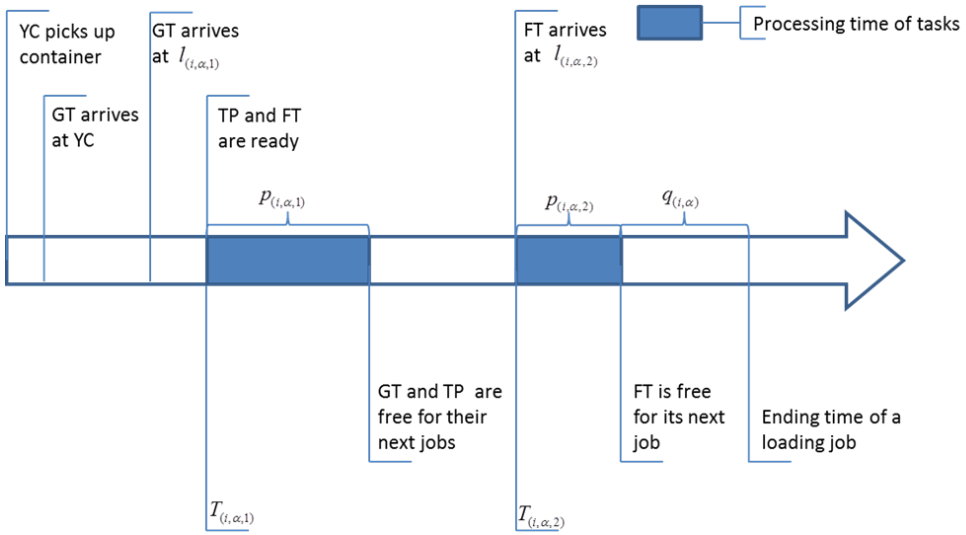


Figure 3.2 Activity flow for a loading job

3.1.4 Conflict Resolution

Two subtasks cannot be carried out simultaneously when they are competing for the same resources, or there is traffic conflict between the FTs. Some subtasks will be delayed to avoid conflicts in resources and traffic.

We introduce the variable $Z_{(i,\alpha,k)(j,\beta,h)}$ to set apart the starting time of subtasks.

$Z_{(i,\alpha,k)(j,\beta,h)}$=1 if the starting time of subtask (j,β,h) is greater than or equal to the finishing time of subtask (i,α,k). The relationship of the processing time of two subtasks can be represented by the equations with variable $Z_{(i,\alpha,k)(j,\beta,h)}$. They are shown in Figure 3.3.

Figure 3.3 illustrates that two subtasks cannot be operated simultaneously and are represented by the equation: $Z_{(i,\alpha,k)(j,\beta,h)} + Z_{(j,\beta,h)(i,\alpha,k)} = 1$. Either $Z_{(i,\alpha,k)(j,\beta,h)}$ or $Z_{(j,\beta,h)(i,\alpha,k)}$ is equal to 1, meaning that either subtask (j,β,h) starts to be operated after subtask (i,α,k) is completed or subtask (i,α,k) is started after subtask (j,β,h) is completed.

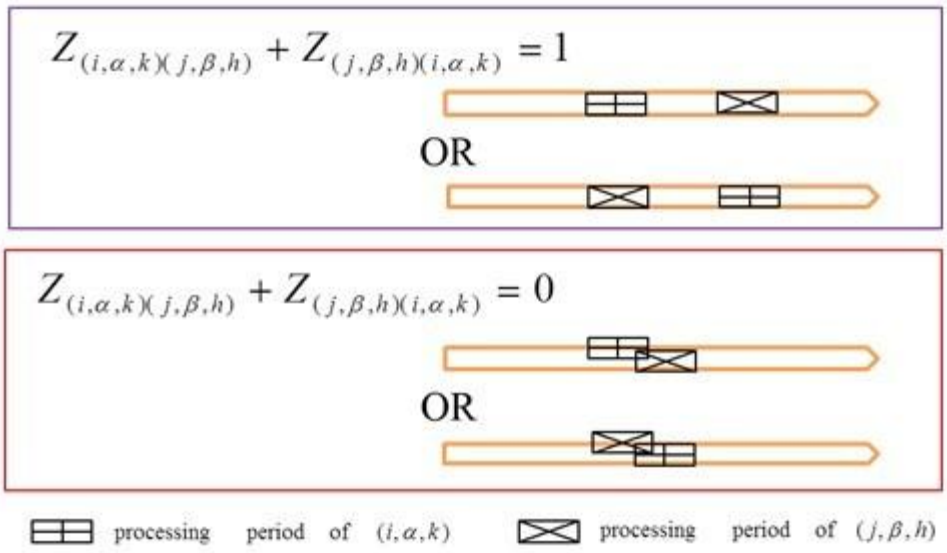


Figure 3.3 Relationship between the processing time of two subtasks

The resources in our problems are the FT, QC, TP and GT. For the pair of subtasks that are competing for the same resources, some constraints are proposed to ensure that they will not be operated simultaneously, as shown below:

$$Z_{(i,\alpha,k)(j,\beta,h)} + Z_{(j,\beta,h)(i,\alpha,k)} = 1;$$
$$T_{(j,\beta,h)} \geq T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + M(Z_{(i,\alpha,k)(j,\beta,h)} - 1)$$

Moreover, any two successive subtasks that use the same QC or GT must also be set apart by a minimum handling time. For these two successive subtasks that belong to different types of jobs (loading or unloading), the minimum separation time is different. t_{ld}^{qc} is defined as the minimum separation time of two successive subtasks that use the same QC, with the former subtask belonging to a loading job and the latter subtask belonging to a discharging job. Similarly, we can define the other minimum separation times as t_{dl}^{qc}, t_{ll}^{qc}, and t_{dd}^{qc}. Any two successive subtasks that use the same GT will also be set apart by a certain time interval. They are denoted by t_{ld}^{gt}, t_{dl}^{gt}, t_{ll}^{gt}, and t_{dd}^{gt}.

In summary, the starting time of any two successive subtasks that use the same QC or GT should satisfy the following constraint:

$$T_{(j,\beta,r)} \geq T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t_{ms} + M(Z_{(i,\alpha,k)(j,\beta,h)} - 1),$$

where t_{ms} is the minimum separation time defined above. Because the QC sequence is given, the value of $Z_{(i,\alpha.k)(j,\beta,h)}$ is known when the subtasks (i,α,k) and (j,β,h) are competing for the same QC (thus α and β are the same). The constraints can be presented as:

$T_{(i+1,\alpha,h)} \geq T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t_{ms}, \quad \forall (i,\alpha,k),(i+1,\alpha,h) \in QL$

Additionally, the travel time should be considered when the subtasks are competing for the same resources, such as the TP and FT. We have the constraints:

$T_{(j,\beta,h)} \geq T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t^{ft}_{(i,\alpha,k)(j,\beta,h)} + M(Z_{(i,\alpha,k)(j,\beta,h)} - 1)$ if these two subtasks are competing for the same FT;

$T_{(j,\beta,h)} \geq T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t^{tp}_{(i,\alpha,k)(j,\beta,h)} + M(Z_{(i,\alpha,k)(j,\beta,h)} - 1)$ if they are competing for the same TP.

FT conflict will also contribute to delay because two subtasks cannot be conducted simultaneously when there is an FT conflict between them. One will be operated first, whereas the other one will be delayed. The variable $Z_{(i,\alpha.k)(j,\beta,h)}$ is used to avoid conflict among FTs. To build the constraints that avoid the conflict among FTs, the FTs on the same rail are set in increasing order from left to right. Suppose that FT k_1 performs subtask (j,β,h) and that FT k_2 performs subtask (i,α,k).

Consider the following conditions:

(1) FT k_1 and FT k_2 are on the same rail;

(2) $k_1 < k_2$, i.e., FT k_1 is on the left-hand side of FT k_2;

(3) $l_{(j,\beta,h)} > l_{(i,\alpha,k)}$, i.e., the location of subtask (j,β,h) is on the right-hand side of subtask (i,α,k).

If conditions (1)-(3) are met, there is an FT conflict between these two subtasks. Consequently, subtask (i,α,k) and subtask (j,β,h) cannot be run simultaneously to avoid FT conflict.

The number of FTs on rail r is indicated as N_r, and the index of the FTs on the same rail is increased from left to right. The set of FTs on each rail can be presented as

$$Rail1:\{1,2,...,N_1\}, Rail2:\{N_1+1, N_1+2,...,N_1+N_2\},...,$$
$$Rail\ r:\{\sum_{k=1}^{r} N_{k-1}+1, \sum_{k=1}^{r} N_{k-1}+2,...,\sum_{k=1}^{r} N_k\}$$

The constraint to avoid FT conflict can be represented by (a):

$$\sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{v} X^{m}_{(j,\beta)} - \sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{v} X^{m}_{(i,\alpha)} \le Z_{(i,\beta,k)(j,\beta,h)} + Z_{(j,\beta,h)(i,\alpha,k)} + 2 - \left(\sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{\sum_{i=1}^{r+1} N_{i-1}} X^{m}_{(i,\alpha)} + \sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{\sum_{i=1}^{r+1} N_{i-1}} X^{m}_{(j,\beta)}\right)$$
$$\forall r = 1,2,...,R,\ k=1,2,\ h=1,2,\ l_{(i,\alpha,k)} < l_{(j,\beta,h)},\ v = \sum_{t=1}^{r} N_{t-1}+1,..., \sum_{t=1}^{r+1} N_{t-1}$$

(a)

Condition (1) can be obtained as

$$\sum_{m=1+\sum_{i=1}^{r} N_{i-1}}^{\sum_{i=1}^{r+1} N_{i-1}} X^{m}_{(i,\alpha)} + \sum_{m=1+\sum_{i=1}^{r} N_{i-1}}^{\sum_{i=1}^{r+1} N_{i-1}} X^{m}_{(j,\beta)} = 2, \qquad \exists r \in \{1,2,...,R\}$$

When Condition (1) holds, Constraint (a) can be reduced to (b):

$$\sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{v} X_{(j,\beta)}^{m} - \sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{v} X_{(i,\alpha)}^{m} \le Z_{(i,\alpha,k)(j,\beta,h)} + Z_{(j,\beta,h)(i,\alpha,k)},$$

$$\forall r = 1,2,\ldots,R,\ k = 1,2,\ h = 1,2,\ l_{(i,\alpha,k)} < l_{(j,\beta,h)}, v = \sum_{t=1}^{r} N_{t-1} + 1,\ldots,\sum_{t=1}^{r+1} N_{t-1}$$

(b)

If Condition (1) holds, Conditions (2) and (3) are satisfied. This is denoted as the equation:

$$\sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{v} X_{(j,\beta)}^{m} - \sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{v} X_{(i,\alpha)}^{m} = 1, \quad \exists v \in \{\sum_{t=1}^{r} N_{t-1} + 1,\ldots,\sum_{t=1}^{r+1} N_{t-1}\}$$

This means the FT (denoted as k_1) for subtask (j,β,h) is on the left hand side of the FT (denoted as k_2) for subtask (i,α,k), i.e., $k_1 < k_2$.

From (b), we can see that $Z_{(i,\alpha,k)(j,\beta,h)} + Z_{(j,\beta,h)(i,\alpha,k)} \ge 1$ must be satisfied, which implies either $Z_{(i,\alpha,k)(j,\beta,h)}$ or $Z_{(j,\beta,h)(i,\alpha,k)}$ is equal to one. Therefore, these two subtasks cannot be operated simultaneously. The constraint can ensure that any pair of subtasks with FT conflict will not be operated simultaneously. The time constraint can be presented as

$$T_{(j,\beta,h)} - (T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t^{ft}_{(i,\alpha,k)(j,\beta,h)}) \ge M(Z_{(i,\alpha,k)(j,\beta,h)} - 1), \qquad \forall (i,\alpha,k),(j,\beta,h) \in G$$

Consequently, if there is FT conflict between two subtasks, either

$$T_{(j,\beta,h)} - (T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t^{ft}_{(i,\alpha,k)(j,\beta,h)}) \ge 0, \text{ or}$$

$$T_{(i,\alpha,k)} - (T_{(j,\beta,h)} + p_{(j,\beta,h)} + t^{ft}_{(i,\alpha,k)(j,\beta,h)}) \ge 0 \text{ holds.}$$

3.2 Mathematical model

The mathematical model is shown below.

$$Min: \underset{(Q_\alpha,\alpha)\in L,(Q_\beta,\beta)\in D}{Max} \{T_{(Q_\alpha,\alpha,2)} + h_{1(Q_\alpha,\alpha)} + q_{(Q_\alpha,\alpha)}, T_{(Q_\beta,\beta,2)} + h_{2(Q_\beta,\beta)} + g_{(Q_\beta,\beta)}\}$$

(3.1)

Constraints:

(1) FT dispatching constraints

$$\sum_{m=1}^{|V|} X^{m}_{(i,\alpha)} = 1, \quad \forall (i,\alpha) \in H \tag{3.2}$$

$$X^{m}_{(i,\alpha)} + X^{m}_{(j,\beta)} - 1 \le Z_{(i,\alpha,2)(j,\beta,1)} + Z_{(j,\beta,2)(i,\alpha,1)}, \quad \forall (i,\alpha),(j,\beta) \in H, m \in F \tag{3.3}$$

$$\sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{v} X^{m}_{(j,\beta)} - \sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{v} X^{m}_{(i,\alpha)} \le Z_{(i,\beta,k)(j,\beta,h)} + Z_{(j,\beta,h)(i,\alpha,k)} + 2 - \left(\sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{\sum_{i=1}^{r+1} N_{i-1}} X^{m}_{(i,\alpha)} + \sum_{m=1+\sum_{t=1}^{r} N_{t-1}}^{\sum_{i=1}^{r+1} N_{i-1}} X^{m}_{(j,\beta)} \right)$$

$$\forall r = 1,2,..., R, k = 1,2, h = 1,2, \; l_{(i,\alpha,k)} < l_{(j,\beta,h)}, v = \sum_{t=1}^{r} N_{t-1} + 1,..., \sum_{t=1}^{r+1} N_{t-1} \tag{3.4}$$

$$T_{(j,\beta,h)} - (T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t^{ft}_{(i,\alpha,k)(j,\beta,h)}) \ge M(Z_{(i,\alpha,k)(j,\beta,h)} - 1) \quad \forall h = 1,2, k = 1,2 \tag{3.5}$$

(2) Time constraints for the two tasks of a given job

$$T_{(i,\alpha,1)} + p_{(i,\alpha,1)} + t^{ft}_{(i,\alpha,1)(i,\alpha,2)} \le T_{(i,\alpha,2)}, \qquad \forall (i,\alpha) \in H \tag{3.6}$$

(3) Sequence-dependent times for different resources

QC:

$$T_{(i+1,\alpha,2)} \ge T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + t^{ll}_{qc}, \quad \forall (i,\alpha) \in L, (i+1,\alpha) \in L \tag{3.7}$$

$$T_{(i+1,\alpha,1)} \ge T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + t^{ld}_{qc}, \quad \forall (i,\alpha) \in L, (i+1,\alpha) \in D \tag{3.8}$$

$$T_{(i+1,\alpha,2)} \ge T_{(i,\alpha,1)} + p_{(i,\alpha,1)} + t^{dl}_{qc}, \quad \forall (i,\alpha) \in D, (i+1,\alpha) \in L \tag{3.9}$$

$$T_{(i+1,\alpha,1)} \ge T_{(i,\alpha,1)} + p_{(i,\alpha,1)} + t^{dd}_{qc}, \quad \forall (i,\alpha) \in D, (i+1,\alpha) \in D \tag{3.10}$$

GT (successive tasks belong to the same yard):

$$Z_{(i,\alpha,k)(j,\beta,h)} + Z_{(j,\beta,k)(i,\alpha,h)} = 1, \quad \forall l_{(i,\alpha,k)} = l_{(j,\beta,h)}, k = 1,2, h = 1,2, (i,\alpha,k),(j,\beta,h) \in TP \tag{3.11}$$

$$T_{(j,\beta,1)} \ge T_{(i,\alpha,1)} + p_{(i,\alpha,1)} + t^{ll}_{gt} + M(Z_{(i,\alpha,1)(j,\beta,1)} - 1), \quad \forall (i,\alpha) \in L, (j,\beta) \in L, l_{(i,\alpha,1)} = l_{(j,\beta,1)} \tag{3.12}$$

$$T_{(j,\beta,2)} \geq T_{(i,\alpha,1)} + p_{(i,\alpha,1)} + t_{gt}^{ld} + M(Z_{(i,\alpha,1)(j,\beta,2)} - 1), \quad \forall (i,\alpha) \in L, (j,\beta) \in D, l_{(i,\alpha,1)} = l_{(j,\beta,2)} \tag{3.13}$$

$$T_{(j,\beta,1)} \geq T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + t_{gt}^{dl} + M(Z_{(i,\alpha,2)(j,\beta,1)} - 1), \quad \forall (i,\alpha) \in D, (j,\beta) \in L, l_{(i,\alpha,2)} = l_{(j,\beta,1)} \tag{3.14}$$

$$T_{(j,\beta,2)} \geq T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + t_{gt}^{dd} + M(Z_{(i,\alpha,2)(j,\beta,2)} - 1), \quad \forall (i,\alpha) \in D, (j,\beta) \in D, l_{(i,\alpha,2)} = l_{(j,\beta,2)} \tag{3.15}$$

TP:

$$\sum_{m \in F_r} X_{(j,\beta)}^{m} + \sum_{m \in F_r} X_{(i,\alpha)}^{m} - 1 \leq Z_{(i,\alpha,k)(j,\beta,h)} + Z_{(j,\beta,h)(i,\alpha,k)} \quad \forall r = 1,2,\ldots,R,\ k = 1,2,\ h = 1,2,\ \forall (i,\alpha,k), (j,\beta,h) \in TP_v, v = 1,\ldots,B. \tag{3.16}$$

$$T_{(j,\beta,h)} \geq T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t_{(i,\alpha,k)(j,\beta,h)}^{tp} + M(Z_{(i,\alpha,k)(j,\beta,h)} - 1) + M(\sum_{m \in F_r} X_{(j,\beta)}^{m} + \sum_{m \in F_r} X_{(i,\alpha)}^{m} - 2) \quad \forall k = 1,2, h = 1,2, \forall (i,\alpha,k), (j,\beta,h) \in TP_v, v = 1,\ldots,B \tag{3.17}$$

(4) Domain of variables:

$$X_{(i,\alpha)}^{m} = 0 \text{ OR } 1; \quad \forall (i,\alpha) \in H, m \in F \tag{3.18}$$

$$Z_{(i,\alpha,k)(j,\beta,h)} = 0 \text{ OR } 1; \quad \forall (i,\alpha), (j,\beta) \in H,\ k = 1,2,\ h = 1,2 \tag{3.19}$$

$$T_{(i,\alpha,1)},\ T_{(i,\alpha,2)} \geq 0, \quad \forall (i,\alpha) \in H \tag{3.20}$$

Objective (3.1) minimizes the makespan of a given set of container jobs. Constraints (3.2) – (3.5) are the constraints of FT dispatching. Constraint (3.2) ensures that every container job must be completed by exactly one FT. Constraint (3.3) means that each FT can only carry one container at a time. Conflict among FTs can be avoided via Constraint (3.4). Constraint (3.5) is the time constraint for two tasks if they cannot be performed simultaneously. Constraint (3.6) is the time constraint for the subtasks of each container job. Constraints (3.7) – (3.10) guarantee that the starting time of two successive

subtasks of the same QC must be separated by a minimum handling time. Constraint (3.11) ensures that each GT can only carry one container at a time. Constraints (3.12) – (3.15) ensure that the starting time of two successive subtasks of the same GT must be separated by a minimum handling time. Constraints (3.16) – (3.17) represent the time of constraint for the subtasks that use the same TP. It ensures that each TP can only operate one container at a time. Constraints (3.18) – (3.20) define the domains of the decision variables.

3.3 Filtered Beam Search-based Algorithm

To minimize the makespan of a given set of container jobs, we need to make the following decisions:

(i) The assignment of container jobs to FTs;

(ii) The job sequence among resources, including FTs, TPs and GTs.

(iii) The prior ordering of FTs when there is FT conflict.

The solution space of the decision variables increases exponentially with increases in the number of jobs. Existing commercial solvers are not able to solve the MIP model. Therefore, we develop a heuristic algorithm using the filtered beam search concept to address our problem.

3.3.1 Filtered beam search algorithm

Before presenting our algorithm, we first introduce the analogous algorithm called the filtered beam search (FBS) algorithm. The filtered beam search algorithm was proposed by Ow et al. (1988). It is widely used to solve combinatorial optimization problems in various areas, such as berth allocation problems (BAP) (Wang et al. (2007)), job shop scheduling (Sabuncuoglu et al. (1999)), load sequencing of outbound containers (Kim et al. (2004)), product

line design and selection (Nair et al. (1995)), and the single machine early/tardy problem (Ow et al. (1989)).

FBS is a type of tree structure-based algorithm. The beam nodes represent the partial solutions with good performance and are further expanded to generate new nodes. Evaluation functions are used to assess the performance of these nodes. The nodes with good performance are most likely to be selected to be the beam nodes of the next level in the tree. By continually expanding the tree, a good and complete solution can be obtained at the leaf level. However, the computation time can be quite large due to the effort of evaluating large numbers of nodes generated from the beam nodes. To make the algorithm efficient, a local evaluation (a cost estimated by a simple or greedy rule) will first be applied to help screen the nodes. The local evaluation is simple and computationally fast. After the filtering procedure, based on the local evaluation, many nodes with poor performance are screened out, whereas a smaller number of nodes are retained. Another function, called global evaluation (a cost estimation projecting the current partial solution to a complete solution), will be applied to evaluate the retained nodes. Beam nodes are selected based on the results of the global evaluation. The significance of a FBS can be summarized as follows. The local method is computationally inexpensive but less accurate. The global method is more accurate but computationally expensive. The filtering procedure removes the poor candidates so that the global method does not have to evaluate them. It can produce high-quality solutions with an efficient computational effort.

3.3.2 Development of the tree structure algorithm

Our algorithm is also constructed by tree structure and is analogous to FBS when using the screening procedure. However, the most remarkable point distinct from FBS is the mechanism of the global evaluation. In FBS, the global evaluation is to project the current partial solution to one complete solution or a certain number of complete solutions, and then the result of this complete solution or average of the results of those complete solutions is considered as the estimated cost of the partial solution. Thus it can be seen that the way to generate complete solutions is substantially crucial to the accuracy of the estimated cost. Because of the complexity of our problem, finding a way that generates promising complete solutions from partial solutions is intractable. Therefore, we will propose an optimal method or near optimal method to evaluate the nodes again after the screening procedure instead of the global evaluation. In other words, we will not project our partial solutions to a complete solution.

The expansion of nodes in the tree is also different. In our tree structure algorithm, a unit called the intermediate node will be generated from upper level first before a new beam node is created.

To illustrate our tree structure algorithm clearly, some related terms need to be defined first.

Beam node: the nodes in the tree which will be further explored. It represents the solution of involved subtasks showing how these subtasks are assigned to vehicles and how the vehicles transport the containers without collisions.

Partial solution: A beam node in the tree structure, except at the leaf level, represents a partial solution when not all of the subtasks have been assigned.

Complete solution: A beam node at the leaf level in the tree structure represents a complete solution that all subtasks are accomplished with the feasible schedule.

Assigned subtask: The subtask in the beam node is called an assigned subtask. The involved decision variables are determined for these subtasks.

Unassigned subtask: A subtask that has not been assigned to the beam node is called an unassigned subtask.

Assigned subtask with partial decisions: This is a subtask that is assigned to a beam node during the procedure of expanding a beam node. Only the decision about the vehicle assignment, as well as the sequence of this subtask in the corresponding vehicle, is determined.

We can give the explicit procedure of how the tree structure algorithm searches for a complete solution. At the root level of the tree, there is only one beam node that is an empty set. A large number of intermediate nodes are generated by selecting unassigned subtasks and assigning them to vehicles. A method called the surrogate method is developed to assess the fitness of these intermediate nodes. The surrogate method should be computationally fast. A related small number of intermediate nodes are retained after the screening procedure based on the results from the surrogate method. A method called the detail method is developed to solve the intermediate nodes to optimal solutions or near optimal solutions. Because the intermediate nodes represent

the solution of vehicle dispatching problem, the detail model solves the conflict-free routing problem under the solution of vehicle dispatching problem. After the calculation of detail model, vehicle dispatching and conflict-free routing problems of the subtasks in the intermediate nodes are solved. The best *m* intermediate nodes according to the results of the detail method are selected as new beam nodes at next level. The subtasks in the beam nodes become assigned subtasks and are removed from the set of unassigned subtasks. The tree is expanded in this way until the leaf level is approached. The framework of the filtered beam search based algorithm can be described by Figure 3.4.

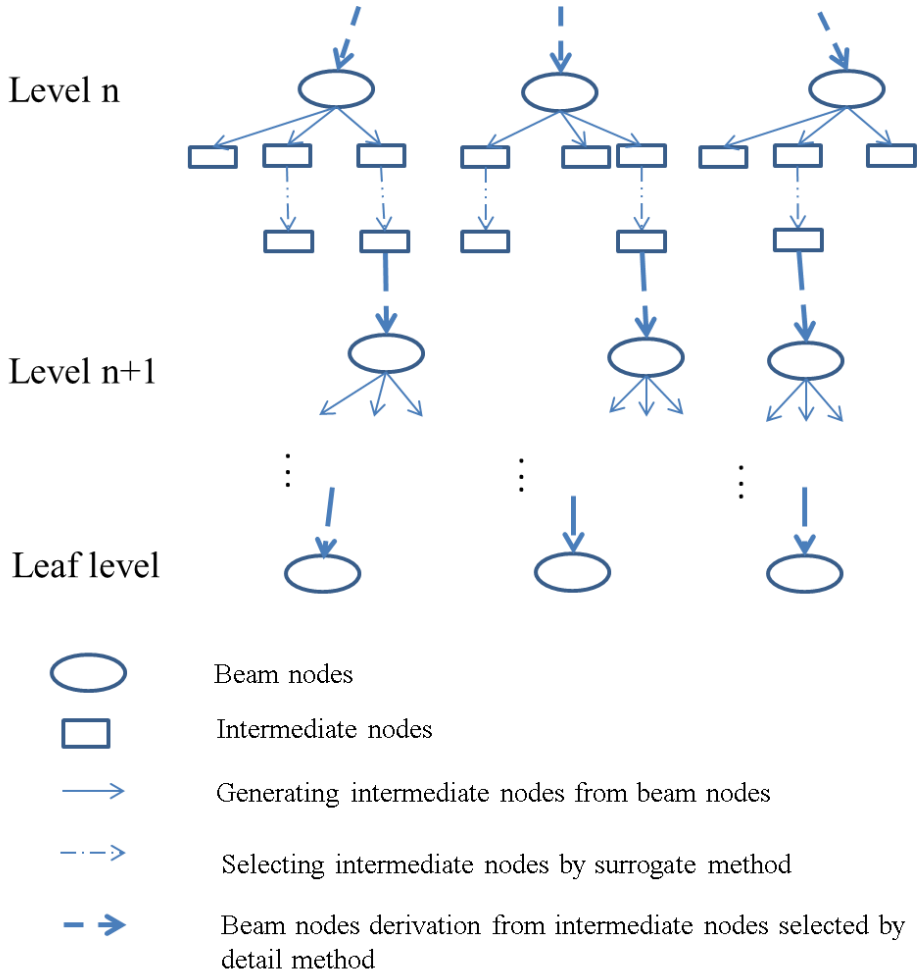


Figure 3.4 Framework of the tree structure algorithm

3.3.3 Discussion of the performance of the tree structure algorithm

It is intractable to simultaneously solve the vehicle dispatching and conflict-free routing problems when the planning horizon is large. However, we can try to obtain promising solutions by dynamically rolling the planning horizon. Each time we solve the problem to optimality or near optimality within a related small planning horizon. The planning horizon is extended step by step until it becomes complete. It is believed that longer decomposed planning horizon brings the solution closer to optimality. However, the computational efforts are exponentially increased when the decomposed planning horizon gets longer. The tree structure algorithm is in line with the strategy of rolling the planning horizon. A complete planning horizon is equivalent to the tree from root to leaf. Each expansion of the tree is similar to rolling the planning horizon.

The number of retained partial planning horizons is also significantly affects the final result. It is intuitive that when more number of promising partial planning horizons are retained for further exploration, the final result will be closer to optimality. This number of retained partial planning horizons is reflected as the beam number in the tree structure algorithm. Considering the computational efforts, the number of beam nodes has to be controlled. The performance of the tree structure algorithm is deteriorated if the number of possible intermediate nodes if extensively large. The possibility of losing promising solutions will increase because of missing the promising intermediate nodes among the huge set. Fortunately, the number of intermediate nodes is arbitrarily large because of the concept of QC list under the container terminal environment. In the vehicle dispatching problem under

the container terminal environment, a work schedule of QCs is constructed first based on the bay profile, which is sent as a guideline for discharging and loading operations by a shipping agent. Then, a sequence list called the QC list is made, which specifies the sequence of discharging and loading operations of individual containers by the corresponding QC. Thus, there exists a precedence relationship among the subtasks operated by the same QC. Consequently, we should avoid assigning a container job to a vehicle if a certain number of container jobs that preceding this container job are not accomplished. Therefore, the possible number of the intermediate nodes can be dramatically reduced.

We implement the tree structure algorithm to solve the three problems under different container terminal systems. According to the characteristics of container terminal systems, different surrogate methods and detail methods are proposed. The next three chapters will introduce the new container terminal systems and discuss how we implement the algorithm to solve the complicated problems of vehicle dispatching and conflict-free routing.

3.3.4 The fitness calculation

We want to examine the efficiency of the schedule represented by the intermediate nodes. A good schedule should not contain long delay times or large number of unproductive moves. The delay time is involved in the handshakes among different types of equipment or during the FT conflict. The unproductive moves of FTs include empty FTs travelling to pick-up their next container, or in making way for other FTs to prevent FT conflict. Therefore, the fitness of the intermediate node, denoted as f, is defined as the ratio of

unproductive time to the length of the time window. A smaller fitness leads to a better intermediate node. The unproductive time includes the delay time and the time of unproductive FT moves. To determine the unproductive time, we must define the time window. The start time of the time window begins at the end time of the beam node from which the intermediate node is expanded. The end time of the time window is the time when the earliest FT finishes all look-ahead subtasks. In the fitness calculation, we only consider the unproductive time within the time window. To obtain the end time of the time window, we must calculate the completion time of the assigned subtasks with partial decisions. Two methods are proposed to make the decisions for calculating the completion time of the assigned subtasks with partial decisions in the intermediate nodes. The surrogate model approach is the first approximation method, and the reduced MIP model approach is the other more accurate method. The surrogate model approach is used to filter out the intermediate nodes with poor performance, saving the best w intermediate nodes, where w is the filter width. The reduced MIP model approach will then calculate the fitness of the saved intermediate nodes again. The new generation of beam nodes will be derived from the intermediate nodes with the first b smallest fitness value, where b is the beam width. The details of these two approaches will be discussed later in this section.

A simple example is given below to illustrate the procedure of calculating the fitness of an intermediate node. We assume that there are three FTs and that the look-ahead number is two. The QC handling time is 90 s, whereas the TP handling time is 60 s. The index of subtask (i,α,k) means the k th subtask of the i th container in the QC α list. We randomly select the unassigned

subtasks and assign them to the FTs. After the assignment procedure, we obtain the FT sequence of the newly assigned subtasks as follows: subtasks $(3,3,1),(3,3,2)$ assigned to FT1, subtasks $(4,1,2),(5,1,1)$ assigned to FT2 and subtasks $(4,2,1),(4,2,2)$ assigned to FT3. The starting time of the current time window is the completion time of assigned subtask $(4,3,2)$ (referring to Figure 3.5), which is also the end time of the previous time window. After the calculation by the surrogate model, the completion time of each subtask is obtained and the Gantt Chart is compiled, as shown in Figure 3.5. The completion time of subtask $(5,1,1)$ is the end time of the time window because FT2 is the earliest FT that completes all of its look-ahead subtasks.

Hence, the fitness can be calculated:

$$f = \frac{15+145+110+145+75}{500-220} = 1.75.$$

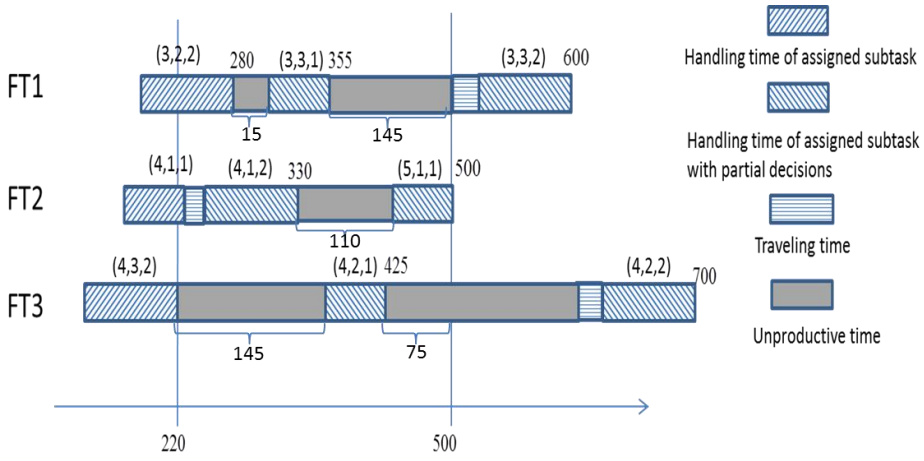


Figure 3.5 Gantt chart based on the surrogate model approach

3.3.5 Surrogate model approach

There are two types of subtasks in an intermediate node: assigned subtasks and assigned subtasks with partial decisions. The completion time of the

assigned subtasks is determined and will not change. We only need to calculate the completion time of the assigned subtasks with partial decisions.

The surrogate model approach calculates the completion time of one subtask at a time. According to the sequence in each FT, a set that consists of the first subtask in each FT is constructed. This set is called Set P. We will select one subtask in Set P whose completion time will be determined. Then, Set P will be updated by removing the selected subtask and adding the subtask following the selected subtask belonging to the same FT. The procedure is executed until the completion time of all assigned subtasks with partial decisions is determined. The assigned subtasks with partial decisions are the subtasks that are assigned to the FTs when expanding the beam node.

The way to select the subtask such that its completion time will be determined is based on the cost of the subtask in Set P. The cost of a subtask is defined as the delay time the subtask brings to other subtasks in Set P. For example, two subtasks (denoted as A1 and A2) cannot be executed simultaneously. Subtask A2 will be delayed if we allow subtask A1 to be conducted first. The delay time of subtask A2 is brought from subtask A1. We then select another subtask (denoted as A3) in Set P. The delay time of subtask A3 is calculated in the same manner that we let subtask A1 be conducted first. When the delay time of subtask A3 is calculated based on subtask A1, we do not consider the impact of other subtasks, such as A2, on subtask A3. In summary, the sum of the delay time of all subtasks in Set P, except subtask A1, represents the cost of subtask A1. The costs of other subtasks in Set P are calculated in the same manner.

After $m*\Delta$ selections, where m is the number of FTs and Δ is the look-

ahead number of subtasks for one FT, the completion time of all the assigned subtasks with partial decisions can be calculated. Therefore, the fitness of the intermediate node can be calculated based on the completion time of the subtasks.

Some notations must first be introduced to interpret the pseudo-code of the surrogate model approach.

The sequence of subtasks in FT i is represented as $SF_i = (f_{i,1}, f_{i,2}, .., f_{i,\Delta})$, where $f_{i,j}$ is the index of the subtask, the subscript i is the index of FT and j is the index of the order in the sequence. $f_{i,j}$ represents the j th subtask that FT i will carry out. Δ is the look-ahead number of the subtasks that each FT will carry out. A candidate set, denoted as S, is the set of subtasks among which there is one subtask, which will be selected to calculate its completion time. The cost of subtask $f_{i,j}$ is denoted as $C_{f_{i,j}}$. The completion time of subtask $f_{i,j}$ is denoted as $CT_{f_{i,j}}$. The pseudo-code is shown below:

Initialization of candidate set, $S = \{f_{i,1} \mid i \in M\}$, where M is the set of FTs.

while $S \neq \Phi$, **do**

(1) calculate $C_{f_{i,j}}$, where $f_{i,j} \in S$;

(2) select the subtask $f_{i,j}$, where $C_{f_{i,j}} = \min_{f_{i,j} \in S}\{C_{f_{i,j}}\}$;

(3) determine the completion time of subtask $f_{i,j}$,

$CT_{f_{i,j}} = \max\{CT_k + t^{ft}_{k,f_{i,j}} + p_{f_{i,j}}, CT_h + t^{ft}_{h,f_{i,j}} + p_{f_{i,j}}\}$, where subtask k is the previous

subtask of subtask $f_{i,j}$ in the same FT, and subtask h is the subtask

with which subtask

$f_{i,j}$ cannot be operated simultaneously;

(4) update S:

Remove subtask $f_{i,j}$ from S;

if there exists subtask $f_{i,j+1}$, **do**

choose the subtask $f_{i,j+1}$ to enter S;

end if

end while

3.3.6 Reduced MIP model approach

The original MIP model, described in Section 3.2, cannot be solved by commercial software for large-scale problems. However, the MIP model can be greatly reduced when the assignment of subtasks to FTs and the sequence of subtasks in FTs are given. For clarity, we propose a definition denoted as the FT schedule. The FT schedule is the assignment of subtasks to FTs as well as the subtask sequence in each FT. We can re-model the problem given the FT schedule, which is deemed a reduced MIP model. It can be found in Appendix A.

To show the reduction of variables in the reduced MIP model with the given FT schedule, some examples are created. The results are shown in Table 3.1.

The number of FTs is three in each instance in Table 3.1.

The number of variables in the reduced MIP varies depending on the FT schedule, whereas the number of variables in the original MIP is fixed. A certain number of FT schedules are generated for each case. The number of

variables in the reduced MIP is the average number among the generated FT schedules.

Table 3.1 Comparison of the number of variables between the reduced and original MIPs

Case	Average no. of variables in the reduced MIP	No. of variables in the original MIP	Ratio
10 jobs	79	448	0.1746
12 jobs	102	635	0.1612
15 jobs	152	972	0.1570
20 jobs	231	1699	0.1364
30 jobs	471	3750	0.1261
40 jobs	804	6585	0.1226
50 jobs	1,251	10,244	0.1224

Because we only calculate the completion time of a certain number of subtasks, the number of subtasks in the reduced MIP model is limited so that the computing time will be small. The completion time of each look-ahead subtask can be obtained from the optimal solution of the reduced MIP model. The fitness value can be found according to the completion time of the subtasks. The reduced MIP model can be used in the second-stage selection. This method is more accurate than the surrogate model, but the computing time is longer.

The method in the first-stage selection should be easy and computationally efficient. However, it should also be precise so that good solutions will not be discarded. A large number of calculation examples are created to show that the correlation coefficient between the fitness values is high based on the surrogate model and reduced MIP model. The computing time of these two approaches is also compared. The results are provided in Table 3.2.

Table 3.2 Correlation coefficient between surrogate model and reduced MIP model

	Correlation coefficient	Computing time ratio
3 subtasks look-ahead	0.8432	30.12
4 subtasks look-ahead	0.8131	32.88
5 subtasks look-ahead	0.7892	37.23
6 subtasks look-ahead	0.7732	43.69

The correlation coefficient is high and most of the unsatisfying partial solutions can be discarded by the first stage selection. The computing time of the surrogate model is much shorter than that of reduced MIP. Therefore, a large number of partial solutions can be evaluated in the first stage selection.

In summary, for each beam node, we randomly generate a certain number of intermediate nodes during the expanding procedure. For each generated intermediate node, we first use a surrogate model approach to calculate the fitness. The best w (filter width) intermediate nodes are retained after the filtering procedure. These intermediate nodes are assessed by the reduced MIP model approach. The best b (beam width) intermediate nodes are retained. Beam nodes are derived from these best b intermediate nodes. The algorithm is terminated when the completion time of all subtasks is determined, where the beam nodes represent a complete solution. The best solution can then be selected from the beam nodes.

3.4 Computation experiments

Experiments are conducted to assess the proposed approaches. The experiments are divided into two parts: the first part is to assess the proposed algorithm by comparison to the optimal results. The second part analyzes the

layout that impacts the performance of the FB-ACT.

3.4.1 Comparison study of the performance of the proposed methods

The parameters involved with the TP, GT, QC and FT are shown as follows: The speeds of the FT, GT, and TP are 4, 4 and 1 m/s, respectively. The QC handling time is 40 moves per hour, and the TP handling time is 60 moves per hour. The layout of the terminal in our experiments is shown in Figure 3.6. All of the numerical studies use the same parameters.

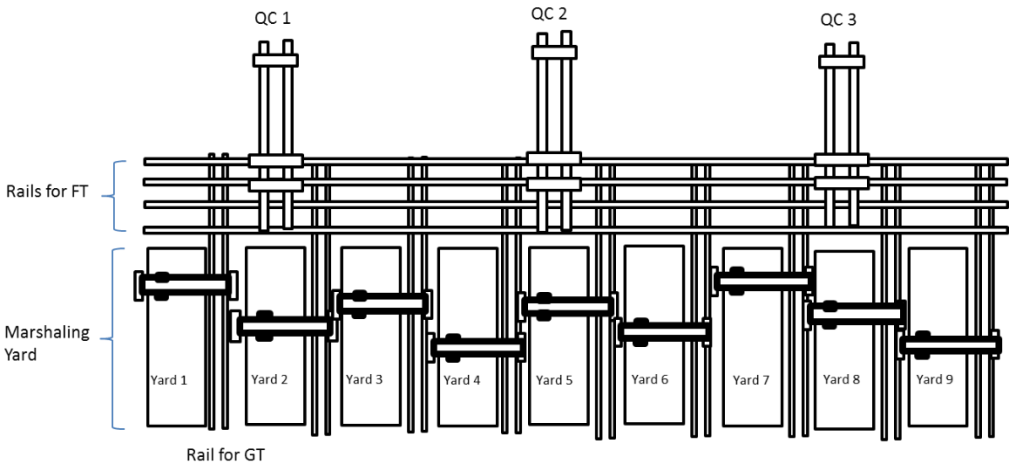


Figure 3.6 The layout of the terminal

As our problem is focus on the operation in the area of rails for FTs, the movement of QCs is not considered. Each QC is assumed to be fixed. The horizontal distance between two contiguous rails for is 20 meters. Like the layout shown in Figure. 3.6, the distance between the QCs and yards are given in Table 3.3.

Table 3.3 distance between QC and yard

	Yard1	Yard2	Yard3	Yard4	Yard5	Yard6	Yard7	Yard8	Yard9
QC1	10 (meters)	10	30	50	70	90	110	130	150
QC2	70	50	30	10	10	30	50	70	90
QC3	130	110	90	70	50	30	10	10	30

As there is only one GT in each rail, there is no conflict of GTs. The operation in the yard area is simplified. We assume the location of the container in the yard is in the middle. We also assume that the yard crane is always ready for the GT. Hence, the travel time of a GT between the pickup/draw point by TP and pickup/draw point by yard crane can be set to be 12.5 second. The handling time that the yard crane picks up (draws) a container from (to) a GT is set to be 10 second. Similarly, the handling time that the TP picks up (draws) a container from (to) a GT is 10 second. The minimum separation time of GT can be shown in Table 3.4. Since the location is assumed to be fixed, the minimum separation time of QC is shown in Table 3.4

Table 3.4 minimum separation time of QC

Notation	Time (second)	Notation	Time (second)
t_{qc}^{ll}	45	t_{gt}^{ll}	45
t_{qc}^{ld}	145	t_{gt}^{ld}	0
t_{qc}^{dl}	0	t_{gt}^{dl}	185
t_{qc}^{dd}	45	t_{gt}^{dd}	85

The look-ahead number in our experiments is set to 4. There is a tradeoff between the computation time and accuracy of the algorithm when choosing the look-ahead number. A larger look-ahead number leads to a longer computation time in the reduced MIP model. In addition, when generating the intermediate nodes from beam nodes, the number of possible combinations will increase greatly if the look-ahead number is larger. Because we will only

generate a certain number of intermediate nodes, the likelihood of losing the optimal assignment will increase if the number of possible candidates of the intermediate nodes is large. However, if the look-ahead number is not sufficiently large, the accuracy of the algorithm will decrease. Therefore, we set the look-ahead number to 4.

Table 3.5 Comparison with the optimal results

Case	CPLEX		FBS		Optimal Gap
	Makespan	CPU Time (min)	Makespan	CPU Time (min)	
P1 (8 jobs, 3FT)	14.45	0.05	14.45	3.00	0
P2 (9 jobs, 3FT)	14.61	0.05	14.61	2.91	0
P3 (10 jobs, 3FT)	14.40	0.61	14.42	4.67	0.14
P4 (10 jobs, 3FT)	16.08	0.62	16.09	4.45	0.06
P5 (12 jobs, 3FT)	16.75	2.34	16.8	4.50	0.30
P6 (12 jobs, 3FT)	17.25	2.40	17.83	4.52	3.36
P7 (12 jobs, 4FT)	16.10	2.50	16.15	4.62	0.31
P8 (15 jobs, 3FT)	20.33	32.47	20.75	8.67	2.07
P9 (15 jobs, 3FT)	22.00	22.26	22.07	8.17	0.32
P10 (15 jobs, 4FT)	19.89	25.08	19.90	9.50	0.05
P11 (15 jobs, 4FT)	19.87	24.42	20.42	8.62	2.77
P12 (15 jobs, 4FT)	19.50	36.34	19.77	8.77	1.13
P13 (20 jobs, 3TF)	24.07	3 h 21	24.79	11.52	2.99
P14 (20 jobs, 4FT)	23.07	4 h 31	23.72	12.01	2.64
			Average Optimality Gap		1.153

When the number of jobs is small, CPLEX can solve our MIP model and obtain the optimal result. Fourteen instances are randomly generated. There is only one rail in each instance. Three quay cranes serve the jobs and containers are distributed among nine yard blocks. The comparison is shown in Table 3.5. The optimality gap is relatively small when the problem scale is small. The computation time of the CPLEX method increases exponentially. Our

algorithm can solve the problems in polynomial time. The optimal results in large-scale problems cannot be obtained by commercial software. Therefore, we assess our algorithm via comparison with the results calculated by the FCFS rule.

Table 3.6 Comparison with the FCFS rule

	FCFS rule	FBS	Gap (%)
P1 (20 jobs, 3FT)	36.53	24.95	31.70
P2 (20 jobs, 4FT)	36.75	22.28	39.37
P3 (20 jobs, 5FT)	34.03	20.43	39.96
P4 (20 jobs, 6FT)	34.54	20.13	41.72
P5 (20 jobs,7FT)	30.03	17.77	40.83
P6 (20 jobs, 3FT, 2Rail)	24.55	17.67	28.02
P7 (20 jobs, 2FT, 3Rail)	20.75	15.03	27.57
P8 (20 jobs, 3FT, 3Rail)	19.99	13.64	31.77
P9 (30 jobs,3FT)	61.82	42.68	30.96
P10 (30 jobs, 4FT)	60.82	41.50	31.77
P11 (30 jobs, 5FT)	55.73	38.10	31.63
P12 (30 jobs, 6FT)	54.13	36.28	32.98
P13 (30 jobs, 7FT)	54.84	38.90	29.06
P14 (30 jobs, 3FT, 2Rail)	40.56	25.00	38.36
P15 (30 jobs, 2FT, 3Rail)	31.55	24.67	21.81
P16 (30 jobs, 3FT, 3Rail)	31.27	22.89	26.80
P17 (40 jobs, 3FT)	83.68	50.70	39.41
P18 (40 jobs, 4FT)	79.88	46.27	42.08
P19 (40 jobs, 5FT)	81.74	42.60	47.88
P20 (40 jobs, 6FT)	80.92	42.23	47.81
P21 (40 jobs, 7FT)	73.61	41.25	43.96
P22 (40 jobs, 3FT, 2Rail)	55.73	38.28	31.31
P23 (40 jobs, 2FT, 3Rail)	46.44	30.97	33.31
P24 (40 jobs, 3FT, 3Rail)	44.46	28.48	35.94
P25 (50 jobs, 3FT)	110.45	66.97	39.37
P26 (50 jobs, 4FT)	109.56	65.20	40.49

Continue Table 3.6

P27 (50 jobs, 5FT)	99.63	56.77	43.02
P28 (50 jobs, 6FT)	104.14	55.57	46.64
P29 (50 jobs, 7FT)	102.12	54.47	46.66
P30 (50 jobs, 3FT,2Rail)	73.79	45.35	38.54
P31 (50 jobs, 2FT, 3Rail)	63.38	44.55	29.71
P32 (50 jobs, 3FT,3Rial)	61.99	36.57	41.01

In Table 3.6, the index like (3FT, 2 Rail) means that there are 2 rails and 3 FTs are mounted on each rail.

Table 3.6 suggests that our algorithm significantly outperforms the FCFS rule in every instance, especially in cases with a relatively large number of FTs (i.e., five, six or seven) on the same track. The average difference of the instances that is less than or equal to three FTs on the same track is 32.85% and it is 40.37% when there are more than three FTs on the same track. The FCFS rule performs poorly in its handling of the FT conflict. The FCFS rule is a greedy heuristic rule that considers a relatively small number of combinations of solutions, whereas our algorithm uses the MIP to optimize the solutions of the assigned subtasks with partial decisions.

3.4.2 Effects of the parameters on the efficiency of FB-ACT

Effects of the allocation of FTs on the makespan

Table 3.7 Effects of the allocation of FTs on the makespan

	1 Rails, 6 FTs (min)	2 Rails, 3 FTs (min)	3 Rails, 2 FTs (min)
20 jobs	21.55	18.24	16.23
30 jobs	35.44	27.60	26.33
40 jobs	42.12	37.05	33.24
50 jobs	56.04	46.34	43.87

15 instances are created in each case in Table 3.7. The result is the average of the instances.

The results indicate that for a given total number of FTs, the system performs better with fewer FTs on the same rail. This finding is intuitive because when the number of FTs on one rail is reduced, the number of potential FT conflicts decreases. However, the difference between columns 4 and 3 in Table 3.5 is considerably less than that between columns 3 and 2. Because the delay time due to FT conflict is not so much in the case of 3 FTs on a rail, the reduction of FTs on the same rail will not yield significant benefits.

Effects of the average job distance on the makespan

The allocation of containers can affect the performance of the system by increasing the delay due to FT conflict. Because container jobs with long distances will increase the traveling time, we set the traveling time to zero to eliminate its impact on the makespan. In addition, there is no conflict on the traveling route of GTs. The distance between the transfer point by the TP and the yard storage point will not affect the makespan. Therefore, the job distance in our numerical experiments is between the QC pick-up or delivery point and the transfer point by the TP. Four groups of numerical experiments are conducted. The horizontal coordinate is the distance. Each experiment contains seven intervals of distance, i.e., [30, 45], [45,60], [60,75], [75,90], [90,105], [105,120] and [120,135] (meters). Ten instances are generated in each interval. The value of the vertical coordinate is the mean of the makespan of these 10 instances.

The results in Figure 3.7 suggest that the makespan increases with increases in the average distance. This trend occurs because the delay

increases due to FT conflict. The likelihood that one FT will be delayed to make way for another FT may increase if its route is longer. Therefore, to improve the performance of the FB-ACT system, the containers should be well allocated. A container should not be stored in a yard that is far away from its location in the vessel.

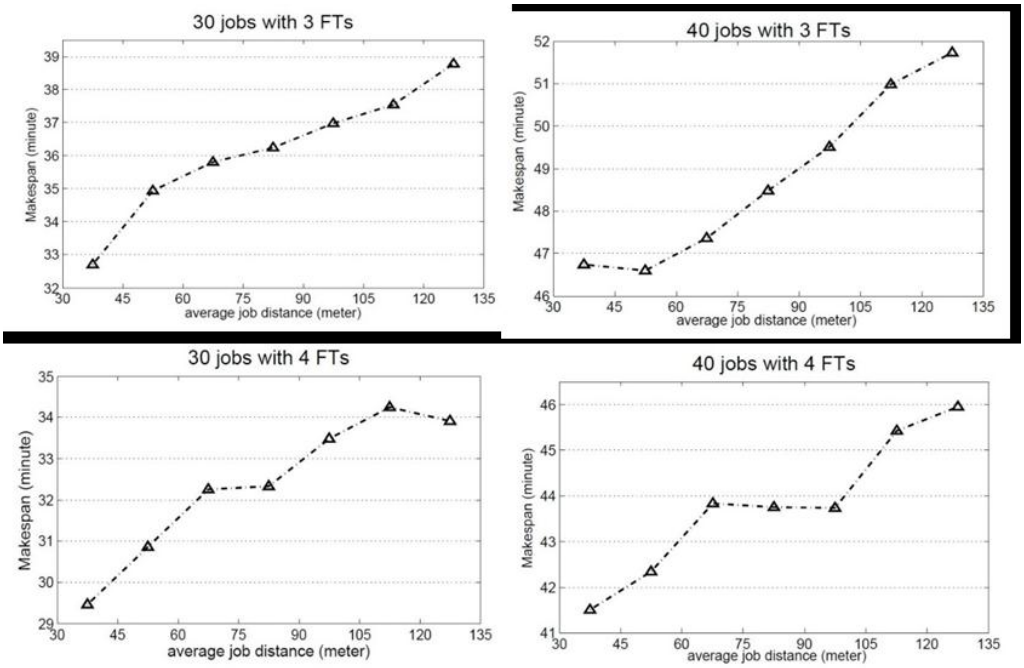


Figure 3.7 Effects of the average job distance on the makespan

Effects of the number of FTs and TPs on the makespan

Some instances are created to study the effects of the number of FTs and TPs on the makespan. There are 18 yards, 5 quay cranes and 40 jobs. We set different numbers of FTs and TPs and run the experiments. 12 instances are created in each case. The result for each case is the average of the 12 instances. The results are shown in Table 3.8.

Table 3.8 Effects of the numbers of FTs and TPs on the makespan

	1 Rail 3 FT	1 Rail 4 FT	1 Rail, 5 FT	1 Rail 6 FT	1 Rail, 7 FT
6 TPs	57.00	50.22	48.67	48.00	47.7
12 TPs	54.45	47.08	45.65	44.48	43.65
18 TPs	53.28	45.22	43.73	43.60	43.44

The results indicate that increasing the numbers of FTs and TPs can decrease the makespan. This trend is obvious because increasing a resource can reduce the waiting time. However, the rate of reduction is decreased when increasing the number of FTs. This trend can be observed from Table 3.9. When the number of FTs on the same rail is small, an increase in the number of FTs can reduce the waiting time. However, an increase in the number of FTs will increase the delay time to prevent FT conflict when the number of FTs is large.

Table 3.9 Rate of reduction by increasing the number of FTs by one

	4FT vs. 3FT	5FT vs. 4FT	6FT vs. 5FT	7FT vs. 6FT
6 TPs	11.89 (%)	3.09	1.38	0.62
12 TPs	13.54	3.04	2.56	1.87
18 TPs	15.13	3.33	0.31	0.30

In summary, FT conflict is the main variable that contributes to delay time. The containers should be well allocated to reduce the movement of FTs. The increase in job distance will create more delay time to prevent FT conflict. The FTs should be distributed to different rails. When the number of FTs on the same rail is sufficiently large, an increase in the number of FTs can only reduce the makespan slightly.

3.5 Conclusion

In this chapter, we focus on finding a promising schedule for containers while considering the handshakes and interference among the equipment involved. A mathematical model is proposed to achieve the objective. Because the large-scale models cannot be solved by commercial software, an algorithm was developed that uses an idea similar to the filtered beam search method. The proposed algorithm is tested with the 14 instances, which can be solved by CPLEX. The results show that the algorithm can achieve a near-optimal solution. The results from another 32 instances at a relatively large scale illustrate that our algorithm significantly outperforms the FCFS rule.

In future studies, a tight lower bound should be found. Because the optimal solution of our problem cannot be found for a large number of jobs, a tight and strong lower bound must be established to assess the best solution found by the proposed algorithm. In our current model, we set the number of layers of the frame bridge to one. However, multi-layer frame bridges can increase productivity, which is one of the most attractive characteristics of an FB-ACT system. The comparison of productivity between single- and multi-layer FB-ACT systems is an interesting field for future research. Additionally, in our study, only one GT serves a yard block. We can increase the number of GTs that serve yard blocks. These GTs can be on the same rail or increase the layer of yard rails.

CHAPTER 4 Vehicle Dispatching and Conflict-free Routing Problems Under GRID-ACT

4.1 Problem Description

The GRID system consists of three components: transfer units (TUs), overhead rails and transfer tables. The overhead rails cover the whole area of the storage yard. Thus the container can be only stacked under the overhead rails. The TUs move along the rails which are bi-directional. Each TU has two pairs of wheels that are respectively responsible for two mutually perpendicular directions. One pair of wheels is mounted on the rails to move the TU along one direction while the other pair of wheels is not connected with the rails. When changes to the perpendicular direction, the idle wheels are mounted on the rails and the other wheels will leave the rails and become idle. At this time, the TU completes the procedure of making a 90 degrees turn. The TU can perform like a yard crane to lift up a container or release a container. The activity flow of a discharging job can be shown as follows:

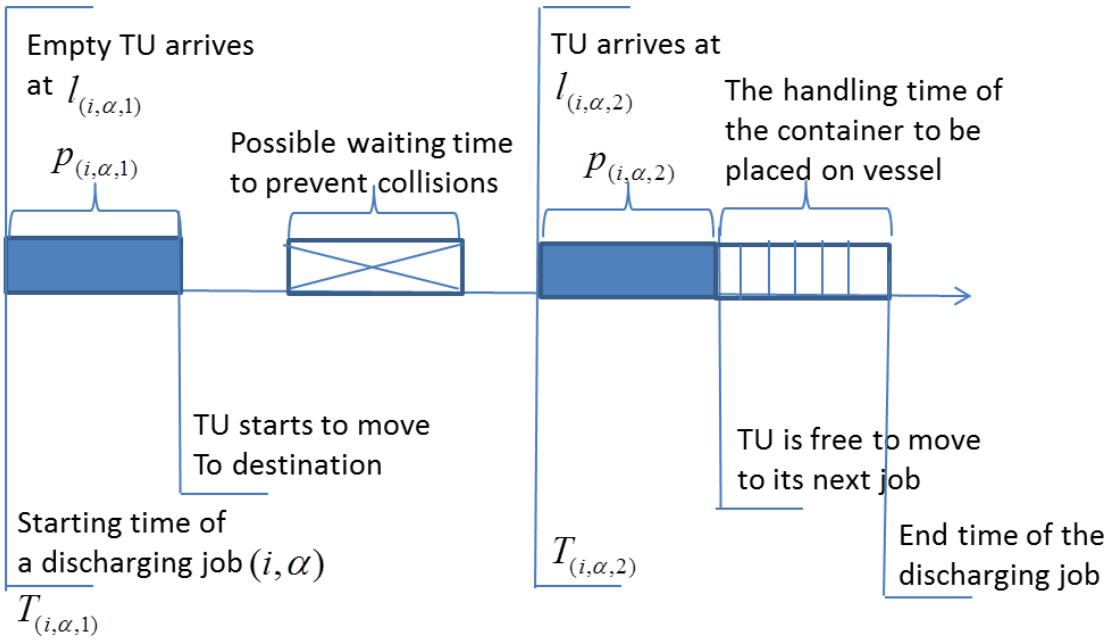


Figure 4.1 The activity flow of a discharging job under the GRID system

An empty TU arrives at the location $l_{(i,\alpha,1)}$ and starts to pick up the container (i,α). After a period equaling to $p_{(i,\alpha,1)}$, the TU starts to move to its destination at location $l_{(i,\alpha,2)}$. The TU will choose a proper way or stop to wait if needed to prevent vehicle collisions. When the TU arrives at location $l_{(i,\alpha,2)}$, it starts to put down the container to the transfer table. After a period equaling to $p_{(i,\alpha,2)}$, the TU is free and can start to move to its next job. The container (i,α) will be picked up by the QC and placed on the vessel. The activity of a loading job is similar in a reverse order.

In this study, we focus on the problem of vehicle dispatching and conflict-free routing simultaneously. In order to solve the routing of TUs, we propose to convert the physical layout of this system into a pure node layout and then into a time space network. Because the term of node has been used in the tree structure algorithm, another term called the cell will take the place of the node. The purpose of this transformation is to measure the position of TUs at every time, so that we can model them to capture the conflict-free routings. We will implement the technique of set partitioning based model to model our problem. Because of the intrinsic difficulty of the model, our problem will be decomposed into two sub-problems: vehicle dispatching problem and conflict-free-routing problem. A tree structure algorithm is used to solve the vehicle dispatching problem, while a column generation algorithm based method will be applied to solve the routing problem. In order to model our problem, we will first discuss the way to construct the time-space network. The methodology will be discussed after this.

4.1.1 Convert the physical layout into a pure cell layout

The shape of a TU is a little bit larger than a 40ft standard container. The length of the TU is 13.2 meters and the width is 3.3 meters. The speed of the TU is 2 meters per second. To model the routing of the TUs, we define the time that one TU needs to traverse the distance equaling to the width of a TU as a unit of time. The unit of time equals to 3.3/2=1.65 seconds. We use this time unit to measure all the activities in this system. The activities are including:

I. TUs move on the rails;

II. TUs stop to wait to prevent collisions;

III. TUs make a turn;

IV. TUs pick up containers from the stacking areas or buffers;

V. TUs put down containers to the stacking areas or buffers;

VI. QCs load and unload containers

We format the container stacking area as a gridding. We call the grid in the gridding layout as a cell. The size of one cell in the gridding is of length 3.3 meters and width 3.3 meters. Thus, we can indicate the location of the containers and TUs using the index of the cells. The gridding layout and the index of the cells are shown in Figure 4.2. Each cell is assigned with a unique number.

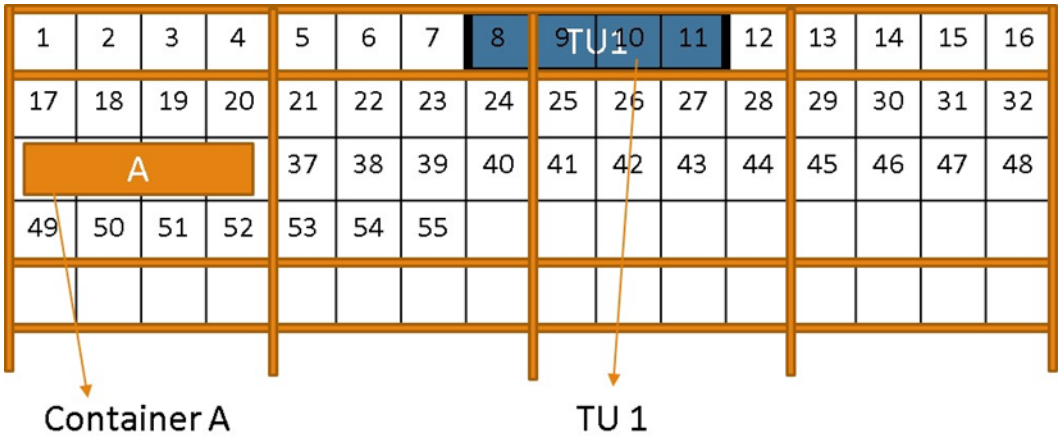


Figure 4.2 The gridding layout of GRID system

A 40ft standard container occupies four cells. As shown in Figure 4.2, Container A occupies Cells 33, 34, 35 and 36. The distance of two tracks belonging to a vertical rail (vertical to the berth) is the width of four cells and the distance of two tracks belonging to a horizontal rail (parallel to the berth) is the width of one cell. To indicate the location of the containers, we define the index of the first cell (counted from left to right) occupied by the container to be the index of the location of this container. For example, the location of Container A is indicated by Cell 33. Similarly, a TU also occupies four cells at a time, and the location of the TU can be indicated by the index of the first cell (also counted from left to right) occupied by the TU. The location of TU1 in Figure 4.2 is Cell 8.

To model our problem, we propose a definition of the adjacent set to describe the adjacent relations among the cells, which is used to determine how TUs move.

Adjacent set: Cell A is adjacent to Cell B if the TU can move from Cell A to Cell B at next time unit, which means the TU stays at Cell A at time t and it can stay at Cell B at time $t+1$. For simplicity, define a set denoted by $A(i)$ as

the cells that are adjacent to Cell i. Because the TU can move bi-directionally, thus if $j \in H(i)$, $i \in H(j)$ also holds. In addition, the TU can stay at its current place, and thus $i \in H(i)$ also holds.

For the cells in the row such that a horizontal rail is constructed above, their neighboring cells in the horizontal direction are the adjacent cells of them. As shown in Figure 4.2, a horizontal rail is constructed on the first row of the cells. Cell 5 and Cell 7 belong to the adjacent set $A(6)$. Whether the neighboring cells in the vertical direction belong to the adjacent set or not would depend on whether the TU can move directly between them. From the way we define the location of TUs, we can have, for example, Cell 21 belonging to $A(5)$, whereas Cell 22 is not an element of $A(6)$. The concept of adjacent set will help us construct a time-space network.

4.1.2 Avoidance of vehicle collisions

As the TUs are running on the rails, they cannot cross over each other when running on the same rail. There is one track shared by two contiguous vertical rails, and so the TUs running on the contiguous vertical rails also cannot cross over each other. As an illustration, in Figure 4.3, TU1 cannot move downwards if TU2 is going to keep staying at the current location or to move upwards. Different from vertical rails, each horizontal rail owns its exclusive tracks in order to improve the productivity of the system. The TUs running on two adjacent horizontal rails will not impact each other. For example, in Figure 4.3, TU1 can move rightwards while TU2 stays at its current place or is going to move leftwards. Therefore, we propose two terms to help prevent

vehicle collisions: one is safety zone and the other one is guide-path segment. The details are discussed as follows.

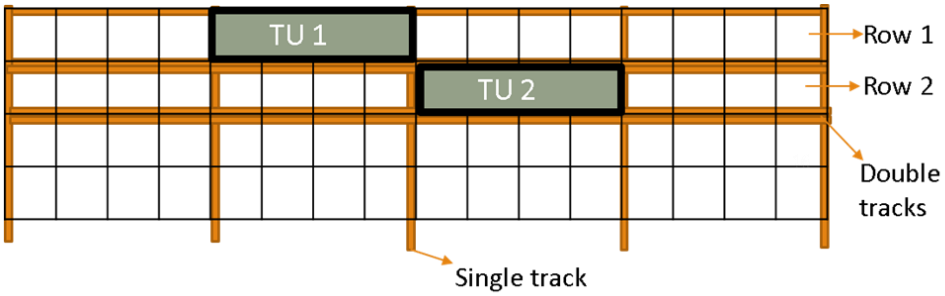


Figure 4.3 An illustration of collisions between TUs

Safety zone

To ensure the safety of the system and prevent the collisions mentioned above, we define the safety zone for each TU. The safety zone of one TU consists of the cells that are neighboring to those cells occupied by this TU. The cells occupied by this TU also belong to the safety zone. It should be highlighted that in some cases, the neighboring cells do not belong to the safety zone. These cases happen when the TUs are on horizontal rails. Because the horizontal rails have their unique tracks, the safety zone of the TU on the horizontal rail is different from the safety zone of the TU on the vertical rail. Figure 4.4 summarizes the safety zone of the TU staying at different locations.

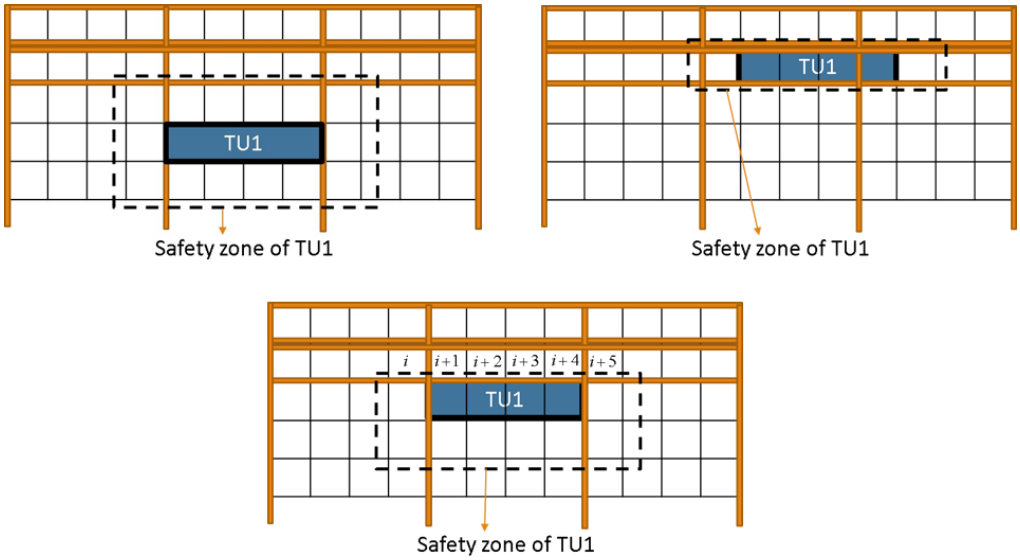


Figure 4.4 Safety zone of TU1 on different locations

There are three types of safety zone of one TU when it stays at different locations. The first type is that all the cells neighboring to the TU are counted in the safety zone. The second type is that only left and right cells neighboring to the TU are counted in the safety zone while the cells that are on the up-side or down-side of the TU are not counted. The third type is that the cells neighboring to the TU but belonging to the row in which there is a horizontal rail above it are not counted in the safety zone. Like the situation shown in the third part of Figure 4.4, the cells indexed as i to $i+5$ are not counted in the safety zone of TU1. The reason why the safety zone does not include all the neighboring cells is that we want to reduce the congestion on the horizontal rails where there is heavy traffic.

In summary, we can use a set of constraints called safety zone constraints to prevent vehicle collisions. Safety zone constraints are described as: at every time, TUs cannot occupy any cells belonging to the safety zone of the other

TUs. However, the cross-over collisions cannot be captured by the safety zone constraints. Thus, guide-path segment is proposed to prevent the collisions.

Guide-path segment

The TUs may still confront cross-over collisions. They cannot be prevented by safety zone constraints, when they are moving vertically in the area where there are horizontal rails. We assume the scenario shown in Figure 4.2 in which TU1 is at Cell 5 at time t and TU2 is at Cell 25. TU1 is going to move downwards while TU2 is going to move upwards. Thus, at time $t+1$, TU1 is at Cell 21 and TU2 is at Cell 9. We can find that both at times t and $t+1$, the safety zone constraints are not violated. However, this scenario must be prevented because of the cross-over of the TUs on the contiguous rails. Similarly, if two TUs are on the same rail, safety zone constraints also cannot prevent this cross-over collision. Therefore, we propose a concept called guide-path segment to help build constraints to prevent this kind of vehicle collision.

If a path satisfies the following two conditions, it is a guide-path segment.

(1) At least one cell of the path is on the row where a horizontal rail is built;

(2) TUs can move vertically along it.

A set denoted as $F(w)$ is proposed to help build the constraints of avoidance of vehicle collisions. $F(w)$ is the set of guide-path segments where TUs cannot travel at the reverse direction when a TU travels on guide-path segment w. Intuitively, we have $w \in F(w)$. We call the other guide-path segments in set $F(w)$ as adjacent to w. By the definition of guide-path segments, we can

propose a set of constraints to prevent vehicle collisions, that is if one TU enters a guide-path segment denoted as w at time t and is going to move downwards, then any other TUs cannot enter the guide-path segments which are adjacent to w as well as w itself, at time t and move upwards. We can find that the scenario mentioned at the beginning of this subsection can be detected by this set of constraints.

4.1.3 Decoupling operations between QCs and TUs

In this system, a buffer is placed at the quayside. The containers can be temporarily stored on the buffer. To simplify the operations involved in the quayside, we assume that the capacity of the buffer is sufficient. Thus, we can decouple the operations between QCs and TUs. TUs can directly put down the containers on the buffer without waiting for the QCs. The QCs will pick up and deliver the containers according to the pre-specified QC lists. However, we should still need to prevent the buffer from becoming overcrowded. Hence, the TUs should operate the container jobs with consideration of the QC lists, which means it should avoid delivering containers whose precedent containers are not completed. In other words, when we solve the assigning problem, the container jobs with fewer predecessors should be assigned first.

4.1.4 Activities for discharging and loading jobs

The container jobs can be categorized into discharging jobs and loading jobs. Similarly, each container job consists of two subtasks. A subtask is the activity that the TU loads or unloads a container. For a loading job, the activities can be described as follows. A TU picks up the container from the slot in the storage yard. This is the start of the first subtask of the container job. Once the container is fixed on the TU, this is the end of the first subtask. Then, the TU

moves along the path which is conflict-free with other TUs. When the TU arrives at the transfer point in the quayside, the TU starts to unload the container to the buffer. This is the start of the second subtask. The end of the second subtask is the time that the container is released on the buffer. At this time, the TU is free to move to its next container job. The container job is then completed. For a discharging job, we assume that the corresponding container is always ready on the buffer. When a TU arrives at the transfer point in the quayside, the TU starts to pick up the container from the buffer. This is the start of the first subtask. Once the container is fixed with the TU, the first subtask ends. Then, the TU moves along a conflict-free path to the place where the container will be stored in the yard. When the TU arrives at this place, it starts to release the container onto the place. This is the start of the second subtask. Once the container is released onto the place, the second subtask ends and the container job is also completed at this time.

4.2 Mathematical Model

The input of our problem is the information about the location of the origin and destination of a set of containers as well as the pre-specified QC lists. Our objective is to minimize the total processing time of a given set of containers, considering the vehicle collisions and capacity of QCs. We need to solve the container assignment problem so as to know each container will be assigned to which TU, and the routing problem to know how the TUs accomplish the transportations without vehicle collisions.

To be able to track the position of a TU at any given time, a time dimension is introduced to the pure cell layout and hence a time space network is created. In a time space network, a cell is duplicated many times (up to the maximum

time unit) to represent the same cell at different times. The maximum time unit can be the upper bound of the makespan of any feasible solution.

Based on this time space network, we can model the problem as a set partitioning problem, where each decision variable represents a path of how the TU transports the containers. The notations of the mathematical model are given below:

Parameters:

K	Set of all TUs
N	Set of all cells
T	Set of all times
J	Set of all container jobs
W	Set of all guide-path segments
$L(n)$	Set of all feasible routes for TU n
$\lambda(n)$	Set of all feasible routes generated in the route generating problem for Vehicle n
$G(i)$	Set of all cells in the physical layout that are within the safety distance to Cell i but excluding Cell i
$F(w)$	Set of guide-path segments that are adjacent to w $(w \in W)$ and including itself, that is $w \in F(w)$
$A(i)$	Set of all cells in the physical layout that are adjacent to Cell i

p_τ Processing time of subtask τ

$c_{l,n}$ The total penalty cost of route l ($l \in L(n)$), where it equals to the total time that the route accomplishes its related subtasks

$A_{(i,t),l,n}$ $\begin{cases} 1, \text{if TU } n, \text{following route } l \text{ is at cell } i \text{ at time } t \\ 0, \text{otherwise} \end{cases}$

$B_{l,n}^{i}$ $\begin{cases} 1, \text{if TU } n, \text{following route } l \text{ covers job } i \\ 0, \text{otherwise} \end{cases}$

$f_{w^-,l.n}^{t}$ $\begin{cases} 1, \text{if TU } n, \text{following route } l \text{ enters segment } w \text{ at} \\ \quad \text{an upwards direction at time } t \\ 0, \text{otherwise} \end{cases}$

$f_{w^+,l.n}^{t}$ $\begin{cases} 1, \text{if TU } n, \text{following route } l \text{ enters segment } w \text{ at} \\ \quad \text{a downwards direction at time } t \\ 0, \text{otherwise} \end{cases}$

Decision Variables:

$X_{l,n}$ $\begin{cases} 1, \text{if TU } n \text{ follows route } l \\ 0, \text{otherwise} \end{cases}$

The assumptions of the problem are made:

(1) Yard slot of each job is known;

(2) The traveling speed of empty and loaded TUs is the same, and the related acceleration and deceleration are not considered;

(3) The capacity of the buffer on quayside is set to be efficient;

(4) Number of container jobs, number of TUs and QCs are all known.

The problem is formulated as follows:

Problem (P_0)

Objective:

$$Minimize \quad \sum_{n\in K}\sum_{l\in L(n)} c_{l,n}X_{l,n} \tag{4.1}$$

Constraints:

$$\sum_{l\in L(n)} X_{l,n} = 1, \quad \forall n \in K \tag{4.2}$$

$$\sum_{l\in L(n)}\sum_{n\in K} B^{i}_{l,n}X_{l,n} = 1, \quad \forall i \in J \tag{4.3}$$

$$\sum_{l\in L(n)}\sum_{n\in K} A_{(i,t)l,n}X_{l,n} + 0.5\sum_{j\in G(i)}\sum_{l\in L(n)}\sum_{n\in K} A_{(j,t)l,n}X_{l,n} \le 1, \quad \forall i \in N, t \in T \tag{4.4}$$

$$\sum_{l\in L(n)}\sum_{n\in K} f^{t}_{w^{-},l,n}X_{l,n} + 0.5\sum_{w\in F(w)}\sum_{l\in L(n)}\sum_{n\in K} f^{t}_{w^{+},l,n}X_{l,n} \le 1, \quad \forall w \in W, t \in T \tag{4.5}$$

$$X_{l,n} \in \{0,1\}, \quad \forall l \in L(n), n \in K \tag{4.6}$$

Objective function (4.1) minimizes the total time to accomplish all subtasks. Constraint set (4.2) restricts a TU to select only one route from the set of the possible routes. Constraint set (4.3) ensures that every job is visited only once. Constraint set (4.4) ensures the safety distance is maintained between TUs in order to prevent vehicle collisions. Constraint set (4.5) prevents the cross over among TUs in the vertical direction. For example, one TU enters a segment denoted as w_i at time t in the "-" direction, that is $f^{t}_{w_i^{-},l,n}X_{l,n} = 1$. Then other TUs cannot enter the segment w_j and w_k in the reverse direction, where w_j

and $w_k \in F(w_i)$. However, if there is no TU entering the segment w_i in the "-" direction at time t, the situation is acceptable that one TU (n_j) enters the segment w_j in the "+" direction at time t and another TU (n_k) enters the segment w_k in the "+" direction at time t. Thus $0.5*(f^t_{w_j^+,l,n} X_{l,n_j} + f^t_{w_k^+,l,n_k} X_{l,n_k}) = 1$, which does not violate the constraint. Finally, constraint set (4.6) specifies the variables to be binary.

Note that the second components of the left-side in Constraint sets (4.4) and (4.5) have a coefficient equaling to 0.5. This is because when the first component of the left-side equals to zero, the situation that the TUs move near the location is allowed. Let us give an illustration on Constraint set (4.5). We have $w_i \in F(w_k)$, $w_j \in F(w_k)$. If no TU moves on w_k^- at time t, we allow one TU to move on w_i^+ and another one TU to move on w_j^+ at time t. Thus the second component equals to one where the constraint still holds. The explanation is similar for Constraint set (4.4).

4.3 Heuristic Method

In this section, we discuss the solution approach to solve our problem. The algorithm is developed based on the tree structure, and two methods are embedded to help solve the conflict-free routing problems under the given schedules of vehicle dispatching problems.

4.3.1 Decomposition

There are two aspects that determine the solvability of the mathematical model proposed in Section 4.2. One is the number of jobs and the other one is the complexity of routes including the number of TUs, the length of routes and the

potential collisions among TUs. Unfortunately, the proposed mathematical model is intrinsically difficult to be solved under the environment of the GRID system in container terminals. Firstly, the number of containers that need to be transferred is quite large. Secondly, the layout of the container terminal is large, resulting in a complex routing problem. Thirdly and the most significant factor, the large amount of potential collisions would require an elaborate route for each TU. Therefore, we will decompose our problem into two sub-problems: assignment problem and routing problem. We will solve the assignment problem first, and then solve the routing problem based on the results from the assignment problem.

The assignment problem can be considered as a combinatorial problem. As the solution space of the assignment problem is dramatically large, it is difficult to find the optimal assignment even when ignoring the routing problem. One promising way is to determine the scheduling only within a short-term look-ahead planning horizon and then rolling the planning horizon step by step while the scheduling made in the previous planning horizon is kept. The complete scheduling can be obtained when all the jobs are involved. During one short-term planning horizon, the assignment problem is solved first, then the routing problem is solved under the solution of the assignment problem. The tree structure algorithm can fit this mechanism, because the tree structure algorithm expands the tree step by step and the nodes of next level in the tree inherit the results of the upper level. The procedure of expanding the tree is the counterpart of the rolling the planning horizon. The leaf nodes in the tree are complete solutions while the nodes of other levels are partial solutions.

In conclusion, we will implement the tree structure algorithm to solve the problem in this study. Different from the one in Chapter 3, another two embedded methods are proposed. One is a heuristic sequential path generation (HSPG), which is easily implemented to solve the routing problem, while the other one is a column generation algorithm with further search procedure which can solve the routing problem more accurately. This second method is called CGA-FS in short. The first method HSPG is used in the screening procedure, while the second method CGA-FS is used in the selection of new beam nodes. The related contents of the algorithm will be discussed as follows.

4.3.2 Development of the tree structure algorithm

To illustrate our tree structure algorithm clearly, some related terms are needed to be defined first.

Beam node: the nodes in the tree which will be further explored. It represents the solution of involved subtasks showing how these subtasks are assigned to vehicles and how to route the vehicles on the layout without collisions.

Partial solution: A beam node in the tree structure, except at the leaf level, represents a partial solution when not all of the subtasks have been assigned.

Complete solution: A beam node at the leaf level in the tree structure represents a complete solution that all subtasks are accomplished.

Assigned subtask: The subtask in the beam node is called an assigned subtask. The involved decision variables are determined for these subtasks.

Unassigned subtask: A subtask that has not been assigned to the beam node is called an unassigned subtask.

Assigned subtask with partial decisions: This is a subtask that is assigned to a beam node during the procedure of expanding a beam node. Only the decision about the vehicle assignment, as well as the sequence of this subtask in the corresponding vehicle, is determined, while the involved decisions about the routing are not determined.

Now, we can give the explicit procedure of how the tree structure algorithm searches for a complete solution. At the root level of the tree, there is only one beam node that is an empty set. A large number of intermediate nodes are generated by selecting unassigned subtasks and assigning them to vehicles. In order to avoid the situation that the buffer becomes over-crowded by storing too many loading containers, or the situation that the TUs have to wait for the discharging containers because those containers cannot be picked up by the QCs until their predecessors have been picked up, we will select the unassigned containers which are on the top of the QC lists. After assigning Δ subtasks to each TU, an intermediate node is created. Δ is the look-ahead number of subtasks of each TU during the next short-term planning horizon. Then, the HSPG method will be used to assess the fitness of this intermediate node. A related small number of intermediate nodes are retained after the screening procedure based on the results from the HSPG method. The CGA-FS method is developed to solve the routing problem of the retained intermediate nodes again. The best m intermediate nodes according to the results of the CGA-FS method are selected as new beam nodes at next level. The subtasks in the beam nodes becomes assigned subtasks and are removed from the set of unassigned subtasks. The tree is expanded in this way until the

leaf level is approached. The framework of the tree structure algorithm can be described by Figure 4.5.

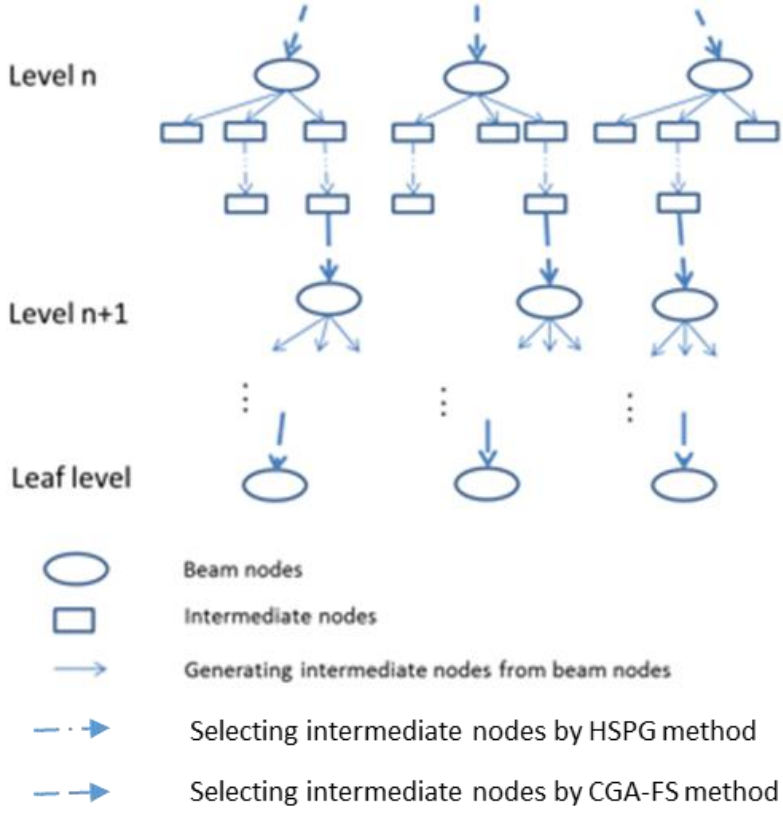


Figure 4.5 Framework of filtered beam search-based algorithm

The following parts will give the details of the HSPG method and CGA-FS method. These two methods aim to find conflict-free routes for the TUs. In order to solve the routing problem, the time-space network is built. The mechanism of determining the cost of arcs in the time-space network is the core of these two methods.

4.3.3 Time-space network

The network is a time-space acyclic network $G = (V, A)$, where V and A represent the set of vertices and the set of directed arcs, respectively (shown in Figure 4.6).

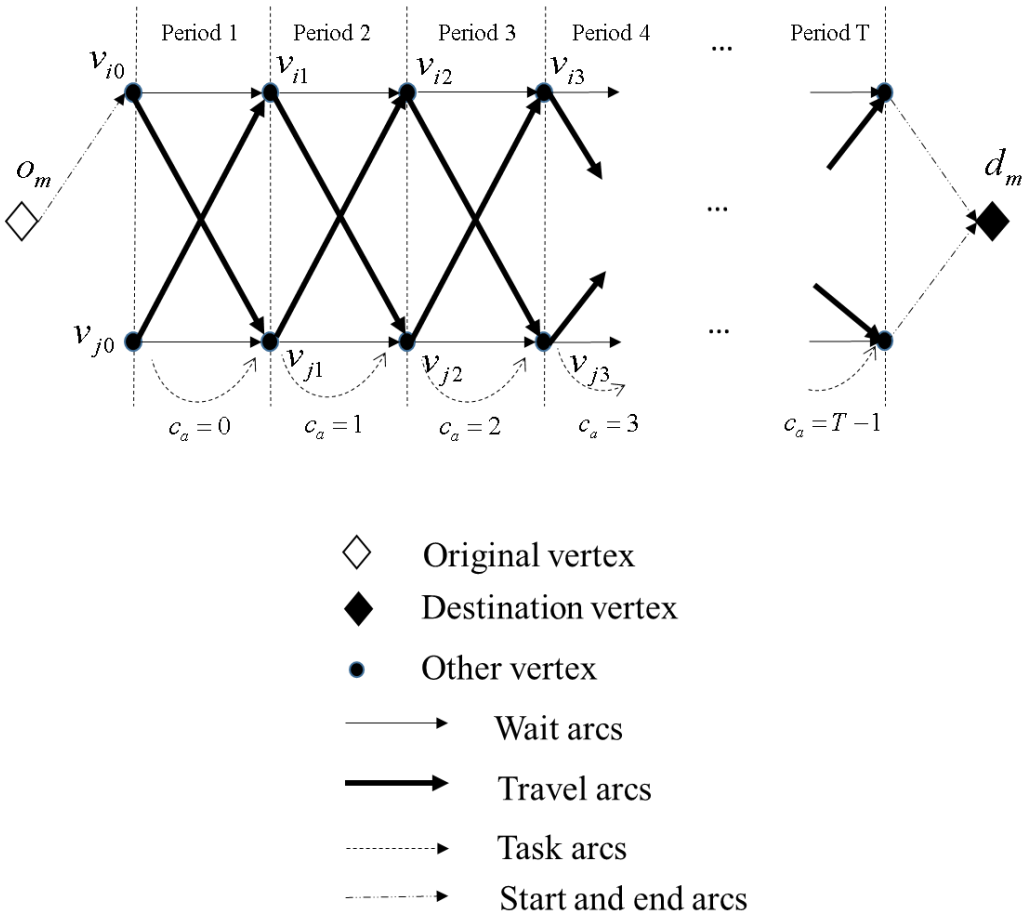


Figure 4.6 Part of a mathematical network

There are two types of vertices:

- To each location $n \in N \cup S$, there corresponds $T+1$ vertices $v_{nl} \in V$ (one vertex for each time t).
- An origin vertex $o_m \in V$ and a destination vertex $d_m \in V$ are associated with each vehicle $m \in K$ to represent the start and the end of its route respectively. The set of origin and destination vertices is denoted as OD.

The arc set A includes five types of arcs which correspond to moves in space and/or time:

- The start arcs link the origin vertex o_m of each vehicle to the vertex $v_{nl} \in V$ corresponding to its initial position.

- The end arcs link all vertices v_{nT} (T is the last period of the horizon) to each destination vertex $d_m, m \in K$.
- The waiting arcs are defined for all pairs of vertices $(v_{nt}, v_{n,t+1}) \in V \setminus OD \times V \setminus OD$. These arcs represent a vehicle waiting at a given location for one period of time.
- The travel arcs represent the movements between two adjacent locations. These arcs are represented as $(v_{it}, v_{j,t+1})$ where i and j are adjacent locations.
- Let $\tau \in QL$ be a subtask to be operated at the quayside transfer point and the slot in storage yard. A task arc $(v_{nt}, v_{n,t+h}) \in V \setminus OD \times V \setminus OD$ is defined if this subtask can begin at time t.

The sets of start and task arcs are denoted by A^S and A^T, respectively, while the set of travel arcs is denoted as A^H. A^H contains two subsets denoted as A^W and A^G. A^W represents the travel arcs whose paths are guide-path segments, while A^G represents the travel arcs whose paths are not guide-path segments. A^W is divided into subsets A^{W-} and A^{W+} according to the direction (+ or -) used to travel along the corresponding guide-path segment. Similarly, A^G is divided into subsets A^{G-} and A^{G+}.

4.3.4 HSPG method

The HSPG method should be fast in computation. It generates the route of each TU in a decentralized manner. The way of decentralization means that we determine the routes of TUs sequentially. We fix the route of one TU, and

the following routes of the other TUs cannot violate the previous determined routes. However, it is not a good way to determine the route of one TU from its first job to its last job and then move to fix the route of another TU, because the TUs at the back of the sequence will confront a high chance to sustain a large delay time. Therefore, this sequence of TUs is changed after all TUs accomplish one job. Some concepts are introduced below to help illustrate the HSPG method.

TU sequence: We define a sequence called the TU sequence which determines the order of TUs to solve their routing problem. The TU sequence is indicated as Seq, where $Seq=(a,b,..,m)$ and the element in the sequence is the index of the TUs. The number of the elements in the sequence is the number of TUs. The TU in the first place of the sequence will find its route first. After the scheduling of the route for the first TU, the TU with the second order in the sequence will start to find its route.

Short-term Planning horizon: The short-term planning horizon in our method is not a time window with a fixed length. The end time of the horizon is the time that all TUs complete their assigned jobs. The start time of the horizon is the end time of last short-term planning horizon.

Unproductive time: Unproductive time of a TU includes the waiting time, empty movement and the additional movement time. The time of waiting is that the TU stays at the current place to prevent the collisions. When a TU completes a job, it will start to move to the location of the next job. This movement is called empty movement. Empty movement is inevitable. If the collisions among the TUs are neglected, the shortest path of each TU is

intuitive. The time of the shortest path can be calculated. The difference of the time of a feasible route and shortest path is defined as the additional movement time.

Cost of arcs: In the HSPG method, the cost of the arcs in the time-space network can only be one or infinite. If the arc is available for the TU, it is valued with one, otherwise it becomes infinite. The arc is available which means the TU can transfer from the head node to the tail node. There are two situations that the cost of the arc is set to be infinite.

I. If the cell corresponding to the head node of the arc is occupied by other TUs at the same time, the cost of the arc is set to be infinite.

II. If the TU needs to stay at the current cell for a certain time to accomplish the process such as making a turn, the cost of the arc is infinite when the cell of the tail node is not the current cell that the TU is staying at.

The total unproductive time of the TUs should be minimized to improve the efficiency of the system. To pursue this goal, the TU sequence is determined by the unproductive time of the TUs. The TU with larger unproductive time will be assigned with a smaller order in the TU sequence. The route of the TU is obtained according to the cost of the arcs in the network. Once a route is obtained, the cost of the arcs whose corresponding cells are involved in the route will be infinite. Therefore, we can find that there are more arcs whose cost is infinite if the order of the TU in the TU sequence is large. Consequently, the unproductive time may increase.

We will discuss the way to determine the routes of the TUs as follows.

When the cost of the arcs in the network is determined, we will find the paths with minimum cost in the time-space network as the routes of the TUs. A recursive formula is adopted to find the path with minimum cost. The recursive formula is presented as:

$$f(i,t) = \min_{j \in I}(f(j,t-1) + R_{(j,t-1)(i,t)}),$$

where $f(i,t)$ is the cost of the path that arrives at node (i,t), and $R_{(j,t-1)(i,t)}$ is the cost of the arc that connecting the nodes $(j,t-1)$ and (i,t). The value of $R_{(j,t-1)(i,t)}$ is either one or infinite according to the criteria mentioned before.

Given the pair of origin and destination, the route with minimum cost is also the feasible route with minimum time. Because we set the cost of infeasible arcs to infinite, the route obtained by the recursive formula will only contain the arcs with cost equaling to one. Therefore, the route is a feasible route. In addition, the cost of available arcs equals to one. Hence, the cost of the route is also the time the route consumes from the origination to the destination. Consequently, the route from the origination to the destination with a minimum cost is the shortest route.

The HSPG method can be summarized by the following pseudo-code:

For i =1 to Δ, **do**

(i) Generate TU sequence Seq based on the unproductive time of the TUs;

For i =1 to m , **do**

(a) Determine the route of the ith TU in the Seq by the method of recursive formula;

(b) Update the cost of the arcs in the network;

End for

(ii) Update the unproductive time of each TU;

End for

Δ is the number of look-ahead jobs of each TU. The look-ahead jobs are the jobs that will be scheduled during the short-term planning horizon. m is the number of TUs.

4.3.5 CGA-FS method

The CGA-FS method solves the routing problem more accurately than the HSPG method. Different from the HSPG method, column generation algorithm is implemented to determine the cost of the arcs in the network. However, because of the high potential of vehicle collisions as well as the congestions in transfer area, it is hard to get an improved feasible solution from column generation algorithm. Therefore, we will continue to find new columns by adding new constraints after the typical column generation algorithm meets the stopping criteria. The further search will be terminated when we obtain an integer solution. The following contents include the implementation of column generation algorithm in the problem, and the way

we use the information from the results of column generation algorithm to obtain an improved feasible solution.

Master model

The mathematical model is built in the section of problem definition. Because we decompose the problem into two sub-problems, the mathematical model in this section will not consider the assignment problem anymore. The model is reduced to concentrate on the routing problem, which is shown in a model denoted as RP_0. The model RP_0 is the same as P_0 but without the set of constraints (4.3). Thus, we will not show RP_0 here.

We will implement column generation algorithm to help solve RP_0. RP_0 is divided into two interrelated problems; the master problem (MP) and route generating problem (RGP). The MP is a relaxed, as non-integer values are allowed, and restricted problem of RP_0 because only a subset of the decision variables (or columns), $X_{l,n}$ are generated by RGP. In each iteration of the column generation procedure, the MP is solved to optimality. According to the dual values of the constraints, RGP generates columns (routes) for TUs that have the potential to improve the objective function of MP. This iterative process stops when no more improving columns are generated.

The master problem is presented as:

Problem (MP)

Objective:

$$Minimize \quad \sum_{n\in K}\sum_{l\in\lambda(n)} c_{l,n}X_{l,n} \tag{4.8}$$

Constraints:

$$\sum_{l\in\lambda(n)} X_{l,n} = 1, \quad \forall n \in K \tag{4.9}$$

$$\sum_{l\in\lambda(n)}\sum_{n\in K} A_{(i,t)l,n}X_{l,n} + 0.5 * \sum_{j\in G(i)}\sum_{l\in\lambda(n)}\sum_{n\in K} A_{(j,t)l,n}X_{l,n} \le 1, \quad \forall i \in N, t \in T \tag{4.10}$$

$$\sum_{l\in\lambda(n)}\sum_{n\in K} f^{t}_{w^{-},l,n}X_{l,n} + 0.5\sum_{v\in F(w)}\sum_{l\in\lambda(n)}\sum_{n\in K} f^{t}_{v^{+},l,n}X_{l,n} \le 1, \quad \forall w \in W, t \in T \tag{4.11}$$

$$X_{l,n} \ge 0, \quad \forall l \in \lambda(n), n \in K \tag{4.12}$$

Route Generating Problem

In this part, we will discuss how to generate new routes which will be added to the master model. The route generating problem consists of finding the shortest path (that is, the route with the least reduced cost) in network G. To compute the reduced cost of a path, the arc costs c_a are replaced by arc reduced cost $\bar{c}_a$ derived from the dual solution of the current restricted master model. Such a reduced cost is given by $\bar{c}_a = c_a - \delta(a)$, where $\delta(a)$ is the sum of all the dual contributions for this arc a. These contributions are calculated according to which type the arc belongs to. The contributions are shown in Table 4.1, where $\delta^{(1)}_k$ $(k \in K)$, $\delta^{(2)}_{n,t}$ $(n \in N \cup S, t \in T)$, $\delta^{(3)}_{w,t}$ $(w \in W, t \in T)$, denote the dual variables associated with constraints (4.9)-(4.11), respectively. Furthermore, τ_a represents the subtask associated with arc $a \in A^T$, t^s_a and t^e_a

are the start and end times associated with arc $a \in A$, i_a is the tail vertex of arc $a \in A$, and w_a is the guide-path segment associated with arc $a \in A^W$.

Table 4.1 Dual contributions to arc reduced costs

If arc $a \in A$ satisfies	Add this contribution to its reduced cost
$a \in A^S, i_a = o_m$	$\delta_m^{(1)}$
$a \notin A^T$	$\delta_{j_a,t_a^i}^{(2)} + 0.5 * \sum_{j \in G(i_a)} \delta_{j,t_a^i}^{(2)}$
$a \in A^T$	$\sum_{t=t_a^s}^{t_a^e-1} (\delta_{j_a,t}^{(2)} + 0.5 * \sum_{j \in G(i_a)} \delta_{j,t}^{(2)})$
$a \in A^W$	$\delta_{w_a,t_a^i}^{(3)} + \delta_{j_a,t_a^i}^{(2)} + 0.5 * \sum_{j \in G(i_a)} \delta_{j,t_a^i}^{(2)}$
$a \in A^G$	$\delta_{j_a,t_a^i}^{(2)} + 0.5 * \sum_{j \in G(i_a)} \delta_{j,t_a^i}^{(2)}$

A dynamic programming algorithm is used to solve the shortest path problem. This dynamic programming algorithm is a simple pushing algorithm that is performed after ordering the vertex of G in topological order.

Procedure of further search

In this section, we will first discuss the reason why column generation algorithm cannot solve the routing problem well in the GRID system. After that, we will present the procedure of further search to obtain a promising and feasible solution.

There are two reasons why column generation algorithm cannot guarantee an optimal solution for a problem modelled as a set partitioning model. Firstly, when the algorithm stops, the optimal solution is found for the relaxation of the model, not the model itself. Secondly, we concentrate on searching the columns with negative reduced costs. But it is possible that a column of the optimal solution to the routing model RP_0 has a positive reduced cost. However, this column will not be generated during the iterations. The optimal solution to the master model actually is a lower bound to the routing model RP_0. The optimal solution of the master model is likely to be fractional, because vehicles have high collision potential under the GRID-ACT system.. The solution of the routing model RP_0 when the procedure of column generation algorithm is stopped, may not improve a lot compared to the initial solutions generated by the sequential paths generation algorithm, or may even be the same as the initial solutions. Thus, a procedure of further search is proposed to improve the solution. The motivation of the further search is to add new constraints to the master model (MP) so that we can get new dual values to continue the column generation iterations. The new constraints aim to reserve the node or guide-path segment to one TU and prevent other TUs from using it, which helps to approach feasibility. We will stop the procedure until the solution of the master model is integer.

We define a set to record the information of new constraints which is denoted as A, where the elements in set A is (α,t,k). (α,t) represents the constraint whose corresponding dual values are most negative in the current model MP. If the constraint belongs to the set of constraints (4.10), we have $\alpha \in N$, while

if it belongs to the set of constraints (4.11), we have $\alpha \in W$. The indicator k represents the TU whose routes contribute to the constraint most, that is the value denoted as $\delta(m)$ is largest, where $\delta(m)$ equals to

$\sum_{l\in\lambda(m)} X_{l,m} A_{(\alpha,t)l,m} + 0.5 \sum_{l\in\lambda(m)} \sum_{j\in G(\alpha)} X_{l,m} A_{(j,t)l,m}$ if $\alpha \in N$ or

$\sum_{l\in\lambda(m)} X_{l,m} f^{t}_{w^-,l,m} + 0.5 \sum_{l\in\lambda(m)} \sum_{v\in F(w)} X_{l,m} f^{t}_{v^+,l,m}$ if $\alpha \in W$. Thus, a new constraint is created and added to the model MP, where the constraint is represented as:

$$\sum_{m\in K, m\neq k} \sum_{l\in\lambda(m)} X_{l,m} A_{(\alpha,t)l,m} + \sum_{m\in K, m\neq k} \sum_{l\in\lambda(m)} \sum_{j\in G(\alpha)} X_{l,m} A_{(j,t)l,m} = 0, \quad \forall(\alpha,t,k) \in A, \alpha \in N,$$

or

$$\sum_{m\in K, m\neq k} \sum_{l\in\lambda(m)} X_{l,m} f^{t}_{w^-,l,m} + \sum_{m\in K, m\neq k} \sum_{l\in\lambda(m)} \sum_{v\in F(\alpha)} X_{l,m} f^{t}_{v^+,l,m} = 0, \quad \forall(\alpha,t,k) \in A, \alpha \in W.$$

We define these new constraints as (4.8a). These new constraints aim to ensure certain TUs will not use the nodes or guide-path segments so as to prevent vehicle collisions. After the new constraint is added to model MP, we solve the model again. We will update the cost of the arcs according to the dual values. When $(\alpha,t,k) \in A$, we want to prevent the TUs which are not the TU k from occupying the node (β,t) where $\beta \in G(\alpha)$ and $\alpha \in N$, or using the segment (β,t) where $\beta \in F(\alpha)$ and $\alpha \in W$. Thus, we set the cost of the arcs involved with (β,t) to be infinitely large. Consequently, the new generated routes will not violate the constraints (4.8a).

However, if there are more than one TUs whose value of $\delta(m)$ is close to the maximum value, we will create branches. For example, assume $\delta(m_i)$ is the largest among $\delta(m)$, where $m \in K$. For any TU n such that $\delta(m_i) - \delta(n) \leq \varepsilon$,

where ε is a threshold value that should be set to a small value, we will create the branches where the new constraint is created under a different TU. The further search will stop if the solution of the model *MP* in any branch is integer. The TU will select the route such that the corresponding variable value equals to one.

We generate 20 instances to see how much improvement the further search can obtain. Each instance contains the information including the number of TUs, the starting point of each TU, the container jobs of each TU, the origination and destination of each container job. Table 4.2 presents the results of CGA without further search and CGA with further search. The result is the total time to accomplish the container jobs. The lower bound is the optimal solution of the master problem of the instance. Gap1 in the fifth column is the difference of the results between CGA and CGA-FS. Gap2 in the seventh column is the difference of the results between CGA-FS and lower bound. The fourth column in Table 4.2 is the number of new constraints that added to the master problem during the further search.

In cases P1-P10, each TU is assigned with two container jobs, while each TU is assigned with three container jobs in the cases P11-P20. In cases P1, P2, P11 and P12, there are 3 TUs and 2 QCs. In the cases P3, P4, P13 and P14, there are 4 TUs and 2 QCs. In cases P5, P6, P15 and P16, there are 6 TUs and 3 QCs. In cases P7, P8, P17 and P18, there are 8 TUs and 4 QCs. In cases P9, P10, P19 and P20, there are 10 TUs and 4 QCs.

Table 4.2 Comparison of CGA-FS, CGA and LB

Case	CGA	CGA-FS	# of new constraints	Gap1 (%)	Lower bound	Gap2 (%)
P1	13.31	12.62	5	5.18	12.34	2.27
P2	13.70	13.20	7	3.65	12.88	2.48
P3	17.82	16.76	9	5.95	16.35	2.51
P4	19.02	17.55	12	7.73	16.92	3.72
P5	28.32	25.86	15	8.69	25.11	2.99
P6	30.11	27.22	18	9.60	25.87	5.22
P7	45.23	39.35	22	13.00	37.32	5.44
P8	44.00	38.12	24	13.36	36.70	3.87
P9	67.32	58.31	34	13.38	55.22	5.60
P10	66.05	56.20	32	14.91	54.00	4.07
P11	19.66	18.91	8	3.81	18.42	2.66
P12	19.65	18.20	8	7.38	17.88	1.79
P13	27.80	26.32	9	5.32	25.54	3.05
P14	29.00	27.12	9	6.48	26.23	3.39
P15	45.35	40.35	13	11.03	38.72	4.21
P16	44.11	38.76	17	12.13	37.50	3.36
P17	66.22	56.30	24	14.98	53.88	4.49
P18	69.05	54.62	20	20.90	51.88	5.28
P19	106.25	89.92	28	15.37	85.22	5.52
P20	110.11	93.68	26	14.92	89.01	5.25

The results show that the CGA-FS can improve the solution obtained from the CGA. The number of additional new constraints is increased when the number of TUs gets larger, because we need more efforts to handle the vehicle collisions when the more TUs run on the system.

A summary of the CGA-FS algorithm is shown next:

1.Initialize a certain number of routes for each TU

2. **while** $Imp\_num \leq Imp\_num\_max$ or $Ite\_num \leq Ite\_num\_max$, **do**

(1) solve the restricted master problem with relaxation of the integer variables;

(2) calculate the reduced costs of arcs in the network according to the dual values;

(3) find new column for each TU by the shortest path dynamic programming algorithm under the network;

(4) add the new generated routes to the master model;

(5) $Ite\_num = +Ite\_num$;

(6) **if** the optimal value of the relaxation of the restricted master model is same as previous iteration, **do**

$Imp\_num = +Imp\_num$;

end if.

end while

3. Solve the master model without relaxation of the integer variables; If the objective value is the same as the optimal value of the relaxation of the restricted master model, Stop; Otherwise go to Step 4;

4. Start the further search procedure;

4.1 find the constraint whose dual value is most negative;

4.2 find the TU which contributes to this constraint most;

4.3 create branches, and add a new constraint to MP of each branch;

4.4 solve the MP, and update the cost of the arcs;

4.5 generate a new route for each TU and add these new columns to MP;

4.6 solve the MP; if the solution of any branch is integer, Stop; otherwise, go back to Step 4.1.

Here, Imp_num_max is the maximum number that we allow for the successive iterations to be without improvement, and Ite_num_max is the maximum number of iterations for generating new routes.

4.4 Heuristic rules algorithm

We propose an algorithm which implements several heuristic rules. The results of the heuristic algorithm are used to compare with that of our algorithm. In this heuristic algorithm, we will also use the HSPG method introduced in Section 4.3.4 to solve the routing problem. However, the TU sequence is randomly generated instead of basing on the unproductive time of TUs. The way of job dispatching is based on two vehicle-initiated rules: FCFS (first come first serve) rule and STT (shortest travel time) rule. The STT rule tries to reduce the unproductive move of TUs because they will be assigned with the jobs which are most near them. The FCFS rule is applied to prioritize the waiting transportation orders.

The framework of the heuristic algorithm can be presented by Figure 4.7. The algorithm runs repeatedly until the stopping criteria is met. The stopping

criteria can be set as the computation time or the number of solutions we obtain. Finally, we select the best solution obtained by the algorithm.

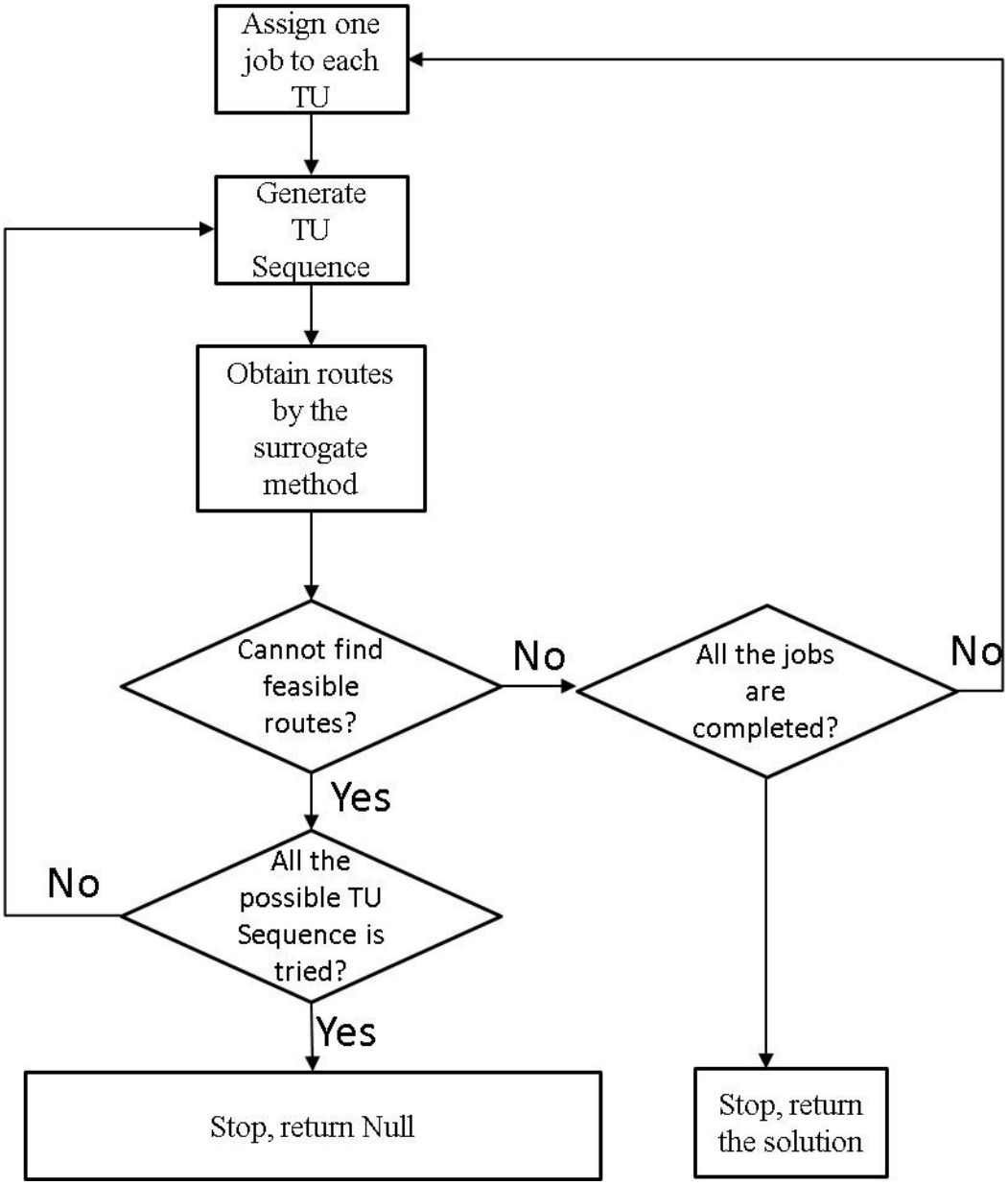


Figure 4.7 The framework of the heuristic rules algorithm

4.5 Numerical experiments

The numerical experiments consists of two major parts. The first part aims at accessing the effectiveness of our algorithm. The second part is the study on the GRID-ACT system.

4.5.1 Effect of the filter-width on the performance of the algorithm

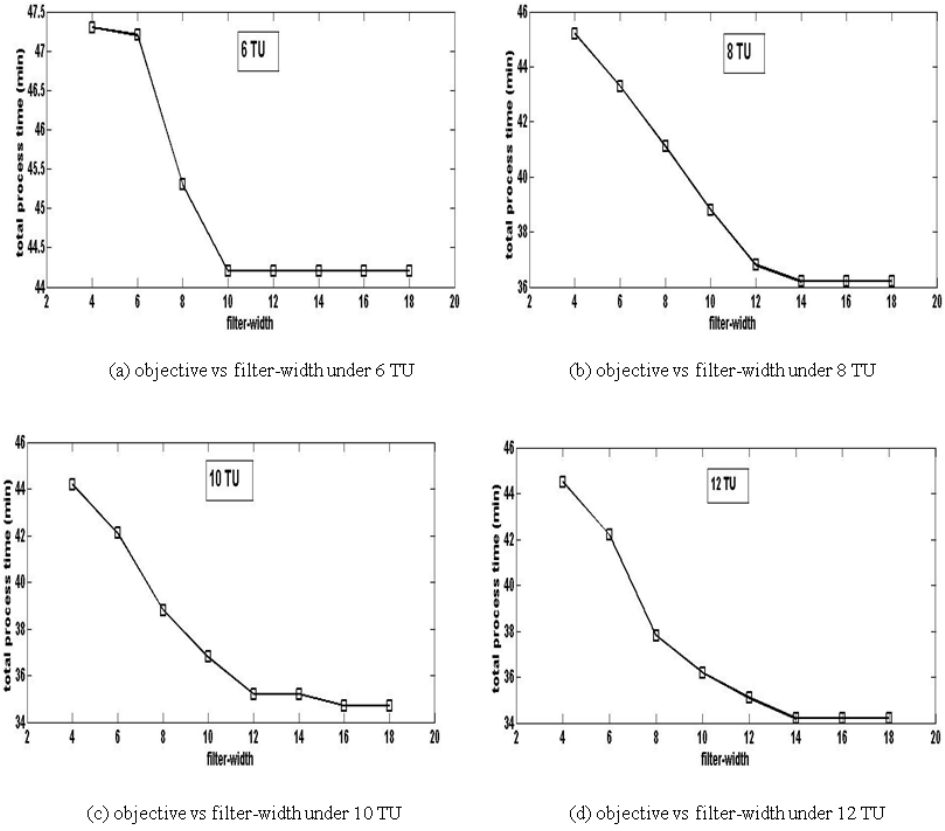
The filter-width is critical parameters that affect the performance of the algorithm. It is intuitive that increasing filter-width can help to obtain a more promising solution. However, the computation time will be increased. Thus, we need to handle the trade-off between the accuracy and computing time. In our algorithm, we use the HSPG method to screen the intermediate nodes, and then the fitness of the retained nodes is calculated by the CGA-FS method. If the correlation coefficient of the fitness calculated by HSPG and CGA-FS is small, the filter-width must be set to a large number in order to prevent screening out the good intermediate nodes. On the contrary, the filter-width can be set to a relatively small number if the correlation coefficient is large. Table 4.3 shows the correlation coefficient of the results calculated by the HSPG method and CGA-FS method.

Table 4.3 Correlation coefficient between HSPG method and CGA-FS method

	4 TU	6 TU	8 TU	10 TU	12 TU
Correlation coefficient	0.9517	0.9423	0.9011	0.8923	0.8733

The term “4 TU” in Table 4.3 means the intermediate nodes are for the case that there are four TUs in the system. In each case, there are 100 intermediate nodes that are randomly generated. The correlation coefficient is calculated by the results of these 100 intermediate nodes.

From the results, we can find that the correlation coefficient is large which means we can set a relatively small number for the filter-width to reduce the computing time while the good solutions may not be discarded. Figure 4.8 shows the results with different filter-width.



(a) objective vs filter-width under 6 TU

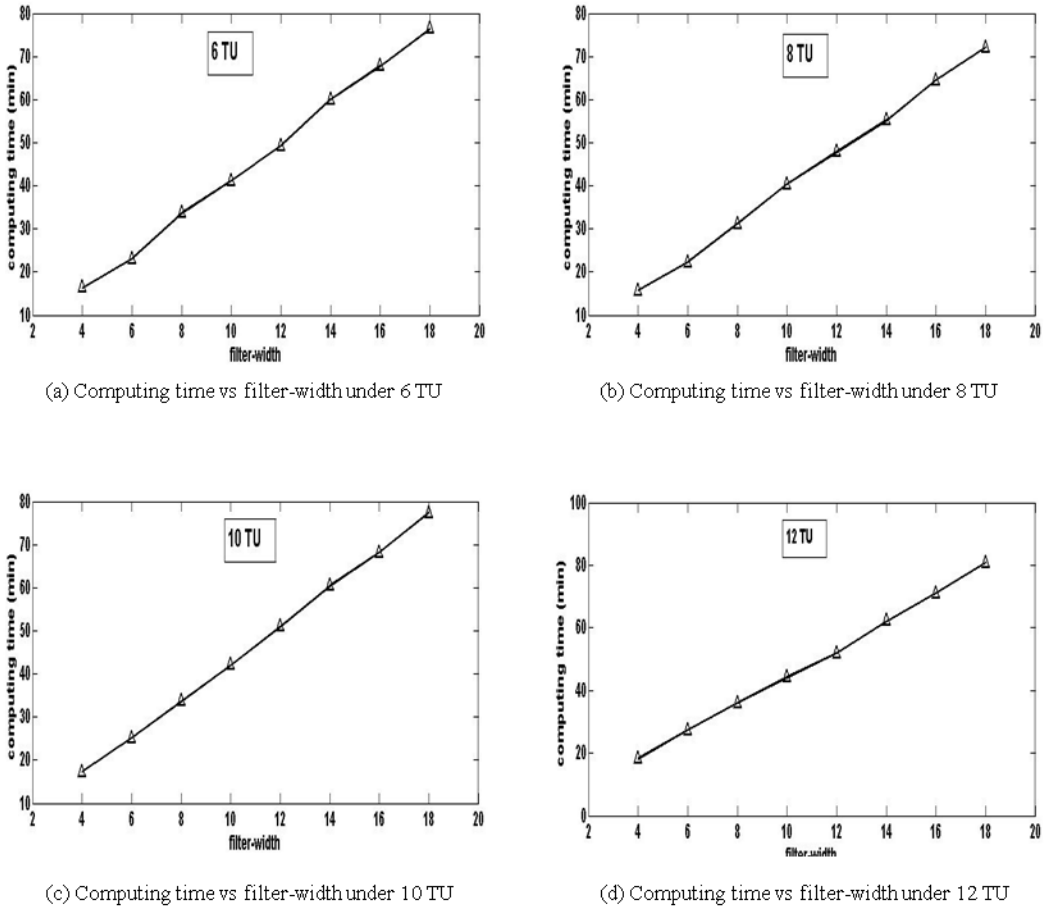
(b) objective vs filter-width under 8 TU

(c) objective vs filter-width under 10 TU

(d) objective vs filter-width under 12 TU

Figure 4.8 Objective vs filter-width

From Figure 4.8, we can find that when the filter-width is large enough, the improvement of the objective is minor. However, the raise of filter-width will increases the computing time. Figure 4.9 shows the relationship between filter-width and computing time.



(a) Computing time vs filter-width under 6 TU

(b) Computing time vs filter-width under 8 TU

(c) Computing time vs filter-width under 10 TU

(d) Computing time vs filter-width under 12 TU

Figure 4.9 Computing time vs filter-width

From the results shown in Figure 4.8 and Figure 4.9, we can determine the filter-width with consideration of computing time. Therefore, the experiments in this study will select the number around 14 as the filter-width.

4.5.2 Comparison study of the performance of the proposed method

The algorithm with heuristic rules is widely used in a variety of industrial applications. We will conduct a comparison study between the algorithm described in Section 4.4 and our proposed method. The results are shown in Table 4.4. All the cases involve 100 container jobs. The run time of the heuristic algorithm is set to be equal to the time that our algorithm takes.

Table 4.4 Comparison of CGA-FS and heuristic rules algorithm

Case	CGA-FS Makespan (min)	Heuristic algorithm Makespan (min)	Gap (%)
P1, 2 QC 3 TU	130.16	146.12	10.92
P2, 2 QC 3 TU	124.02	138.08	10.18
P3, 2 QC 4 TU	96.00	110.22	12.90
P4, 2 QC 4 TU	98.44	110.57	12.13
P5, 3 QC 3 TU	124.09	134.23	8.17
P6, 3 QC 3 TU	116.78	125.55	6.99
P7, 3 QC 4 TU	92.21	104.05	11.38
P8, 3 QC 4 TU	91.99	102.63	10.37
P9, 3 QC 5 TU	78.24	88.43	11.52
P10, 3 QC 5 TU	77.67	89.87	13.58
P11, 4 QC 5 TU	64.40	72.13	10.72
P12, 4 QC 5 TU	66.22	75.26	12.01
P13, 4 QC 7 TU	42.50	52.12	18.46
P14, 4 QC 7 TU	41.12	50.26	18.19
P13, 4 QC 10 TU	31.48	38.22	17.63
P14, 4 QC 10 TU	32.01	38.20	16.05

From the results, we can find that our method outperforms the heuristic algorithm, especially in the cases that the TU number is large. From the results, we can also find that the number of TUs and QCs affects the performance of the system. Thus, in the second part, we will study the effects of the parameters on the efficiency of the system.

4.5.3 Effects of the parameters on the efficiency of GRID-ACT

In this section, we will discuss the effects of the parameters like the number of TUs, QCs and horizontal rails.

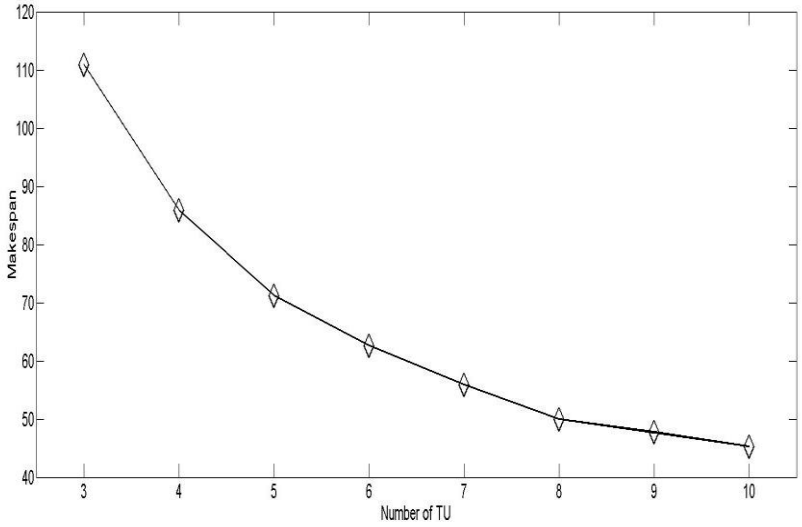


Figure 4.10 Effects of number of TUs on makespan

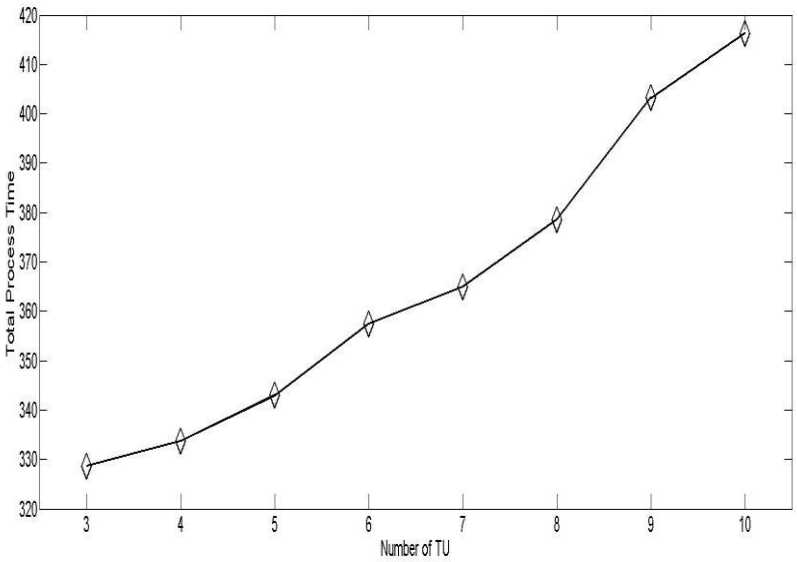


Figure 4.11 Effects of number of TUs on total processing time

Figures 4.10 and 4.11 are the results of different number of TUs. It is intuitive that the makespan is decreased when increasing the number of TUs. However, the total processing time gets larger. This is because the system becomes congested, resulting in more delay when TUs are transporting the containers. In addition, TUs have to travel a longer distance in order to prevent vehicle collisions, which also increases the total processing time.

The number of horizontal rails also affects the performance of the system. Figure 4.12 shows the results of makespan under different number of horizontal rails.

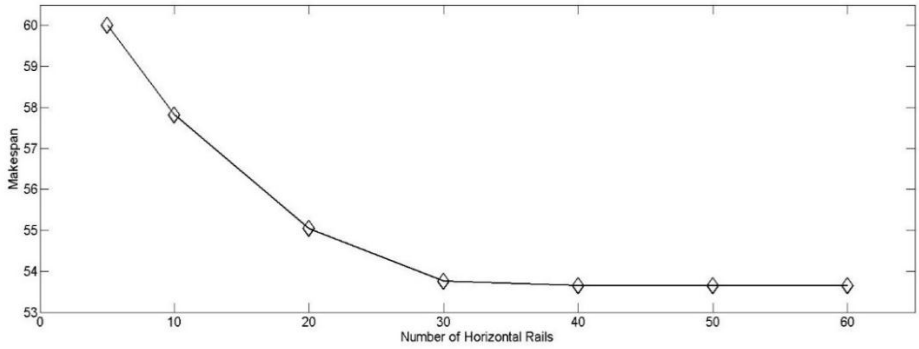


Figure 4.12 Effects of number of horizontal rails on makespan

We can find that when the number of horizontal rails is small, increasing this resource can significantly reduce the makespan. This is because it can make the routes more flexible so that TUs can make a change on the paths to prevent vehicle collisions other than staying to wait until they can move through the congested area. However, this benefit cannot be obtained when the number of horizontal rails is large. The increment of horizontal rails does not reduce the makespan. This is because it cannot reduce the delay that happens in the transfer area. Every TU will approach the transfer points in the quayside to pick up or deliver the containers. Thus, the transfer area will be congested especially at the transfer points. Therefore, increasing the number of QCs can significantly reduce the makespan, because the traffic of each transfer point is reduced when increasing the number of transfer points. Table 4.5 shows the results of makepan under different number of QCs.

When the number of TUs is small, the reduction of makespan by increment of one QC is not large. When the number of TUs is large, the reduction of

makespan by increasing of one QC is significant. Because when number of TU is large, increase the number of QC can spread out TUs in the transfer area, and the delay because of the busy of transfer points is also reduced.

Table 4.5 Effects of number of QCs on makespan

	QC2 VS QC3	**QC3 VS QC4**
TU 4	5.46%	2.69%
TU 5	5.61%	5.39%
TU 6	8.60%	12.4%
TU 7	11.93%	14.52%
TU 8	12.80%	16.21%

4.6 Conclusion

In this study, we describe the operations of the GRID-ACT and propose an algorithm to solve the problem consisting of vehicle dispatching and conflict-free routing. To access our algorithm, we conduct a comparison study with the heuristic algorithm. From the results, our algorithm can solve problems efficiently. In order to maintain a high productivity of the system, we should invest enough TUs and horizontal rails. Because the speed of the TU is slow, the system is low productivity when the number of TU is small. However, when the number of TU is large, the marginal benefit is small by additional one TU. It is because the system becomes more congested which resulting in more delay.

CHAPTER 5 A Study On A New Design ACT

From the studies on FB-ACT and GRID-ACT, we can observe the main drawbacks of these two kinds of ACTs. In the FB-ACT system, the vehicles mounted on the same rail can only wait or give way to prevent collisions, which results in a large waiting time of the vehicles. Moreover, the handshakes among the equipment also bring waiting time to the system. In the GRID-ACT, the vehicles can flexibly select the paths to prevent collisions rather than keep waiting at a place. But the speed of the vehicles is relatively slow. Thus, a lot of time would be spent on traveling on the paths to accomplish the transportation demands. On the other hand, the advantages of these two ACTs attract lots of attention. In FB-ACT, the yard crane productivity is high because the ground trolleys can move to the designate slot. This leads to the reduction of the movement of YCs whose speed is much slower than ground trolleys. In GRID-ACT, the handshakes are greatly reduced, because the vehicles can lift and release the containers by themselves. A hybrid design of ACT is proposed which combines the advantages of these two ACT systems while weeds out the drawbacks. The idea of the design is to implement GRID system to take place of transportation activities at the quayside, while the transportation activities within the storage yard are performed by the same way in FB-ACT system.

5.1 Introduction of the new design ACT

The layout of the new design ACT is shown in Figure. 5.1. We call this design HFG-ACT (an abbreviation of Hybrid of Frame Bridge and GRID systems based ACT).

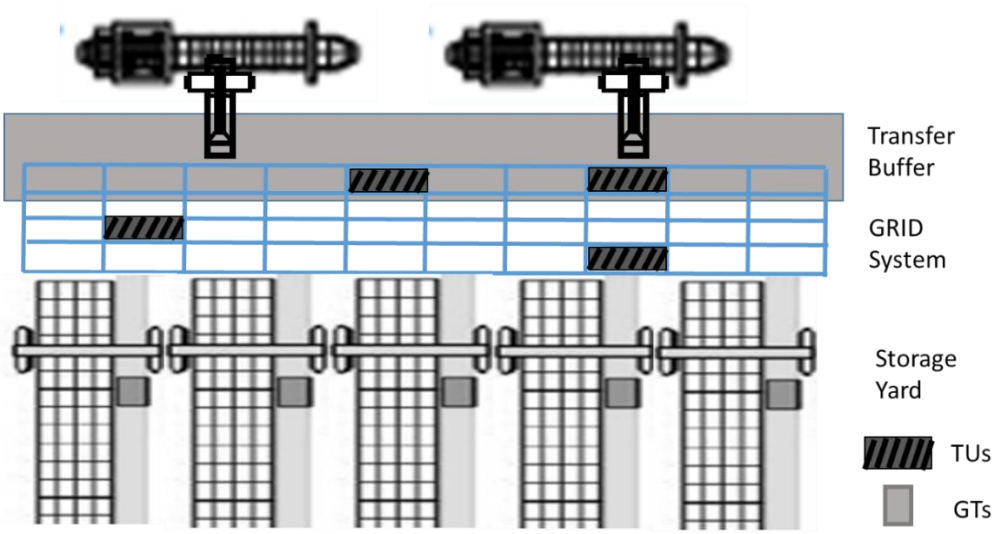


Figure 5.1 Layout of HFG-ACT

Compared to the layout of FB-ACT, we can observe that we implement the GRID system to accomplish the transportation of containers at the quayside. Because the positions of containers in GRID system and on GTs are mutually perpendicular, we need to turn the container to 90 degrees when transfer it between a TU and a GT. Therefore, the TUs in HFG-ACT are different from those in GRID-ACT. These TUs are designed to be capable to rotate the container by 90 degrees.

The GRID system plays a role on transferring the containers between quayside and storage yard. We define two kinds of transfer points: quayside transfer point and yard-side transfer point, where TUs can unload or load containers. The quayside transfer points are set on the first lane of the horizontal lanes which is the nearest to the vessels. The explicit location of the points are determined by the location of the QCs, whereas the transfer points at yard-side are explicitly appointed. They are allocated on the last lane of the horizontal lanes which is nearest to storage yard. Specifically, they are the places above

the ground rails. Each transfer point at yard-side serves the corresponding yard block.

In order to improve the efficiency of TUs in GRID system, neither the horizontal nor the vertical contiguous lanes will share one track, which means TUs on the contiguous lanes can move in diverse directions. The illustration is shown in Figure 5.2. Both situations shown in Figure 5.2 are permitted.

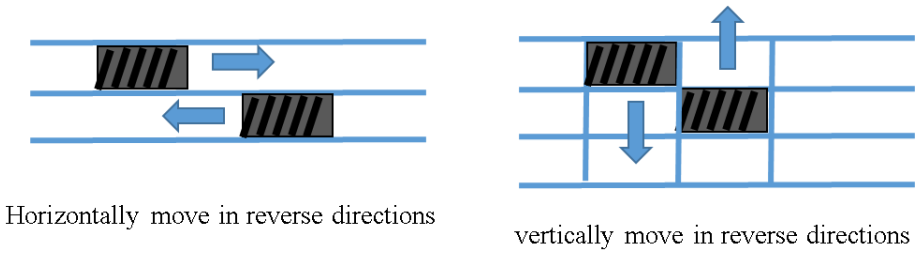


Figure 5.2 Movement in reverse directions

In order to prevent the head-on or rear-end collisions, a safety distance between the TUs on the same horizontal lanes is defined. Because the speed of the vehicle is relatively slow, the safety distance can be set short so that vehicles can move more flexibly.

In summary, the operation of a loading container can be described as following. One YC picks up the container from the slot in the storage yard and puts it on a GT. Then the GT carrying the container will move to the transfer point. Once a TU arrives at the transfer point, it will pick up the container from the GT and then rotate the container to 90 degrees before lifting the container. The TU carrying the container moves along the lanes to approach the transfer point at quayside. The TU will put down the container onto the transfer buffer. At this time, the TU is free to handle its next job and a specified QC will load the container on the transfer buffer to the vessel. The

operation of an unloading container is performed similarly but in a reverse order.

In order to model the conflict-free routing problem, the definition of cell is proposed meshing with the structure of GRID system, which is the same as that shown in Chapter 4. The shape of a TU is slightly larger than a 40ft standard container. We set the length of the TU to 13.2 meters and the width to 3.3 meters. The GRID system is constructed by horizontal tracks and vertical tracks, which form homogeneous lattice. Thus the length of a lattice is four times to the width of it. Therefore, we define the time that the TU spends on the movement with a distance of 3.3 meters as a unit time. The speed of the TU is set to be 2 m/s. Thus, a unit time is equivalent to 1.65 seconds. We create the dummy cells meshing on the GRID system. One cell is a square area. The position of the TUs can be represented by the cells. Distinctively, one TU occupies four cells at a time. We choose the first cell counted from left to right as the position of the TU. Based on the definition of cells, we can quantify the safety zone of the TUs at any time. If we set the safety distance to one unit time distance, the safety zone can be presented as the set:

$$G(i) = \{ j \mid \max(i-4, L(i)) \le j \le \min(i+4, R(i)) \},$$

where i is the position of the TU, $R(i)$ is the right-most cell of the row that Cell i belongs to and similarly $L(i)$ is the left-most cell. Once a TU stays at Cell i, the cells in set $G(i)$ are reserved so other TUs cannot occupy. Moreover, the position of the transfer points mentioned previously can also be presented by the cells.

Another useful term guide-path segment introduced in Chapter 4, is also used to help model our problem. A guide-path segment is the smallest unit that the TU travel through. It is bi-directional. Moreover, the guide-path segments can be categorized into two types: vertical guide-path segments and horizontal guide-path segments. Intuitively, vertical guide-path segments are those TUs move in vertical direction while horizontal guide-path segments are those TUs move horizontally. The guide-path segment can be represented by the cells at its two sides. The set of horizontal guide-path segments is denoted as W^H, where $W^H = \{(i, i+1) \mid L(i) \le i \le R(i) - 4, \forall i \in N\}$. The set of vertical guide-path segments is denoted as W^V. Unlike the horizontal movement, the vertical movement can only occurs at certain cells where vertical tracks exist for TUs to travel along. Thus, we have:

$$W^V = \{(i, i+R) \mid i = (k-1)*R + (j-1)*4 + 1, L(i) \le i \le R(i) - 3,$$
$$\forall k \in Row \setminus N^R, j \in Integer\},$$

where R is the number of cells on one lane, Row is the set of lanes, and N^R is the index of last lane. The last lane is the rail nearest to the storage yard while the first lane is the rail nearest to the quayside.

The definition of guide-path segments is used to help build constraints to prevent vehicle collisions in the mathematical model, and construct the mathematical network for searching solutions. They will be discussed later.

5.2 Model development

Similar to the previous studies, we will address the vehicle dispatching and conflict-free routing problems in HFG-ACT, while the quay crane and yard

crane capacities are considered. The modeling technique is similar to that in study on the GRID-ACT.

Modeling assumptions

The following assumptions are made:

- Job sequence and job types for each QC are given; Subtasks handled by QCs must be carried out in the exact order in the QC list;
- Yard block of each job is known;
- The traveling speeds of empty and loaded TUs are the same;
- Only one GT is running on each yard rail, i.e., there is no conflict among GTs;
- Number of container jobs, number of TUs, GTs, QCs and YCs are all known;

Because there is only one GT on the ground rail, we can simplify the operations in storage yard as a constant time. This constant time consists of GT traveling time and YC handling time. To make it simple, we assume that the travel distance of GTs is set between the transfer point and the middle of the yard block.

Notations

The model parameters are as follows:

Q the set of QCs;

K the set of TUs;

Y the set of yard blocks;

N_k the number of jobs in the QC list of QC k;

L the set of loading jobs;

D the set of discharging jobs;

H the set of all the jobs, $H = L \cup D$;

QL the set of subtasks that handled at quayside transfer points;

QL^q the subtasks whose corresponding container belongs to the sequence list of QC q, $q \in Q$;

YL the set of subtasks that handled at yard-side transfer points;

YL^y the subtasks whose corresponding container belongs to the yard block with index of y, $y \in Y$;

W^V the set of vertical path segments (w^+ and w^-, indicate the two directions along the segment $w \in W^V$).

(i, α) container job index. The job (i, α) refers to the i th job in the sequence list of QC α.

$L(n)$ Set of all feasible routes for TU n

$\lambda(n)$ Set of all feasible routes generated in the pricing problem for the vehicle n

Binary parameters:

$$A_{(i,t)lm} \begin{cases} 1, \text{if TU } m, \text{following route } l \text{ is at location } i \text{ at time } t \\ 0, \text{otherwise} \end{cases}$$

$$B_{l,m}^{i} \begin{cases} 1, \text{if TU } m, \text{following route } l \text{ covers job } i \\ 0, \text{otherwise} \end{cases}$$

$$f_{w^+,lm}^{t} \begin{cases} 1, \text{if TU } m, \text{following route } l \text{ enters segment } w \text{ in a direction } w^+ \\ \quad \text{at time } t \\ 0, \text{otherwise} \end{cases}$$

$$f_{w^-,lm}^{t} \begin{cases} 1, \text{if TU } m, \text{following route } l \text{ enters segment } w \text{ in a direction } w^- \\ \quad \text{at time } t \\ 0, \text{otherwise} \end{cases}$$

$$g_{\tau,lm}^{t} \begin{cases} 1, \text{if TU } m, \text{following route } l \text{ starts subtask } \tau \text{ at time } t \\ 0, \text{otherwise} \end{cases}$$

The decision variables are as follows:

X_{lm} =1: TU m select the route l; otherwise 0.

To solve the problem, three decisions need to be determined, as shown below:

(1) The assignment of container jobs to TUs, i.e. each container job is assigned to which TU.

(2) The location of the TUs at every time unit to ensure that there are no vehicle collisions;

(3) The starting time of the subtasks because of the constraints of QC and YC capacities.

We model the problem as a set partitioning problem. A route is defined as a set of the decisions mentioned above for a TU. It should be highlighted that the decisions of a container job must be complete if a route contains this job. For example, if route l contains the assignment decision that a container job (i,α) is assigned to TU m, the decisions including how the TU travels along the lanes and when the subtasks $(i,\alpha,1)$ and $(i,\alpha,2)$ start, must also be contained in route l.

Model Formulation

Mathematical model $P0$

Objectives:

$$Minimize: \sum_{m\in K}\sum_{l\in L(m)} c_{lm} X_{lm} \tag{5.1}$$

Constraint

$$\sum_{l\in L(m)} X_{lm} = 1, \quad \forall m \in K \tag{5.2}$$

$$\sum_{l\in L(m)}\sum_{m\in K} B_{lm}^{\tau} X_{lm} = 1, \quad \forall \tau \in H \tag{5.3}$$

$$\sum_{l\in L(m)}\sum_{m\in K} A_{(i,t)lm} X_{lm} + 0.5 * \sum_{j\in G(i)}\sum_{l\in L(m)}\sum_{m\in K} A_{(i,t)lm} X_{lm} \le 1, \quad \forall i \in N, t \in T \tag{5.4}$$

$$\sum_{l\in L(m)}\sum_{m\in K} f_{w^-,lm}^{t} X_{lm} + \sum_{l\in L(m)}\sum_{m\in K} f_{w^+,lm}^{t} X_{lm} \le 1, \quad \forall w \in W^V, t \in T \tag{5.5}$$

$$\sum_{l\in L(m)}\sum_{m\in K} t_{lm}^{\tau} X_{lm} + t_{(\tau,\tau')}^{qc} \le \sum_{l\in L(m)}\sum_{m\in K} t_{lm}^{\tau'} X_{lm}, \quad \forall q \in Q, (\tau,\tau') \in QL^q \tag{5.6}$$

$$\sum_{l\in L(m)}\sum_{m\in K} g^{t}_{lm} X_{lm} + \sum_{l\in L(m)}\sum_{m\in K}\sum_{u=t-t^{yc}_{(\tau',\tau)}}^{t+t^{yc}_{(\tau,\tau')}} g^{u}_{\tau' lm} X_{lm} \le 1, \quad \forall y \in Y, (\tau,\tau') \in YL^{y} \tag{5.7}$$

$$X_{lm} \in \{0,1\}, \quad \forall l \in L(m), m \in K \tag{5.8}$$

Formulation (5.1)-(5.8) is explained as follows. As c_{lm} represents the penalty generated by route l, the objective function (5.1) minimizes the sum of penalties for the full horizon. The constraints (5.2) enforce the selection of exactly one route for each vehicle. Constraints (5.3) ensure that each container job is assigned to exactly one vehicle route.

The conflict-free routing constraints are modeled with (5.4)-(5.5). Constraints (5.4) ensure that there is at most one vehicle visiting a cell in N at time $t \in T$. Constraints (5.5) are used to prevent the cross-over among the vehicles in the same lane. The details are illustrated below.

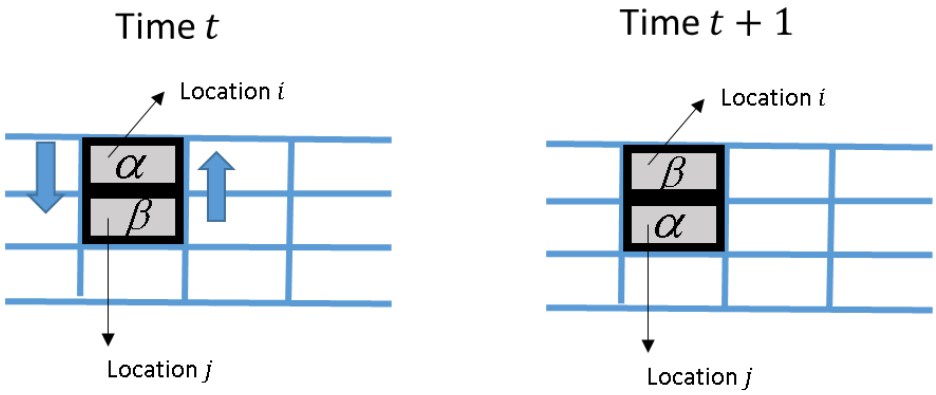


Figure 5.3 Cross-over collision

Considering the scenario shown in Figure. 5.3, TU α and TU β currently at time t stay on cells i and j, respectively. They are going to move in the reverse direction. Thus, at time $t+1$, TU α will stay at cell j while TU β will stay at cell i. This kind of scenario cannot be detected by constraints (5.4).

However, it is infeasible because the TUs on the same lane cannot cross-over each other. Therefore, constraints (5.5) are proposed to prevent this kind of vehicle collision. In addition, this scenario will not happen if these two TUs are on a horizontal lane. This is because of the restriction of safety zone, constraints (5.4) can prevent two TUs from staying contiguously. In conclusion, only the vertical guide-path segments have possibilities of head-on collisions, which cannot be prevented by constraints (5.4).

Constraints (5.6)-(5.7) are proposed considering the capacities of QCs and YCs. Constraints (5.6) impose a time interval between two consecutive subtasks using the same QC. The values of the time intervals depend on the type of subtasks. They are the same as those defined in the study on FB-ACT. Similarly, constraints (5.7) impose a time interval between the pair of consecutive subtasks using the same yard block. We assume that there is no pre-defined sequence for YCs. This means that we only need to ensure that the YC serves one subtask at a time, but the sequence of subtasks being carried out is unknown. Similarly, the values of the time intervals are the same as those defined in the study on FB-ACT. Finally, constraints (5.8) specify the binary character of the variables.

In practice, the number of feasible routes is substantially large and the model (5.1)-(5.8) cannot be solved directly. In order to solve the problem efficiently, we will apply the tree structure algorithm where two efficient approaches are embedded. Before discussing the algorithm in detail, the mathematical network will be presented first. It is implemented to solve the routing problem when we decompose our original problem into vehicle dispatching problem and vehicle conflict-free routing problem.

5.3 Mathematical network

The conflict-free routes will be generated based on a time-space network. We will introduce this network first before we present our algorithms. The mathematical network is a time-space acyclic network $G = (V, A)$, where V and A represent the set of vertices and the set of directed arcs, respectively (shown in Figure 5.4).

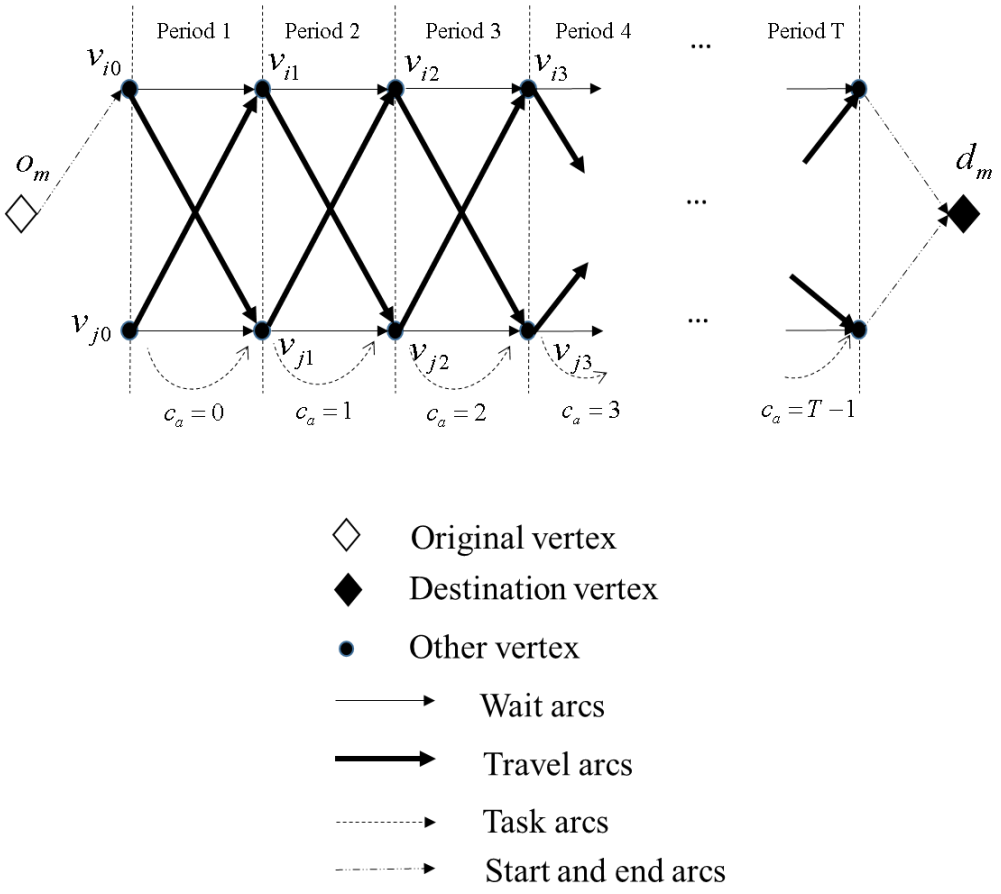


Figure 5.4 Part of a mathematical network

There are two types of vertices:

- To each location $n \in N \cup S$ corresponds $T+1$ vertices $v_{nl} \in V$ (one vertex for each time t).

- An origin vertex $o_m \in V$ and a destination vertex $d_m \in V$ are associated with each vehicle $m \in K$ to represent the start and the end of its route, respectively. The set of origin and destination vertices is denoted as OD.

The arc set A includes five types of arcs, corresponding to moves in space and/or time:

- The start arcs link the origin vertex o_m of each vehicle to the vertex $v_{nl} \in V$ corresponding to its initial position, i.e. a location or a transfer point on the guide-path.

- The end arcs link all vertices v_{nT} (T is the last period of the horizon) to each destination vertex $d_m, m \in K$.

- The wait arcs are defined for all pairs of vertices $(v_{nt}, v_{n,t+1}) \in V \setminus OD \times V \setminus OD$. These arcs represent a vehicle waiting at a given location for one period of time.

- The travel arcs represent the movements along the guide-path. These arcs are represented as $(v_{it}, v_{j,t+1})$ where i and j are adjacent locations.

- Let $\tau \in QL \cup YL$ be a subtask to be operated at the quayside transfer point or yard side transfer point. A task arc $(v_{nt}, v_{n,t+h}) \in V \setminus OD \times V \setminus OD$ is defined if this subtask can begin at time t, where n is the location of the transfer point.

The sets of start and task arcs are denoted by A^S and A^T, respectively, while the set of travel arcs A^W is divided into subsets A^{W-} and A^{W+} according to

the direction (+ or -) used to travel along the corresponding guide-path segment.

To calculate the cost of a route $l \in \lambda(m)$, a cost c_a is associated with each arc $a \in A$. This cost is null for all arcs other than the task arcs. For a task arc a representing task τ starting at time t, this cost corresponds to the penalty incurred for beginning τ at time t.

5.4 Solution methodology

The complexity of vehicle dispatching and conflict-free routing leads to intrinsic difficulty of this problem. The tree structure algorithm is adopted to solve the problem, where two methods are embedded during the procedure of expanding the tree. The structure and mechanism of expanding the tree are the same as that in the study shown in Chapters 3 and 4. But the embedded methods are different which are developed according to the characters of the problem in this study. The problem is still divided into two sub-problems: vehicle dispatching problem and vehicle conflict-free routing problem. As the conflict-free routing problem is still complicated even though the decisions of vehicle dispatching problem are given, we propose two heuristic based algorithms to solve this routing problem. In Chapter 2, we can find that because each FT moves on a unique rail, the routing problem can be solved by determining the subtask sequence. The subtask sequence is a prior ordering of the subtasks when they are competing for the resources. The TUs in this problem can move more flexibly so that the routing problem cannot be solved by using the same method as that in the Chapter 3. However, the subtask sequence seems to be a crucial factor for the performance of the solutions.

Thus, we will solve the routing problem by determining the subtask sequence. The first method is easy and fast and it is called heuristic sequential path generation (HSPG), which is used to screen the intermediate nodes. The second method is more accurate but needs more computation efforts and it is called column generation algorithm based sequential path generation (CGASPG), which is used to generate new beam nodes. The procedure of expanding the tree is the same as that in the previous study. We will only discuss these two embedded methods in this study.

5.4.1Heuristic sequential path generation

A greedy algorithm is used to play the role of the surrogate method, which is used to screen the intermediate nodes. This algorithm consists of generating the conflict-free route for each subtask sequentially. The way to give a priori ordering of the subtasks is crucial to the performance of the algorithm. In addition, this algorithm is applied to solve the conflict-free routing problem in a large number of intermediate nodes, which implies that the computation time must be short. Considering the impact from the pre-specified QC list, we should maintain consistency of the prior ordering of the subtasks and the QC list. Intuitively, if a subtask is ranked high in the QC list but ranked a low position in the prior ordering, this will incur an additional delay since other subtasks succeeding to it have to be delayed when the vehicles already arrive at the transfer point. In order to obtain a prior ordering which tends to prevent this additional delay, a mathematical model is built, which is denoted as $P^S(n)$, where n indicates the intermediate node. The objective of the model is to determine a prior ordering so that the delay is minimized based on the assumption that the vehicle collisions are not considered. Thus, the traveling

time between the transfer points is represented by the time using the shortest path. Because the number of subtasks involved is relatively small and the model is simple, the computation efforts of the model will be small. After the ordering is solved, the conflict-free route of each subtask is solved by sequential path generation algorithm based on the ordering.

We first define the additional parameters in this model while the parameters that are the same as previous will not be presented here again.

Parameters

p_α processing time of subtask α.

$t_{\alpha\beta}^{TU}$ The traveling time for a TU from the location of subtask α to the location of subtask β using the shortest path ignoring vehicle collisions

$V(m)$ the set of subtasks that belongs to vehicle $m, m \in K$

$C(\alpha)$ the set of subtasks that compete the same yard transfer point, where $\alpha \in YL$.

$Q(\alpha)$ the set of subtasks that cannot be performed until subtask α is accomplished according to the QC list.

SP the set of pairs of successive subtasks belonging to the same vehicle

Decision variables

T^m the time for vehicle m to complete its subtasks

T_α the starting time of subtask α

$z_{\alpha\beta}$ $\begin{cases} 1, \text{if subtask} \alpha \text{ starts earlier than subtask} \beta \\ 0, \text{otherwise} \end{cases}$

Mathematical model $P^S(n)$

Objective:

Minimize: $\sum_{m \in K} T^m - \sum_{\alpha \in S} p_\alpha - \sum_{(\alpha,\beta) \in SP} t_{\alpha\beta}^{TU}$ (5.9)

Constraint

$$T^m \geq T_\alpha + p_\alpha, \quad \forall m \in K, \alpha \in V(m) \tag{5.10}$$

$$T_\alpha + p_\alpha + t_m^{intval} - T_\beta \leq M(1 - z_{\alpha\beta}), \quad \forall \alpha, \beta \in C(\alpha) \cup Q(\alpha)$$

(5.11)

$$T_\alpha + p_\alpha + t_{\alpha\beta}^{TU} - T_\beta \leq 0, \quad \forall (\alpha, \beta) \in SP$$

(5.12)

$$z_{\alpha\beta} + z_{\beta\alpha} = 1, \quad \forall \alpha \in YC, \beta \in C(\alpha)$$

(5.13)

$$z_{\alpha\beta} = 1, \quad \forall (\alpha, \beta) \in SP$$

(5.14)

$$z_{\alpha\beta} = 1, \quad \forall \alpha \in QL^q, q \in Q, \beta \in Q(\alpha)$$

(5.15)

$$z_{\alpha\beta} \in \{0,1\}, T_\alpha \in T \tag{5.16}$$

The objective function (5.9) minimizes the sum of delay during the horizon of accomplishing all the subtasks in the intermediate node. Constraints (5.10) define the complete time for vehicles to accomplish their subtasks. Constraints (5.11) impose a time interval of at least $p_\alpha + t_m^{intval}$ periods between two subtasks that compete for the same resources at the transfer points. The definition of t_m^{intval} is presented in the study of FB-ACT, which is the necessary handling time for the resources like QC, YC and GT between two consecutive subtasks. Constraints (5.12) impose a time interval of at least the processing time and traveling time between two subtasks using the same vehicle. Constraints (5.13) ensure that two subtasks that competing for the same resources in yard block cannot be performed simultaneously. Constraints (5.14) and (5.15) ensure that the subtasks are performed consistently with the sequence in each vehicle and QC list. Finally, constraints (5.16) specify the domain of the decision variables.

Once the mathematical model P^S is solved, the priori ordering of the subtasks can be determined according to the values of variables $T_\alpha (\alpha \in S)$. The subtask with smaller value of T_α is assigned with a higher ranking in the prior ordering.

Given the network G and the subtask set S, the main steps of the algorithm are shown as following:

(1) Position all vehicles at their source nodes in G.

(2) Route the vehicle with the subtask τ which is the first rank on the priori ordering. Remove from G all arcs and vertices in this path to prevent

collisions when routing other vehicles. Remove also all arcs corresponding to task τ. Update the priori ordering by removing the task τ.

(3) If there are still some unscheduled subtasks, return to step 2.

(4) Otherwise, Stop.

After these steps, we can obtain a feasible solution. The total processing time of subtasks computed based on the feasible solution is treated as the fitness of the corresponding intermediate node. As discussed before, a screening procedure is performed when the fitness of the intermediate nodes is calculated. Another method is applied to the retained intermediate nodes to resolve the conflict-free routing problem. This method will be discussed in the next section.

5.4.2Column generation algorithm based sequential path generation

It is intractable to solve the conflict-free routing problem to optimality when the number of vehicle gets large. We implement the column generation technique to help solving the routing problem. We first present how we use column generation algorithm for our routing problem. Then, we will introduce how we use the results from the column generation algorithm to help solving the routing problem.

Column Generation Algorithm

The mathematical model for solving the conflict-free routing problem associated with the intermediate node n, is denoted as $P^C(n)$. It is presented as following.

Mathematical model $P^C(n)$

Objectives:

$$Minimize: \sum_{m\in K}\sum_{l\in L(m)} c_{lm} X_{lm} \tag{5.17}$$

Constraint

$$\sum_{l\in L(m)} X_{lm} = 1, \quad \forall m \in K \tag{5.18}$$

$$\sum_{l\in L(m)}\sum_{m\in K} A_{(i,t)lm} X_{lm} + 0.5 * \sum_{j\in G(i)}\sum_{l\in L(m)}\sum_{m\in K} A_{(i,t)lm} X_{lm} \le 1, \quad \forall i \in N, t \in T \tag{5.19}$$

$$\sum_{l\in L(m)}\sum_{m\in K} f^{t}_{w^-,lm} X_{lm} + \sum_{l\in \lambda Lm)}\sum_{m\in K} f^{t}_{w^+,lm} X_{lm} \le 1, \quad \forall w \in W^V, t \in T \tag{5.20}$$

$$\sum_{l\in L(m)}\sum_{m\in K} t^{\tau}_{lm} X_{lm} + t^{qc}_{(\tau,\tau')} \le \sum_{l\in L(m)}\sum_{m\in K} t^{\tau'}_{lm} X_{lm}, \quad \forall (\tau,\tau') \in QL(n) \tag{5.21}$$

$$\sum_{l\in L(m)}\sum_{m\in K} g^{t}_{\tau lm} X_{lm} + \sum_{l\in L(m)}\sum_{m\in K}\sum_{u=t-t^{yc}_{(\tau',\tau)}}^{t+t^{yc}_{(\tau,\tau')}} g^{u}_{\tau' lm} X_{lm} \le 1, \quad \forall (\tau,\tau') \in YL(n) \tag{5.22}$$

$$X_{lm} \in \{0,1\}, \quad \forall l \in L(m), m \in K \tag{5.23}$$

The model $P^C(n)$ aims to solve the conflict-free routing problem in the intermediate node indicated by n. A column generation algorithm is used to find its optimum solution. Such an algorithm divides the problem into a restricted master problem and several sub-problems. In our study, the restricted master problem is simply the linear relaxation restricted to a small subset of its variables and the sub-problems corresponding to shortest path problems with additional constraints. There is one sub-problem for each

vehicle. The role of the restricted master problem is to coordinate the various proposals made by the sub-problems to satisfy all linking constraints (5.18)-(5.22). The role of the sub-problems consists of identifying variables that may contribute to improve the objective function.

The column generation procedure is iterative. It starts with a small subset of the path variables. The restricted master problem is solved optimally at each iteration to provide current primal and dual solutions. Each sub-problem is solved by dynamic programming with the objective of finding path variables with negative reduced costs, under the dual information obtained from the restricted master problem. After that, the new paths variables are added to the restricted master problem, which will be solved again. The primal solution to the restricted master problem is optimal to the linear relaxation of $P^C(n)$, when there are no path variables that with negative reduced cost. The iterative procedure is then stopped.

The sub-problems

The key factor in solving the sub-problems is to derive the arc reduced costs $\bar{c}_a$ from the dual solution of the current restricted master problem. Such a reduced cost is given by $\bar{c}_a = c_a - \delta(a)$, where $\delta(a)$ is the sum of all the dual contributions for this arc a. These contributions are calculated according to which type that arc belongs to. The contributions are shown in Table 5.1, where $\delta_k^{(1)}\ (k \in V)$, $\delta_{n,t}^{(2)}\ (n \in N \cup S, t \in T)$, $\delta_{w,t}^{(3)}\ (w \in W^V, t \in T)$, $\delta_{(\tau,\tau')}^{(4)}\ ((\tau,\tau') \in QL)$, $\delta_{(\tau,\tau'),t}^{(5)}\ ((\tau,\tau') \in YL, t \in T)$ denote the dual variables associated with constraints (5.18)-(5.22), respectively. Furthermore, τ_a

represents the subtask associated with arc $a \in A^T$, t_a^s and t_a^e are the start and end times associated with arc $a \in A$, i_a is the tail vertex of arc $a \in A$, and w_a is the guide-path segment associated with arc $a \in A^{W^V}$.

Table 5.1 Dual contributions to arc reduced costs

If arc $a \in A$ satisfies	Add this contribution to its reduced cost
$a \in A^S, i_a = o_m$	$\delta_m^{(1)}$
$a \notin A^T$	$\delta_{j_a,t_a^i}^{(2)} + 0.5 * \sum_{j \in G(i_a)} \delta_{j,t_a^i}^{(2)}$
$a \in A^T$	$\sum_{t=t_a^s}^{t_a^e-1} (\delta_{j_a,t}^{(2)} + 0.5 * \sum_{j \in G(i_a)} \delta_{j,t}^{(2)})$
$a \in A^{W^V}$	$\delta_{w_a,t_a^i}^{(3)} + \delta_{j_a,t_a^i}^{(2)} + 0.5 * \sum_{j \in G(i_a)} \delta_{j,t_a^i}^{(2)}$
$a \in A^T, \tau_a \in YC$	$\sum_{(\tau_a,\tau') \in YL} \delta_{(\tau_a,\tau')t_a^i}^{(5)} + \sum_{(\tau',\tau_a) \in YL} \sum_{u=t-t_m^{intval}}^{t+t_m^{intval}} \delta_{(\tau',\tau_a)u}^{(5)}$
$a \in A^T, (\tau_a, \tau') \in QL$	$t_a^i \delta_{(\tau_a,\tau')}^{(4)}$
$a \in A^T, (\tau', \tau_a) \in QL$	$-t_a^i \delta_{(\tau',\tau_a)}^{(4)}$

A dynamic programming algorithm is used to solve the shortest path problem. This dynamic programming algorithm is a simple pushing algorithm that is performed after ordering the vertices of G in topological order.

As discussed in Chapter 4, we need to spend great efforts to prevent vehicle collisions in the GRID system. It is always the case that the solution from the

model $P^C(n)$ is the same as the initial feasible solution when no more new paths variables are added. The initial feasible solutions are computed by the HSPG method. One of the ways to obtain the optimal solution is the branch-and-cut technique. However, those problems mainly focus on the vehicle dispatching while the prevention of vehicle collisions tends to be easier to handle. Unfortunately, in our problem, vehicle collision is a hazard, which needs an explicit scheduling to prevent it. In addition, the detail method is not used for only once. It is applied on every retained intermediate node at every level of the spreading tree. The computation time of the detail method should be controlled so that the whole problem can be solved within a reasonable timescale. On the other hand, the detail method should be accurate enough to afford a satisfying performance of the whole algorithm. Different from the problem in Chapter 4, the transfer points at the quayside and yard-side are much more congested. The method CGA-FS developed in Chapter 4 fails to improve the solutions. We can find that the task sequence is crucial for solving the routing problem. Thus, we will provide a method based on the column generation algorithm to obtain a promising task sequence. The main idea is to estimate the starting time of the tasks using the results obtained by the column generation algorithm. Then the task sequence is determined by the estimated starting time. We call this method as column generation algorithm based sequential paths generation (CGASPG).

For each TU, we can estimate the starting time of its subtasks by summating the starting time of the subtask multiplied by the flow of its corresponding route. For instance, route l_i, l_j (where $l_i, l_j \in \lambda(n)$) are the routes of vehicle n

with the positive flow $X_{l_i,n}$ and $X_{l_j,n}$, respectively. The flow of the other routes of vehicle n is zero. One subtask is carried by vehicle n, denoted as τ. Then, the estimated starting time can be calculated by: $\bar{t}^{\tau} = t^{\tau}_{l_i,n} * X_{l_i,n} + t^{\tau}_{l_j,n} * X_{l_j,n}$. Thus, similarly, we can obtain the estimated starting time of all the subtasks in a similar manner. The subtask with an earlier estimated starting time will be ranked at a high position on the prior ordering. However, to maintain a certain level of flexibility, a threshold is introduced as H^{SPG}. Therefore, a branch will be created when there exists subtasks that $|\bar{t}^{\alpha} - \bar{t}^{\beta}| \leq H^{SPG}$, where one branch is with the decision that subtask α is prior to β when searching the conflict-free routing, and the other one branch is with the decision that subtask β is prior to α.The sequential paths generation algorithm is implemented to search the conflict-free routes according to the prior ordering.

This paragraph will discuss the insight of the CGASPG method. The crucial factor for the routing problem is to determine which vehicle can use the locations where more than one vehicle is competing for them. The branch-and-bound algorithm appears to be a way to optimally solve the routing problem. However, the scale of the variables is always substantially large considering the number of locations and the length of time dimension. Apart from that, a tight lower bound is also intractable to be found. Instead of making decisions of vehicle usage on every location at the time, we make a compromise that a prior subtask will have the right to occupy the locations when we find the route for it. Thus, the problem is reduced to make a prior ordering of the subtasks. The worst case of the number of the orderings is the

value equaling to the factorial of n, where n is number of subtasks. In fact, the number is much smaller than $factorial(n)$ because of the precedence relationship pre-specified by the QC lists and the sequence on the same vehicle. However, the number of the feasible ordering is still substantially large, resulting in unpredictable computation efforts to enumerate all these feasible orderings. Therefore, we will follow the information obtained from column generation algorithm to capture the promising orderings.

5.5 Computational experiments

In this section, we will conduct the numerical experiments to access the performance of this hybrid system. We will compare the performance of FB-ACT and HFG-ACT. Although the algorithm for each ACT system is not the same which may affects the fairness of the comparison, it can still provide some useful information that the hybrid system seems to have a performance.

The experiments are conducted based on the layout that there are 15 yard blocks and 4 QCs allocated along the quayside. For the FB-ACT system, there are 3 horizontal rails. For the HFG-ACT system, there are 4 horizontal rails and one of the rails is for transferring containers between TUs and GTs. Thus, both system have 3 rails for the vehicles to travel. There are 100 container jobs in each instance. The results are shown in Table 5.2. The results are the makespan.

Table 5.2 Comparison results of FB-ACT and HFG-ACT

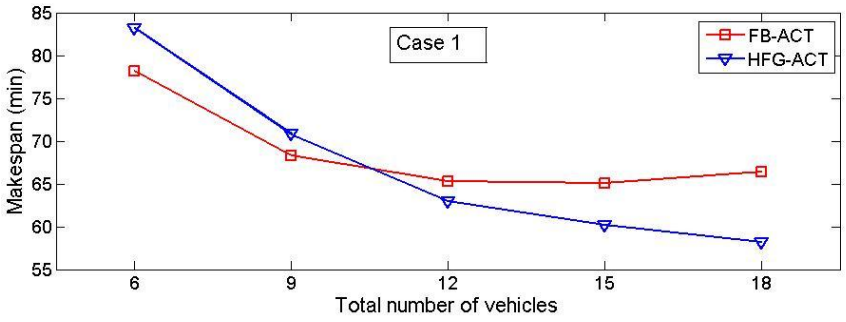
Case	FB-ACT (min)	HFG-ACT (min)	Difference (%)
2 FTs			
P1.1	78.2	83.2	-6.39
P2.1	76.4	84.1	-10.1
P3.1	81.4	85.8	-5.41
P4.1	80.9	83.4	-3.09
P5.1	82.3	84.9	-3.16
Average:			-5.62
3 FTs			
P1.2	68.3	70.8	-3.66
P2.2	70.6	71.5	-1.27
P3.2	71.1	72.6	-2.11
P4.2	72.2	73.7	-2.08
P5.2	73.2	75.4	-3.01
Average:			-2.43
4 FTs			
P1.3	65.3	63.0	3.52
P2.3	68.6	63.5	7.43
P3.3	68.4	64.4	5.85
P4.3	67.0	63.1	5.82
P5.3	65.8	61.6	6.38
Average:			5.80
5 FTs			
P1.4	65.1	60.2	8.91
P2.4	65.2	61.8	6.60
P3.4	67.6	61.1	11.0
P4.4	65.2	60.4	8.74
P5.4	64.5	60.5	7.60
Average:			8.57
6 FTs			
P1.5	66.4	58.2	12.35
P2.5	65.3	59.9	8.27

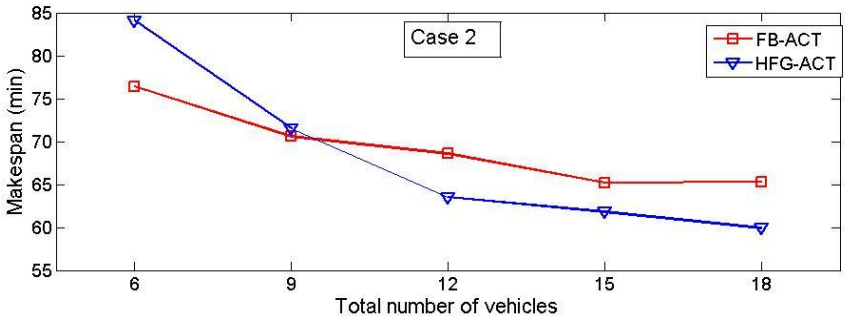
Continue Table 5.2

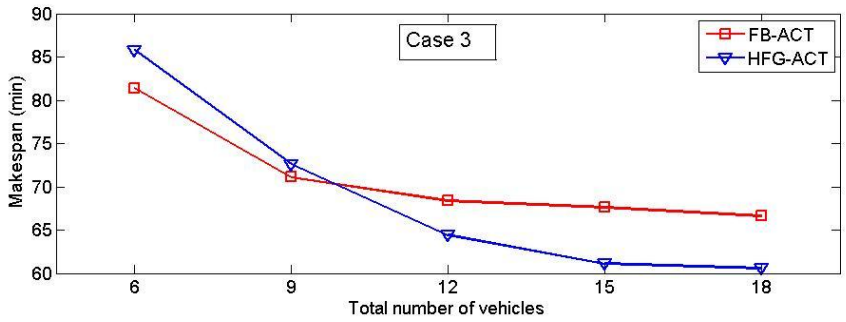
P3.5	66.6	60.6	9.01
P4.5	67.3	60.8	9.66
P5.5	63.5	57.2	9.92
Average:			9.82

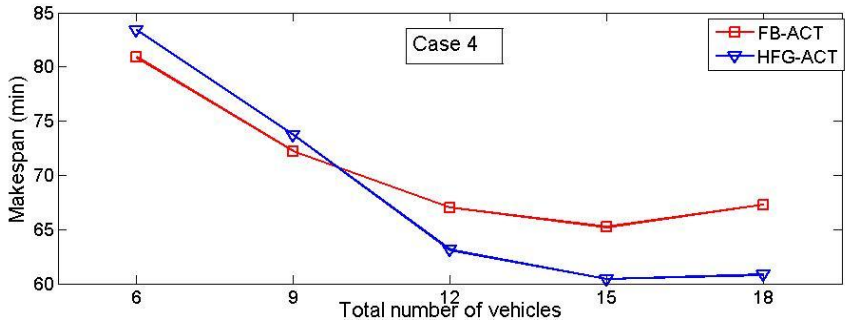
We randomly generate five cases and each case is calculated when there are 2 FTs, 3 FTs, 4 FTs, 5 FTs and 6 FTs. The index of 2 FTs means that there are 2 FTs on the rail in the FB-ACT and there are 6 FTs in this system. Thus, there are also 6 TUs in HFG-ACT. The cases of P1.1 and P1.2 are under the same case but with different number of FTs. There are 6 FTs in total in FB-ACT in the case P1.1 while there 9 FTs in total in FB-ACT in the case P1.2. The difference in Table 5.2 is calculated as following: $different = (column2 - column3)/column2$.

From the results, we can find that when the number of FTs on each rail is small, the FB-ACT performs better than HFG-ACT. This is because the traffic congestion on the rail is not much and the travelling speed of FT is higher than that of TU. However, when the number of FTs on the rail is large, HFG-ACT performs better than FB-ACT. The HFG-ACT can handle the traffic congestion on the rails more efficiently by flexibly selecting the paths while the FTs on the FB-ACT can only be delayed on the rails. Figure 5.5 shows the results of Table 5.1 that we plot the makespan of FB-ACT and HFG-ACT under different cases and number of vehicles.


Case 1
FB-ACT
HFG-ACT
Makespan (min)
Total number of vehicles
85
80
75
70
65
60
55
6
9
12
15
18


Case 2
FB-ACT
HFG-ACT
Makespan (min)
Total number of vehicles
85
80
75
70
65
60
55
6
9
12
15
18


Case 3
FB-ACT
HFG-ACT
Makespan (min)
Total number of vehicles
90
85
80
75
70
65
60
6
9
12
15
18


Case 4
FB-ACT
HFG-ACT
Makespan (min)
Total number of vehicles
85
80
75
70
65
60
6
9
12
15
18

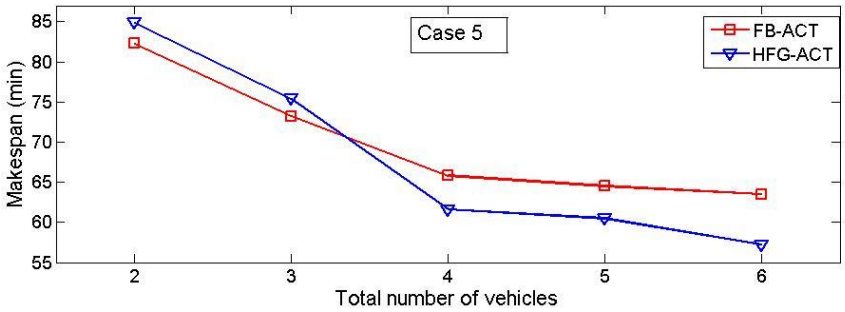


Figure 5.5 Makespan of FB-ACT and HFG-ACT

From Figure 5.5, we can find that the FB-ACT will not get benefit from the increase of vehicle numbers when it gets congested. However, HFG-ACT outperforms FB-ACT in the scenario that the number of vehicles is relative large.

5.6 Conclusion

In this study, we observe some operational limitations that impede on the performance of the FB-ACT. Thus, we propose a hybrid FB-ACT system which uses some good ideas from the GRID-ACT and implement in the transportation system at the quayside. From the numerical results, we can find that the new design system can handle the vehicle congestion more efficiently than FB-ACT. When the number of vehicles is small, FB-ACT performs better than HFG-ACT. However, when the number of vehicles gets large, FB-ACT seems to be stunned in the vehicle congestion. Thus HFG-ACT is able to provide a good solution in this situation.

CHAPTER 6 Conclusion

With the development of technology, container terminal operators tend to design new container terminals to improve the productivity as well as maintain efficient operations. In this thesis, we study the details of the operations of two new conceptual ACT systems which have triggered interests in industrial as well as academic area. To the best of our knowledge, the work in the thesis is the first work to simultaneously solve the vehicle dispatching and conflict-free routing in these two ACT systems. As we observe some operational limitations that impede on the performance of the FB-ACT, we propose a hybrid FB-ACT system which integrate some good ideas from the GRID-ACT into the transportation system at the quayside

We first study the system called Frame Bridge based Automated Container Terminal proposed by Shanghai Zhenhua Heavy Industries Co. Ltd. In FB-ACT system, several different types of equipment are involved in the transportation of one container. We emphasize on the management of resources in order to reduce the delay during the handshakes and avoid vehicle collisions, which significantly affects the efficiency of the system. We develop a mathematical programming model which integrates the vehicle dispatching problem and vehicle conflict-free routing problem. Because the model cannot be solved when the scale of the problem is large, we develop a tree structure algorithm. Our algorithm can solve this complicated problem efficiently by dividing it into the sub-problems with small scale. By comparing with the heuristic algorithm, our algorithm can solve the problem and give a more satisfying performance.

Another new conceptual ACT called Goods Retrieval and Inventory Distribution (GRID) based ACT was proposed by BEC industries LLC. In GRID-ACT, the main issues are the vehicle dispatching and conflict-free routing problems. Due to the meshed structure of the paths, the problem is modeled as a set partitioning problem. Given the intrinsic difficulty of this problem, the tree structure algorithm is also adopted to solve the complicated problem. The conflict-free routing problem should be concentrated on because of its significance in reducing traveling time including the delay. Thus two methods are proposed to solve this problem. One is fast in computation, which will be used in screening procedures. And the other one is more accurate, which will be used in the selection of beam nodes (the partial solutions which will be further exploration). Because of the complexity of the conflict-free routing problem, the typical column generation algorithm cannot solve it well. However, we can continue the search by using the information obtained by the typical column generation algorithm. New columns can be generated by adding new constraints which aim to prevent TUs from running in congested area. After the additional searching, we can obtain a promising and feasible solution for the conflict-free routing problem. Our algorithm can obtain satisfying solutions when compared to a heuristic algorithm.

By analyzing the performance of FB-ACT and GRID-ACT, we proposed a new design of container terminal. The main drawback of FB-ACT is the relative long waiting time because of the prevention of vehicle collisions. The vehicles in GRID-ACT can flexibly select paths to prevent vehicle collisions, which can reduce the waiting time. However, because of the slow speed of the vehicle and the long distance of the paths, GRID-ACT has to spend a

relatively long time on traveling. This newly designed terminal is a hybrid of FB-ACT and GRID-ACT, which appears to offset the drawbacks of these two kinds of terminals. This newly designed terminal outperforms FB-ACT and the result is shown by a comparison study.

There are still many interesting topics related to these new conceptual ACT systems. One of them is the container allocation problem. Because the stacking mechanism in GRID-ACT is very different from the traditional container terminals, how to allocate the containers in the storage yard seems to be one crucial strategy that affects the performance of the system. The second one is the operations related to the yard in FB-ACT system. The system can use more than one GT on the ground rail. The ground rails can also be constructed as two layers. Thus, we should spend more efforts on the scheduling problem in the yard area.

Appendix A

Some new parameters are introduced to present the reduced MIP model.

Parameters:

$S_m(i)$: the *ith* subtask in the FT schedule of FT m. The last subtask in the FT schedule of FT m can be indicated as $S_m(\Delta)$, where Δ is the look-ahead number.

C_{GT} : the set of pairwise subtasks which competes for the same GT.

C_{TP} : the set of pairwise subtasks which competes for the same TP.

C_{FT} : the set of pairwise subtasks where there is FT conflict between them.

C : the set of pairwise subtasks which cannot be operated simultaneously. We can have: $C = C_{GT} \bigcup C_{TP} \bigcup C_{FT}$.

Decision variables:

$Z^*_{(i,\alpha,k)(j,\beta,h)}$: $Z^*_{(i,\alpha,k)(j,\beta,h)}$=1 if the starting time of subtask (j,β,h) is greater than or equals to the finishing time of subtask (i,α,k) ; $Z^*_{(i,\alpha,k)(j,\beta,h)}$ =0, conversely. It is different from the variable $Z_{(i,\alpha,k)(j,\beta,h)}$ in the original MIP model which the value of zero does not force the starting time of subtask (i,α,k) to be greater than or equals to the finishing time of subtask (j,β,h).

The reduced MIP model can be shown as below:

$$Min: \underset{m \in M}{Max}\{T_{(S_m(\Delta))} + p_{(S_m(\Delta))}\}$$

Constraints:

(1) Time constraints for the two subtasks of a given job

$$T_{(i,\alpha,1)} + p_{(i,\alpha,1)} + t^{ft}_{(i,\alpha,1)(i,\alpha,2)} \le T_{(i,\alpha,2)}, \quad \forall (i,\alpha) \in H$$

(2) Sequence dependent times for different resources

QC:

$$T_{(i+1,\alpha,2)} \ge T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + t^{ll}_{qc}, \quad \forall (i,\alpha) \in L, (i+1,\alpha) \in L$$

$$T_{(i+1,\alpha,1)} \ge T_{(i,\alpha,2)} + p_{(i,\alpha,2)} + t^{ld}_{qc}, \quad \forall (i,\alpha) \in L, (i+1,\alpha) \in D$$

$$T_{(i+1,\alpha,2)} \ge T_{(i,\alpha,1)} + p_{(i,\alpha,1)} + t^{dl}_{qc}, \quad \forall (i,\alpha) \in D, (i+1,\alpha) \in L$$

$$T_{(i+1,\alpha,1)} \ge T_{(i,\alpha,1)} + p_{(i,\alpha,1)} + t^{dd}_{qc}, \quad \forall (i,\alpha) \in D, (i+1,\alpha) \in D$$

GT:

$$T_{(j,\beta,h)} \ge T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t_{ms} + M(Z_{(i,\alpha,k)(j,\beta,h)} - 1),$$
$$\forall ((i,\alpha,k),(j,\beta,h)) \in C_{GT}$$

$$T_{(i,\alpha,k)} \ge T_{(j,\beta,h)} + p_{(j,\beta,h)} + t_{ms} - M * Z_{(i,\alpha,k)(j,\beta,h)},$$
$$\forall ((i,\alpha,k),(j,\beta,h)) \in C_{GT}$$

TP:

$$T_{(j,\beta,h)} \ge T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t^{tp}_{(i,\alpha,k)(j,\beta,h)} + M(Z_{(i,\alpha,k)(j,\beta,h)} - 1),$$
$$\forall ((i,\alpha,k),(j,\beta,h)) \in C_{TP}$$
$$T_{(i,\alpha,k)} \ge T_{(j,\beta,h)} + p_{(j,\beta,h)} + t^{tp}_{(i,\alpha,k)(j,\beta,h)} - M * Z_{(i,\alpha,k)(j,\beta,h)},$$
$$\forall ((i,\alpha,k),(j,\beta,h)) \in C_{TP}$$

FT conflict:

$$T_{(j,\beta,h)} \geq T_{(i,\alpha,k)} + p_{(i,\alpha,k)} + t^{ft}_{(i,\alpha,k)(j,\beta,h)} + M(Z_{(i,\alpha,k)(j,\beta,h)} - 1),$$
$$\forall((i,\alpha,k),(j,\beta,h)) \in C_{FT}$$
$$T_{(i,\alpha,k)} \geq T_{(j,\beta,h)} + p_{(j,\beta,h)} + t^{ft}_{(i,\alpha,k)(j,\beta,h)} - M * Z_{(i,\alpha,k)(j,\beta,h)},$$
$$\forall((i,\alpha,k),(j,\beta,h)) \in C_{FT}$$

(3) Domain of variables

$$Z_{(i,\alpha,k)(j,\beta,h)} = 0 \quad OR \quad 1, \quad \forall((i,\alpha,k),(j,\beta,h)) \in C$$

$$T_{(i,\alpha,1)}, T_{(i,\alpha,2)} \geq 0, \quad \forall(i,\alpha) \in H$$

References

Agarwal, Y., K. Mathur, et al. (1989). "A set‐partitioning‐based exact algorithm for the vehicle routing problem." Networks 19(7): 731-749.

Amato, F., Basile, F., Carbone, C., & Chiacchio, P. (2005). An approach to control automated warehouse systems. Control Engineering Practice, *13*(10), 1223-1241.

Appelgren, L. H. (1969). "A column generation algorithm for a ship scheduling problem." Transportation science 3(1): 53-68.

Asef-Vaziri, A. and B. Khoshnevis (2006). Automated technologies in maritime container terminals. METRANS national urban freight conference, Long Beach.

Baldacci, R., N. Christofides, et al. (2008). "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts." Mathematical Programming 115(2): 351-385.

Bard, J. F. and H. W. Purnomo (2005). "Preference scheduling for nurses using column generation." European journal of operational research 164(2): 510-534.

Bierwirth, C. and F. Meisel (2009). "A fast heuristic for quay crane scheduling with interference constraints." Journal of Scheduling 12(4): 345-360.

Bish, E. K., F. Y. Chen, et al. (2005). "Dispatching vehicles in a mega container terminal." OR spectrum 27(4): 491-506.

Briskorn, D., A. Drexl, et al. (2006). "Inventory-based dispatching of automated guided vehicles on container terminals." OR spectrum 28(4): 611-630.

Canonaco, P., P. Legato, et al. (2008). "A queuing network model for the management of berth crane operations." Computers & Operations Research 35(8): 2432-2446.

Chan, C. T., L.H. Huat (2002). "Containers, container ships and quay cranes: a practical guide." Genesis Typesetting & Publication Services.

Chang, D., Z. Jiang, et al. (2010). "Integrating berth allocation and quay crane assignments." Transportation Research Part E: Logistics and Transportation Review 46(6): 975-990.

Chang, D., Z. Jiang, et al. (2011). "Developing a dynamic rolling-horizon decision strategy for yard crane scheduling." Advanced Engineering Informatics 25(3): 485-494.

Chen, Z.-L. and W. B. Powell (1999). "Solving parallel machine scheduling problems by column generation." INFORMS Journal on Computing 11(1): 78-94.

Choi, E. and D.-W. Tcha (2007). "A column generation approach to the heterogeneous fleet vehicle routing problem." Computers & Operations Research 34(7): 2080-2095.

Corréa, A. I., A. Langevin, et al. (2004). Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming, Springer.

De, S. and A. Lee (1990). "Flexible manufacturing system (FMS) scheduling using filtered beam search." Journal of Intelligent Manufacturing 1(3): 165-183.

Deng, L., Lu, G., Yang, W., & Fei, M. (2013). Automated warehouse path optimization based on immunity discrete particle swarm optimization. Chinese Automation Congress (CAC), *2013* (pp.703-707). IEEE.

Desaulniers, G., J. Desrosiers, et al. (2002). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems, Springer.

Desaulniers, G., A. Langevin, et al. (2003). "Dispatching and conflict-free routing of automated guided vehicles: An exact approach." International Journal of Flexible Manufacturing Systems 15(4): 309-331.

Desrochers, M. and F. Soumis (1989). "A column generation approach to the urban transit crew scheduling problem." Transportation science 23(1): 1-13.

Desrosiers, J., F. Soumis, et al. (1984). "Routing with time windows by column generation." Networks 14(4): 545-565.

Geerlings, H. and R. Van Duin (2011). "A new method for assessing CO2-emissions from container terminals: a promising approach applied in Rotterdam." Journal of Cleaner Production 19(6): 657-666.

Ghasemzadeh, H., E. Behrangi, et al. (2009). "Conflict-free scheduling and routing of automated guided vehicles in mesh topologies." Robotics and Autonomous Systems 57(6): 738-748.

Gilmore, P. C. and R. E. Gomory (1961). "A linear programming approach to the cutting-stock problem." Operations research 9(6): 849-859.

Grunow, M., H.-O. Günther, et al. (2006). "Strategies for dispatching AGVs at automated seaport container terminals." OR spectrum 28(4): 587-610.

He, J., D. Chang, et al. (2010). "A hybrid parallel genetic algorithm for yard crane scheduling." Transportation Research Part E: Logistics and Transportation Review 46(1): 136-155.

He, S. J., Cheng, F., & Luo, J. (2007). Modeling and Implementing of an Automated Warehouse via Colored Timed Petri Nets; a Behavior Perspective. Control and Automation, 2007. ICCA 2007. IEEE International Conference on (pp.2823-2828). IEEE.

Imai, A., H. C. Chen, et al. (2008). "The simultaneous berth and quay crane allocation problem." Transportation Research Part E: Logistics and Transportation Review 44(5): 900-920.

Javanshir, H. and S. S. Ganji (2010). "Yard crane scheduling in port container terminals using genetic algorithm." Journal of Industrial Engineering International 6(11): 39-50.

Jiang, L. L., & Zhang, C. (2010). Stacker picking path optimization of the automated warehouse based on graph theory. Logistics Sci-Tech.

Jeon, S. M., K. H. Kim, et al. (2011). "Routing automated guided vehicles in container terminals through the Q-learning technique." Logistics Research 3(1): 19-27.

Jung, S. H. and K. H. Kim (2006). "Load scheduling for multiple quay cranes in port container terminals." Journal of Intelligent Manufacturing 17(4): 479-492.

k., P. N. (2007). "A study on the efficiency of transportation equipment at automated container terminals." Proceedings of the 2007 annual Conference on International Conference on Computer Engineering and Applications, World Scientific and Engineering Academy and Society (WSEAS).

Kim, C. W. and J. M. Tanchoco (1991). "Conflict-free shortest-time bidirectional AGV routeing." THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 29(12): 2377-2391.

Kim, C. W. and J. TANCHOCOJ (1993). "Operational control of a bidirectional automated guided vehicle system." THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 31(9): 2123-2138.

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

Kim, K. H. and J. W. Bae (2004). "A look-ahead dispatching method for automated guided vehicles in automated port container terminals." Transportation science 38(2): 224-234.

Kim, K. H., J. S. Kang, et al. (2004). "A beam search algorithm for the load sequencing of outbound containers in port container terminals." OR Spectrum 26(1): 93-116.

Kim, K. H. and K. Y. Kim (1999). "An optimal routing algorithm for a transfer crane in port container terminals." Transportation science 33(1): 17-33.

Kim, K. H. and Y.-M. Park (2004). "A crane scheduling method for port container terminals." European journal of operational research 156(3): 752-768.

Lee, D.-H., H. Q. Wang, et al. (2008). "Quay crane scheduling with non-interference constraints in port container terminals." Transportation Research Part E: Logistics and Transportation Review 44(1): 124-135.

Lee, L. H., E. P. Chew, et al. (2010). "Vehicle dispatching algorithms for container transshipment hubs." OR spectrum 32(3): 663-685.

Li, H. P., Fang, Z. F., & Wang, Y. (2011). Research on path optimization of automated warehouse based on ant colony algorithm. Advanced Materials Research, *201-203*, 1112-1115.

Li, W., Y. Wu, et al. (2009). "Discrete time model and algorithms for container yard crane scheduling." European journal of operational research 198(1): 165-172.

Liang, C., Y. Huang, et al. (2009). "A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning." Computers & Industrial Engineering 56(3): 1021-1028.

Liu, C.-I. and P. Ioannou (2002). A comparison of different AGV dispatching rules in an automated container terminal. Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on, IEEE.

Liu, C.-I., H. Jula, et al. (2002). "Design, simulation, and evaluation of automated container terminals." Intelligent Transportation Systems, IEEE Transactions on 3(1): 12-26.

Liu, S. N. (2011). Modeling and optimization of integrated scheduling of automated warehouse system. Advanced Materials Research, *230-232*, 35-39.

Maza, S. and P. Castagna (2001). Conflict-free AGV routing in bi-directional network. Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on, IEEE.

Maza, S. and P. Castagna (2005). "A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles." Computers in Industry 56(7): 719-733.

Moccia, L., J. F. Cordeau, et al. (2006). "A branch - and - cut algorithm for the quay crane scheduling problem in a container terminal." Naval Research Logistics 53(1): 45-59.

Möhring, R. H., E. Köhler, et al. (2005). Conflict-free real-time AGV routing. Operations Research Proceedings 2004, Springer.

Mourgaya, M. and F. Vanderbeck (2007). "Column generation based heuristic for tactical planning in multi-period vehicle routing." European journal of operational research 183(3): 1028-1041.

Nair, S. K., L. S. Thakur, et al. (1995). "Near optimal solutions for product line design and selection: beam search heuristics." Management Science 41(5): 767-785.

Ng, W. (2005). "Crane scheduling in container yards with inter-crane interference." European journal of operational research 164(1): 64-78.

Ng, W. and K. Mak (2005). "An effective heuristic for scheduling a yard crane to handle jobs with different ready times." Engineering Optimization 37(8): 867-877.

Ng, W. and K. Mak (2005). "Yard crane scheduling in port container terminals." Applied mathematical modelling 29(3): 263-276.

Ng, W. and K. Mak (2006). "Quay crane scheduling in container terminals." Engineering Optimization 38(6): 723-737.

Nishi, T., Y. Hiranaka, et al. (2011). "A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles." Computers & Operations Research 38(5): 876-888.

Nishi, T. and Y. Tanaka (2012). "Petri net decomposition approach for dispatching and conflict-free routing of bidirectional automated guided vehicle systems." Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 42(5): 1230-1243.

Oboth, C., R. Batta, et al. (1999). "Dynamic conflict-free routing of automated guided vehicles." International Journal of Production Research 37(9): 2003-2030.

Ow, P. S. and T. E. Morton (1988). "Filtered beam search in scheduling†." THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 26(1): 35-62.

Ow, P. S. and T. E. Morton (1989). "The single machine early/tardy problem." Management Science 35(2): 177-191.

Park, Y.-M. and K. H. Kim (2005). A scheduling method for berth and quay cranes. Container Terminals and Automated Transport Systems, Springer: 159-181.

Qiu, L. and W.-J. Hsu (2001). "A bi-directional path layout for conflict-free routing of AGVs." International Journal of Production Research 39(10): 2177-2195.

Ribeiro, C. C. and F. Soumis (1994). "A column generation approach to the multiple-depot vehicle scheduling problem." Operations research 42(1): 41-52.

Sabuncuoglu, I. and M. Bayiz (1999). "Job shop scheduling with beam search." European journal of operational research 118(2): 390-412.

Sammarra, M., J.-F. Cordeau, et al. (2007). "A tabu search heuristic for the quay crane scheduling problem." Journal of Scheduling 10(4-5): 327-336.

Singh, S. and M. Tiwari (2002). "Intelligent agent framework to determine the optimal conflict-free path for an automated guided vehicles system." International Journal of Production Research 40(16): 4195-4223.

Stahlbock, R. and S. Voß (2008). "Operations research at container terminals: a literature update." OR spectrum 30(1): 1-52.

Steenken, D., S. Voß, et al. (2004). "Container terminal operation and operations research-a classification and literature review." OR spectrum 26(1): 3-49.

Taillard, É. D. (1999). "A heuristic column generation method for the heterogeneous fleet VRP." Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle 33(1): 1-14.

Vacca, I., M. Salani, et al. (2013). "An exact algorithm for the integrated planning of berth allocation and quay crane assignment." Transportation science 47(2): 148-161.

Van den Akker, J., C. A. Hurkens, et al. (2000). "Time-indexed formulations for machine scheduling problems: Column generation." INFORMS Journal on Computing 12(2): 111-124.

Vanderbeck, F. (2005). Implementing mixed integer column generation. Column Generation, Springer: 331-358.

Vangeri, A., & Hebbal, S. S. (2014). Route optimization of automated warehouse with the aid of modified genetic algorithms (mga). International Review of Mechanical Engineering, *8*(4), 667-679.

Vis, I. F. and R. De Koster (2003). "Transshipment of containers at a container terminal: An overview." European journal of operational research 147(1): 1-16.

Wang, F. and A. Lim (2007). "A stochastic beam search for the berth allocation problem." Decision Support Systems 42(4): 2186-2196.

Wang, T. B., & Zhu, Z. H. (2008). Research of modeling automated warehouse with colored timed petri net. Mechanical & Electrical Engineering Technology.

Wilhelm, W. E. (2001). "A technical review of column generation in integer programming." Optimization and Engineering 2(2): 159-200.

Zeng, J. and W.-J. Hsu (2008). "Conflict-free container routing in mesh yard layouts." Robotics and Autonomous Systems 56(5): 451-460.

Zhang, L.-W., R. Ye, et al. (2005). "Mixed integer programming models for dispatching vehicles at a container terminal." Journal of Applied Mathematics and Computing 17(1-2): 145-170.

Zhen, L., L. H. Lee, et al. (2012). "A comparative study on two types of automated container terminal systems." Automation Science and Engineering, IEEE Transactions on 9(1): 56-69.