Computers & Industrial Engineering 166 (2022) 107968

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A branch-and-bound approach for AGV dispatching and routing problems in automated container terminals

Purple denotes assumptions
Red is purpose
blue is Methodology
orange is fasle 5

Zehao Wang [a], Qingcheng Zeng [b,*]

[a] School of Maritime Economics and Management, Dalian Maritime University, Dalian, PR China
[b] School of Maritime Economics and Management, Dalian Maritime University, Dalian, PR China

ARTICLE INFO

Keywords:
Automated container terminals
AGV dispatching
Path planning
Bidirectional path
Branch-and-bound

ABSTRACT

The efficiency of automated container terminals primarily depends on the automated guided vehicle (AGV) scheduling scheme. This paper investigates the AGV dispatching and routing problem with multiple bidirectional paths to generate conflict-free routes. A mixed-integer programming (MIP) model is formulated to minimize the completion time of all jobs (i.e., minimize the makespan). A tailored branch-and-bound (B&B) algorithm is developed to solve the problem within a reasonable amount of time, where the B&B process is used to obtain job assignments and job sequences, and a heuristic algorithm is proposed to generate conflict-free routes. To expedite the proposed algorithm, some accelerating techniques are designed in view of the characteristics of the problem. Computational experiments indicate that the proposed algorithm can obtain near-optimal solutions in small-scale instances, and in large-scale instances, the B&B algorithm significantly outperforms the port scheduling strategies. The proposed algorithm can be applied to real-world cases and dynamic cases, and a two-stage greedy heuristic algorithm is also developed to obtain a satisfactory solution in a short amount of time. Some managerial insights into AGV scheduling are offered to guide terminal operations in this study.

1. Introduction

Container terminals play an important role in international container trade and the global economy. The volume of containers worldwide has continued to grow for decades, and is expected to continue growing in the future. To meet the growing trade demands, ports have gradually moved to the construction of automated container terminals or semi-automated container terminals, considering sustainable development and the significant benefits brought by automation. Automated container terminals contribute to the reduction of manual operation and the enhancement of operation stability, which can reduce the number of workers by at least 45% and operating expenses by 25%-55%, thereby improving terminal productivity by 10%-35% (McKinsey & Company, 2018, Chen et al., 2020). However, the rapid development of global containership scale has posed great challenges to the capacities of automated container terminals, and there is thus an urgent demand to improve the operation efficiency and service quality. Due to the limitation of the terminal layout and the high cost of quay cranes (QCs) and automated stacking cranes (ASCs), simply increasing the number of cranes is not an economical and effective way to improve terminal efficiency. Meanwhile, an excessive number of cranes and vehicles may lead to conflicting equipment waiting, which in turn limits the improvement of operation efficiency. Therefore, the terminal prefers to propose more efficient equipment scheduling solutions to improve efficiency and economic performance. A possible solution is to address the vehicle scheduling problem of how to assign containers with conflict-free routes so as to minimize the completion time of all jobs, thus achieving a high productivity (Zhong et al., 2020). The vehicle scheduling problem has attracted the increasing attention of scholars and terminal operators.

or, stacker cranes

Def of AGV

Automated guided vehicles (AGVs) are important transportation vehicles equipped with automatic guiding devices that can transport containers between the QC on the quayside and the ASC on the yard side. Fig. 1 shows a typical layout of an automated container terminal, where the QC is engaged for loading (unloading) containers onto (from) the ship at the quayside, and the ASC is responsible for container stacking and retrieval in the yard. In front of each block in the yard are a few handover points (HPs) where the container transfer between the AGV and ASC is conducted, and HPs also exist under the back-reach of each QC for the container transfer to (from) AGV, as shown in Fig. 1. The AGV-support (Tian et al., 2018) is designed to reduce the equipment waiting time and improve the operation efficiency, and Fig. 2 illustrates

* Corresponding author.
E-mail addresses: wzh_wlgc@163.com (Z. Wang), qzeng@dlmu.edu.cn (Q. Zeng).

https://doi.org/10.1016/j.cie.2022.107968
Received 15 January 2021; Received in revised form 6 January 2022; Accepted 23 January 2022
Available online 30 January 2022

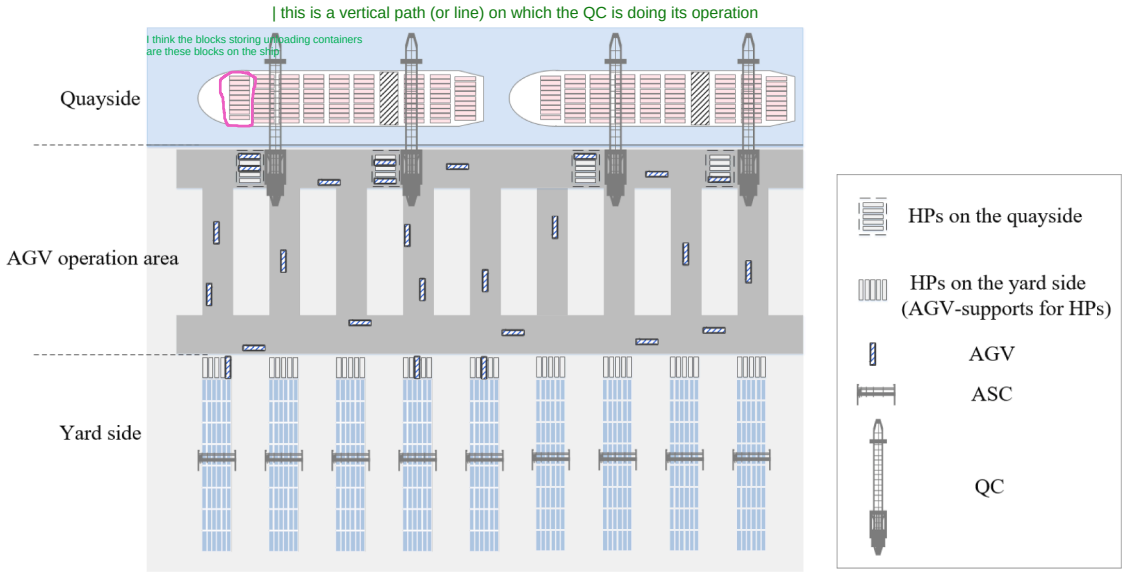


Fig. 1. The typical layout of an automated container terminal.

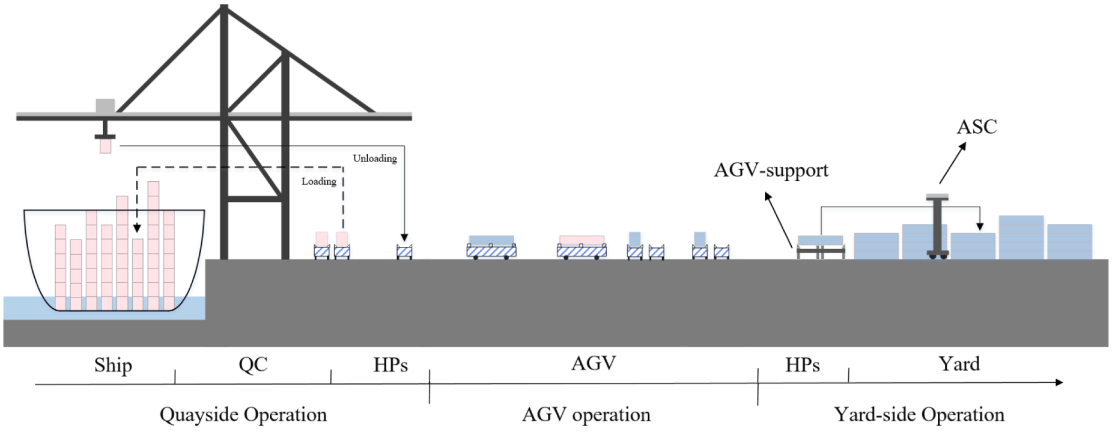


Fig. 2. The operation in automated container terminals.

the operation in automated container terminals. The AGV-supports are used for the temporary storage of containers, which are located in the HPs on the yard side, and the AGVs and ASCs can autonomously place containers on the AGV-supports or pick them up from the AGV-supports. When picking up or placing a container at the AGV-support, the AGV can lift the container and place it on the AGV-support, or lift the container and remove it from the AGV-support. By using the AGV-support, both the ASC and the AGV can transfer containers autonomously without waiting for each other. The operation mode of automated container terminals can be divided into a loading process and an unloading process. In the unloading process, a QC discharges a container onto an AGV, which will transport the container to the yard. The AGV places the container on the AGV-support and continues its next operation job. Then, an ASC collects the container from the AGV-support and places it to the corresponding location in the yard. The loading process is the reverse of the unloading process.

AGV is the key piece of equipment that connects the quayside operation and yard side operation, and its transportation has to simultaneously consider the cooperation of the QC and ASC. The degree of synchronization between AGVs and cranes is one of the key factors influencing quayside productivity, and the AGV scheduling is directly related to the whole operation of the terminal. Ideally, cranes do not need to wait for AGVs, and container operation can be performed without delay, which cannot be achieved by simply increasing the number of AGVs (Choe et al., 2016). Therefore, an excellent AGV scheduling scheme is required to achieve efficient terminal operations. In addition, the AGV scheduling problem is extremely complicated, which not only determines the job assignment to the AGV, but also includes the AGV path planning to avoid congestion and conflict. Combining these two factors to develop AGV scheduling schemes can better serve the cranes, thus improving the operation efficiency of the terminal.

Some studies related to the AGV scheduling problem focus on the job assignment of the AGV, and thus ignore the traffic congestion of the AGVs on the path (Choe et al., 2016; Luo and Wu, 2015; Luo et al., 2016; Hu et al., 2019; Luo and Wu, 2020). In those studies, the travelling time of AGVs between any two processing locations (HPs of blocks and QCs) is assumed to be known in advance. Although Chen et al. (2020) modeled the AGV congestion, the potential conflicts among vehicle movements were not taken into consideration. Considering congestion and conflict during vehicle transportation, layouts with unidirectional travel paths are used at various terminals in Europe, where multiple parallel travel paths and multiple shortcuts are adopted to minimize the travel time between the quayside and the yard side (Debjit et al., 2020). Fig. 3 illustrates six layouts of AGVs used in the container terminal research. Fig. 3 (a) illustrates a single unidirectional travel layout used by all vehicles (Vis and Harika, 2004), and Fig. 3 (b) shows a unidirectional travel layout with multiple parallel travel paths at the quayside

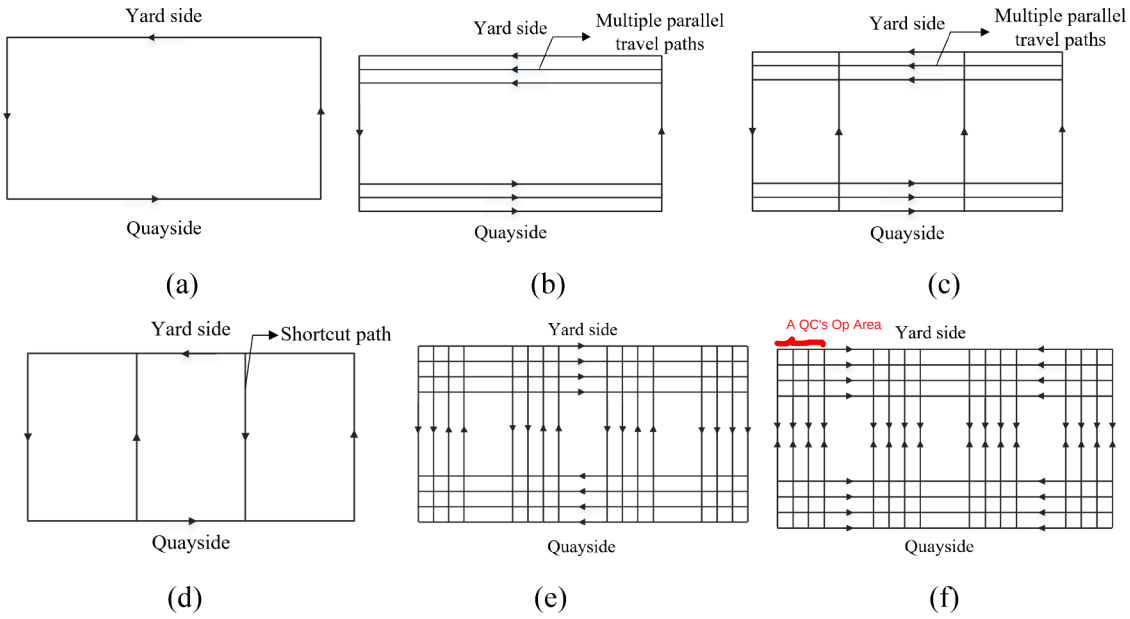


Fig. 3. Types of travel path used in the AGV transport area. Notes: (a) Single unidirectional loops; (b) multiple unidirectional loops; (c) multiple unidirectional loops with shortcuts between quayside and yard side; (d) unidirectional path structure with multiple shortcuts; (e) unidirectional path structure with multiple parallel travel paths and multiple shortcuts; (f) bidirectional path structure with multiple parallel travel paths and multiple shortcuts.

and at the yard side (Hoshino et al., 2004). Debjit et al. (2020) advocate the layout from Fig. 3 (c), which is the multiple unidirectional loops with shortcuts between quayside and yard side. Ji et al. (2020) use the layout with a unidirectional path structure and multiple shortcuts, as shown in Fig. 3 (d). The unidirectional path structure with multiple parallel travel paths and multiple shortcuts is adopted by some studies (Yang et al., 2018; Zhong et al., 2020), as shown in Fig. 3 (e). In this research, we adopt the layout from Fig. 3 (f), which is the bidirectional path structure with multiple parallel travel paths and multiple shortcuts.

The multiple parallel travel paths at the quayside and at the yard side provide more path options for AGV transportation, while multiple shortcuts between the quayside and the yard side have a substantial effect on minimizing the travel time and improving the system performance. In Fig. 3 (a) to (d), there are fewer optional routes between any two processing locations (HPs of blocks and QCs), which has a high potential for congestion and conflict, and is not conducive to efficient transportation. As shown in Fig. 3 (e) and Fig. 3 (f), the number of optional routes between any two processing locations can be increased by using multiple parallel travel paths and multiple shortcuts, thus reducing congestion and conflict, as well as reducing delays. Therefore, it is crucial for terminal operators to determine an efficient AGV scheduling scheme based on multiple parallel travel paths and multiple shortcuts.

Advantage of using bidirectional layout

In addition, the AGV is a vehicle that can travel in both directions, so the travel paths used in the AGV transport area should also be bidirectional. Although unidirectional travel paths are advocated for some automated container terminals, bidirectional travel paths can better guarantee efficient horizontal transport operations. Compared to unidirectional path structure, bidirectional path structure makes AGV transport more flexible, offers more optional routes between any two processing locations, and reduces unnecessary detours. However, a bidirectional path structure makes avoiding AGV congestion and conflict more complicated. As illustrated in Fig. 3 (f), the bidirectional path structure with multiple parallel travel paths and multiple shortcuts is used in the automated container terminal of Qingdao Qianwan, where the AGV scheduling problem with conflict-free routes has received great attention from terminal operators. Therefore, it is important to incorporate the effects of the bidirectional travel paths in the AGV scheduling problem.

**** Disadvantage of using bidirectional layout

The main contributions of this paper are twofold. First, this paper focuses on the AGV dispatching and routing problem (**ADRP**) in an automated container terminal. Compared to previous related studies, we adopt the bidirectional path structure with multiple parallel travel paths and multiple shortcuts. Considering the congestion and conflict during AGV transportation, we present a MIP model for **ADRP** to determine the operation job, operation sequence, and conflict-free travel route. Second, considering that this model is usually computationally challenging for even small-scale problem instances because of the extremely large number of feasible assignment schemes and routes, we develop a tailored branch-and-bound (B&B) algorithm based on the characteristics of the problem to solve the **ADRP** efficiently. Several accelerating techniques are also proposed to enhance the efficiency of the B&B algorithm. Computational experiments based on the filed data of Qingdao Port are conducted to evaluate the performance of the proposed method. The experimental results show that the proposed B&B algorithm significantly outperforms the commercial solver and two port scheduling strategies. Moreover, a two-stage greedy heuristic algorithm is developed to obtain a satisfactory solution in a short amount of time.

The remainder of this paper is structured as follows. Section 2 reviews the related literature. Section 3 describes the problem considered in this study. Section 4 formulates a model for the problem. Section 5 provides a tailored branch-and-bound algorithm. Section 6 discusses the results of the computational experiments. Section 7 concludes this study and suggests some possible directions for future studies.

2. Literature review

There have been numerous studies on the equipment scheduling of automated container terminals. Most of these studies are devoted to promoting the handling efficiency by optimizing equipment scheduling, including cranes and vehicles. Moreover, studies on vehicle scheduling mainly include AGV scheduling, ALV scheduling, and integrated scheduling, whereas little attention has been given to the congestion and conflict of vehicles. Vehicle dispatching, routing and scheduling can be described as follows (Vis, 2006; Miyamoto and Inoue, 2016). A dispatching problem involves selecting a vehicle for each transportation job to execute it. If the dispatching decision is made, a route and schedule should be planned for the vehicle to transport the job from its

def of dispatching problem

Table 1
Publications related to the vehicle scheduling problem in automated container terminals.

Refs.	Transportation equipment	path structure	Vehicle conflict	Modeling approach	Solution algorithm	Objective	Container operation
Luo et al. (2016)	AGV	–	No	MIP	GA	Min ship's berth time	U
Yang et al. (2018)	AGV	unidirectional	Yes	BPM	RHP + GA	Min makespan	L&U
Hu et al. (2019)	AGV & ALV	–	No	MIP	PSO + H	Min operational cost	L&U
Chen et al. (2020)	AGV	–	No	STN	ADMM	Min total turn time	L&U
Ji et al. (2020)	AGV	unidirectional	Yes	BPM	GA + H	Min makespan	L&U
Zhong et al. (2020)	AGV	unidirectional	Yes	MIP	GA + PSO	Min delay time	L&U
Luo and Wu. (2020)	AGV	–	No	MIP	GA	Min ship's berth time	L
Li et al. (2020)	ALV	–	No	MIP	GA	Min makespan	L&U
Debjit et al. (2020)	AGV & ALV	unidirectional	Yes	QNM	Simulation	Min throughput time	L&U
This paper	AGV	bidirectional	Yes	MIP	B&B	Min makespan	L&U

Notes: "MIP": mixed integer programming; "GA": genetic algorithm; "RHP": rolling horizon procedure; "PSO": particle swarm optimization; "H": heuristics; "STN": space–time network; "ADMM": alternating direction method of multiplier-based dual decomposition; "B&B": branch-and-bound; "BPM": bi-level programming model; "QNM": queuing network model; "L": loading; "U": unloading; "L&U": loading and unloading.

This could be the definition of the set of routes.

origin to its destination over the vehicle network. A route indicates the path taken by the vehicle when it makes a pickup and delivery. The related schedule gives the arrival and departure times of the vehicle at each segment, the pickup and delivery point, and intersections during the route to ensure conflict-free routing. A dispatching and routing problem involves making the dispatching and routing decisions simultaneously. Vehicle scheduling problems can be seen as routing problems with additional constraints concerning times at which certain activities (e.g., delivery of a load) have to be executed. This section first reviews the studies highly related to the vehicle scheduling problem, and then a review of studies related to automated container terminals is discussed.

And also this!

From the perspective of the scheduling problem in automated seaport container terminals, there are numerous studies on AGVs (Grunow et al., 2006; Angeloudis and Bell, 2010; Luo and Wu, 2020). Xin et al. (2015) studied the scheduling problem for free-ranging AGVs with the goal of minimizing the makespan. A hierarchical control architecture was proposed to solve the problem. Choe et al. (2016) proposed an online preference learning algorithm for dispatching AGVs to changing situations in an automated container terminal. The automated lifting vehicle (ALV) is a transportation vehicle that is widely used in automated container terminals, and its scheduling problem has received extensive attention (Nguyen and Kim, 2009; Li et al., 2020). Gupta et al. (2017) used an integrated queuing network to analyze the performance of container terminals with ALV. Roy and De (2018) used a network of open and semi-open queues to research the operation efficiency at an automated container terminal with ALVs. Some studies also investigated the operation comparison between AGVs and ALVs (Vis and Harika, 2004; Bae et al., 2009).

پیشینه

Studies that take into account congestion and conflict

The abovementioned studies focus on the vehicle scheduling problem without considering congestion and conflict. However, the vehicle operations of automated container terminals need to avoid congestion and conflict in practice, and some studies have investigated the vehicle scheduling problem with conflict-free paths. Yang et al. (2018) formulated a bi-level programming model for an integrated scheduling problem. The upper level model is the integrated scheduling problem of AGVs, and the lower level is AGV path planning. A congestion prevention rule-based bi-level genetic algorithm was designed to solve the model. Tommaso et al. (2018) addressed the conflict-free pickup and delivery problem to determine the vehicle paths and speeds on each arc of the path, with AGV scheduling in automated container terminals being one area that appeared in this problem. Zhong et al. (2020) studied the integrated scheduling of multi-AGVs with conflict-free path planning with cranes. A hybrid genetic algorithm-particle swarm optimization with a fuzzy logic controller was proposed. For rapid design evaluation of container terminals with AGVs and ALVs, Debjit et al. (2020) built an integrated queuing network model. These studies have formulated some vehicle transportation rules to avoid conflict, but the path structure in the studies is unidirectional, and there are few optional paths for vehicles.

Other studies related to the equipment scheduling of automated container terminals mostly focus on the crane scheduling problem (Park et al., 2010; Gharehgozli et al., 2017; Jenny et al., 2018; Lu and Wang, 2019) and integrated scheduling problem (Lau and Zhao, 2008; Zhong et al., 2020). In the crane scheduling problem, Hu et al. (2016) built three models with the goal of minimizing the makespan. An exact algorithm and a genetic algorithm were proposed to solve the scheduling problem of twin automated stacking cranes. Ulf and Kathrin (2017) studied the scheduling of automated yard cranes by using a simulation model with realistic scenario data. A branch-and-bound procedure was proposed to investigate the effects of four different automated yard crane systems on the terminal efficiency. To solve the yard crane scheduling problem with the uncertainty of the task groups' arrival times and handling volumes, He et al. (2019) proposed a GA-based framework combined with a three-stage algorithm. In the integrated scheduling problem of vehicles and other equipment, Hu et al. (2019) considered two types of popular vehicles in automated container terminals: automated lifting vehicles (ALVs) and AGVs. To solve the joint vehicle dispatching and storage allocation problem, a three-stage decomposition method was proposed based on a greedy search. An integrated problem of AGV scheduling and container storage was studied by Luo et al. (2016), where a genetic algorithm was proposed to solve the problem, focusing on the unloading process at an automated container terminal. Another study for this integrated problem considered a dual-cycle strategy at automated container terminals (Luo and Wu, 2015). Chen et al. (2020) studied an integrated rail-mounted yard crane and AGV scheduling problem. A multi-commodity network flow model based on a discretized virtualized network was proposed.

Newly designed automated container terminals, that is the terminals with new layout designs (such as FB-ACT, rail-shuttle-based ACT, and GRID-based ACT) and advanced equipment (including frame trolleys, transfer units, and ground transporters), have been researched in some studies (Zhen et al., 2012; Yang et al., 2015). Jiang et al. (2018) studied a newly designed container terminal named the frame bridge-based automated container terminal (FB-ACT). An algorithm based on a filtered beam search was proposed to solve the frame trolley dispatching problem. Zhen et al. (2018) studied the scheduling problems for two types of cranes in FB-ACT. A meta-heuristics algorithm was designed to improve the operation efficiency. A storage allocation strategy was proposed for a new automated container terminal by Zhou et al. (2018). The study introduced a novel approach for a transshipment container hub called the hybrid GRID system.

Different from previous related studies, we consider the AGV dispatching and routing problem with multiple bidirectional paths. Related studies and the work of this paper are summarized in Table 1 in terms of transportation equipment, path structure, vehicle conflict, modeling approach, solution algorithm, objective, and container operation. These

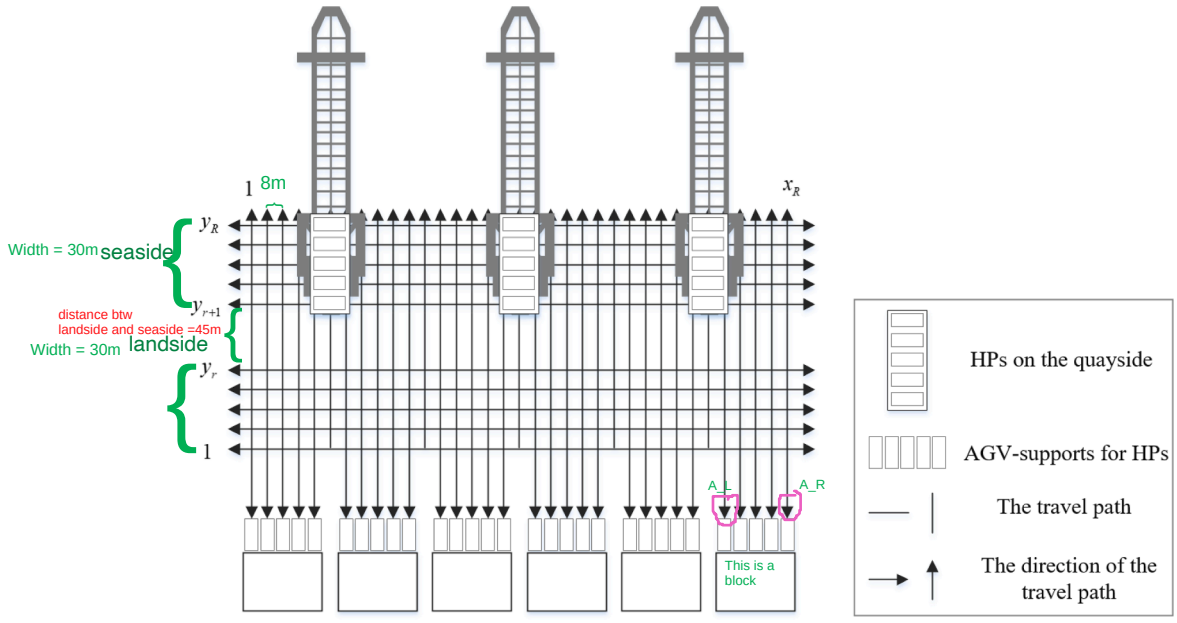


Fig. 4. The layout of the AGV operation area.

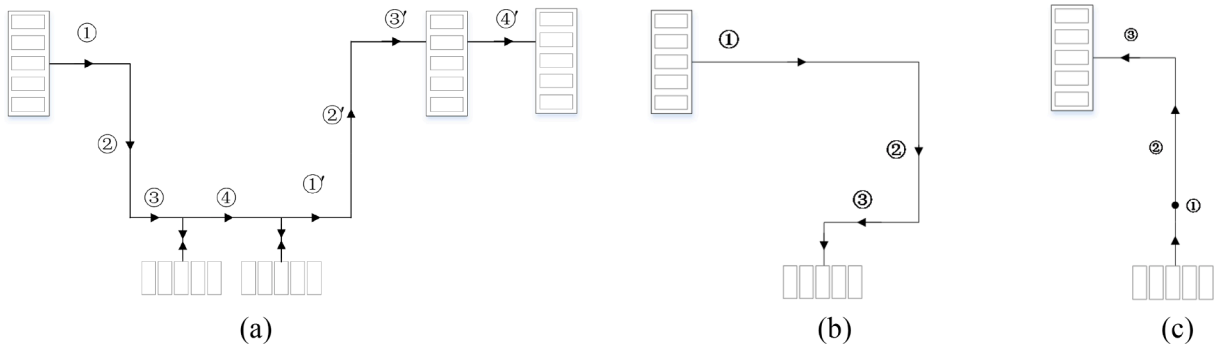


Fig. 5. The AGV transportation actions.

studies have mainly simultaneously focused on loading and unloading operations, and some studies have ignored conflicts during vehicle operation. In studies considering conflict, the travel paths are unidirectional, there are few optional paths for vehicles, and heuristic algorithms are usually proposed to solve the problem.

In view of the above discussions, this paper studies the AGV scheduling problem with bidirectional paths, and there are multiple optional routes between any two processing locations (HPs of blocks and QCs). The operation sequence of each QC and the storage location of each container are assumed to be known in this paper. The main decisions include the job assignment to each AGV, the job sequence on the AGV, the AGV transportation route, the action sequence of the AGVs to avoid conflict, and the timing of each event. The job assignment and the job sequence indicate which container the AGV transports and the sequence in which the AGV transports the containers, respectively. The AGV transportation route refers to the travel paths of the AGV, and the action sequence of the AGVs determines the time sequence of AGV movements, which is used to avoid conflict when two AGVs are located on the same path. A MIP model is presented for the problem to minimize the makespan, and a tailored branch-and-bound algorithm is developed to solve the problem.

Def of 5 Decisions

3. Problem description

The equipment considered in this paper for the **ADRP** includes QCs, AGVs, and AGV-supports. The QCs are responsible for placing containers on the AGV or removing them from the AGV, and the QC operation sequence for containers is known. The AGVs are responsible for the horizontal transportation between the quayside and the yard side in the bidirectional path structure with multiple parallel travel paths and multiple shortcuts. The AGV-supports are located in front of each block, and AGVs can autonomously place the containers on the AGV-supports or pick them up from the AGV-supports, which means that AGVs and AGV-supports can complete the container transfer without ASCs. Therefore, the ASC is not considered in the modeling process.

Def of QC

def of AGV

Def of AGV-support

The layout of the AGV operation area is shown in Fig. 4, which is based on the automated container terminal of Qingdao Qianwan. AGVs deliver the loading and unloading containers simultaneously, while the QC operation lists and container storage locations are given. There are a few AGV-supports for HPs in front of each block. The AGV or ASC places the container on the AGV-support and continues its operation, which can avoid equipment waiting. As shown in Fig. 4, the index of vertical paths is x, which increases from 1 (most left) to x_R (most right), and the index of horizontal paths is y, which ranges from 1 (nearest to yard side) to y_R (furthest to yard side). The boundary between the landside and seaside in the horizontal path is y_r.

Movement rules for AGVs

In this paper, we study the AGV dispatching and routing problem (**ADRP**) at the operational level. The AGV cannot turn multiple times in the seaside and landside operation areas, and the AGV travel paths are bidirectional. Therefore, the operation process of an AGV for a container

Meaning towards the QCs and ASCs, in a vertical path

can be divided into four actions, as shown in Fig. 5 (a). The multiple QC double cycling model is used to reduce the empty-travel distance and the energy consumption, where a loading (unloading) job is completed before an unloading (loading) job (Ku and Arthanari, 2016). In Fig. 5 (a), actions 1, 2 and 3 are AGV loaded travel, while action 4 is AGV empty travel. Action 2 is vertical action on vertical paths, and the remaining three actions are horizontal actions on horizontal paths. Action 1 of the unloading container is located on the AGV seaside operation area (horizontal path, the index is y_{r+1} to y_R), and action 1 of the loading container is located on the AGV landside operation area (horizontal path, the index is 1 to y_r). Actions 3 and 4 of the unloading container are assumed to have the same horizontal path. Fig. 5 (b) and Fig. 5 (c) are two special cases of AGV transportation. The AGV loaded travel distance in Fig. 5 (b) is increased, which rarely occurs. The distance of action 3 in Fig. 5 (c) is zero.

The AGV conflict is the most challenging issue in the AGV routing problem. When the AGVs are arranged on the same path at the same time, they cannot cross each other. The purpose of the AGV routing problem is to avoid AGV conflict and decide which AGV action should be conducted first. The AGV operation sequence and the routing result will affect the timing of all events. To minimize the makespan of all containers, we need to decide on the job assignment to the AGV, the action sequence of the AGVs, the AGV travel path, the container sequence for one AGV, the time of AGV action, and the operation time on the quay-side and the yard side.

decision variables

AGV conflict problem: multiple AGVs having the same route, >> avoiding it

Invovles deciding who's gonna go first

4. Model formulation

In this section, we present a mixed integer programming model for **ADRP** in automated container terminals.

4.1. Assumptions

To facilitate the presentation of the essential ideas without loss of generality, the following assumptions are made in this paper (see e.g., Jiang et al., 2018; Luo and Wu, 2020; Zhong et al., 2020; Li et al., 2020).

(1) The QC operation sequence and container storage locations are given by a higher-level decision.
(2) The QC handling time between two containers can be estimated according to their storage locations. S^Q
(3) The speed of empty and loaded AGVs is known and uniform.
(4) The AGV-supports are set in front of each block to avoid waiting between the ASC and AGV.
(5) AGVs and QCs can only carry one container at a time. not multiloaded
(6) The multiple QC double cycling model is used, which means that a loading (unloading) container is completed before an unloading (loading) container (Ku and Arthanari, 2014; Yang et al., 2018). In the multiple QC double cycling, the operation is carried out between two or more QCs located in separate hatches, first loading by one QC and then unloading by another QC, and the transport of the respective container is handled by the same AGV.

4.2. Notation

The parameters used in the model can be summarized as follows:

m The QC index

(m,i) The container index, which means the i-th job of QC m

(m,i_d) The index of the last job on QC m

l The AGV index

QC operation

(m,i,α) The α - th action of container (m,i), $\alpha \in \{1,2,3,4\}$, which refers to the four actions in the AGV transportation process, as shown in Fig. 5

B The set of AGVs

D, L The sets of unloading and loading containers

C The set of all containers, $C = D \cup L$

M A very large positive number

AGV Actions

W^V The set of AGV vertical actions, $W^V = \{(m,i,\alpha)|(m,i) \in C, \alpha \in \{2\}\}$

W^H The set of AGV horizontal actions, $W^H = \{(m,i,\alpha)|(m,i) \in C, \alpha \in \{1,3,4\}\}$

W^T The set of AGV actions, $W^T = W^H \cup W^V$

X^R The set of vertical paths in the AGV operation area, $X^R = \{1,2,\ldots,x_R\}$

Y^R The set of horizontal paths in the AGV operation area, $Y^R = \{1,2,\ldots,y_R\}$

Y^S The set of horizontal paths in the AGV seaside operation area, $Y^S = \{y_{r+1}, y_{r+1}+1, \ldots, y_R\}$

Y^L The set of horizontal paths in the AGV landside operation area, $Y^L = \{1,2,\ldots,y_r\}$

v^{AGV} The speed of AGVs

$G^Q_{(m,i)}$ The required time for a QC to operate container (m,i)

$G^Y_{(m,i)}$ The required time for an AGV to place (take) container (m,i) on (from) AGV-support

$O_{(m,i)}$ The vertical path of a QC to operate container (m,i), $O_{(m,i)} \in X^R$

$A^L_{(m,i)}$ The left-most vertical path of the block that stored container (m,i) \in X^R

Does QCs move or are they immobile?

$A^R_{(m,i)}$ The right-most vertical path of the block that stored container (m,i) \in X^R

$S^Q_{(m,i)(m,i+1)}$ The required time for a QC to switch from container (m,i) to $(m,i+1)$

These parts where it can be improved!

ψ_1, ψ_2 The sets of container pairs (m,i,n,j) that (m,i) must precede (n,j), where ψ_1 is for QC operation and ψ_2 is for ASC operation.

The decision variables in the model are as follows:

$Z_{(m,i)(n,j),l} = 1$ if container (n,j) is conducted immediately after (m,i) and both are assigned to AGV l; $= 0$ otherwise. Note that (m,i) and (n,j) are not equal

Assignment decision variable

$U^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)} = 1$ if action (m,i,α_1) is conducted before (n,j,α_2); $= 0$ otherwise. Note that (m,i,α_1) and (n,j,α_2) are not equal

for conflict freeness!

$U^{QC}_{(m,i)(n,j,\alpha)} = 1$ if the QC operation of container (m,i) and horizontal action (n,j,α) are conducted simultaneously; $= 0$ otherwise

$P^X_{(m,i,\alpha),x} = 1$ if the finishing location of action (m,i,α) is on the vertical path x; $= 0$ otherwise

$P^Y_{(m,i,\alpha),y} = 1$ if the finishing location of action (m,i,α) is on the horizontal path y; $= 0$ otherwise

$P^X_{(m,i,0),x} = 1$ if the starting location of container (m,i) (i.e., the starting location of action $(m,i,1)$) is on the vertical path x; $= 0$ otherwise

$P^Y_{(m,i,0),y} = 1$ if the starting location of container (m,i) is on the horizontal path y; $= 0$ otherwise. Note that the $(m,i,\alpha), (m,i) \in C, \alpha \in \{0,1,2,3,4\}$ in variables $P^X_{(m,i,\alpha),x}$, $P^Y_{(m,i,\alpha),y}$ and $t^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)}$ indicate the starting or finishing locations of AGV actions, and the $(m,i,\alpha), (m,i) \in C, \alpha \in \{1,2,3,4\}$ in the remaining variables represents the AGV actions.

$T^Q_{(m,i)}$ The starting time for a QC to operate container (m,i)

$T^Y_{(m,i)}$ The starting time for an AGV to place (take) container (m,i) on (from) AGV-support

$T^{Start}_{(m,i,\alpha)}$ The starting time for an AGV to conduct action (m,i,α)

We introduce the following intermediate variables:

$t^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)}$ The required time for an AGV to switch from the finishing location of an action (m,i,α_1) to the finishing location of another action (n,j,α_2). This can be obtained from decision variables $P^X_{(m,i,\alpha),x}$ and $P^Y_{(m,i,\alpha),y}$. Note that $(m,i,0)$ and $(n,j,0)$ in this variable indicate the starting locations of containers (m,i) and (n,j) (i.e., the starting location of actions $(m,i,1)$ and $(n,j,1)$).

$X^{position}_{(m,i,\alpha)}$ The vertical path where the finishing location of action (m,i,α) is located

$Y^{position}_{(m,i,\alpha)}$ The horizontal path where the finishing location of action (m, i, α) is located. Note that variables $X^{position}_{(m,i,\alpha)}$ and $Y^{position}_{(m,i,\alpha)}$ are used to link variables $P^{X}_{(m,i,\alpha),x}$ and $P^{Y}_{(m,i,\alpha),y}$ with $t^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)}$, as shown in constraints (37) to (39), and $(m, i, 0)$ in those two variables indicates the starting location of container (m, i).

4.3. Mathematical model

Def of makespan

The object of the model is to minimize the makespan of all containers, i.e., the time when the last container has been completed, as shown in objective function (1). The finishing time is $T^{Q}_{(m,i_d)} + G^{Q}_{(m,i_d)}$ if the last container (m, i_d) of QC m is a loading job; otherwise, the makespan is $T^{Y}_{(m,i_d)} + G^{Y}_{(m,i_d)}$.

کمینه نمودن طولانی ترین کار کانتینری=تابع هدف ما

[**ADRP model**] Min : $\max_{m}\left\{T^{Q}_{(m,i_d)} + G^{Q}_{(m,i_d)}, T^{Y}_{(m,i_d)} + G^{Y}_{(m,i_d)}\right\}$ (1)

The constraints of the model can be divided into five groups as follows: the job assignment constraints, the location constraints of AGV actions, the conflict-free constraints of AGV actions, the time constraints, and the constraints on the range of variables. The explanations of the five groups and the constraints of this model are shown as follows.

Every feasible solution in here is a usable in cnstr\-8 to 46

(i) **Job assignment constraints.** The job assignment constraints determine which container the AGV transports and the sequence in which the AGV transports the containers, and also ensure that the multiple QC double cycling model is adopted. Constraint (2) ensures that each container is handled by one AGV, where 0 is a virtual node. Constraint (3) ensures the operation continuity of the AGVs, which means that each container has either one predecessor or one successor for AGV operation. Constraint (4) ensures that each AGV must start its operation from the virtual node. Constraint (5) ensures that each AGV must finish its operation at the virtual node. Constraints (6) and (7) ensure that the multiple QC double cycling model is used for the AGV operation.

$$\sum_{l\in B}\sum_{(n,j)\in C\cup\{0\}} Z_{(m,i)(n,j),l} = 1, \forall (m,i) \in C \quad (2)$$

This is and in Condition in gams

$$\sum_{(n,j)\in C\cup\{0\}} Z_{(n,j)(m,i),l} = \sum_{(h,k)\in C\cup\{0\}} Z_{(m,i)(h,k),l}, \forall (m,i) \in C, \forall l \in B \quad (3)$$

$$\sum_{(m,i)\in C} Z_{0(m,i),l} = 1, \forall l \in B \quad (4)$$

$$\sum_{(m,i)\in C} Z_{(m,i)0,l} = 1, \forall l \in B \quad (5)$$

$$\sum_{l\in B}\sum_{(n,j)\in D\cup\{0\}} Z_{(m,i)(n,j),l} = 1, \forall (m,i) \in L \quad (6)$$

$$\sum_{l\in B}\sum_{(n,j)\in L\cup\{0\}} Z_{(m,i)(n,j),l} = 1, \forall (m,i) \in D \quad (7)$$

(ii) **Location constraints of AGV actions.** The location constraints of AGV actions determine the location of the AGV during its transportation. The container storage locations are known in this model, therefore the vertical path where the QC operation location of each container is located is given, and the vertical path range of the yard side operation location of each container is also known. The starting location of the unloading container for AGV transportation is on the quayside and the finishing location is on the yard side, and the loading container is the reverse of the unloading container, which means that the location constraints for the unloading and loading containers are also different. The location constraints of AGV actions can be explained as follows.

Constraints (8) and (9) indicate that if container (m, i) is the preceding job of (n, j) transported by the same AGV, the finishing location of (m, i) is the same as the starting location of (n, j). Constraint (8) is for vertical path and constraint (9) is for horizontal path. Constraints (10) and (11) ensure that there can only be one location for AGV action, including the starting location of action $(m, i, 1)$ and the finishing locations of actions $(m, i, 1)$ to $(m, i, 4)$. Constraint (12) indicates that the starting horizontal path of the loading container is in the AGV landside operation area. Constraint (13) indicates that the starting horizontal path of the unloading container is in the AGV seaside operation area. Constraint (14) indicates that the starting vertical path of the unloading container is the vertical path of the QC operation location. Constraint (15) indicates that the starting vertical path of the loading container is the vertical path corresponding to the block that stored container (m, i). Constraint (16) indicates that the finishing horizontal path of the unloading container is in the AGV landside operation area. Constraint (17) indicates that the finishing horizontal path of the loading container is in the AGV seaside operation area. Constraint (18) indicates that the finishing vertical path of the loading container is the vertical path of the QC operation location. Constraint (19) indicates that the finishing vertical path of the unloading container is the vertical path corresponding to the block that stored container (m, i). Constraints (20) and (21) ensure the locational continuity of the AGV transportation. Constraint (20) ensures that the starting location and finishing location of a horizontal action are on the same horizontal path. Constraint (21) ensures that the starting location and finishing location of a vertical action are on the same vertical path.

There is a very serious problem here, that is why z_{(m,i)(n,j)l} is used?? these are our decision variables!

$$P^{X}_{(m,i,4),x} = P^{X}_{(n,j,0),x}, \sum_{l\in B} Z_{(m,i)(n,j),l} = 1, \forall (m,i),(n,j) \in C, \forall x \in X^{R} \quad (8)$$

$$P^{Y}_{(m,i,4),y} = P^{Y}_{(n,j,0),y}, \sum_{l\in B} Z_{(m,i)(n,j),l} = 1, \forall (m,i),(n,j) \in C, \forall y \in Y^{R} \quad (9)$$

wherever you see an agv action, you must check wt, not C!

$$\sum_{x\in X^{R}} P^{X}_{(m,i,\alpha),x} = 1, \forall (m,i) \in C, \forall \alpha \in \{0,1,2,3,4\} \quad (10)$$

$$\sum_{y\in Y^{R}} P^{Y}_{(m,i,\alpha),y} = 1, \forall (m,i) \in C, \forall \alpha \in \{0,1,2,3,4\} \quad (11)$$

$$\sum_{y\in Y^{L}} P^{Y}_{(m,i,0),y} = 1, \forall (m,i) \in L \quad (12)$$

$$\sum_{y\in Y^{S}} P^{Y}_{(m,i,0),y} = 1, \forall (m,i) \in D \quad (13)$$

$$P^{X}_{(m,i,0),O_{(m,i)}} = 1, \quad \forall (m,i) \in D \quad (14)$$

It is a bit changed in the GAMS, Look out!

$$\sum_{x=A^{L}_{(m,i)}}^{A^{R}_{(m,i)}} P^{X}_{(m,i,0),x} = 1, \forall (m,i) \in L \quad (15)$$

$$\sum_{y\in Y^{L}} P^{Y}_{(m,i,3),y} = 1, \forall (m,i) \in D \quad (16)$$

$$\sum_{y\in Y^{S}} P^{Y}_{(m,i,3),y} = 1, \forall (m,i) \in L \quad (17)$$

$$P^{X}_{(m,i,3),O_{(m,i)}} = 1, \forall (m,i) \in L \quad (18)$$

$$\sum_{x=A^{L}_{(m,i)}}^{A^{R}_{(m,i)}} P^{X}_{(m,i,3),x} = 1, \forall (m,i) \in D \quad (19)$$

According to the onConflicts.pdf, (a) is deadlock and (b) is livelock

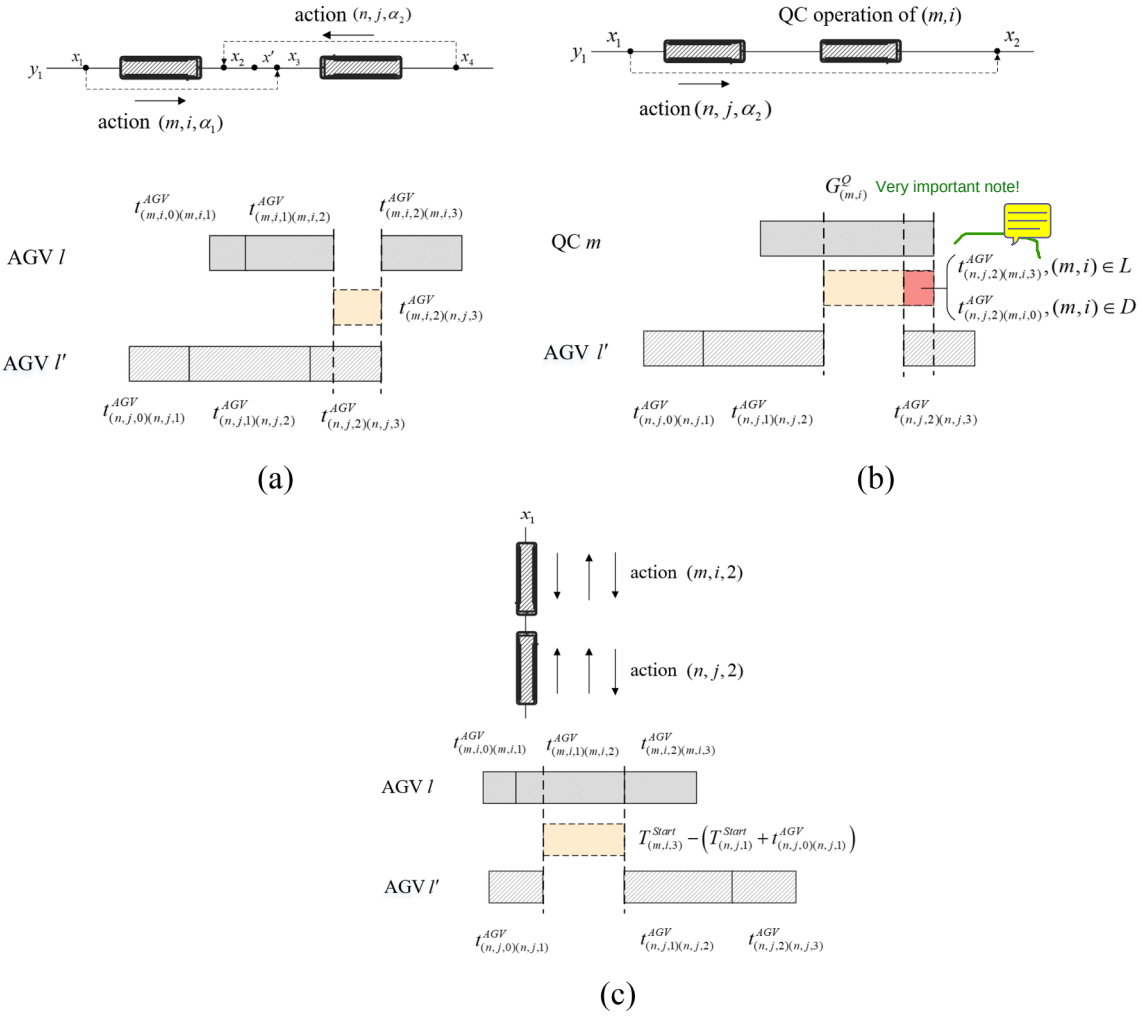


Fig. 6. Examples of AGV conflicts. Notes: (a) the opposite direction conflict of different AGVs on the same horizontal path; (b) the conflict between the AGV quayside operation and the AGV transportation on the same horizontal path; (c) the vertical path conflict of different AGVs.

$$P^{Y}_{(m,i,\alpha),y} = P^{Y}_{(m,i,\alpha-1),y}, \forall(m,i,\alpha)\in W^{H}, \forall y\in Y^{R} \quad (20)$$

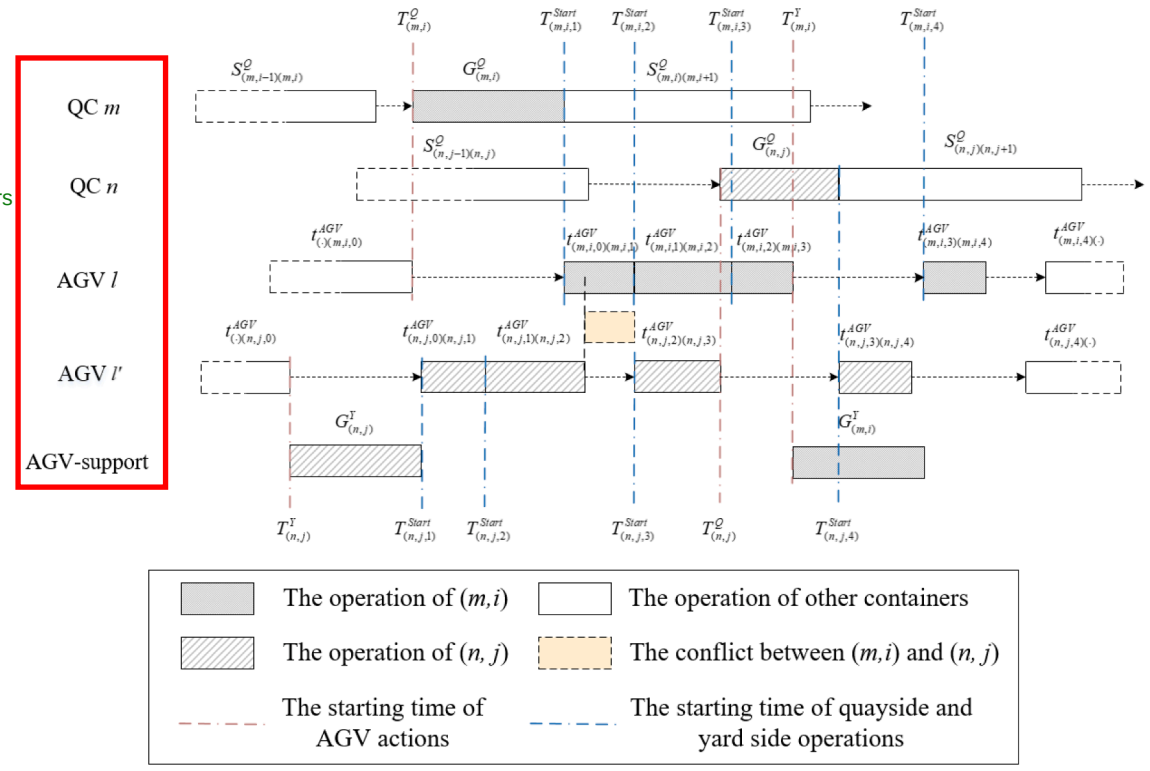
$$P^{X}_{(m,i,\alpha),x} = P^{X}_{(m,i,\alpha-1),x}, \forall(m,i,\alpha)\in W^{V}, \forall x\in X^{R} \quad (21)$$

(iii) **Conflict-free constraints of AGV actions.** The conflict-free constraints of AGV actions avoid conflicts during AGV transportation, and the core constraints include constraints (23), (26), and (27), which avoid the opposite direction conflict, the conflict between the AGV quayside operation and the AGV transportation on the same horizontal path, and the vertical path conflict respectively, as shown in Fig. 6. The conflict will occur when two AGV actions are completed by different AGVs at the same time and are located in the same path. The conflict-free constraints of AGV actions can be explained as follows. Constraint (22) indicates that if container (m,i) is the preceding job of (n,j), the finishing action of (m,i) must be conducted before the starting action of (n,j). Constraint (23) avoids the opposite direction conflict of different AGVs on the same horizontal path, as shown in Fig. 6 (a). The explanations of constraint (23) and Fig. 6 (a) are presented in **Appendix A**. Constraints (24) to (26) avoid the conflict between the AGV quayside operation and the AGV transportation on the same horizontal path, as shown in Fig. 6 (b). This situation only occurs in the AGV seaside operation area, where one AGV is waiting for the quayside operation of another AGV ahead. Variable $U^{QC}_{(m,i)(n,j,\alpha)}$ appears only in constraints (24) to (26), and is used to indicate the time relationship between the QC operation at the quayside and the AGV action. Constraints (24) and (25) define the variable $U^{QC}_{(m,i)(n,j,\alpha)}$, which models the time relationship between the quayside operation of container (m,i) and the horizontal action (n,j,α) when they are conducted at the same time. The explanations of constraint (26) and Fig. 6 (b) are presented in **Appendix B**. Constraint (27) avoids the vertical path conflict of different AGVs, which ensures that two vertical actions conducted by different AGVs cannot simultaneously be on the same vertical path, as shown in Fig. 6 (c). In Fig. 6 (c), actions $(m,i,2)$ and $(n,j,2)$ conflict in the same vertical path x_1, and the waiting time for action $(n,j,2)$ is $T^{Start}_{(m,i,3)}-\left(T^{Start}_{(n,j,1)}+t^{AGV}_{(n,j,0)(n,j,1)}\right)$. Constraint (28) ensures the action sequence of a container.

When Conflict occurs

$$U^{AGV}_{(m,i,4)(n,j,1)} \geq \sum_{l\in B} Z_{(m,i)(n,j),l}, \forall(m,i),(n,j)\in C \quad (22)$$

3 essential transporters
Of this problem



Gantt Chart

Fig. 7. An illustrative example of the sequence and the timing of two exemplary containers.

take a look at the constraint 23 in GAMS for dealing with this

This constraint is making the model HUGE!!

$$U^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)} + U^{AGV}_{(n,j,\alpha_2)(m,i,\alpha_1)} + 3 - P^{Y}_{(m,i,\alpha_1),y} - P^{Y}_{(n,j,\alpha_2),y} - \sum_{x=1}^{x'} \left(P^{X}_{(m,i,\alpha_1-1),x} + P^{X}_{(n,j,\alpha_2),x} - P^{X}_{(m,i,\alpha_1),x} - P^{X}_{(n,j,\alpha_2-1),x} \right) \geq 0, \quad (23)$$

For each of these, there should be a wt(...) corresponding to it!

$$\forall (m,i,\alpha_1),(n,j,\alpha_2) \in W^H, \forall y \in Y^R, \forall x' \in X^R$$

every time a-1 is used without summation, this code should be used in equation declaration: `ord(a1_1)<ord(a1)`
when used inside summation, that code should be written after `..`

$$T^{Q}_{(m,i)} + G^{Q}_{(m,i)} + M\left(1 - U^{QC}_{(m,i)(n,j,\alpha)}\right) \geq T^{Start}_{(n,j,\alpha)}, \forall (m,i) \in C, \forall (n,j,\alpha) \in W^H \quad (24)$$

Big -M method

$$T^{Start}_{(n,j,\alpha)} + t^{AGV}_{(n,j,\alpha-1)(n,j,\alpha)} + M\left(1 - U^{QC}_{(m,i)(n,j,\alpha)}\right) \geq T^{Q}_{(m,i)}, \forall (m,i) \in C, \quad \forall (n,j,\alpha) \in W^H \quad (25)$$

$$U^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)} + U^{AGV}_{(n,j,\alpha_2)(m,i,\alpha_1)} \geq P^{X}_{(m,i,\alpha_1),x} + P^{X}_{(n,j,\alpha_2),x} - 1, \forall (m,i,\alpha_1),(n,j,\alpha_2) \in W^V, \forall x \in X^R \quad (27)$$

$$U^{AGV}_{(m,i,\alpha-1)(m,i,\alpha)} = 1, \forall (m,i) \in C, \forall \alpha \in \{2,3,4\} \quad (28)$$

$= t$

$$\left(3 - U^{QC}_{(m,i)(n,j,\alpha_2)} - P^{Y}_{(m,i,\alpha_1),y} - P^{Y}_{(n,j,\alpha_2),y} + \left\| \sum_{x=1}^{O_{(m,i)}} P^{X}_{(n,j,\alpha_2),x} - \sum_{x=O_{(m,i)}+1}^{x_R} P^{X}_{(n,j,\alpha_2-1),x} \right\| \right) M + T^{Start}_{(n,j,\alpha_2)} + t^{AGV}_{(n,j,\alpha_2-1)(m,i,\alpha_1)} \geq T^{Q}_{(m,i)} + G^{Q}_{(m,i)}, \quad (26)$$

$$\forall (n,j,\alpha_2) \in W^H, \forall y \in Y^S, \forall (m,i) \in D \; \forall \alpha_1 \in \{0\} \; or \forall (m,i) \in L \forall \alpha_1 \in \{3\}$$

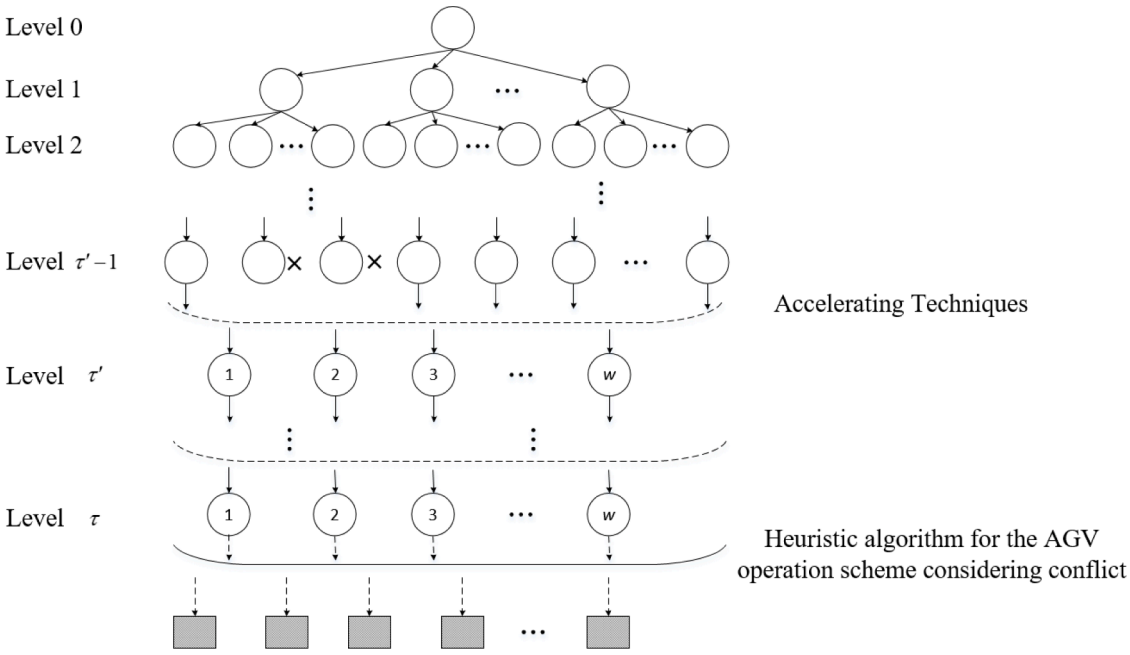


Fig. 8. The structure of the B&B algorithm.

(iv) **Time constraints.** The time constraints ensure the time relationship of the QC and the AGV to operate containers. Constraint (29) ensures the time relationship of the QC operation, and constraints (30), (31), (34) to (36) ensure the time relationship of the AGV to transport a container. Constraints (32) and (33) ensure the time relationship of the AGV to transport two consecutive containers. Fig. 7 shows the time relationship between the variables, which can be used to understand the time constraints. In Fig. 7, (m,i) is an unloading container and (n,j) is a loading container. The time constraints of AGV actions can be explained as follows. Constraint (29) ensures that the quayside operation is under the given job sequence for each QC. Constraint (30) ensures the loading and unloading sequences of containers on the ship for QC operation, and constraint (31) ensures the sequences of containers in the yard for ASC operation. Note that containers (m,i) and (n,j) in set ψ_2 are located in the same block. Constraints (32) and (36) ensure the minimum time gap between the yard side operation and AGV action for the AGV to transport a container. Constraints (33) and (37) ensure the minimum time gap between the quayside operation and AGV action for the AGV to transport a container. Constraints (34) and (35) ensure the minimum time gap between the AGV action and quayside (or yard side) operation for the AGV to transport two consecutive containers. Constraint (38) models the minimum time gap between two sequential actions. Constraints (39) to (41) model the relationship between intermediate variables and decision variables, which are used to link variables $P^X_{(m,i,\alpha),x}$ and $P^Y_{(m,i,\alpha),y}$ with $t^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)}$.

$$T^{Q}_{(m,i+1)} \geq T^{Q}_{(m,i)} + G^{Q}_{(m,i)} + S^{Q}_{(m,i)(m,i+1)}, \forall (m,i),(m,i+1) \in C \quad \text{QC} \quad (29)$$

$$T^{Q}_{(n,j)} \geq T^{Q}_{(m,i)} + G^{Q}_{(m,i)}, \forall (m,i,n,j) \in \psi_1 \quad \text{QC} \quad (30)$$

$$T^{Y}_{(n,j)} \geq T^{Y}_{(m,i)} + G^{Y}_{(m,i)}, \forall (m,i,n,j) \in \psi_2 \quad \text{AGV and ASC} \quad (31)$$

$$T^{Y}_{(m,i)} \geq T^{Start}_{(m,i,3)} + t^{AGV}_{(m,i,2)(m,i,3)}, \forall (m,i) \in D \quad \text{AGV} \quad (32)$$

$$T^{Q}_{(m,i)} \geq T^{Start}_{(m,i,3)} + t^{AGV}_{(m,i,2)(m,i,3)}, \forall (m,i) \in L \quad \text{AGV} \quad (33)$$

$$T^{Y}_{(n,j)} + M\left(1 - \sum_{l \in B} Z_{(m,i)(n,j),l}\right) \geq T^{Start}_{(m,i,4)} + t^{AGV}_{(m,i,3)(m,i,4)}, \forall (m,i) \in D, \forall (n,j) \in L \quad \text{AGV} \quad (34)$$

$$T^{Q}_{(n,j)} + M\left(1 - \sum_{l \in B} Z_{(m,i)(n,j),l}\right) \geq T^{Start}_{(m,i,4)} + t^{AGV}_{(m,i,3)(m,i,4)}, \forall (m,i) \in L, \forall (n,j) \in D \quad \text{AGV} \quad (35)$$

$$T^{Start}_{(m,i,\alpha)} \geq T^{Y}_{(m,i)} + G^{Y}_{(m,i)}, \forall (m,i) \in D, \alpha \in \{4\} \text{ or } \forall (m,i) \in L, \alpha \in \{1\} \quad \text{AGV} \quad (36)$$

$$T^{Start}_{(m,i,\alpha)} \geq T^{Q}_{(m,i)} + G^{Q}_{(m,i)}, \forall (m,i) \in D, \alpha \in \{1\} \text{ or } \forall (m,i) \in L, \alpha \in \{4\} \quad \text{Quay side AGV Op} \quad (37)$$

$$T^{Start}_{(n,j,\alpha_2)} + M\left(1 - U^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)}\right) \geq T^{Start}_{(m,i,\alpha_1)} + t^{AGV}_{(m,i,\alpha_1-1)(m,i,\alpha_1)}, \forall (m,i,\alpha_1),(n,j,\alpha_2) \in W^T \quad (38)$$

These are not constraint, these are `.l` relations.

$$X^{position}_{(m,i,\alpha)} = x, P^{X}_{(m,i,\alpha),x} = 1, \forall (m,i) \in C, \forall \alpha \in \{0,1,2,3,4\}, \forall x \in X^R \quad \text{(P_X and P_Y) <=> t_AGV} \quad (39)$$

$$Y^{position}_{(m,i,\alpha)} = y, P^{Y}_{(m,i,\alpha),y} = 1, \forall (m,i) \in C, \forall \alpha \in \{0,1,2,3,4\}, \forall y \in Y^R \quad \text{(P_X and P_Y) <=> t_AGV} \quad (40)$$

★ Each member of the scheme is an assignment.

$$t^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)} = (\left|X^{position}_{(m,i,\alpha_1)} - X^{position}_{(n,j,\alpha_2)}\right| + \left|Y^{position}_{(m,i,\alpha_1)} - Y^{position}_{(n,j,\alpha_2)}\right|)/v^{AGV}, \quad \forall (m,i),(n,j) \in C, \forall \alpha_1,\alpha_2 \in \{0,1,2,3,4\} \tag{41}$$

(P_X and P_Y) <=> t_AGV

$$Z_{(m,i)(n,j),l} \in \{0,1\}, \forall (m,i),(n,j) \in C, \forall l \in B \tag{42}$$

$$U^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)} \in \{0,1\}, \forall (m,i,\alpha_1),(n,j,\alpha_2) \in W^T \tag{43}$$

$$U^{QC}_{(m,i)(n,j,\alpha)} \in \{0,1\}, \forall (m,i) \in C, (n,j,\alpha) \in W^H \tag{44}$$

$$P^X_{(m,i,\alpha),x}, P^Y_{(m,i,\alpha),y} \in \{0,1\}, \forall (m,i) \in C, \forall \alpha \in \{0,1,2,3,4\}, \forall x \in X^R, \forall y \in Y^R \tag{45}$$

$$T^Q_{(m,i)}, T^Y_{(m,i)}, T^{Start}_{(m,i,\alpha)} \geq 0, \forall (m,i) \in C, \forall \alpha \in \{1,2,3,4\} \tag{46}$$

(v) **Constraints on the range of variables.** Constraints (42) to (46) are the nonnegativity and binary constraints for the decision variables.

For a better demonstration of the modeling concepts, a feasible solution example in Fig. 7 can be used. In the example, the sequence and timing of all events are shown. The time points of quayside operations, yard side operations, and AGV actions for containers (m,i) and (n,j) are also marked. The modeling concept to avoid AGV conflict is to control the location, sequence, and timing of the AGV actions. As shown in Fig. 7, an unloading container (m,i) is assigned to AGV l, and a loading container (n,j) is assigned to AGV l'. Each container induces four actions (i.e., $(m,i,1)$ to $(m,i,4)$ and $(n,j,1)$ to $(n,j,4)$), where the second actions are vertical actions and all others are horizontal actions. An opposite direction conflict occurs between actions $(m,i,1)$ and $(n,j,3)$ in Fig. 7, and the values of the decision variables for this example are given in **Appendix C**.

5. Solution approach

The mixed integer programming formulation of the **ADRP** is extremely unwieldy and complex. When the size of the instances becomes large, the problem cannot be directly solved in polynomial time using commercial solvers, such as Gurobi. Therefore, a tailored branch-and-bound (B&B) algorithm is proposed, as shown in Fig. 8. Generally, the branch-and-bound algorithm is a search method that systematically enumerates candidate solutions (Tomazella and Nagano, 2020; Bai et al. 2017). To reduce the computational time, accelerating techniques are used in the B&B algorithm.

Why is B&B unique & cool!!

★

ابعاد مسئله

The size of the solution space mainly depends on the number of containers under consideration. To reduce the solution space, the B&B algorithm solves the problem level by level instead of solving all containers at once. In Fig. 8, level 0 contains an initial node. For level τ' from level 1 to level τ, each node contains a solution of τ' containers without considering conflict, that is, partial job assignment and job sequence of AGVs. Each node in level τ contains a complete solution for all containers without considering conflict. The heuristic algorithm for the AGV operation scheme considering conflict in section 5.4 is used for nodes in level τ to obtain conflict-free routes. Note that the route of each container mentioned below includes four actions (as shown in Fig. 5 (a)), and the four actions are processed in sequence during conflict detection in section 5.4.

The overall framework of the proposed B&B algorithm is illustrated in **Algorithm 1**. Note that parameters v, $N_{\tau'}$, $C_{\tau',v}$, and $C'_{\tau',v}$ are the node index, the set of nodes in level τ', the set of assigned containers for node v in level τ', and the set of containers to be assigned for node v in level τ', respectively.

Algorithm 1 B&B algorithm

01: data: data file, B- the set of AGVs, C- the set of containers, C' - the set of containers to be assigned, i.e., the current jobs in the QC operation lists, ψ_1, ψ_2- the sets of container pairs (m,i,n,j) that (m,i) must precede (n,j)

This is where we need "Good Data" from Job Generator!

02: **Result:** Ω^*- solution
03: **procedure**
04: generate the upper bound T^{UB} based on the algorithm proposed in section 5.1
05: generate the scheme set P
06: accelerating technique 1 is used to remove the repeated schemes in set P
07: $v = 0, N_1 \leftarrow \varnothing$
08: **for** $p \in P$ **do** # branch from level 0 to level 1
09: **for** $l \in B$ **do**
10: **for** $(m,i) \in C'$ **do**
11: $v = v + 1$
12: $C_{1,v} = \{(m,i)\}$, $N_1 = N_1 \cup \{v\}$, update $C'_{1,v}$
13: (m,i) is assigned to AGV l, $scheme(v) = p$
14: **end for**
15: **end for**
16: **end for**
17: **for** $\tau' = 2 \rightarrow \tau$ **do** # branch from level 1 to level τ
18: **if** $|N_{\tau'-1}| > w$ **do** #w is the filter width
19: accelerating technique 4 is used to filter out nodes with poor performances
20: **end if**
21: **for** $v' \in N_{\tau'-1}$ **do**
22: **for** $l \in B$ **do**
23: $num(l) \leftarrow$ the number of jobs assigned to AGV l
24: **if** $num(l)$ reaches the maximum according to $scheme(v')$ **do**
25: **continue**
26: **end if**
27: **for** $(m,i) \in C'_{\tau'-1,v'}$ **do**
28: **if** the assignment of container (m,i) and AGV l conflicts with ψ_1, ψ_2 **do**
29: **continue**
30: **end if**
31: pruning rules 1 and 2 are used to disregard infeasible nodes
32: accelerating techniques 2 and 3 are used to disregard repeated nodes
33: compute the lower bound $T^{LB}_{\tau',v}$ (section 5.3)
34: **if** $T^{LB}_{\tau',v} \geq T^{UB}$ **do**
35: **continue** #disregard the node with larger lower bound
36: **end if**
37: $v = v + 1$
38: $C_{\tau',v} = C_{\tau'-1,v'} \cup \{(m,i)\}$, $N_{\tau'} = N_{\tau'} \cup \{v\}$, update $C'_{\tau',v}$
39: (m,i) is assigned to AGV l, $scheme(v) = scheme(v')$
40: **end for**
41: **end for**
42: **end for**
43: **end for**
44: the heuristic algorithm in section 5.4 is used to generate the final solution Ω^*
45: **return** Ω^*
46: **end procedure**

B&B 1st level

B&B 2nd level+

level 1 and v= node index
=> there is 1 container assigned to node v of level 1

So, scheme(v) is just a way of making sure that the job assigned to the AGV does cause any max-assigned-Container problems

This is where children of v' are born!
And branching strategy is used!

transferring the scheme to the next level,
v (before the addition in l.37) is in the last node from level / tau'
all of the child of v' have the same scheme(v')

5.1. *Upper bound* This deals with Min makespan (hence TIME)

In the B&B algorithm, the upper bound (UB) represents the known feasible solution that has the minimum makespan. The use of a UB allows the B&B algorithm to prune active nodes with larger lower bound values, reducing the size of the search region and shortening the computer processing time. Therefore, a higher-precision upper bound can prune more nodes that cannot find a better feasible solution, thereby improving the efficiency of the algorithm. Before starting the B&B algorithm, we propose a heuristic algorithm to find a UB, as shown in **Algorithm 2**.

In **Algorithm 2**, to ensure that AGV l^* and container $(m,i)^*$ can be obtained (rows 06 through 22), the assignment must be compatible with the multiple QC double cycling model and the sets ψ_1, ψ_2. If the selected container $(m,i)^*$ is the first job of AGV l^*, the multiple QC double cycling model is formed (row 12). To obtain the conflict-free route r^* with the shortest operation time from $R_{(m,i)^*}$ (rows 23 through 25), the heuristic algorithm in section 5.4 is used. The sets are updated in row 27: delete container $(m,i)^*$ from set C, update set C' based on the QC operation lists, and update the time for AGVs to complete their assigned containers. The

upper bound obtained with **Algorithm 2** will be used in the B&B algorithm.

Algorithm 2 Upper bound heuristic algorithm

data: B- the set of AGVs, C- the set of containers, C' - the set of containers to be assigned, i.e., the current jobs in the QC operation lists, $R_{(m,i)}$- the set of optional routes for container (m,i), sorted by operation time from smallest to largest, ψ_1, ψ_2- the sets of container pairs (m,i,n,j) that (m,i) must precede (n,j), M- a very large positive number
Result: Φ- initial solution
procedure
  $\Phi \leftarrow \varnothing$
  $T(l) \leftarrow 0, \forall l \in B$ # the time for AGV l to complete current container operation
  while $C \neq \varnothing$ **do**
    $index = 0$ # equal to 1 if an AGV and container are found
    while $index = 0$ **do**
      $l^* = \underset{l \in B}{argmin} T(l)$
      for $(m,i) \in C'$ **do**
        if the assignment of container (m,i) and AGV l^* is compatible with ψ_1, ψ_2 **do**
          if (m,i) can form multiple QC double cycling with AGV l^* **do**
            $index = 1$
            $(m,i)^* = (m,i)$
            break # the AGV l^* and container $(m,i)^*$ are selected
          end if
        end if
      end for
      if $index = 0$ **do**
        $T(l^*) = M$
      end if
    end while
    for $r \in R_{(m,i)^*}$ **do**
      based on the heuristic in section 5.4, generate the route r^* and the time list h^* of each event with the shortest operation time that are compatible with Φ.
      end for
      $\Phi = \Phi \cup ((m,i)^*, l^*, r^*, h^*)$
      update $C, C', T(l)$
    end while
    return Φ
end procedure

5.2. Branching and pruning strategy

The branching and pruning strategies of the B&B algorithm are as follows.

5.2.1. Branching strategy

The branching strategy defines how new nodes are generated from existing nodes. In this problem, an infeasible solution will appear if the AGV operation sequence conflicts with the QC operation sequence. For example, two AGVs transport four containers of two QCs, where (QC m, CON i) represents the i-th job of QC m. The conflict between the AGVs and QCs occurs when the containers are conducted as follows: AGV 1 = [(QC 1, CON 2), (QC 2, CON 1)] AGV 2 = [(QC 2, CON 2), (QC 1, CON 1)]. To avoid conflicts and traverse all AGV operation situations, we designed a tailored branching strategy. The branching strategy first generates the set of container numbers for AGV operation and then traverses the set of AGVs and the set of containers to be assigned. The container is assigned to the AGV based on the scheme of the container numbers for AGV operation, and the details are described as follows.

The nodes in the B&B algorithm contain information of container numbers for AGV operation, which will be transferred from the upper node to the lower node. Let T^{UB} be the UB value. Let t^{min} be the shortest operation time for containers in set C without considering conflict. Let q be the maximum number of containers conducted by each AGV when the completion time does not exceed T^{UB}.

$$maxq$$

$$q \cdot t^{min} \leq T^{UB}$$

According to $|B|$ AGVs, $|C|$ containers and parameter q, the set P of the container numbers for AGV operation is generated. For example, if $|B| = 2$, $|C| = 4$ and $q = 3$, then set $P = \{(1,3), (2,2), (3,1)\}$. Each element in the set P represents the number of containers conducted by AGV 1 and AGV 2. The branching process is shown in **Algorithm 1**.

When the nodes branch from level 0 to level 1, the algorithm traverses the scheme set P, the AGV set B and the set of containers to be assigned C'. Scheme $p \in P$, AGV $l \in B$ and container $(m,i) \in C'$ corresponding to node v are generated. The node set $N_1 = N_1 \cup \{v\}$ is updated in level 1. The set of assigned containers $C_{1,v}$ and the set of containers to be assigned $C'_{1,v}$ are generated for node v in level 1.

When the nodes branch from level $\tau' - 1$ to level τ' $(2 \leq \tau' \leq \tau)$, the algorithm traverses the node set $N_{\tau'-1}$, the AGV set B and the container set $C'_{\tau'-1,v}$. The AGV $l \in B$ and the container $(m,i) \in C'_{\tau'-1,v}$ that comply with scheme p are selected to generate a new node, where the scheme p is transferred from the node in level $\tau' - 1$ to the node in level τ'. The node set $N_{\tau'}$ in level τ' is updated. The set of assigned containers $C_{\tau',v}$ and the set of containers to be assigned $C'_{\tau',v}$ for node $v \in N_{\tau'}$ in level τ' are generated.

5.2.2. Pruning strategy

During the branching process of the B&B algorithm, we propose three pruning rules that can efficiently disregard numerous nodes and significantly reduce the computational load.

Pruning rule 1 is to disregard the node if the newly assigned container does not comply with the multiple QC double cycling model.

Pruning rule 2 is that after the container of a new node is assigned, if the number of unassigned loading and unloading containers cannot meet the AGV operation of scheme p (i.e., unassigned and assigned containers cannot meet the multiple QC double cycling model), disregard the node. For example, two AGVs transport four containers of two QCs, where the loading and unloading container sets are {(QC 1, CON 1), (QC 2, CON 1)} and {(QC 1, CON 2), (QC 2, CON 2)}, respectively. For scheme $p = (1,3)$, if the job assignment for a node in level 2 is AGV 1 = [(QC 1, CON 1)] and AGV 2 = [(QC 2, CON 1)], then the node is disregarded. This is because two containers need to be assigned to AGV 2, but the unassigned containers (QC 1, CON 2) and (QC 2, CON 2) are both unloading containers, which cannot comply with the multiple QC double cycling model.

The pruning rule 3 is that if the lower bound (LB) is worse than UB, then the node cannot produce a better solution, and the node is disregarded. The details of the LB are shown in section 5.3.

5.3. Lower bound

In the B&B algorithm, if the lower bound of a candidate node dominates the upper bound, then the node is disregarded to efficiently save running time. Therefore, a well-designed lower bound is the core of the algorithm. The LB formula is calculated by relaxing the conflict constraints of the original model. For a candidate node, part of the job assignment and job sequence of AGVs have been determined. The AGV transportation routes are then generated based on **Proposition 1**.

Proposition 1. *If the loaded travel route of each container is the shortest, then the obtained AGV operation scheme is the optimal solution without considering conflict.*

Proof. The proof is presented in **Appendix D**.

For a candidate node v in level τ', part of the job assignment and the job sequence of the AGVs are known, and the shortest loaded travel route is selected for each container. The lower bound model is proposed, and the parameters used in the model can be summarized as follows:

$C_{\tau',v}$ The set of assigned containers for node v in level τ'

$D_{\tau',v}$ The set of assigned unloading containers for node v in level τ'

$L_{\tau',v}$ The set of assigned loading containers for node v in level τ'

$\hat{Z}_{(m,i)(n,j),l}$ The sequence of two containers assigned to the same AGV.

This is given information in a specific node v

b_l The unassigned container number of AGV l, which is determined by scheme p and set $C_{\tau',v}$

$\widehat{t}^{min}$ The shortest operation time for containers in the set $C-C_{\tau',v}$ without considering conflict

$\widehat{t}_{(m,i)}$ The minimum time required to complete all jobs for the AGV that conducts container (m,i), $\widehat{t}_{(m,i)} = \sum_{l\in B}\sum_{(n,j)\in C\cup\{0\}} \widehat{t}^{min} b_l \widehat{Z}_{(n,j)(m,i),l}, \forall (m,i) \in C_{\tau',v}$

$\widehat{t}^{empty}_{(m,i)(n,j)}$ The empty travel time for container (m,i) to (n,j) without considering conflict

$\widehat{t}^{load}_{(m,i)}$ The loaded travel time for container (m,i) without considering conflict

The remaining parameters and the decision variables $T^Q_{(m,i)}$ and $T^Y_{(m,i)}$ have been defined in section 4.2, and the lower bound model (**LB model**) is shown as follows. This is the relaxed problem

$$[\textbf{LB model}]\text{Min}: \max_{(m,i)\in C_{\tau',v}} \left\{T^Q_{(m,i)} + G^Q_{(m,i)} + \widehat{t}_{(m,i)}, T^Y_{(m,i)} + G^Y_{(m,i)} + \widehat{t}_{(m,i)}\right\} \quad (47)$$

$$T^Q_{(m,i+1)} \geq T^Q_{(m,i)} + G^Q_{(m,i)} + S^Q_{(m,i)(m,i+1)}, \forall (m,i),(m,i+1) \in C_{\tau',v} \quad (48)$$

$$T^Y_{(m,i)} \geq T^Q_{(m,i)} + G^Q_{(m,i)} + \widehat{t}^{load}_{(m,i)}, \forall (m,i) \in D_{\tau',v} \quad (49)$$

$$T^Q_{(m,i)} \geq T^Y_{(m,i)} + G^Y_{(m,i)} + \widehat{t}^{load}_{(m,i)}, \forall (m,i) \in L_{\tau',v} \quad (50)$$

$$T^Q_{(n,j)} + M\left(1 - \sum_{l\in B}\widehat{Z}_{(m,i)(n,j),l}\right) \geq T^Q_{(m,i)} + G^Q_{(m,i)} + \widehat{t}^{empty}_{(m,i)(n,j)}, \quad \forall (m,i) \in L_{\tau',v}, \forall (n,j) \in D_{\tau',v} \quad (51)$$

conflict types:

$$T^Y_{(n,j)} + M\left(1 - \sum_{l\in B}\widehat{Z}_{(m,i)(n,j),l}\right) \geq T^Y_{(m,i)} + G^Y_{(m,i)} + \widehat{t}^{empty}_{(m,i)(n,j)}, \quad \forall (m,i) \in D_{\tau',v}, \forall (n,j) \in L_{\tau',v} \quad (52)$$

$$T^Q_{(m,i)}, T^Y_{(m,i)} \geq 0, \forall (m,i) \in C_{\tau',v} \quad (53)$$

The objective function (47) minimizes the completion time of all containers in the node. Constraint (48) ensures that the quayside operation is under the given job sequence for each QC. Constraints (49) and (50) model the minimum time gap between the quayside and yard side operations of each container. Constraints (51) and (52) model the minimum time gap between two consecutive containers assigned to the same AGV. Constraint (53) is the nonnegativity constraint for decision variables. The **LB model** is solved and the LB value of the candidate node is obtained.

5.4. Heuristic algorithm for the AGV operation scheme considering conflict

For the level τ of the B&B algorithm shown in Fig. 8, the job assignment and job sequence of AGVs are known, that is, the parameter $\widehat{Z}_{(m,i)(n,j),l}, \forall (m,i),(n,j) \in C, \forall l \in B$ is known. The next step is to generate a conflict-free scheme, which can be modeled as follows.

$$[\textbf{Conflict} - \textbf{free model}]\text{Min}: \max_{m}\left\{T^Q_{(m,i_d)} + G^Q_{(m,i_d)}, T^Y_{(m,i_d)} + G^Y_{(m,i_d)}\right\} \quad (54)$$

Every container job should be conducted one after another

$$Z_{(m,i)(n,j),l} = \widehat{Z}_{(m,i)(n,j),l}, \forall (m,i),(n,j) \in C, \forall l \in B \quad (55)$$

Constraints (8)-(41) and (43)-(46). ****

When the number of AGVs is small, conflict rarely occurs because there are many optional routes. The possibility that a conflict occurs increases as the number of AGVs increases, which will lead to an expensive computational time. Therefore, we propose a heuristic algorithm for the AGV operation scheme considering conflict, as shown in **Algorithm 3**. **Algorithm 3** handles the three types of conflicts shown in Fig. 6.

- Conflict 1. The opposite direction conflicts on the same horizontal path, as shown in Fig. 6 (a). If Conflict 1 occurs between two actions, the action that arrives later will wait for the action that arrives earlier.
- Conflict 2. The conflict between the AGV quayside operation and the AGV transportation on the same horizontal path is shown in Fig. 6 (b). If Conflict 2 occurs between an action and a quayside operation, the time when the action reaches the quayside operation location must be later than the finishing time of the quayside operation.
- Conflict 3. The vertical path conflict, as shown in Fig. 6 (c). Similar to Conflict 1, the action that arrives later will wait for the action that arrives earlier.

Three consequences of three types of conflicts considered!

In **Algorithm 3**, the container $(m,i)^*$ and AGV l^* are obtained based on

Table 2
Performance comparison under small-scale problems.

Instance	Scale	B&B		Gurobi solver			Two strategies		Gap 1(%)	Gap 2(%)	Gap 3(%)	Gap 4(%)
	(AGVs, Containers)	Obj(s)	CPU time(s)	Obj(s)	LB(s)	CPU time(s)	FCFS	SETTF				
1	(2,6)	401	1.49	401	286	3600	522	436	0	40.21	30.17	8.73
2	(3,6)	280	0.23	280	218	3600	280	280	0	28.44	0.00	0.00
3	(2,7)	518	0.93	518	406	3600	562	537	0	27.59	8.19	3.67
4	(3,7)	363	0.74	363	248	3600	374	374	0	46.37	3.03	3.03
5	(3,8)	416	16.61	416	278	3600	518	518	0	49.64	24.52	24.52
6	(3,8)	399	19.76	399	343	3600	520	520	0	16.33	30.33	30.33
7	(4,8)	323	25.73	321	278	3600	411	411	0.62	16.19	27.24	27.24
8	(3,9)	476	7.59	476	338	3600	516	516	0	40.83	8.40	8.40
9	(4,9)	392	15.98	391	332	3600	412	435	0.26	18.07	5.10	10.97
10	(4,9)	344	31.27	344	270	3600	422	402	0	27.41	22.67	16.86
11	(3,10)	479	57.36	–	310	3600	605	605	–	54.52	26.30	26.30
12	(4,10)	440	74.19	433	362	3600	489	489	1.62	21.55	11.14	11.14
13	(4,11)	430	38.16	430	310	3600	480	469	0	38.71	11.63	9.07
14	(4,12)	490	61.97	–	335	3600	536	536	–	46.27	9.39	9.39
15	(4,14)	512	103.36	–	402	3600	610	668	–	27.36	19.14	30.47
16	(5,16)	526	186.31	–	476	3600	586	613	–	10.50	11.41	16.54
17	(4,20)	805	176.49	–	626	3600	1033	949	–	28.59	28.32	17.89
18	(5,20)	668	320.14	–	626	3600	828	781	–	6.71	23.95	16.92
19	(6,20)	670	436.68	–	626	3600	702	780	–	7.03	4.78	16.42
20	(7,20)	636	257.95	–	626	3600	699	658	–	1.60	9.91	3.46
Average	–	–	–	–	–	–	–	–	0.21	27.70	15.78	14.57

Notes: Gap 1 is defined as (B&B - Gurobi)/Gurobi; Gap 2 is defined as (B&B - LB)/LB; Gap 3 is defined as (FCFS - B&B)/B&B; Gap 4 is defined as (SETTF - B&B)/B&B.

very very important note, the intuition for the algorithm
A new way of solving is done! (combinatorial solution)

the known job assignment and the job sequence (rows 08 through 14). The route with the shortest operation time is obtained from rows 15 through 39. Conflict detection is performed in sequence for four actions of a container (row 20). The container operation time t' and the starting time list of each event $\boldsymbol{h}'$ are generated considering the QC operation sequence. Rows 21 through 24 indicate that if there is no conflict then the current route is selected. Rows 25 through 27 indicate that if all routes in set $R_{(m,i)^*,s}$ have conflicts then the route with the shortest operation time in the set is recorded. If there are conflicts for all routes in set $R_{(m,i)^*,s}$ and the route with shorter operation time cannot be found in route set $R_{(m,i)^*,s'}$, $s' \in S_{(m,i)^*}$, route r_1 is selected (rows 31 through 35). Rows 37 through 39 indicate that if all routes for container $(m,i)^*$ have conflicts, the route with the shortest operation time is selected. If the new solution dominates the existing solution, then the solution Ω^* is updated (rows 42 through 44). Note that the conflict-free route in **Algorithm 2** is generated according to rows 15 through 39 in **Algorithm 3**.

Algorithm 3 Heuristic algorithm for the AGV operation scheme considering conflict

01: data: N_τ- the set of nodes in level τ, $C_{\tau,v}$- the set of assigned containers for node v in level τ, sorted by AGV operation sequence, $C'_{\tau,v}$- the set of containers to be assigned, i.e., the current jobs in the QC operation lists, $S_{(m,i)}$- the set of operation time for container (m,i) without considering conflict, sorted from smallest to largest, $R_{(m,i),s}$- the set of routes for container (m,i) whose operation time is $s \in S_{(m,i)}$ without considering conflict, M- a very large positive number, $l_{(m,i)}$- the AGV that conducts container (m,i) in the node
02: **Result:** Ω^*- solution, $T^{obj}(\Omega^*)$- the objective of solution Ω^*
03: **procedure**
04: $\Omega^* \leftarrow$ *initial solution* # the initial solution is obtained from **Algorithm 2**
05: **for** $v \in N_\tau$ **do**
06: $\Omega \leftarrow \varnothing$
07: **while** $C_{\tau,v} \neq \varnothing$ **do**
08: **for** $(m,i) \in C_{\tau,v}$ **do**
09: **if** $(m,i) \in C'_{\tau,v}$ **do**
10: $(m,i)^* = (m,i)$, $l^* = l_{(m,i)}$ #obtain selected container and AGV
11: update $C_{\tau,v}, C'_{\tau,v}$
12: **break**
13: **end if**
14: **end for**
15: $index = 0$ # equal to 1 if the route with the shortest operation time is found
16: $t_1 = M$ # define the initial value of t_1
17: **for** $s \in S_{(m,i)^*}$ **do**
18: $s' \leftarrow$ the next value of s in set $S_{(m,i)^*}$
19: **for** $r \in R_{(m,i)^*,s}$ **do**
20: $(t', \boldsymbol{h}') = conflict\_detection(r, s, (m,i)^*, l^*, \Omega)$
21: **if** $t' = s$ **do** #there is no conflict
22: $r^* = r$, $t^* = t'$, $\boldsymbol{h}^* = \boldsymbol{h}'$
23: $index = 1$
24: **break**
25: **else if** $t' \leq t_1$ **do**
26: $r_1 = r$, $t_1 = t'$, $\boldsymbol{h}_1 = \boldsymbol{h}'$
27: **end if**
28: **end for**
29: **if** $index = 1$ **do**
30: **break**
31: **else if** s' *exists* **and** $t_1 \leq s'$ **do**
32: $r^* = r_1$, $t^* = t_1$, $\boldsymbol{h}^* = \boldsymbol{h}_1$
33: $index = 1$
34: **break**
35: **end if**
36: **end for**
37: **if** $index = 0$ **do**
38: $r^* = r_1$, $t^* = t_1$, $\boldsymbol{h}^* = \boldsymbol{h}_1$
39: **end if**
40: $\Omega = \Omega \cup ((m,i)^*, l^*, r^*, \boldsymbol{h}^*)$
41: **end while**
42: **if** $T^{obj}(\Omega) \leq T^{obj}(\Omega^*)$ **do**
43: $\Omega^* = \Omega$ #update the solution
44: **end if**
45: **end for**

(continued on next column)

(continued)

Algorithm 3 Heuristic algorithm for the AGV operation scheme considering conflict

46: **return** Ω^*
47: **end procedure**

5.5. Accelerating techniques

problem of the B&B alg. (those arrangements in AGV op scheme P)

The B&B algorithm contains all of the possible assignment plans for the containers. Therefore, the computational time needed to solve the problem grows exponentially with the instance size. To improve the efficiency of the algorithm, we designed four accelerating techniques. During the branching process of the algorithm, duplicate AGV operation schemes are generated due to the different AGV indexes. However, the AGVs in this paper are identical. These duplicate schemes have exactly the same makespan, and the accelerating techniques 1, 2 and 3 are developed to remove these duplicate AGV operation schemes. The algorithm in this paper uses the Breadth-First-Search strategy, and we develop a node filtering method (i.e., accelerating technique 4) during the branching process, which can retain the nodes with lower criteria values. The accelerating technique 4 can reduce the computational time but affect the solution quality, which is essentially a heuristic rule that allows the B&B algorithm to be solved in large-scale problems. The four accelerating techniques are shown as follows.

with lower than upper bound value

Accelerating technique 1 disregards the repeated schemes of the container numbers for AGV operation in the set P. All AGVs in the problem are exactly the same. Therefore, the repetition of schemes caused by different AGVs can be eliminated. For example, if the number of AGVs $|B| = 2$, the number of containers $|C| = 4$, and the maximum number of containers conducted by each AGV $q = 3$, then scheme $(1,3)$ is the same as $(3,1)$. The set P can be expressed as follows: $P = \{(1,3),(2,2)\}$.

Accelerating technique 2 disregards the repeated job assignment and job sequence for the same scheme p. Due to the particularity of the branching strategy, for the same scheme p, two nodes may have the same job assignment and job sequence. For example, there are two nodes (node N1 and node N2) with the same job assignment in level 2, which is shown as follows: AGV 1 = [(QC 1, CON 1)], AGV 2 = [(QC 2, CON 1)]. The generation of node N1 is AGV 1 = [(QC 1, CON 1)], AGV 2 = [] to AGV 1 = [(QC 1, CON 1)], AGV 2 = [(QC 2, CON 1)]. The generation of node N2 is AGV 1 = [], AGV 2 = [(QC 2, CON 1)] to AGV 1 = [(QC 1, CON 1)], AGV 2 = [(QC 2, CON 1)]. Therefore, node N1 and node N2 are exactly the same and can perform deduplication.

Accelerating technique 3 also disregards repeated nodes. The AGVs with the same number of containers for the same scheme are exactly the same, with the following example used for illustration. For two nodes (node N1 and node N2) with the same scheme $p = (2,2)$, the job assignment in node N1 is AGV 1 = [(QC 1, CON 1), (QC 2, CON 2)], AGV 2 = [(QC 2, CON 1), (QC 1, CON 2)], while in node N2 it is AGV 1 = [(QC 2, CON 1), (QC 1, CON 2)], AGV 2 = [(QC 1, CON 1), (QC 2, CON 2)]. Since all the AGVs are exactly the same, the operation information of node N1 and node N2 are the same.

Accelerating technique 4 is a node filtering method. A good job assignment and job sequence of the AGVs shall contain a shorter waiting time and empty travel time. The waiting time incurs when the AGV waits for the quayside operation or the QC waits for the AGV to arrive at the operation location. An empty travel time incurs when an empty AGV travels to pick up its next container. Therefore, the filtering criteria c of a node are proposed as follows. A higher c means more time is wasted on waiting and empty travel. Thus, nodes with lower criteria values are preferable.

$$c = \frac{waiting\ time + empty\ time}{container\ number}$$

The possibility of generating better solutions is different for different schemes in set P. Let α_p be the maximum number of containers for scheme $p \in P$. For example, $\alpha_p = 2$ for $p = (2,2)$ and $\alpha_p = 3$ for $p = (1,$

Table 3
Performance comparison under large-scale problems.

Instance	Scale	B&B		TGH		Two strategies		Gap 1(%)	Gap 2(%)	Gap 3(%)	Gap 4(%)	Gap 5(%)
	(AGVs, Containers)	Obj(s)	CPU time(s)	Obj(s)	CPU time(s)	FCFS	SETTF					
21	(5,30)	946	976.17	1123	0.94	1162	1464	18.71	22.83	54.76	3.47	30.37
22	(6,30)	892	136.93	931	1.29	946	1236	4.37	6.05	38.57	1.61	32.76
23	(7,30)	896	462.84	935	1.54	961	1141	4.35	7.25	27.34	2.78	22.03
24	(8,30)	858	934.62	896	2.56	920	1014	4.43	7.23	18.18	2.68	13.17
25	(6,40)	1323	394.38	1352	1.63	1369	1689	2.19	3.48	27.66	1.26	24.93
26	(7,40)	1239	978.63	1323	2.17	1404	1389	6.78	13.32	12.11	6.12	4.99
27	(8,40)	1238	1136.38	1307	2.73	1348	1432	5.57	8.89	15.67	3.14	9.56
28	(9,40)	1230	1629.51	1327	3.39	1371	1427	7.89	11.46	16.02	3.32	7.54
29	(6,50)	1636	367.72	1805	2.49	1843	2170	10.33	12.65	32.64	2.11	20.22
30	(7,50)	1657	1492.16	1778	3.71	1845	1884	7.30	11.35	13.70	3.77	5.96
31	(8,50)	1703	1273.37	1731	3.93	1776	2052	1.64	4.29	20.49	2.60	18.54
32	(9,50)	1582	1016.53	1627	4.75	1655	1804	2.84	4.61	14.03	1.72	10.88
33	(10,80)	2161	439.91	2249	8.29	2311	2605	4.07	6.94	20.55	2.76	15.83
34	(12,100)	2618	1096.16	2772	11.78	2863	3357	5.88	9.36	28.23	3.28	21.10
35	(12,120)	3204	1316.74	3354	14.41	3416	4227	4.68	6.62	31.93	1.85	26.03
36	(12,150)	3913	1427.63	4069	17.62	4251	4907	3.99	8.64	25.4	4.47	20.59
37	(15,150)	3725	1534.92	4054	22.58	4214	5003	8.83	13.13	34.31	3.95	23.41
38	(15,200)	5174	1931.56	5473	30.32	5602	6800	5.78	8.27	31.43	2.36	24.25
39	(15,250)	6513	2337.37	6619	36.49	6718	8216	1.63	3.15	26.15	1.50	24.13
40	(18,250)	6397	2043.74	7024	44.28	7173	7051	9.80	12.13	10.22	2.12	0.38
Average	–	–	–	–	–	–	–	6.05	9.08	24.97	2.84	17.83

Notes: Gap 1 is defined as (TGH - B&B)/B&B; Gap 2 is defined as (FCFS - B&B)/B&B; Gap 3 is defined as (SETTF - B&B)/B&B; Gap 4 is defined as (FCFS - TGH)/B&B; Gap 5 is defined as (SETTF - TGH)/B&B.

neat trick! When the max number of containers in scheme p1 is higher than the max number of p2, then there would be more nodes for that p2! (Since we are iterating on Containers!

3). To improve the performance of node filtering, for two schemes p_1 and p_2, if $\alpha_{p_1} < \alpha_{p_2}$, then the number of nodes reserved for p_1 is higher than p_2. If $\alpha_{p_1} = \alpha_{p_2}$, the number of nodes reserved for the two schemes is the same. For two nodes with the same schemes, the node with a lower c is preferable. The node filtering method filters out the nodes with poor performances and keeps the best w nodes, where w is the filter width.

6. Computational experiments

In this section, we describe the test instance and provide results to evaluate the performance of the proposed algorithm. The experiments were run on a PC equipped with Intel Core i5 CPU 1.60 GHz and 16 GB RAM. The algorithm was programmed in Python 3.6.5, and Gurobi 8.0.1 was chosen as the MIP solver.

In the computational experiments, when the number of containers reaches thousands, the B&B algorithm cannot obtain solutions in a short amount of time, which indicates that the B&B algorithm sacrifices a large amount of CPU time while improving the solution quality. Even metaheuristic algorithms, such as genetic algorithm and particle swarm optimization algorithm (Hu et al., 2019; Zhong et al., 2020), are not able to obtain solutions in a short amount of time. To reduce the solution time, some advanced heuristic strategies are needed, so we design a two-stage greedy heuristic (TGH) algorithm. In the first stage of the TGH, the evaluation matrix of container and AGV operation schemes is set up to find the local optimal solution for the current situation without considering conflict. Based on the results of the first stage, the second stage of the TGH obtains conflict-free routes according to the first come first service principle, that is, if the travel path is occupied by the first-arriving AGV, the later-arriving AGV will wait for the first-arriving AGV or choose another path. Following the greedy strategy, the route with the shortest travel time for the current container will be selected. The details of the TGH are presented in **Appendix E**, and the results of the TGH are shown in sections 6.3 and 6.7.

def of FCFS

6.1. Instance generation and parameter estimation

The actual operating conditions of Qingdao Qianwan port are utilized as parameters, and the system configuration in the experiments is consistent with Fig. 4. The widths of the seaside and landside AGV operation areas are both set to 30 m (i.e., the distance from 1 to y_r or y_{r+1} to y_R). The distance between the seaside and landside AGV operation area is set to 45 m. The distance between adjacent vertical paths is set to 8 m (i.e., the distance from x_r to x_{r+1}). The number of HPs for each crane is set to 5. The vertical path of a QC to operate containers can be marked according to the storage locations in the ship. The job switching time $S^Q_{(m,i)(m,i+1)}$ for a QC between two jobs is determined by the storage locations of two containers, which is known in our model. All the operation times follow uniform distribution. The operation time of each QC is an integer that follows uniform distribution U (60, 90) s, and the container transfer time between the AGV and the AGV-support follows uniform distribution U (20, 30) s. The physical upper limit of the AGV speed is 6 m/s, and the speed drops when transporting containers. Therefore, the travel speed of the AGV is set to 4 m/s. The filter width is set to 1500, and the numbers of QCs and blocks are set to 3 and 6, respectively.

6.2. Results for small-scale problems

In this section, we compare the proposed B&B algorithm with the Gurobi solver and two strategies in small-scale instances. The Gurobi solver can obtain the optimal solution and lower bound based on the **ADRP model**, but it cannot finish the solution process within the time limit. To further assess the performance of the proposed algorithm, the B&B algorithm is compared with two port scheduling strategies: first come first service (FCFS), and shortest empty travel time first (SETTF).

- The FCFS strategy consists of two rules. First, when an AGV completes its current job, a pre-sequence container in the QC operation lists is allocated to this AGV. Second, the travel routes of the assigned containers are fixed. The unassigned containers need to avoid conflict with assigned containers.
- The SETTF strategy is revised based on the FCFS strategy. When an AGV completes its current job, an unassigned container with the shortest empty travel time is allocated to this AGV. The generation of transportation routes is the same as the FCFS strategy.

Twenty instances are shown in Table 2. The columns in the table provide the instance, the scale (the number of AGVs and containers), the objective value (Obj), the running time (CPU time), the lower bound of the Gurobi solver (LB), the makespan of two strategies, and the

Table 4
The impact of different accelerating techniques.

Instance	Scale	B&B		B&B without techniques 1, 2 and 3		B&B without technique 4		Gap 1(%)	Gap 2(%)
	(AGVs, Containers)	Obj(s)	CPU time(s)	Obj(s)	CPU time(s)	Obj(s)	CPU time(s)		
41	(3,9)	476	7.59	493	107.36	476	8.93	3.57	0.00
42	(3,10)	479	57.36	542	217.47	479	69.26	13.15	0.00
43	(4,11)	430	38.16	480	36.73	430	37.19	11.63	0.00
44	(4,12)	490	61.97	536	154.79	490	218.04	9.39	0.00
45	(4,13)	451	59.42	451	167.12	–	3600	0.00	–
46	(4,14)	512	103.36	515	304.56	–	3600	0.59	–
47	(5,16)	526	186.31	526	319.87	–	3600	0.00	–
48	(5,20)	668	320.14	685	237.91	–	3600	2.54	–
49	(6,20)	670	436.68	690	573.48	–	3600	2.99	–
50	(7,20)	636	257.95	658	237.16	–	3600	3.46	–
51	(5,30)	946	976.17	998	736.39	–	3600	5.50	–
52	(6,30)	892	136.93	946	391.06	–	3600	6.05	–
Average	–	–	–	–	–	–	–	4.91	0.00

Notes: Gap 1 is defined as (B&B without techniques 1, 2 and 3 - B&B)/B&B; Gap 2 is defined as (B&B without technique 4 - B&B)/B&B.

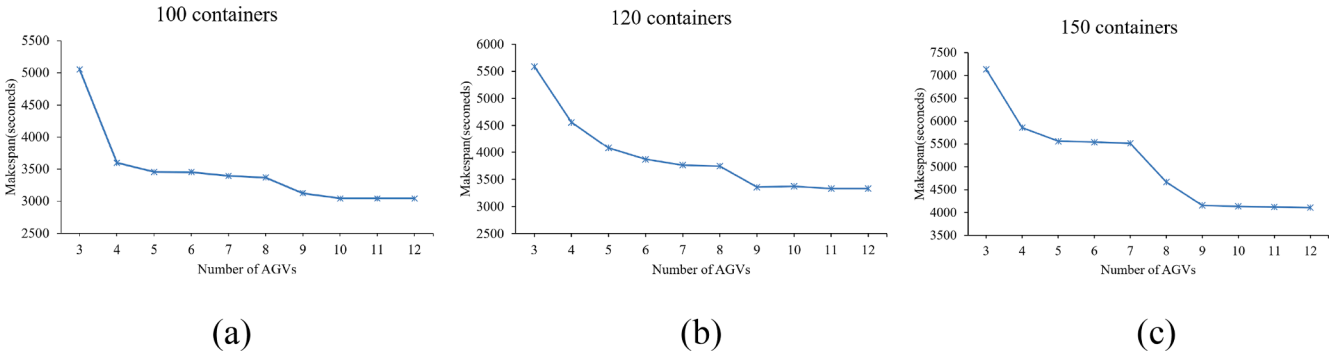


Fig. 9. Effects of the number of AGVs on the makespan.

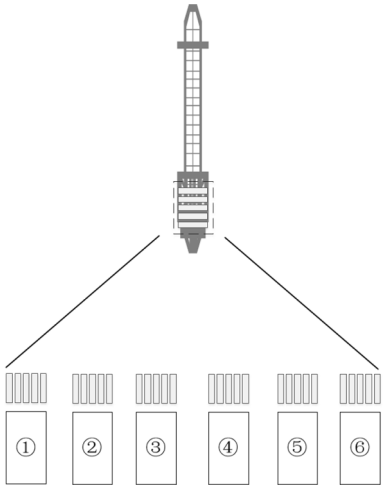


Fig. 10. The storage locations for containers.

optimality gap. The time limit of Gurobi is one hour (3,600 s). We define the instances with less than 20 containers as small-scale cases and those with more than 20 containers as large-scale cases. In the small-scale instances, the Gurobi solver can obtain the lower bound (all instances) and the optimal solution (instances 1–10 and 12–13) within the time limit. However, in the large-scale instances, the Gurobi solver cannot obtain the results within the time limit.

It can be observed from Table 2 that Gurobi can obtain the solutions of twelve instances within the time limit, while our proposed B&B algorithm can obtain the results in 6 min. The optimality gap is relatively small when the problem scale is small, and the average optimal gap is 0.21%, which verifies the effectiveness of the proposed algorithm. When the scale is large, the MIP solver of Gurobi reaches the time limit and cannot obtain a feasible solution, and the B&B algorithm can solve the problems in a short amount of time. The optimality gap between the B&B algorithm and the LB of Gurobi is 27.70%, and the optimality gaps between the B&B algorithm and two strategies are 15.78% and 14.57%, respectively, which indicates that the proposed algorithm outperforms the existing strategies in small-scale instances. The short computational time and high solution quality demonstrate the effectiveness of the proposed B&B algorithm.

6.3. *Results for large-scale problems*

The twenty instances for large-scale problems are shown in this section, and the container operation information is different among different instances. The analyses of the same container operation information and the different AGV numbers (i.e., the sensitivity analysis for the number of AGVs) are given in section 6.5.1. Table 3 illustrates the comparison results for large-scale instances. In large-scale instances, the Gurobi solver cannot obtain the feasible solution and the lower bound within the time limit (3,600 s), and thus the relevant columns of the Gurobi solver in the table are not shown. Moreover, to reduce the computational time, the filter width is set to 500 when the scale of the instance reaches (10, 80) (i.e., instances 33–40). The results of the TGH for large-scale problems are also presented in Table 3. results

The results in Table 3 suggest that the FCFS outperforms the SETTF in most instances, and the B&B algorithm is better than the two port scheduling strategies, where the optimality gaps are 9.08% and 24.97%, respectively. The main reason is that the FCFS prefers to assign pre-sequence containers in the QC operation lists for AGVs, and the SETTF assigns containers with short empty travel time. The SETTF may assign

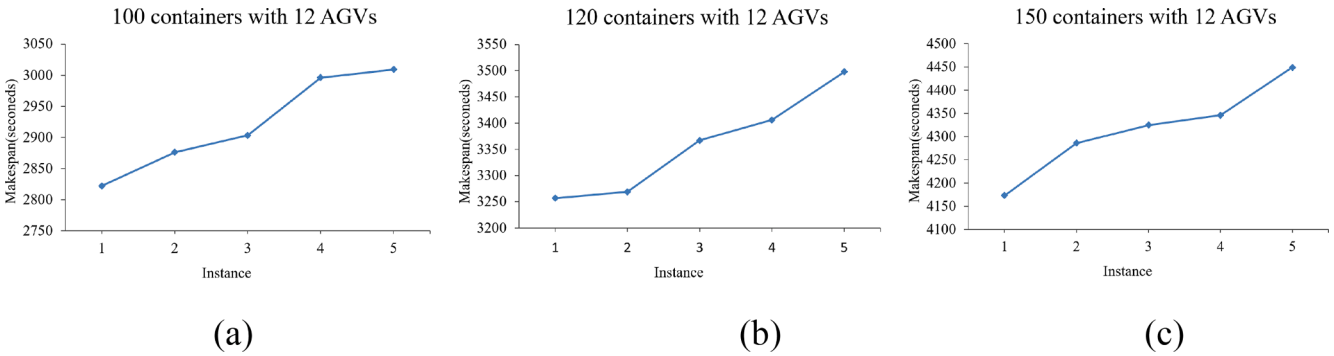


Fig. 11. Effects of the storage locations on the makespan.

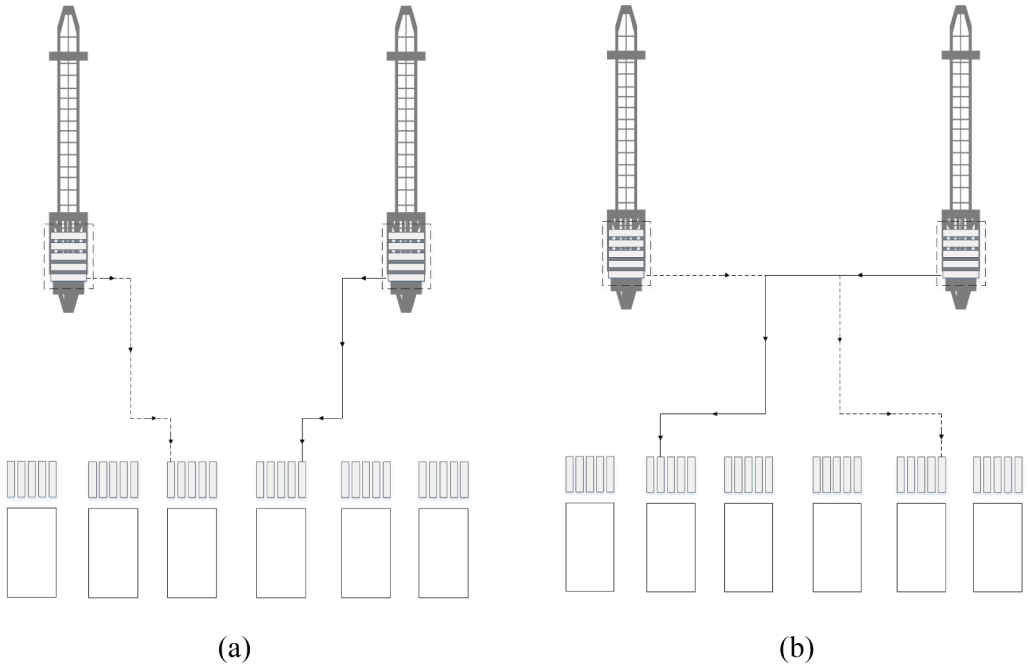


Fig. 12. The AGV routes for different storage locations.

Table 5
Effects of the filter width on the makespan and CPU time.

Filter width	20 jobs with 7 AGVs		30 jobs with 7 AGVs		40 jobs with 7 AGVs	
	Obj (s)	CPU time (s)	Obj (s)	CPU time (s)	Obj (s)	CPU time (s)
100	658	10.36	915	26.81	1398	2.96
500	656	72.59	915	137.40	1300	142.07
800	636	139.27	882	213.15	1298	896.13
1000	636	204.18	882	280.09	1298	537.60
1500	636	257.95	896	462.84	1239	978.63
2000	636	483.64	896	560.73	1276	2153.74
2500	636	540.16	896	629.64	1277	2561.08
3000	636	687.09	896	803.92	1277	3493.41

subsequent containers in the QC operation lists to AGVs, resulting in a long time for AGVs to wait for QC operations. The two strategies only consider a relatively small number of feasible solutions, whereas the B&B algorithm explores many possible feasible solutions. Therefore, the B&B algorithm outperforms the two strategies. In addition, the B&B algorithm outperforms the TGH with an optimality gap of 6.05%, but the CPU time of the TGH is much less than that of the B&B algorithm. The TGH outperforms the two strategies with optimality gaps of 2.84% and 17.83, respectively, providing a method to solve **ADRP** in a short amount of time for large-scale problems.

6.4. *Improvement of the accelerating techniques*

In the proposed B&B algorithm, accelerating techniques 1, 2 and 3 disregard the repeated operation schemes, which can reduce the computational time without affecting the solution quality. Accelerating technique 4 filters out operation schemes with a poor performance, which can reduce the computational time but will affect the solution quality. Therefore, we compare the B&B algorithm with the B&B algorithm without accelerating techniques 1, 2 and 3 and the B&B algorithm without accelerating technique 4. The impact of different accelerating techniques on the solution efficiency is shown in Table 4.

In Table 4, we observe that the optimality gap between the B&B algorithm and the B&B algorithm without accelerating techniques 1, 2 and 3 is 4.91%. The reason is that accelerating techniques 1, 2 and 3 can disregard many repeated schemes, and accelerating technique 4 will reserve a large number of repeated schemes with a poor performance when accelerating techniques 1, 2 and 3 are not considered. The gap between the B&B algorithm and the B&B algorithm without accelerating technique 4 is zero, while the algorithm without technique 4 cannot obtain solutions within the time limit when the scale reaches (4,13). This is because the algorithm without technique 4 spends considerable

Table 6
How these data are generated is a secret!!
Performance comparison of a dynamic case.

Stage d	$\lvert C_{d-1}^{U}\rvert$	$\lvert C_{d}^{N}\rvert$	N_d	B&B		Two strategies		Gap 1(%)	Gap 2(%)
				Obj(s)	CPU time(s)	FCFS	SETTF		
1	0	79	50	1351	387.15	1425	1585	5.48	17.32
2	29	66	50	2504	537.94	2748	3207	9.74	28.08
3	45	52	50	3895	316.69	4169	4811	7.03	23.52
4	47	48	50	5093	266.42	5492	6442	7.83	26.49
5	45	52	50	6326	319.62	6882	7813	8.79	23.51
6	47	47	50	7453	343.25	8313	9388	11.54	25.96
7	44	56	50	8587	432.92	9557	10,832	11.30	26.14
8	50	48	50	9614	567.52	10,890	12,389	13.27	28.86
9	48	56	50	10,952	388.91	12,272	13,791	12.05	25.92
10	54	42	50	12,181	269.08	13,618	15,233	11.80	25.06
11	46	49	50	13,428	311.78	14,930	16,701	11.19	24.37
12	45	37	50	14,373	253.25	16,261	18,340	13.14	27.60
13	32	53	50	15,152	454.86	17,618	19,959	16.28	31.73
14	35	44	50	17,074	308.25	19,002	21,389	11.29	25.27
15	29	48	50	18,286	313.51	20,368	22,924	11.39	25.36
16	27	62	50	19,515	418.95	21,605	24,648	10.71	26.30
17	39	56	50	20,624	316.31	23,007	26,235	11.55	27.21
18	45	51	50	21,862	371.65	24,258	27,677	10.96	26.60
19	46	54	50	22,879	490.73	25,539	30,064	11.63	31.40
20	50	0	50	24,017	332.98	26,945	34,002	12.19	41.57
Average	–	–	–	–	370.09	–	–	10.96	26.91

Notes: Gap 1 is defined as (FCFS - B&B)/B&B; Gap 2 is defined as (SETTF - B&B)/B&B.

Table 7
Performance comparison of static real-world cases.
results for static cases

Instance	Scale	B&B		TGH		Two strategies		Gap 1(%)	Gap 2(%)	Gap 3(%)	Gap 4(%)	Gap 5(%)
	(AGVs, Containers)	Obj(s)	CPU time(s)	Obj(s)	CPU time(s)	FCFS	SETTF					
53	(12,100)	2795	1036.07	2973	12.17	3046	3493	6.37	8.98	24.97	2.46	17.49
54	(15,100)	2501	893.14	2612	14.95	2734	3207	4.44	9.32	28.23	4.67	22.78
55	(18,100)	2376	1192.43	2598	19.03	2893	3162	9.34	21.76	33.08	11.35	21.71
56	(12,130)	3398	1409.36	3743	16.38	3869	4705	10.15	13.86	38.46	3.37	25.70
57	(15,130)	3162	1513.91	3541	19.68	3663	3663	11.99	15.84	15.84	3.45	3.45
58	(18,130)	3584	1039.28	3801	24.61	3943	4481	6.05	10.02	25.03	3.74	17.89
59	(12,160)	3914	1592.60	4315	21.01	4496	4788	10.25	14.87	22.33	4.19	10.96
60	(15,160)	4079	1237.59	4266	22.57	4338	4476	4.58	6.35	9.73	1.69	4.92
61	(18,160)	3892	1560.34	4189	28.63	4399	5908	7.63	13.03	51.80	5.01	41.04
62	(12,200)	4837	1976.83	5234	24.04	5541	5962	8.21	14.55	23.26	5.87	13.91
63	(15,200)	4556	2098.17	5185	29.46	5434	6269	13.81	19.27	37.60	4.80	20.91
64	(18,200)	4836	1803.62	5247	35.39	5434	6596	8.50	12.37	36.39	3.56	25.71
Average	–	–	–	–	–	–	–	8.44	13.35	28.89	4.51	18.87

Notes: Gap 1 is defined as (TGH - B&B)/B&B; Gap 2 is defined as (FCFS - B&B)/B&B; Gap 3 is defined as (SETTF - B&B)/B&B; Gap 4 is defined as (FCFS - TGH)/B&B; Gap 5 is defined as (SETTF - TGH)/B&B.

Table 8
Performance comparison of dynamic real-world cases.
results for dynamic cases

Instance	Scale	$\overline{N}$	B&B		TGH		Two strategies		Gap 1(%)	Gap 2(%)	Gap 3(%)	Gap 4(%)	Gap 5(%)
	Containers		Obj(s)	CPU time(h)	Obj(s)	CPU time(s)	FCFS	SETTF					
65	1000	20	25,814	0.95	28,174	178.91	29,281	32,709	9.14	13.43	26.71	3.93	16.10
66	1000	10	27,078	0.54	28,174	178.91	29,281	32,709	4.05	8.14	20.80	3.93	16.10
67	1000	6	26,893	0.16	28,174	178.91	29,281	32,709	4.76	8.88	21.63	3.93	16.10
68	3000	20	78,463	2.47	82,305	544.59	85,613	90,647	4.90	9.11	15.53	4.02	10.14
69	3000	10	79,867	1.50	82,305	544.59	85,613	90,647	3.05	7.19	13.50	4.02	10.14
70	3000	6	80,351	0.58	82,305	544.59	85,613	90,647	2.43	6.55	12.81	4.02	10.14
71	5000	20	117,635	4.11	130,256	954.76	133,184	146,274	10.73	13.22	24.35	2.25	12.30
72	5000	10	126,894	2.67	130,256	954.76	133,184	146,274	2.65	4.96	15.27	2.25	12.30
73	5000	6	129,183	0.79	130,256	954.76	133,184	146,274	0.83	3.10	13.23	2.25	12.30
74	7000	20	186,940	6.64	197,037	1245.24	203,762	223,490	5.40	9.00	19.55	3.41	13.43
75	7000	10	192,318	3.68	197,037	1245.24	203,762	223,490	2.45	5.95	16.21	3.41	13.43
76	7000	6	195,126	1.02	197,037	1245.24	203,762	223,490	0.98	4.43	14.54	3.41	13.43
Average	–	–	–	–	–	–	–	–	4.28	7.83	17.84	3.40	12.99

Notes: Gap 1 is defined as (TGH - B&B)/B&B; Gap 2 is defined as (FCFS - B&B)/B&B; Gap 3 is defined as (SETTF - B&B)/B&B; Gap 4 is defined as (FCFS - TGH)/B&B; Gap 5 is defined as (SETTF - TGH)/B&B.

time in the operation schemes with a poor performance, thereby increasing the computational time.

6.5. Sensitivity analysis

In this section, we conduct a sensitivity analysis by varying the number of AGVs, the storage locations and the filter width. The experiments and insights are demonstrated in sections 6.5.1–6.5.3.

6.5.1. The sensitivity analysis for the number of AGVs

To observe the impact of the number of AGVs on operation efficiency, we conduct a sensitivity analysis by varying the number of AGVs in this section, as shown in Fig. 9. The number of containers is set to 100, 120 and 150, and the number of AGVs varies from 3 to 12.

The results indicate that a greater number of AGVs is not necessarily ideal. Fig. 9 shows that when the number of AGVs is small, the makespan decreases significantly as the number of AGVs increases. When the number of AGVs is 9 to 12, the efficiency reaches its peak. The continued increase in the number is not conducive to AGV operations. The reasons are as follows: 1) too many AGVs will cause a path conflict and waiting delay; and 2) AGVs will wait for QC operations. When the number of AGVs is large, the QC operations restrict the improvement of the operation efficiency. The number of QCs in this paper is set to 3, and Fig. 9 illustrates that the ideal number of AGVs is 6 to 8. Therefore, the ratio of the number of AGVs to the number of QCs is preferably 1:3 to 1:4.

6.5.2. The sensitivity analysis for the storage locations

The storage locations in the yard are assumed to be given in **ADRP**, and we could vary the input to study the impact of storage locations. In practice, containers are preferentially stored in blocks near their unloading or loading QC. In this section, the distance between the yard storage location and the QC operation location for each container is set at certain levels. As shown in Fig. 10, the containers in each QC operation list can be stored in six adjacent blocks. In these experiments, the storage blocks are set to five ranges, that is, [③, ④], [②, ③, ④, ⑤], [②, ⑤], [①, ②, ⑤, ⑥], and [①, ⑥]. The number of AGVs is set to 12, and the number of containers is set to 100, 120 and 150, as shown in Fig. 11.

The results in Fig. 11 suggest that the makespan increases with the distance between the QC and the block, and the reason is that the containers will be stored in the blocks further away from their loading or unloading QCs. This means that the loaded travel distance increases and the containers are more likely to have a path conflict, which leads to an increasing makespan. Fig. 12 shows the AGV routes for different storage locations, which can illustrate the relationship between storage locations and AGV conflicts. In Fig. 12 (a), AGVs have short loaded travel distances, and the possibility of conflict is small. In Fig. 12 (b), the yard storage locations are further away from the QC operation locations, and AGVs travel further and occupy the path longer, which makes it more likely that different AGVs occupy the same path at the same time, thus leading to an increased possibility of conflicts among AGVs. Thus, to improve the performance of the automated container terminals, the storage location should be carefully planned.

6.5.3. The sensitivity analysis for the filter width

In this section, we conduct a sensitivity analysis by varying the filter width, which directly affects the computational time and the solution quality of the algorithm. The number of AGVs and the number of containers are set to (7, 20), (7, 30) and (7, 40). We then experiment on the filter width by setting the parameter to 100, 500, 800, 1000, 1500, 2000, 2500, and 3000. The results are summarized in Table 5.

In general, an increase in the filter width will increase the solution quality of the proposed algorithm and consume more computational time. As shown in Table 5, when the filter width is too large, poor results are produced. This is because the following situation occurs during the branching process of the middle level in the B&B algorithm. Some nodes with poor final solutions have small c values, which branch more nodes with small c values but poor final solutions. These new nodes are more likely to be reserved, and the nodes with excellent final solutions may be disregarded, resulting in a large filter width but poor results. Therefore, it is important to choose an appropriate filter width, and we chose 1500 in our proposed algorithm.

6.6. Case study for dynamic problems

To demonstrate the applicability of the proposed algorithm in dynamic problems, we conduct computational experiments with dynamic cases. In the static problem, all the container information is given at once and the algorithm obtains AGV scheduling scheme for all containers. In the dynamic problem, the container information is given at different stages, and the algorithm obtains AGV scheduling scheme based on the container information and equipment operation states in each stage. The dynamic case is divided into multiple stages, and each stage is solved independently based on the B&B algorithm or the two port scheduling strategies. When one stage is completed, the information of the unassigned containers, AGVs, and QCs is passed to the next stage. The containers are not given at once but are obtained at different stages. For each stage, the set of containers to be assigned includes the newly obtained containers in this stage and the unassigned containers in the previous stage, and the unassigned containers in this stage will become the containers to be assigned in next stage. The dynamic problem algorithm of a stage is presented in **Appendix F**.

In the case study for dynamic problems, the scale (containers, AGVs) is set to (1000, 18), the dynamic case is divided into 20 stages, and the filter width is set to 500. The results are shown in Table 6. $|C_{d-1}{}^{U}|$, $|C_d{}^{N}|$, and N_d in Table 6 indicate the number of unassigned containers in the previous stage, the number of new containers in this stage, and the number of containers to be assigned in this stage, respectively. The objective values of the B&B algorithm and the two strategies refer to the total completion time proceeding to this stage, and the CPU time refers to the computational time of the B&B algorithm in this stage.

From Table 6, we can observe that the dynamic case is divided into 20 stages, the number of newly obtained containers in each stage is different, and the number of assigned containers is 50. For example, the number of newly obtained containers in stage 1 and stage 2 are 79 (0 + 79) and 95 (29 + 66), respectively, and the number of assigned containers is 50. We further observe that the B&B algorithm outperforms the two port scheduling strategies in terms of the solution quality, where the average gaps are 10.96% and 26.91%. The B&B algorithm can solve the dynamic problem within a reasonable amount of time, and the average computational time is 370.09 s. The excellent performance shows that the proposed B&B algorithm is an effective method for dynamic cases.

6.7. Real-world case study

In this section, we conduct computational experiments with static and dynamic real-world cases based on the physical layout and operational data of Qingdao Qianwan, where the planning horizon is 12 h. In the static cases, all container information is known at the beginning and the proposed algorithm takes into account global information to obtain the AGV scheduling scheme. In the dynamic cases, the algorithm solves the problem in stages, with each stage considering a limited number of containers. The large filter width results in a long computational time, and thus the filter width is set to 500 in the realistic scale instances to

improve the efficiency of the algorithm. To validate the effectiveness of the proposed algorithm, the scheme provided by the B&B algorithm is compared to the real port scheduling strategies (FCFS and SETTF) and the TGH, and the comparison results between the TGH and the two strategies are also given. The results of static and dynamic cases are shown in Table 7 and Table 8.

In the static cases, the number of AGVs varies from 12 to 18, and the number of containers varies from 100 to 200, as shown in Table 7. The results in Table 7 demonstrate that with small filter width, the algorithm is able to solve the static problem in real-world instances within a reasonable amount of time. Although the reduction of the filter width affects the solution quality, the algorithm outperforms the two port scheduling strategies, and the average gaps are 13.35% and 28.89%, respectively. Moreover, although the solution quality of the B&B algorithm is better than the TGH with an optimality gap of 8.44%, the TGH is able to obtain a satisfactory solution within 1 min, where the optimality gaps compared with FCFS and SETTF strategies are 4.51% and 18.87%, respectively.

To further improve the efficiency of the algorithm and reduce the computational time, we solve the problem dynamically when the scale of the instances reaches thousands of containers, as shown in Table 8, where the column "$\overline{N}$" represents the number of containers considered and to be assigned in each stage. For example, the number of containers is 1000 and $\overline{N}$ is 20 in instance 65, which means that this instance is divided into 50 stages, and the proposed algorithm only considers the information of 20 containers in each stage to obtain the AGV scheduling scheme for these 20 containers. Once the solution for one stage is obtained, the information of equipment and container is updated and the algorithm proceeds to the next stage. The number of AGVs in the dynamic real-world cases is set to 18. In Table 8, we observe that as the $\overline{N}$ decreases, the computational time is significantly reduced and the solution quality of the proposed algorithm is slightly decreased. This is because a large $\overline{N}$ value represents more container information being considered by the algorithm at each stage. When the scale of the instances reaches 7000 containers, the algorithm is still able to obtain results in about one hour. This result indicates that the proposed model can indeed help port operators to significantly reduce the completion time of the equipment, thus achieving higher productivity. To obtain a satisfactory solution in a short amount of time, we also conducted computational experiments of the TGH for thousands of containers. The results in Table 8 show that the TGH is slightly inferior to the B&B algorithm, but the CPU time of the TGH is much lower than that of the B&B algorithm. Moreover, the TGH outperforms the FCFS and SETTF strategies with optimality gaps of 3.40% and 12.99%, respectively.

7. Conclusions

In this paper we investigate the AGV dispatching and routing problem considering congestion and conflict during AGV transportation with bidirectional travel paths and multiple optional routes between any two processing locations (HPs of blocks and QCs). The modeling of AGV movements and conflict is one of the main challenges in this study, and a MIP model is proposed to minimize the makespan of all containers. A tailored branch-and-bound algorithm is developed to solve the problem. The computational experiments indicate that the proposed algorithm can find near-optimal solutions within a reasonable amount of time, while Gurobi cannot obtain a feasible solution within the time limit as the scale increases. The proposed algorithm significantly outperforms the two port scheduling strategies, and the average optimality gaps are 13.39% and 21.12% for all scale instances. The real-world case study is conducted to test the efficiency and effectiveness of the proposed algorithm, and a two-stage greedy heuristic algorithm is developed to obtain a satisfactory solution in a short amount of time.

Final results propositi on

Some insights are also shown through the computational experiments. The accelerating techniques can effectively reduce the computational time, but accelerating techniques 1, 2 and 3 will affect the solution quality, while technique 4 will not. Increasing the number of AGVs can help to reduce the makespan, but a greater number of AGVs is not necessarily ideal. It is also found that the storage locations of containers will affect the makespan. To reduce the AGV conflicts and makespan, containers shall be stored in blocks closer to their loading or unloading QCs.

Future research can focus on uncertain factors, such as the uncertain quayside operation time, which is known in this paper. In practice, the container transfer between the QC and the AGV requires remote operational assistance with the operator to accurately place containers on the AGV or pick them up from the AGV, which leads to the uncertainty of quayside operation time. Moreover, how to reduce the computational time without affecting the solution quality is our future work as well.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science of China (71671021).

Appendix A. An explanation of constraint (23)

Constraint (23) avoids the opposite direction conflict of different AGVs on the same horizontal path. For horizontal actions (m,i,α_1) and (n,j,α_2), the constraint will only activate when the following three conditions are true.

- Condition a. Two horizontal actions are conducted on the same horizontal path.
- Condition b. The starting location of action (m,i,α_1) is on the left of the starting location of action (n,j,α_2).
- Condition c. The finishing location of action (m,i,α_1) is on the right of the finishing location of action (n,j,α_2), and the two actions have opposite directions.

Together these conditions define the situation when two horizontal actions have opposite direction conflicts, and all three conditions are true in Fig. 6 (a). As condition a is true,

$$P^{y}_{(m,i,\alpha_1)} + P^{y}_{(n,j,\alpha_2)} = 2, y \in Y^R$$

As both conditions b and c are true,

$$\sum_{x=1}^{x'}\left(P^x_{(m,i,\alpha_1-1)}+P^x_{(n,j,\alpha_2)}-P^x_{(m,i,\alpha_1)}-P^x_{(n,j,\alpha_2-1)}\right)\Big\langle 2, for x_3 \le x' or x' < x_2$$

$$\sum_{x=1}^{x'}\left(P^x_{(m,i,\alpha_1-1)}+P^x_{(n,j,\alpha_2)}-P^x_{(m,i,\alpha_1)}-P^x_{(n,j,\alpha_2-1)}\right)=2, for x_2 \le x' < x_3$$

Thus, $x_2 \le x' < x_3$ will activate constraint (23), and the opposite direction conflicts can be formulated as follows.

$$\begin{aligned}
&U^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)}+U^{AGV}_{(n,j,\alpha_2)(m,i,\alpha_1)}+2-P^y_{(m,i,\alpha_1)}-P^y_{(n,j,\alpha_2)}+2-\sum_{x=1}^{x'}\left(P^x_{(m,i,\alpha_1-1)}+P^x_{(n,j,\alpha_2)}-P^x_{(m,i,\alpha_1)}-P^x_{(n,j,\alpha_2-1)}\right)\ge 1,\\
&\qquad \forall(m,i,\alpha_1),(n,j,\alpha_2)\in W^H,\forall y\in Y^R,\forall x'\in X^R\\
&U^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)}+U^{AGV}_{(n,j,\alpha_2)(m,i,\alpha_1)}+3-P^Y_{(m,i,\alpha_1),y}-P^Y_{(n,j,\alpha_2),y}-\sum_{x=1}^{x'}\left(P^X_{(m,i,\alpha_1-1),x}+P^X_{(n,j,\alpha_2),x}-P^X_{(m,i,\alpha_1),x}-P^X_{(n,j,\alpha_2-1),x}\right)\ge 0,\\
&\qquad \forall(m,i,\alpha_1),(n,j,\alpha_2)\in W^H,\forall y\in Y^R,\forall x'\in X^R
\end{aligned}\tag{23}$$

Constraint (23) is activated for actions $(m,i,3)$ and $(n,j,3)$ in Fig. 6 (a), where the two actions are located on the same horizontal path (condition a is true), and have opposite directions with overlapping parts of the travel path (both conditions b and c are true). As shown in the example, AGV l' first conducts action $(n,j,3)$. After action $(n,j,3)$ is completed, AGV l will conduct action $(m,i,3)$. The colored square shows the minimum avoidance time $t^{AGV}_{(m,i,2)(n,j,3)}$ of container (m,i).

Appendix B:. An explanation of constraint (26)

For horizontal action (m,i,α_1) and (n,j,α_2) in the AGV seaside operation area, constraint (26) will only activate when the following three conditions are true.

- Condition d. Two horizontal actions are conducted on the same seaside horizontal path, i.e., the quayside operation of container (m,i) is conducted on the same seaside horizontal path as action (n,j,α_2).
- Condition e. The quayside operation location of container (m,i) is between the starting location and finishing location of action (n,j,α_2).
- Condition f. The quayside operation of container (m,i) is conducted simultaneously with action (n,j,α_2).

In Fig. 6 (b), all three conditions are true for actions (m,i,α_1) and (n,j,α_2),

$$1-U^{QC}_{(m,i)(n,j,\alpha_2)}+2-P^y_{(m,i,\alpha_1)}-P^y_{(n,j,\alpha_2)}+\left|\sum_{x=1}^{O_{(m,i)}}P^x_{(n,j,\alpha_2)}-\sum_{x=O_{(m,i)}+1}^{x_R}P^x_{(n,j,\alpha_2-1)}\right|=0,\forall y\in Y^S$$

Thus, the conflict between the AGV quayside operation and AGV transportation can be formulated as follows. If all three conditions are true, the time for action (n,j,α_2) to reach the quayside operation location must be later than the finishing time of the quayside operation for container (m,i), where $t^{AGV}_{(n,j,\alpha_2-1)(m,i,\alpha_1)}$ indicates the time from the starting position of action $(n,j,3)$ to the quayside operation location of container (m,i).

$$\left(3-U^{QC}_{(m,i)(n,j,\alpha_2)}-P^Y_{(m,i,\alpha_1),y}-P^Y_{(n,j,\alpha_2),y}+\left|\sum_{x=1}^{O_{(m,i)}}P^X_{(n,j,\alpha_2),x}-\sum_{x=O_{(m,i)}+1}^{x_R}P^X_{(n,j,\alpha_2-1),x}\right|\right)M+T^{Start}_{(n,j,\alpha_2)}+t^{AGV}_{(n,j,\alpha_2-1)(m,i,\alpha_1)}\ge T^Q_{(m,i)}+G^Q_{(m,i)},$$

$$\forall(n,j,\alpha_2)\in W^H,\forall y\in Y^S,\forall(m,i)\in D\ \forall\alpha_1\in\{0\}\ or\ \forall(m,i)\in L\forall\alpha_1\in\{3\}\tag{26}$$

Constraint (26) is activated for the quayside operation of container (m,i) and action $(n,j,3)$ in Fig. 6 (b). As shown in Fig. 6 (b), the quayside operation and the action are conducted simultaneously ($U^{QC}_{(m,i)(n,j,3)}=1$), the quayside operation location is between the starting location and finishing location of the action ($P^{x_1}_{(n,j,3)}-P^{x_2}_{(n,j,2)}=0$), and the quayside operation is conducted on the same seaside horizontal path as the action ($2-P^{y_1}_{(m,i,\alpha')}-P^{y_1}_{(n,j,3)}=0,\alpha'=1,(m,i)\in D\ or\ \alpha'=3,(m,i)\in L$). As shown in the example, the quayside operation of container (m,i) and action $(n,j,3)$ conflict, and the waiting time for action $(n,j,3)$ is as follow, where $t^{AGV}_{(n,j,2)(m,i,\alpha')}$ indicates the time from the starting position of action $(n,j,3)$ to the quayside operation location of container (m,i).

$$\left(T^Q_{(m,i)}+G^Q_{(m,i)}\right)-\left(T^{Start}_{(n,j,2)}+t^{AGV}_{(n,j,1)(n,j,2)}\right)-t^{AGV}_{(n,j,2)(m,i,\alpha')},\alpha'=0,(m,i)\in D\ or\ \alpha'=3,(m,i)\in L$$

Appendix C:. The explanation of Fig. 7

The sequence and timing of the unloading container (m,i) and the loading container (n,j) are shown in Fig. 7. To show an example of what **ADRP** are according to the model, we give the values of the variables in the figure. Because there are too many variables involved, we only show the binary variables equal to 1, the relationship between the time variables and some other variables. The variables of the job assignment and the job sequence for AGV are as follows, where (h_1,k_1) and (h_2,k_2) are the jobs before and after the AGV transports container (m,i), and (h_3,k_3) and (h_4,k_4) are the jobs

Table C1
The vertical and horizontal paths of container (m,i) for AGV transportation.

Variables	$X^{position}_{(m,i,0)}$	$X^{position}_{(m,i,1)}$	$X^{position}_{(m,i,2)}$	$X^{position}_{(m,i,3)}$	$X^{position}_{(m,i,4)}$	$Y^{position}_{(m,i,0)}$	$Y^{position}_{(m,i,1)}$	$Y^{position}_{(m,i,2)}$	$Y^{position}_{(m,i,3)}$	$Y^{position}_{(m,i,4)}$
Vertical path	x_1	x_2	x_3	x_4	x_5	–	–	–	–	–
Horizontal path	–	–	–	–	–	y_1	y_2	y_3	y_4	y_5

Table C2
The vertical and horizontal paths of container (n,j) for AGV transportation.

Variables	$X^{position}_{(n,j,0)}$	$X^{position}_{(n,j,1)}$	$X^{position}_{(n,j,2)}$	$X^{position}_{(n,j,3)}$	$X^{position}_{(n,j,4)}$	$Y^{position}_{(n,j,0)}$	$Y^{position}_{(n,j,1)}$	$Y^{position}_{(n,j,2)}$	$Y^{position}_{(n,j,3)}$	$Y^{position}_{(n,j,4)}$
Vertical path	x_6	x_7	x_8	x_9	x_{10}	–	–	–	–	–
Horizontal path	–	–	–	–	–	y_6	y_7	y_8	y_9	y_{10}

corresponding to container (n,j).

$$Z_{(h_1,k_1)(m,i),l} = 1Z_{(m,i)(h_2,k_2),l} = 1Z_{(h_3,k_3)(n,j),l'} = 1Z_{(n,j)(h_4,k_4),l'} = 1$$

In the **ADRP model**, the sequence of AGV actions for an AGV must be ensured (constraints (22) and (28)), thus guaranteeing the time relationship between AGV actions during AGV transportation (constraint (36)). The time relationship between quayside operations, yard side operations and AGV actions is guaranteed by constraints (30) to (35). And only when conflict arises, the sequence of AGV actions for different AGVs must be guaranteed to avoid conflict. Therefore, for containers (m,i) and (n,j) in the example, action $(m,i,1)$ needs to be conducted before action $(n,j,3)$, i.e., $U^{AGV}_{(m,i,1)(n,j,3)} = 1$. The binary variables that equal to 1 are shown as follows.

$$\begin{aligned} U^{AGV}_{(n,j,1)(n,j,2)} &= 1U^{AGV}_{(n,j,1)(n,j,3)} = 1U^{AGV}_{(n,j,1)(n,j,4)} = 1U^{AGV}_{(n,j,2)(n,j,3)} = 1U^{AGV}_{(n,j,2)(n,j,4)} = 1U^{AGV}_{(n,j,3)(n,j,4)} = 1U^{AGV}_{(m,i,1)(m,i,2)} = 1U^{AGV}_{(m,i,1)(m,i,3)} = 1U^{AGV}_{(m,i,1)(m,i,4)} \\ &= 1U^{AGV}_{(m,i,2)(m,i,3)} = 1U^{AGV}_{(m,i,2)(m,i,4)} = 1U^{AGV}_{(m,i,3)(m,i,4)} = 1U^{AGV}_{(m,i,1)(n,j,3)} = 1 \end{aligned}$$

We assume that the vertical and horizontal paths of the AGV actions for containers (m,i) and (n,j) are shown in Table C1 and Table C2. The locations of the AGV actions are represented by variables $X^{position}_{(m,i,\alpha)}$ and $Y^{position}_{(m,i,\alpha)}$, where $X^{position}_{(m,i,0)}$ and $Y^{position}_{(m,i,0)}$ indicate the starting paths of container (m,i), and $X^{position}_{(m,i,\alpha)}, Y^{position}_{(m,i,\alpha)}, \alpha \in \{1,2,3,4\}$ indicate the finishing paths. The relevant binary variables that equal to 1 are shown as follows.

$$P^X_{(m,i,0),x_1} = 1P^X_{(m,i,1),x_2} = 1P^X_{(m,i,2),x_3} = 1P^X_{(m,i,3),x_4} = 1P^X_{(m,i,4),x_5} = 1$$

$$P^Y_{(m,i,0),y_1} = 1P^Y_{(m,i,1),y_2} = 1P^Y_{(m,i,2),y_3} = 1P^Y_{(m,i,3),y_4} = 1P^Y_{(m,i,4),y_5} = 1$$

$$P^X_{(n,j,0),x_6} = 1P^X_{(n,j,1),x_7} = 1P^X_{(n,j,2),x_8} = 1P^X_{(n,j,3),x_9} = 1P^X_{(n,j,4),x_{10}} = 1$$

$$P^Y_{(n,j,0),y_6} = 1P^Y_{(n,j,1),y_7} = 1P^Y_{(n,j,2),y_8} = 1P^Y_{(n,j,3),y_9} = 1P^Y_{(n,j,4),y_{10}} = 1$$

The conflict of constraint (26) does not occur in this example, thus variable $U^{QC}_{(m,i)(n,j,\alpha)}$ is equal to 0. The intermediate variable $t^{AGV}_{(m,i,\alpha_1)(n,j,\alpha_2)}$ can be obtained by constraint (39). The time relationship between quayside operations, yard side operations and AGV actions can be derived from Fig. 7. In this example, the conflict occurs for actions $(m,i,1)$ and $(n,j,3)$, and the time relationship between the two actions is as follows.

$$T^{Start}_{(n,j,3)} \geq T^{Start}_{(m,i,1)} + t^{AGV}_{(m,i,0)(m,i,1)}$$

Appendix D:. The proof of Proposition 1

Proposition 1. *If the loaded travel route of each container is the shortest, then the obtained AGV operation scheme is the optimal solution without considering conflict.*

Proof. AGV transportation includes two parts: loaded travel and empty travel. If both the loaded and empty travel routes are the shortest, then the overall AGV travel route is the shortest route. The loaded travel route is already the shortest, and the empty travel route is determined by the vertical path of the quayside and yard side operations.

There are three conditions for the empty travel route of "unloading container to loading container", as shown in Fig. D1. Condition 1 is shown in Fig. D1 (a). The loaded travel route of the first container is the shortest route, and the distance between the finishing vertical path of the first container and the finishing vertical path of the second container is fixed. Condition 2 is shown in Fig. D1 (b). The distance between the starting vertical path of the first container and the finishing vertical path of the second container is fixed. As shown in Fig. D1 (c), condition 3 indicates that the empty travel route is the shortest route if the loaded travel route is the shortest route. The three conditions can explain all the empty travel conditions of "unloading container to loading container". The conditions not shown in Fig. D1 have the same concepts as the three conditions.

Since the vertical path of the quayside operation for each container is known, the empty travel distance of the "loading container to unloading container" is fixed. Therefore, if the loaded travel route is the shortest route, the overall AGV travel route is the shortest route. □

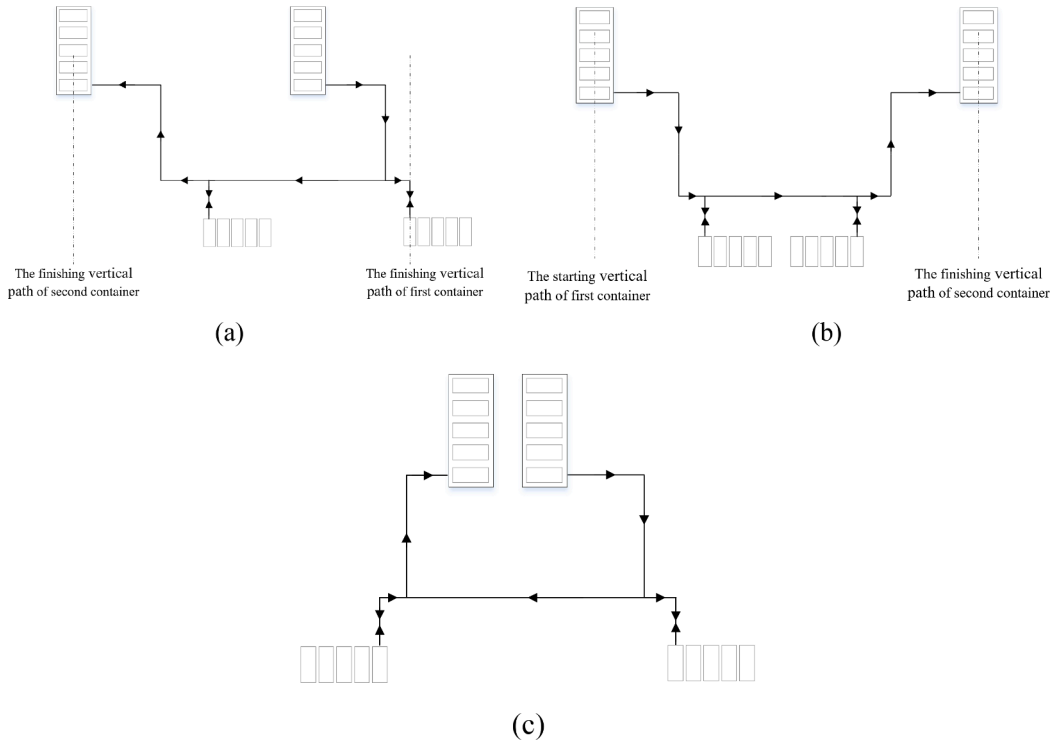


Fig. D1. The conditions for the empty travel route.

Appendix E:. The two-stage greedy heuristic algorithm

The two-stage greedy heuristic (TGH) algorithm solves the **ADRP** in two stages, as shown in **Algorithm E1**. Based on the greedy strategy, the first stage of the algorithm (rows 06 through 20) assigns containers to AGVs without considering conflict, and the second stage (rows 22 through 30) obtains conflict-free routes for AGVs.

In row 07 of the first stage algorithm, $C^{'}$ is the set of containers considered in each algorithm loop, in other words, the algorithm can only select the next job from $C^{'}$. Rows 08 through 16 correspond to the update of the evaluation matrix $\Lambda((m,i),l)$, where $\Lambda((m,i),l)$ is a $|C^{'}| \times |B|$ matrix. Assuming that the container and AGV corresponding to a position in the matrix are $(m,i)_1$ and l_1, respectively, then the value of this position is the current makespan for assigning container $(m,i)_1$ to AGV l_1 without considering conflict (row 14). If the scheme of $(m,i)_1$ and l_1 cannot form multiple QC double cycling or the scheme violates the sets ψ_1,ψ_2, the evaluation value of the scheme is a sufficiently large positive constant, which means that the scheme will not be selected. In rows 17 through 19, the scheme with minimum completion time is selected, while the container set C_1 and the AGV operation list $O(l)$ are updated. Note that when there are multiple identical minimum evaluation values in the evaluation matrix $\Lambda((m,i),l)$, the pre-sequence container in the QC operation list and the AGV that completed the previous job first are selected. Once a scheme is selected in each algorithm loop, the evaluation matrix is updated to obtain the local optimal scheme based on the greedy strategy.

The second stage algorithm mainly obtains conflict-free routes according to the first come first service principle and greedy strategy. Based on the results of the first stage, the AGV that completes the preceding job first will be given priority to occupy travel path (rows 28 and 29). For the current container $(m,i)^*$ and AGV l^*, the conflict-free route with the shortest operation time is selected according to the greedy strategy (rows 25 through 27). The solution set, the container set and the AGV operation information are updated (rows 28 and 29) to obtain the final solution (row 31).

Algorithm E1 Two-stage greedy heuristic algorithm

01: data: B- the set of AGVs, C_1, C_2- the set of containers, $C^{'}$- the set of containers considered in each algorithm loop, $R_{(m,i)}$- the set of optional routes for container (m,i), ψ_1,ψ_2- the sets of container pairs (m,i,n,j) that (m,i) must precede (n,j), M- a very large positive number
02: **Result:** Ω- solution
03: **procedure**
04: Ω←∅,$C^{'}$←∅
05: $O(l)$←∅, $\forall l \in B$ #$O(l)$ is the operation list of AGV l
06: **while** $C_1 \neq \varnothing$ **do** # the first stage greedy heuristic
07: update $C^{'}$ based on C_1
08: **for** $l \in B$ **do** # update evaluation matrix $\Lambda((m,i),l)$
09: **for** $(m,i) \in C^{'}$ **do**
10: **if** the assignment of container (m,i) and AGV l conflicts with ψ_1,ψ_2 **or** container (m,i) cannot form multiple QC double cycling with AGV l **do**
11: $\Lambda((m,i),l) = M$
12: **continue**
13: **end if**

(*continued on next page*)

(*continued*)

Algorithm E1 Two-stage greedy heuristic algorithm

$\Lambda((m,i),l) \leftarrow$ the evaluation value of the assignment $((m,i),l)$
end for
end for
$((m,i)^*,l^*) \leftarrow arg \min_{(m,i)\in C',l\in B} \Lambda((m,i),l)$
$O(l^*) \leftarrow (m,i)^*$ # container $(m,i)^*$ is assigned to AGV l^*
remove $(m,i)^*$ from C_1
end while
$T(l) \leftarrow 0, \forall l \in B$ # the time for AGV l to complete current container operation
while $C_2 \neq \varnothing$ **do** # the second stage greedy heuristic
$l^* = argmin_{l\in B} T(l)$ # The AGV that completed the previous job earliest is selected
$(m,i)^* \leftarrow$ the pre-sequence container in $O(l^*)$
for $r \in R_{(m,i)^*}$ **do**
rows 15 through 39 in **Algorithm 3** are used to generate the route r^* and the time list h^* with the shortest operation time that are compatible with Ω
end for
$\Omega = \Omega \cup ((m,i)^*, l^*, r^*, h^*)$
remove $(m,i)^*$ from C_2, update $T(l), O(l)$
end while
return Ω
end procedure

~~Appendix F:. The dynamic problem algorithm~~

In the dynamic problem, the **Algorithm F1** is conducted to obtain the solution when the set of new containers of stage d is gotten. For stage 1 in **Algorithm F1**, the AGV and QC operation states are both idle, and the set of unassigned containers in stage 0 is $C_0^U \leftarrow \varnothing$. In row 06, the number of containers to be assigned is N_d, which indicates that not all containers in set C will be assigned. The AGV operation states refer to the time, location, and route after the jobs assigned to AGV are completed, and the QC operation states refer to the time and location.

Algorithm F1 Dynamic problem algorithm of stage d

01: data: B- the set of AGVs, C_d^N- the set of new containers in stage d, C_{d-1}^U- the set of unassigned containers in stage $d-1$, N_d- the number of containers to be assigned in stage d
02: **Result:** Θ_d- solution in stage d, C_d^U- the set of unassigned containers
03: **procedure**
04: initialization of AGV and QC operation states
05: $C = C_d^N \cup C_{d-1}^U$ # the set of containers to be assigned in stage d
06: assign N_d containers from set C based on **Algorithm 1** and obtain solution Θ_d
07: update AGV and QC operation states
08: update the set of unassigned containers C_d^U
09: **return** Θ_d, C_d^U
10: **end procedure**

References

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

Bae, H. Y., Choe, R., Park, T., & Ryu, K. R. (2009). Comparison of operations of AGVs and ALVs in an automated container terminal. *Journal of Intelligent Manufacturing, 22*(3), 413–426.

Bai, D., Liang, J., Liu, B., Tang, M., & Zhang, Z. H. (2017). Permutation flow shop scheduling problem to minimize nonlinear objective function with release dates. *Computers & Industrial Engineering, 112*.

Chen, X., He, S., Zhang, Y., Tong, L. C., Shang, P., & Zhou, X. (2020). Yard crane and AGV scheduling in automated container terminal: A multi-robot task allocation framework. *Transportation Research Part C: Emerging Technologies, 114*, 241–271.

Choe, R., Kim, J., & Ryu, K. R. (2016). Online preference learning for adaptive dispatching of AGVs in an automated container terminal. *Applied Soft Computing, 38*, 647–660.

Debjit, R., René, D. K., & René, B. (2020). Modeling and design of container terminal operations. *Operations Research, 68*(3), 686–715.

Gharehgozli, A. H., Vernooij, F. G., & Zaerpour, N. (2017). A simulation study of the performance of twin automated stacking cranes at a seaport container terminal. *European Journal of Operational Research, 261*(1), 108–128.

Grunow, M., Gunther, H. O., & Lehmann, M. (2006). Strategies for dispatching AGVs at automated seaport container terminals. *OR Spectrum, 28*(4), 587–610.

Gupta, A., Roy, D., Koster, R. D., & Parhi, S. (2017). Optimal stack layout in a sea container terminal with automated lifting vehicles. *International Journal of Production Research, 55*(13), 3747–3765.

He, J., Tan, C., & Zhang, Y. (2019). Yard crane scheduling problem in a container terminal considering risk caused by uncertainty. *Advanced Engineering Informatics, 39*, 14–24.

Hoshino, S., Ota, J., Shinozaki, A., Hashimoto, H. (2004). Optimal design methodology for an AGV transportation system by using the queuing network theory. Alami R, Chatila R, Asama H, eds. *Distributed Autonomous Robotic Systems*, vol. 6, (Springer, Tokyo), 411–420.

Hu, H., Chen, X., Wang, T., & Zhang, Y. (2019). A three-stage decomposition method for the joint vehicle dispatching and storage allocation problem in automated container terminals. *Computers & Industrial Engineering, 129*, 90–101.

Hu, Z. H., Sheu, J. B., & Luo, J. X. (2016). Sequencing twin automated stacking cranes in a block at automated container terminal. *Transportation Research Part C: Emerging Technologies, 69*, 208–227.

Jenny, N., Dirk, B., & Erwin, P. (2018). Container dispatching and conflict-free yard crane routing in an automated container terminal. *Transportation Science, 52*(5), 1059–1076.

Ji, S. W., Di, L., Chen, Z. R., & Guo, D. (2020). Integrated scheduling in automated container terminals considering AGV conflict-free routing. *Transportation Letters The International Journal of Transportation Research, 2020*(2), 1–13.

Jiang, X. J., Xu, Y., Zhou, C., Chew, E. P., & Lee, L. H. (2018). Frame trolley dispatching algorithm for the frame bridge based automated container terminal. *Transportation Science, 52*(3), 722–737.

Ku, D., & Arthanari, T. S. (2016). On double cycling for container port productivity improvement. *Annals of Operations Research, 243*(1), 55–70.
Lau, H. Y. K., & Zhao, Y. (2008). Integrated scheduling of handling equipment at automated container terminals. *Annals of Operations Research, 112*(1), 665–682.
Li, H., Peng, J., Wang, X., & Wan, J. (2020). Integrated resource assignment and scheduling optimization with limited critical equipment constraints at an automated container terminal. *IEEE Transactions on Intelligent Transportation Systems*.
Lu, H., & Wang, S. (2019). A study on multi-ASC scheduling method of automated container terminals based on graph theory. *Computers & Industrial Engineering, 129*, 404–416.
Luo, J., & Wu, Y. (2015). Modelling of dual-cycle strategy for container storage and vehicle scheduling problems at automated container terminals. *Transportation Research Part E: Logistics and Transportation Review, 79*, 49–64.
Luo, J., Wu, Y., & Mendes, A. B. (2016). Modelling of integrated vehicle scheduling and container storage problems in unloading process at an automated container terminal. *Computers & Industrial Engineering, 94*, 32–44.
Luo, J., & Wu, Y. (2020). Scheduling of container-handling equipment during the loading process at an automated container terminal. *Computers & Industrial Engineering, 149*, Article 106848.
McKinsey & Company. (2018). The future of automated ports. <https://www.mckinsey.com/industries/travel-transport-and-logistics/our-insights/the-future-of-automated-ports>.
Miyamoto, T., & Inoue, K. (2016). Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems. *Computers & Industrial Engineering, 91*, 1–9.
Nguyen, V. D., & Kim, K. H. (2009). A dispatching method for automated lifting vehicles in automated port container terminals. *Computers & Industrial Engineering, 56*(3), 1002–1020.
Park, T., Choe, R., Ok, S. M., & Ryu, K. R. (2010). Real-time scheduling for twin RMGs in an automated container yard. *OR Spectrum, 32*(3), 593–615.
Roy, D., & De, K. R. (2018). Stochastic modeling of unloading and loading operations at a container terminal using automated lifting vehicles. *European Journal of Operational Research, 266*(3), 895–910.
Tian, Y., Wang, J., Chen, J., & Fan, H. (2018). Collaborative scheduling of QCs, L-AGVs and ARMGs under mixed loading and unloading mode in automated terminals. *Journal of Shanghai Maritime University, 39*(03), 14–21.
Tomazella, C. P., & Nagano, M. S. (2020). A comprehensive review of branch-and-bound algorithms: Guidelines and directions for further research on the flow shop scheduling problem. *Expert Systems with Applications, 158*, Article 113556.
Tommaso, A., Tolga, B., Gianpaolo, G., Emanuela, G., & Emanuele, M. (2018). Path and speed optimization for conflict-free pickup and delivery under time windows. *Transportation Science, 52*(4), 739–755.
Ulf, S., & Kathrin, F. (2017). Scheduling of different automated yard crane systems at container terminals. *Transportation Science, 51*(1), 305–324.
Vis, I. F., & Harika, I. (2004). Comparison of vehicle types at an automated container terminal. *OR Spectrum, 26*(1), 117–143.
Vis, I. F. A. (2006). Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research, 170*, 677–709.
Xin, J., Negenborn, R. R., Corman, F., & Lodewijks, G. (2015). Control of interacting machines in automated container terminals using a sequential planning approach for collision avoidance. *Transportation Research Part C: Emerging Technologies, 60*, 377–396.
Yang, Y., Zhong, M., Dessouky, Y., & Postolache, O. (2018). An integrated scheduling method for AGV routing in automated container terminals. *Computers & Industrial Engineering, 126*, 482–493.
Yang, X., Mi, W., Li, X., An, G., Zhao, N., & Mi, C. (2015). A simulation study on the design of a novel automated container terminal. *IEEE Transactions on Intelligent Transportation Systems, 16*(5), 1–11.
Zhong, M., Yang, Y., Yasser, D., & Octavian, P. (2020). Multi-AGV scheduling for conflict-free path planning in automated container terminals. *Computers & Industrial Engineering, 142*, Article 106371.
Zhen, L., Lee, L. H., Chew, E. P., Chang, D. F., & Xu, Z. X. (2012). A comparative study on two types of automated container terminal systems. *IEEE Transactions on Automation Science and Engineering, 9*(1), 56–69.
Zhen, L., Hu, H., Wang, W., Shi, X., & Ma, C. (2018). Cranes scheduling in frame bridges based automated container terminals. *Transportation Research Part C: Emerging Technologies, 97*, 369–384.
Zhou, C., Chew, E. P., & Lee, L. H. (2018). Information-based allocation strategy for GRID-Based transshipment automated container terminal. *Transportation Science, 52*(3), 707–721.