UNIVERSITY OF Southampton

University of Southampton Research Repository
ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

http://eprints.soton.ac.uk

UNIVERSITY OF SOUTHAMPTON

FACULTY OF BUSINESS AND LAW

School of Management

Modelling of Quayside Logistics Problems at Container Terminals

by

Jiabin Luo

Thesis for the degree of Doctor of Philosophy

November 2013

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF BUSINESS AND LAW

School of Management

Doctor of Philosophy

MODELLING OF QUAYSIDE LOGISTICS PROBLEMS AT CONTAINER TERMINALS

Jiabin Luo

Container terminals serve as an interface between marine and land transportation. Since the introduction of containerisation in 1960s, the number of containers handled worldwide has dramatically grown every year. With the increasing containerisation, nowadays container terminals are working at maximum capacity. Therefore, the efficiency of stacking and transportation of large number of containers to and from the quayside is critical to any container terminal.

We have investigated the integration of container-handling equipment (such as quay cranes, yard cranes, automated guided vehicles and straddle carriers) scheduling and container storage allocation problems in two types of container-handling system: one is automated container terminal, which represents the current container terminal development and the other is straddle-carrier system, which has been used by most European container terminals. For each type of container terminal, we have studied three integrated problems respectively considering container unloading process (during which containers are unloaded from a ship and delivered to the storage yard), container loading process (during which containers are picked up from the yard and delivered to the quayside to be loaded onto a ship) and dual-cycle process (unloading and loading of containers simultaneously). Our aims are to determine the optimal schedules of container-handling equipment and assign optimal yard locations for containers. The objective is to minimise the berth time of the ship, which is the most important factor to evaluate the efficiency of container terminals.

We have developed six models for the above problems. Optimal solutions can be obtained in small sizes of the problems under investigation; however, large-sized problems are hard to solve optimally in a reasonable time. Therefore, genetic algorithms are designed for each model to solve the problem in large sizes. The computational results show the effectiveness of the proposed models and heuristic approaches in dealing with problems in container terminals.

Table of Contents

List of Tables

List of Figures

DECLARATION OF AUTHORSHIP

I, Jiabin Luo

declare that the thesis entitled

Modelling of Quayside Logistics Problems at Container Terminals

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- Parts of this work have been published and presented as:

Publications

1. Luo, J. and Wu, Y. (2013). An integrated scheduling problem of container handling equipment in the loading operation at automated container terminals. In, Klement, E. P., Borutzky, W., Fahringer, T., Hamza, M. H., Uskov, V. (eds.) *Proceeding of the 12th IASTED International Conference on Artificial Intelligence and Applications* 2013, Austria (doi: 10.2316/P.2013.793-047).

2. Luo, J., Wu, Y., Halldorsson, A. and Song, X. (2011). Storage and stacking logistics problems in container terminals. *OR Insight*, Vol. 24, No. 4, pp. 256-275.

3. Luo, J., Wu, Y., Ma, T. and Zhang, D. An integrated modelling approach for the straddle carrier scheduling and storage allocation problem in the unloading process at container terminals. *European Journal of Operational Research* (**under revision**).

Conferences

1. Luo, J. and Wu, Y. (2013) An integrated scheduling problem of container handling equipment in the loading operation at automated container terminals. *The 12th IASTED International Conference on Artificial Intelligence and Applications*, Innsbruck, Austria, February 2013.

2. Luo, J. and Wu, Y. (2011) Optimisation of dual-cycle container handling process at seaport terminals. *INFORMS 2011 annual meeting,* Charlotte, NC, USA, November 2011.
3. Luo, J, Wu, Y. and Zhang, D. (2010) Modelling of vehicle scheduling and storage problems for import containers. *24th European Conference on Operational Research (EURO)*, Lisbon, Portugal, July 2010.
4. Luo, J. and Wu, Y. (2010) Vehicle scheduling and storage management at container terminals. *LASS conference*, University of Southampton, UK, April 2010 (poster).

5. Luo, J., Wu, Y. and Zhang, D (2010) Vehicle scheduling and storage problems at container terminals. *LANCS workshop*, Southampton, March 2010 (poster).

Signed: ……..

Date:……

Acknowledgements

Completing a Doctor of Philosophy degree is not only an academic achievement, but also a unique opportunity to develop personally. This thesis would never have been completed without the precious help and support I received from many people during my time at the University of Southampton and in the UK.

First of all, I would like to express my gratitude to my main supervisor, Dr. Yue Wu who gave me the opportunity to undertake my PhD, for her constant guidance, advice and encouragement throughout the years. She gave me important inspiration on the mathematical modelling and perspectives for solution approaches. I would also like to thank my supervisor, Professor Douglas Macbeth for his useful comments and suggestions, which have greatly helped in improving this work.

Second, many thanks to DP World Southampton, who provided me with an excellent guided tour of Southampton Container Terminal, which gave me invaluable and meaningful insights of the practical aspects and handling of real-life operations.

Third, I am grateful to everyone in CORMSIS (Centre for Operational Research, Management Sciences & Information Systems) at Southampton Management School for the great working environment I have enjoyed. It has been a valuable experience to be part of this leading group and I have benefited immensely from the seminars and discussions that were held.

Finally, a special thanks to my parents. They have always encouraged me to discover my potential and to realise my wishes, with love and enthusiasm.

Definitions and Abbreviations

AGV	Automated Guided Vehicle
ALV	Automated Lifting Vehicle
B&B	Branch and Bound
B&C	Branch and Cut
CTA	Container Terminal Altenwerder
ECT	Europe Combined Terminal
GA	Genetic Algorithm
GAPM	Genetic Algorithm Plus Maximum Matching
GYAP	General Yard Allocation Problem
HIA	Hybrid Insertion Algorithm
LV	Lifting Vehicle
MCF	Minimum Cost Flow
MIP	Mixed-Integer Programming
MLGA	Multi-Layer Genetic Algorithm
MT	Multi-Trailer
NP Hard	Nondeterministic Polynomial-time Hard
OFV	Objective Function Value
OR	Operations Research
PPT	Pasir Panjang Terminal
QC	Quay Crane

RMGC	Rail-mounted gantry Crane
RTGC	Rubber-tyred gantry Crane
SA	Simulated Annealing
SC	Straddle Carrier
SSAP	Storage Space Allocation Problem
TEU	Twenty-foot Equivalent Unit
TS	Tabu Search
YC	Yard Crane

Chapter 1. Introduction

1.1 Background

Introduced about half a century ago, containers are large steel boxes of standard dimensions designed for easy and fast handling of cargos. Containers are eight feet wide, and come in three standard lengths - 20 feet, 40 feet or 45 feet. They have a height of either 8.5 feet or 9.5 feet. A 20-foot container carries up to about 28 tonnes of cargo with a volume up to 1000 cubic feet (Christiansen *et al.*, 2007). Before the introduction of containerisation, cargo was transported piece by piece, which made the transportation of goods very expensive and inefficient. One of the most significant benefits resulting from the introduction of containers is the resulting decrease in the potential risk of damage to goods, and the reduction in the need for re-packing between different transportation modes. Today, over 60% of the world's deep-sea general cargo is transported in containers; on some routes, particularly those between economically strong and stable countries, containerisation is up to 100% (Steenken, Voß and Stahlbock, 2004).

Figure 1.1: Aerial view of a typical container terminal, Hamburg (source: www.maritimejournal.com)

Container terminals have been playing an important role in the global business as the inter-modal interfaces between the seaside and land-side transportation systems. Nowadays, every major container terminal is dominated by stacks of colourful containers. Figure 1.1 is an aerial view of a typical container terminal. For most container terminals, three types of equipment are generally used; these are quay cranes (QCs), yard cranes (YCs) and vehicles, i.e. trucks, automated guided vehicles (AGVs), and straddle carriers (SCs), among others. Although container terminals vary in size, function and geometric layout, essentially, the handling operations in container terminals throughout the world can be divided into three types: vessel operations associated with containerships, receiving/delivery operations of trucks/trains, and container handling and storage operations in the yard (Lee and Kim, 2010).

Recently, there has been a tremendous growing in the container shipping business. In 2011, global container-port throughput has increased by an estimated 5.9 % to 572.8 million TEUs (container port throughput is measured in terms of TEUs - twenty-foot equivalent units), its highest level ever (United Nations Conference on Trade and Development, 2012). As a result of this dramatic increase in container shipping, the number of container terminals and the competition among them has increased considerably. This increased volume is also causing a number of problems at container terminals. As terminals are faced with more and more containers to handle in a limited time, nowadays the world's container terminals are reaching their capacity limits, leading to delay in the handling of containers and thus causing traffic congestion at container terminals. Together with the enormous increase in container shipping, the ship size has increased as well. The first container ship to sail internationally, in 1966, was only able to carry 200, forty-foot containers (Liu, 2010). To date, the largest container ship can carry 11,000 TEUs (Wong and Kozan, 2010). In early 2012, the average capacity of container ships reached 3074 TEUs, a further increase of 6% on the previous year (United Nations Conference on Trade and Development, 2012). However, as ships increase in size, their operating costs at container terminals are also rising. This is because a larger ship carries more containers so it requires more time to unload/load containers from/onto it and also requires more powerful quay cranes to serve it.

There are several approaches to dealing with the above problems. One way is to design a new terminal layout or expand the size of the current terminal, which demands huge investment in the infrastructure of container terminals and may take years to finish. Another way is to increase the volume of container-handling equipment used in the terminal in order to speed up the container-handling processes. However, this may result in interferences between the cranes/vehicles and also cause the traffic congestions, which would further affect the productivity and efficiency of container terminals. Replacing the old equipment with more efficient equipment is also a possible approach, but again this requires a significant financial investment. Therefore, how to use the existing infrastructure and equipment more efficiently remains an essential concern in the container terminals. Currently, container terminal operators are seeking to find optimisation tools and algorithms to improve their productivity, i.e. adopting operations research (OR) techniques, the efficiency of which has been proved in other logistics industries; for example, in the airline industry (Klabjan, 2005).

Maritime logistics and container terminal operations represent a more recent OR field, and have attracted increasing attention from researchers. We study two types of container terminals in this research: the automated container terminal and the straddle-carrier system. Specifically, we study the integration of scheduling cranes/vehicles and container storage allocation problems during container unloading, loading and dual-cycle process. In the literature on container terminal operations, there are many studies on the operational problems in both conventional men-driven container terminals and automated container terminals. The recent development of automated container terminals has led to an increasing interest in studying the scheduling problems in such terminals. This is because the automated container terminal represents the on-going trend of the traditional terminal, in which all or some of the container-handling equipment is automated. Since, in the case of automation, there is a lack of physical human input, efficient scheduling and coordination of different resources (handling equipment, yard locations, etc.) are crucial to improve the overall performance of the automated container terminal. So far, however, there has been little research on the integration of operational problems in automated container terminals. This might due to the difficulty in modelling such problems and providing the integrated solutions. Another type of container terminal considered in this research is the straddle-carrier (SC)

system. To date, only very limited attention has been paid to the optimisation problems in this system; this is because the straddle-carrier system is currently only adopted in Europe and thus only accounts for a small portion of terminals in the world. However, the SC system represents the typical direct transfer system (Kim, 2005) , in which no yard cranes are used. Moreover, as the straddle carrier is the second most commonly used equipment for stacking containers in the container yard (Wiese, Kliewer and Suhl, 2009), it is also important to analyse the problems in the SC system. Therefore, this research seeks to address the operational optimisation problems in these two types of container terminals. Terminal managers are seeking to improve the operational efficiency of container terminals, which is most commonly evaluated by the ship's berth time. Therefore, for all the problems considered in this study, we aim to minimise the ship's berth time by the ways of coordinating different container handling equipment and optimally utilising container storage yard space. This will ultimately increase the operations efficiency and reduce the operating costs of the container terminals.

1.2 Research objectives

The main aim of this research is to provide mathematical models and algorithms for efficient scheduling of container-handling equipment and utilisation of container yard storage space. The specific objectives are:

- To investigate the integrated scheduling of cranes/vehicles and container storage allocation problems in two types of container-handling systems: automated container terminal and straddle-carrier system. In brief, the solutions of the integrated models provide the decisions on the schedule for each piece of equipment (vehicle/crane) and the location of each (import) container.
- To provide heuristic algorithms for solving the real-life large-scale problems. Because scheduling problems at container terminals are usually NP-hard, it is difficult to solve these problems by exact methods (such as branch and bound method) in a reasonable computational time. On the contrary, heuristic methods can deal with this computational difficulty and aim to provide good solutions with the minimum computational effort.

- To help improving the efficiency and throughput of the container terminal by adopting the integrated approaches considered in this research. All of our models aim to minimise the ship's berth time, which is the most critical factor for evaluating the terminal's performance.

1.3 Contributions

This research makes several contributions in the following aspects:

- We are among the first to mathematically model the integration of crane/vehicle scheduling and storage allocation problems in container terminals in three specific problems: (1) the integration of vehicle scheduling and storage allocation problems in the container unloading process; (2) the integrated scheduling of container-handling equipment in the container loading process, and (3) the integration of vehicle scheduling and storage allocation problems when both unloading and loading processes are considered simultaneously.
- Our research is the first to analyse the operations in one type of system: straddle-carrier (SC) system, which has to date received only limited attention from the field. The main characteristics of such a system and how it differs from the indirect system (such as the automated container terminal in this study) are investigated.
- We present new ways to design the genetic algorithms (GAs) with the purpose of providing solutions to the considered problems in reasonable computational times. For each particular problem, we carry out a number of computational tests to verify the performances of GAs.
- Our research contributes to the efficiency of container terminals in several ways: first, the proposed approaches can be applied in real-life terminal operations and the obtained integrated solutions can lead to much better performance of container terminals; second, we present a range of important issues (such as vehicle scheduling, storage problems and others) for the improvement of a terminal's efficiency and thus provide an insight for terminal managers on the operational planning, yard layout and different handling strategies in two handling systems.

1.4 Structure of the thesis

The main part and primary contributions of this thesis are represented by two chapters, chapters 3 and 4, concerning the mathematical models and heuristic algorithms in the automated container terminal and straddle-carrier systems respectively. The thesis consists of five chapters and is organised as follows:

This chapter has set out the basic background to the problem, relating to the development and expansion of container terminals, maritime logistics and container-handling processes, which provides a general introduction to the thesis.

Chapter 2 provides an extensive overview of the problems considered in container-handling operations in the literature. We review the related studies such as the quay crane scheduling, straddle carrier scheduling, automated guided vehicle scheduling, yard crane scheduling and storage allocation problems. Some studies devoted to integrated problems have been identified in the literature. We also give a brief review of the commonly used heuristic methods such as tabu search (TS) and simulated annealing (SA), which have been applied to container terminals.

In chapter 3, we develop three new integrated models for the vehicle/crane scheduling and storage allocation problems at automated container terminals. The automated container terminal considered in our study uses quay cranes, automated guided vehicles and yard cranes for handling containers. The models consider container unloading, container loading and dual-cycle processes separately. All three models are formulated as the mixed-integer programming (MIP) model. The objectives are to minimise the ship's berth time. Due to the computational complexity of the problems, we design different genetic algorithms (GAs) to solve the practical-sized problems. Numerical experiments are carried out to evaluate the performance of the designed GAs and to compare the GA results with the results obtained from the exact solution approach (e.g. Branch and Bound).

In chapter 4, a further three new models for the integration of vehicle/crane scheduling and storage allocation problems in the straddle-carrier system are developed. In such a direct transfer system, only quay cranes and straddle carriers are employed for handling

and delivering containers. We point out the differences with the modelling environment considered in chapter 3 and study container unloading, loading and dual-cycle processes separately in such system. The formulations consist of considerably different variables and constraints; in particular, the restrictions on the buffer area make up a big difference compared to the models in chapter 3. We adopt the MIP modelling approach and GA for the solution methods. For each model, we report the computational results and provide a comprehensive analysis of the outcome and performance of the proposed solution methods.

Finally, chapter 5 presents conclusions and perspectives for future work.

Chapter 2. Literature review on quayside logistics problems

Container terminals not only serve as the multi-modal interface between marine and land transportation systems - they also work as a hub for the trans-shipment of containers between different maritime shipping lines.

Figure 2.1 shows the flow of containers in the unloading and loading processes through a terminal. Typically, during the loading process, after being received from the delivering external trucks and trains, export containers are allocated to the storage yard for temporary storing then loaded onto the ship; during the unloading process, import containers are transported to the storage yard after being discharged from the ships before being loaded onto external trucks and trains for onward delivery.

Therefore, the main productivity of a container terminal can be measured in terms of two types of operations: (1) ship operations, in which containers are discharged from and loaded onto ships; and (2) receiving and delivery operations, in which containers are transferred to and from outside trucks/trains (Kim and Park, 2004). The main objective of a modern container terminal is to minimise the transportation time and costs of containers while maximising the utilisation of container storage space both on the ship and in the yard. In all logistical activities, keeping the goods moving towards their destination is the goal, and the cost of having a ship alongside a port berth while unloading and loading is significant.

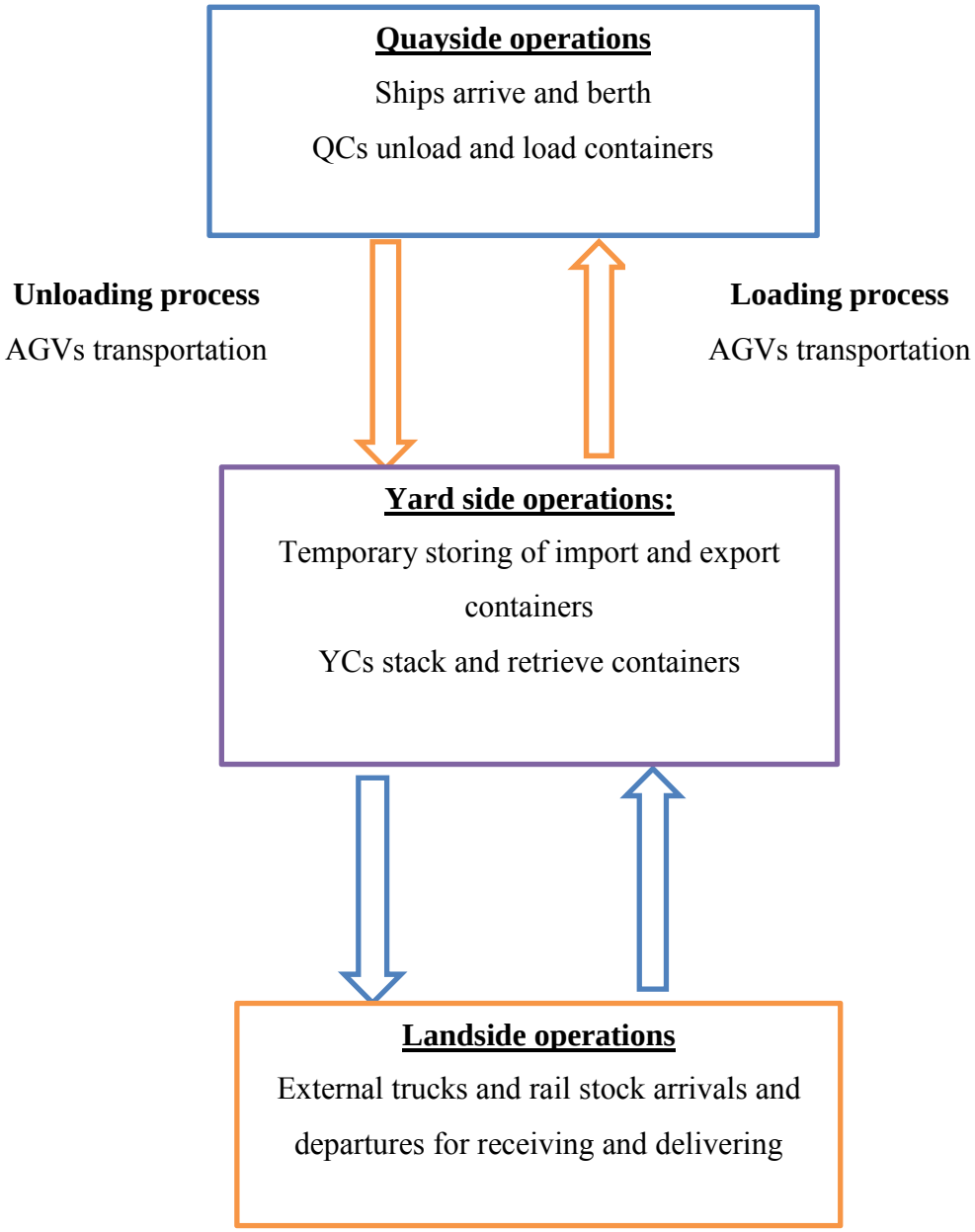


Figure 2.1: Unloading and loading processes in a container terminal

Due to the importance of container logistics, increasing research has been devoted to container terminal operations problems. Steenken, Voß and Stahlbock (2004) provided an overview of the process and classified the problems that occur in container terminals. In particular, they found that effectively allocating storage locations to containers and scheduling container-handling equipment are important factors for improving the

overall performance of the container terminal. Günther and Kim (2006) discussed planning and logistics issues of container terminals, which can be considered at three levels: terminal design, operative planning and real-time control. In this thesis, we study the problems within the level of operative planning consisting of various logistics processes at the terminal. Vacca, Bierlaire and Salani (2007) provided an overview of decision problems arising in the container terminal operations. They also discussed the new trends and topics in the field of shipping/containerisation, such as the competition and cooperation issues. Stahlbock and Voß (2008) updated and extended their review of the literature by presenting papers on the most current logistics processes and operations in container terminals. They also stated that the key factors for the optimum operations of container terminal are the efficiency of the stacking and the transport of this large number of containers to and from the ship's side.

In this thesis, we focus on the quayside logistics problems at container terminals and aim to introduce a set of integrated approaches. Quayside logistics as considered here involves the unloading and loading operations of containers at the quayside, and the related issues, such as the scheduling of vehicles, quay cranes, yard cranes and container storage allocation problems. To address this, this chapter reviews the literature in the following categories: scheduling of container-handling equipment (vehicles, quay cranes, yard cranes) and container storage problems. In addition, previous studies on integrated approaches and heuristic methods are also included.

2.1 Scheduling of container-handling equipment

Most container terminals have different characteristics, which demonstrate the size of the terminal, the equipment types used and the layout of the container yard (Wiese, Kliewer and Suhl, 2009). We consider the following two types of container terminal in this thesis.

Automated container terminal. Nowadays, to increase the terminal's productivity and reduce labour costs, many terminals have started to employ automated equipment so as to cope with the ever-increasing container traffic passing through container terminals. These terminals are known as automated container terminals. Among the different types of automated vehicle that are used in these terminals, automated guided vehicles are the

most representative (Bae *et al.*, 2011). In a typical automated container terminal, there are three main container handling equipment involved: quay cranes (QCs), automated guided vehicles (AGVs) and yard cranes (YCs). QCs are engaged for unloading/loading containers from/to the ship at the quayside, as shown in Figure 2.2. Because QC is the most expensive single unit of container-handling equipment, quayside operations to coordinate with the QCs form the key bottleneck operation at container terminals. AGVs are built with advanced technology to transport containers between the quayside and the storage yard on the pre-defined paths without being driven by men, as described in Figure 2.3 and Figure 2.5. YCs are equipped in the storage yard for the retrieving and stacking of containers, as shown in Figure 2.4. The storage yard is for temporary storing of containers; it is divided into different blocks and each slot within the block is specified by bay, row and tier numbers. Often, stacks are separated into blocks for export, import, special and empty containers.

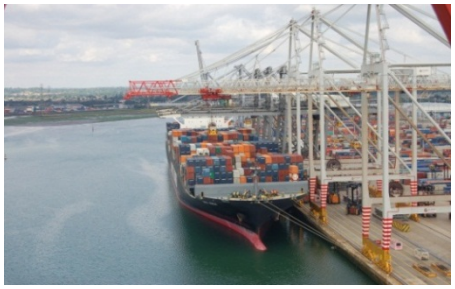


Figure 2.2: Quay crane and container ship

(source: photo taken at Southampton)

Figure 2.3: Automated container terminal

(source: www. mescranes.com)

Figure 2.4: Yard crane

(source: www. dctgdansk.com)

Figure 2.5: Automated guided vehicle

(source: www. fotograf-hamburg.de)

Straddle-carrier system. The straddle carrier (SC) has the ability to lift containers from the ground by itself as shown in Figure 2.6. It functions as the vehicle to transport containers between the vessel and container yard, and also functions as a flexible moving yard crane for stacking and retrieving containers in the yard. Figure 2.7 shows the SC working in the yard. With such features, straddle-carrier systems are very flexible and dynamic (Steenken, Voß and Stahlbock, 2004). As a consequence, in the straddle-carrier system, only two types of equipment are employed; QC and SC, for handling containers. In the survey carried out by Wiese, Kliewer and Suhl (2009), it was found that SC is the second most popular type of equipment used in the yard. These days, many terminals, particularly in Europe, adopt such a system to cope with container handling.

Figure 2.6: Straddle carrier

(source: photo taken at Southampton)

Figure 2.7: Straddle carrier working in the yard

(source: www.dpworldsouthampton.com)

The reason we chose these two types of terminals is that they represent the two groups of handling system (Kim, 2005); one is the 'direct transfer system', which includes the on-chassis system (in which containers are stored on chasses instead of being stacked on top of each other) and the carriers direct system (straddle-carrier system), and the other is called the 'indirect transfer system', which includes the straddle-carrier-relay and the transfer-crane-relay systems (present in a typical automated container terminal).

The problems considered in this thesis are related to the scheduling of QCs, YCs and vehicles (AGVs in the automated container terminal, SCs in the straddle-carrier system). Therefore, in the following subsections, we review the literature regarding the scheduling of QCs, YCs, AGVs and SCs.

2.1.1 QC scheduling problem

At a container terminal, the quayside is equipped with quay cranes (QCs) for unloading and loading of containers. Because QC is the most important and expensive equipment for quayside operations in the container terminals, it is therefore essential to efficiently schedule the operations of QCs in order to achieve the highest possible productivity levels at the terminal.

A container vessel is partitioned into a number of bays, which consists of stacks of containers. Generally, container space on the vessel is divided into two areas: on the deck and in the hold, and separated by hatch covers. The precedence relationship means that containers on the deck must be unloaded before containers in the hold can be unloaded and the containers in the hold must be loaded before containers on the deck should be loaded. A QC schedule specifies the service sequence of each ship-bay and the time schedule for each service; thus the sequence of containers for unloading and loading operations is determined. Many research works have been reported in the literature devoting to the QC scheduling problem. The QC scheduling problem was first introduced in Daganzo (1989) paper , where the ships were partitioned into bays and the movements of containers in a bay was defined as a task. The author suggested an algorithm to determine the number of cranes to be assigned to ship-bays of multiple vessels. Under these assumptions, Peterkofsky and Daganzo (1990) proposed a branch and bound (B&B) algorithm to determine the number of cranes to be assigned to each ship-bay and also obtained the departure times of multiple ships. The objective was to minimise the sum of the delay cost. More recently, Kim and Park (2004) considered a more practical environment, in which it is assumed that there were unloading and loading tasks within the same ship-bay, and that the QC schedule for each container was determined; they also developed an efficient heuristic approach based on a greedy randomised adaptive search procedure to overcome the computational difficulty of the B&B method for the proposed mixed integer programming (MIP) model. Moccia *et al.*

(2006) formulated the QC scheduling problem as a vehicle routing problem with constraints on precedence relationships. They aimed to minimise the vessel completion time as well as the crane-idle times (generated from interferences between cranes). A branch and cut (B&C) algorithm was developed to solve the large-sized instances. Another problem of scheduling QCs was studied by Ng and Mak (2006); they aimed to establish the work schedule for each QC in order to minimise the ship's berth time in port. Lee, Wang and Miao (2008) proposed a genetic algorithm (GA) to find the optimal handling sequence of holds for QCs assigned to a ship while considering the case of interferences between different QCs.

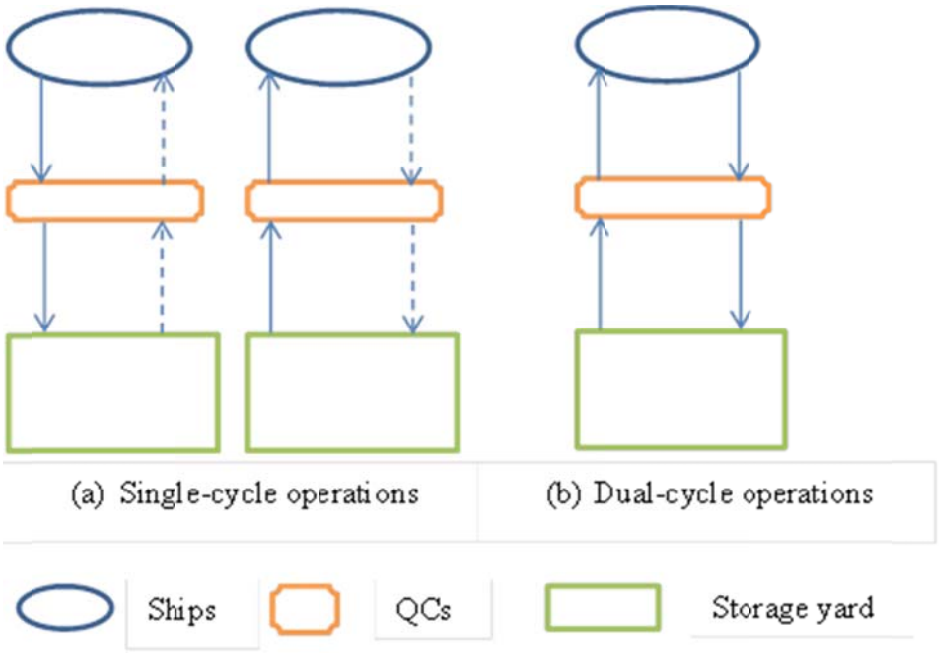


Figure 2.8: Single-cycle and dual-cycle operations strategies in container terminals

Most of the works reviewed consider the single-cycle method for QC operation, in which only the container unloading or loading process is studied, because it is easy to implement and has been adopted by conventional container terminals. However, dual-cycle allows the QCs to load containers in sequence with containers being unloaded, which is proven to improve the efficiency of the QC and the container port. Figure 2.8 compares the differences between these two operational strategies. The solid line refers to the loaded moves of QCs and vehicles while the dashed line describes the empty moves of QCs and vehicles. Obviously, the dual-cycle strategy can reduce the empty moves of all the handling equipment and therefore improve the efficiency of the

container-handling process. The studies on dual-cycle operations to date have not been widely applied. The first studies in this area were carried out by Goodchild (2005), Goodchild and Daganzo (2006). They described the dual-cycle QC-scheduling problem and presented solution algorithms and mathematical models in order to reduce the operations time and the number of operations of the QCs; they also analysed the impact of the dual-cycle operation on the land-side operations (Goodchild and Daganzo, 2007). Zhang and Kim (2009) extended the 2006 study by Goodchild and Daganzo to the sequencing problem not only for the stacks under the same hatch but also for all hatches. The problem is reformulated as a MIP model and decomposed into two parts: hatch sequencing and QC tasks sequencing under the same hatch. A hybrid heuristic approach was proposed to solve this model. Meisel and Wichmann (2010) found that the dual-cycle strategy can further accelerate the container-handling process by repositioning reshuffle containers directly within the bay of vessel, instead of temporarily unloading them.

2.1.2 Vehicle scheduling problem

In container terminals, scheduling of vehicles is becoming one of the major planning issues as inefficient vehicle schedules will cause delays in container-handling processes and thus affect the productivity of the container terminal. Containers are transported between the ship and the yard by different types of vehicles based on the types of container terminal. The most common ones are forklifts, reach stackers, straddle carriers (SCs), trucks, multi-trailers (MTs), automated guided vehicles (AGVs) and automated lifting vehicles (ALVs). In general, the vehicle scheduling problem can be formulated as transportation-type assignment problems, which determine how to dispatch vehicles to tasks (Das and Spasovic, 2003). In this thesis, we consider the scheduling of SCs in the straddle-carrier system and AGVs in automated container terminal.

AGV scheduling. With the development of material handling and information technology, a number of terminals, such as Europe Combined Terminal (ECT) in Rotterdam, the Container Terminal Altenwerder (CTA) in Hamburg, the Thames Port in the UK, the Pasir Panjang Terminal (PPT) in Singapore, the Patrick Container Terminal in Brisbane and the Pusan Eastern Container Terminal, have started to employ automated container-handling equipment so as to satisfy the customers' growing

demands and lower the labour costs. AGVs are robotics that are able to drive on a road-type network which incorporates electric wires or transponders in the ground to control the position of the AGVs.

From the perspective of the scheduling problem in automated container terminals, most studies have focused on the AGV dispatching methods. Here, dispatching can be defined as the assignment of AGVs to deliver containers. For instance, Chen *et al.* (1998) suggested a greedy algorithm for AGVs' scheduling problem with the assumption that all the AGVs are assigned to one single quay crane. Grunow, Günther and Lehmann (2004) carried out a study on multi-load AGVs' (the AGVs which could carry more than one container at a time) dispatching problem in a seaport container terminal. An alternative approach for scheduling AGVs was performed by Kim and Bae (2004), who proposed a mixed integer programming (MIP) model aiming to minimise both the total travel time of AGVs and the delay in the completion time of QCs. Briskorn, Drexl and Hartmann (2007) presented an alternative formulation of the AGV assignment problem. This formulation was based on a rough analogy to inventory management and is solved using an exact algorithm. Angeloudis and Bell (2010) studied an assignment algorithm for AGVs under uncertain conditions, which is suitable for real-time control of AGVs. The developed algorithm was applied to a simulated port environment where it was found to outperform the well-known heuristics.

There are other studies that are concerned with AGVs, but not specific to container terminals. For example, Bilge and Ulusoy (1995) exploited the interactions between the operations of machines and the scheduling of a material-handling system in a flexible manufacturing system, where material transfer between machines is performed by a number of AGVs. Van Der Heijden *et al.* (2002) used several rules and algorithms for scheduling AGVs in an underground cargo transportation system in order to reduce cargo waiting times. Lim *et al.* (2003) introduced another AGV dispatching method by using the bidding concept, which means the decisions were made through the communication between related vehicles and automated machines.

SC scheduling. In general, the amount of research on SC (which is also called the lifting vehicle (LV) scheduling problem) is relatively small. For the problem of scheduling SCs, Kim and Kim (1999c) formulated an MIP model for routing a single

SC during the loading operation of export containers. They aimed to minimise the total travel time of a SC. Das and Spasovic (2003) presented the procedure for scheduling of SCs by using a simulation method. The objective was to minimise the empty travel of SCs and the delay of trucks. In both of these studies, containers are handled by a combination of yard trucks and SCs. However, as mentioned above, our study is carried out in a container terminal based on the straddle-carrier system, where SCs could travel between the quayside and storage yard, and also function as flexible, moving YCs for stacking containers. Very recently, Moussi *et al.* (2011) discussed how to schedule LVs in loading/unloading operations by using information about pickup and delivery locations. The aim was to minimise the total travel time of all LVs.

A few researchers have analysed the SC scheduling problem in the advanced (automated) container terminal. Van Der Meer (2000) developed a simulation method to derive several dispatching rules, including rules using pre-arrival information, for automated LVs. Nguyen and Kim (2009) discussed how to dispatch automated LVs and proposed a MIP model. It was assumed that containers must be handled in the exact same order as in the QC's sequence list. However, their work is not based on a pure SC system. Most recently, Wong and Kozan (2010) investigated the relationship between QCs, automatic SCs and container storage locations. An MIP model was proposed to optimise the container-handling process. Yuan *et al.* (2011) applied a job-grouping approach where they presented a comprehensive mathematical model for the yard operations of an automated container terminal, which employed automatic SCs. Makespan, which refers to the maximum time to complete the scheduled jobs, is always used to evaluate the efficiency of the schedule. The objective was to improve the makespan of the schedule of yard jobs and reduce the total waiting time for the SCs.

2.1.3 YC scheduling problem

Yard cranes are the most popular container-handling equipment to unload containers from or load containers onto transport vehicles. Moreover, YCs are responsible for the stacking and moving of containers within the yard. In the blocks, containers are stacked on top of others, and the maximum stack height is determined by the YCs used. The two typical YCs are the rubber-tyred gantry crane (RTGC) and the rail-mounted gantry crane (RMGC). The RTGC moves on rubber-tyred wheels over a storage block space

and it can move from block to block, offering higher flexibility in container handling. The RMGC moves on rails, which are fixed to each block. Although it has lower flexibility, RMGC allows higher storage capacity. During container-handling processes, if at any time point, there is no YC available to load/unload containers, then the vehicles coming to the yard have to wait. Obviously, this can delay the ship's departure and therefore the overall berth time; hence, terminal managers have to take into consideration the yard cranes' availability and their schedules. On the other hand, yard cranes are also very expensive resources; therefore, a key aim is to optimise the operational cost of yard cranes by drawing up efficient plans for their use.

Deployment and scheduling of YCs have been investigated by many researchers. Zhang *et al.* (2002) formulated a YC-deployment problem aiming at finding the times and routes of YC movements in order to minimise the total delayed workload in the yard. The problem was formulated as an MIP model and solved by Lagrangian relaxation to obtain the optimal solution. Linn *et al.* (2003) explored another model for solving the deployment problem of the rubber-tyred gantry crane (RTGC) with the objective of finding the optimal routing and scheduling crane procedure. Ng and Mak (2005) investigated a novel model to study the YC scheduling problem with several given ready times to minimise the sum of job-waiting times. A branch and bound algorithm was proposed to solve this problem. A set of tests based on real data were generated to evaluate this algorithm. He *et al.* (2010) developed a dynamic YC scheduling model based on a rolling-horizon approach via objective programming. This aimed at minimising the total delayed workloads. A hybrid algorithm, which employed heuristic rules and parallel genetic algorithm, was used to resolve the NP-complete problem. A simulation model and numerical experiments were developed to test this model.

There are some other research areas related to YCs. Considering the YC routing problem, Huang, Liang and Yang (2009) studied the route planning of YCs for container loading/unloading in the automated container terminal and formulated it as mathematical models. To solve this problem, a genetic algorithm was developed which considers the criteria such as route length, smooth degree (angle changes) and safety distance (with obstacles). However, the model presented had been simplified and there were some other parameters which needed to be evaluated further in order to improve

the model. The problem of scheduling two cooperating automated YCs (see Figure 2.9) was studied by Vis and Carlo (2010). The new technology allowed these two cranes to pass each other so as to be able to cooperate in handling containers. The authors formulated a mathematical model to minimise the complete time for both cranes concerning container storage and retrieval within a block. A simulated-annealing approach was proposed to solve the problem efficiently.

Figure 2.9: Dual rail-mounted gantry crane in an automated container terminal, CTA in Hamburg (source: www.conductix.de)

2.1.4 Integrated scheduling problem of container-handling equipment

Each of the studies discussed above focuses on one of the different decision problems in container terminal operations; however, practically these problems are interdependent on each other because the vehicles and cranes are working cooperatively to perform the operational process for moving containers between the ship and their assigned locations. The importance of the integrated approach has been noted by researchers.

There are some studies on the integrated scheduling problem of different handling equipment. Chen *et al.* (2007) presented an integrated model to schedule different types of equipment in order to minimise the makespan at the ship. The model was formulated as a hybrid flow shop scheduling problem with precedence and block constraints and solved with a tabu search algorithm. Lee, Cao and Shi (2008b) proposed an MIP model to treat the scheduling of QC and yard truck as a whole, single problem, with the objective to minimise the makespan for the unloading operation. The problem was solved with a genetic algorithm. Lau and Zhao (2008) investigated an integrated

scheduling model of different types of handling equipment at automated container terminals. Their model aimed to minimise the total travel time of YCs, AGVs and the delays of QC operations. A multi-layer genetic algorithm was generated to obtain a near-optimal solution for the integrated problem. Lee *et al.* (2010) tackled a similar problem to the study undertaken by Chen et al. (2007), but the problem considered was in a trans-shipment hub, where the loading and discharging activities have to be handled simultaneously. Moreover, they used a genetic algorithm instead of the tabu search to solve this problem. Cao *et al.* (2010) addressed an integrated problem for yard truck and YC scheduling for loading operations and formulated it as an MIP model. Two efficient solution methods, based on Benders' decomposition, were developed for problem solution.

2.2 Container storage

In the common yard layout, the yard is divided into rectangular areas called blocks. The width of a block typically consists of six rows of stacks of containers and there is a transfer point in front of each block for vehicles interacting with yard cranes (YCs). In this thesis, we consider this type of yard layout for analysis in the automated container terminal. However, there are three different block structures as in Figure 2.10.

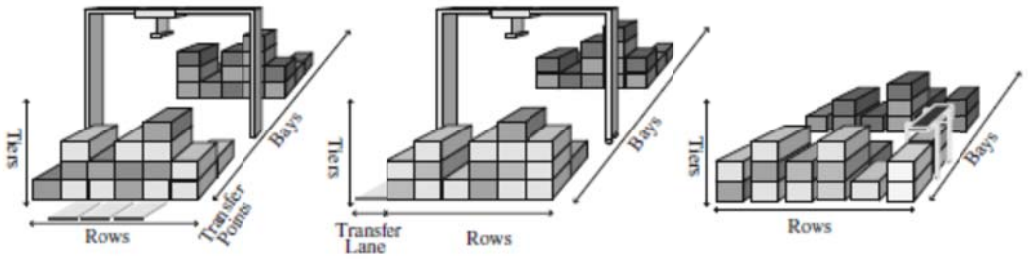


Figure 2.10: Three different block structures: yard cranes with transfer points, yard cranes with transfer lanes and straddle carrier without transfer (Wiese, Suhl and Kliewer, 2010)

2.2.1 Container storage space allocation problem

Storage space is a critical resource in container terminals, and the container storage allocation problem which determines container storage space and locations has been extensively studied. Kim and Kim (1999a) studied how to allocate storage space for

import containers by analysing cases when the arrival rate of containers is constant, cyclic and dynamic. For each arriving vessel, spaces have been allocated to minimise the expected total number of rehandles. Kozan and Preston (1999) considered an optimisation problem of container transfer schedules and storage policies. The genetic algorithm (GA) technique has been used to reduce container-handling times and ships' berth time. Factors that influence the efficiency of a container terminal were analysed at a container terminal with different types of handling equipment, storage capacities and alternative layouts. Preston and Kozan (2001a) modelled the seaport system with the objective of determining the optimal storage strategy for various container-handling schedules, such that the setup times and transport time become minimal.

Chen *et al.* (2003) developed a genetic algorithm for the general yard allocation problem (GYAP), aiming to minimise the yard space used. Zhang *et al.* (2003) made the first attempt to formulate the storage space allocation problem (SSAP) using a rolling-horizon approach. For each planning horizon, the problem was decomposed into two levels: the first level determined the total number of containers associated with each block in the yard, and the second level determined the number of containers associated with each vessel. Murty *et al.* (2005) proposed to incorporate dynamic load attributes into space allocation decisions. Bazzazi, Safaei and Javadian (2009) further extended the work of Zhang *et al.* (2003) by considering refer and empty containers, and developed a GA to solve this problem. Nishimura *et al.* (2009) addressed the storage arrangement of trans-shipment containers on a container yard. An optimisation model was developed to investigate the flow of containers transfers using intermediate storage at the yard.

2.2.2 Integrated problem of container storage allocation and vehicle/crane scheduling

The integrated problem of storage allocation and vehicle scheduling was first proposed by Bish *et al.* (2001). The authors assumed that each container had a number of potential locations in the yard. Containers were delivered from the ship to the yard by trucks. A heuristic method was suggested by dividing this problem into two separate steps in which the first step consisted of location assignments while the second step looked at vehicle scheduling. Bish (2003) extended this work to a combined problem of

determining every container's storage location, each vehicle's scheduling, and the scheduling of each quay crane. Bish *et al.* (2005) further extended this study, however, the integrated problem was still solved separately without considering the interactions between the two sub-problems. Kozan and Preston (2006) presented an iterative search algorithm to determine both optimal locations and corresponding handling schedule. Han *et al.* (2008) provided another way of integrating yard truck and storage allocation problems in trans-shipment hubs, which aimed to minimise the traffic congestions caused by yard trucks. Lee, Cao and Shi (2008a) addressed an MIP model for integrating yard truck operations and the container storage allocation problem for import containers aiming to minimise the completion time of operations. The authors also provided a GA and another heuristic algorithm to solve the formulated MIP problem. Lee *et al.* (2009) further studied this problem and developed a constructive heuristic-hybrid insertion algorithm (HIA) by taking both loading and discharging requests into account, where the objective was to minimise the total turnaround time of yard trucks and the weighted sum of total delay. Wong and Kozan (2010) investigated the relationship between quay cranes, yard machines (forklifts, straddle carriers, reach stackers and tractors) and container storage locations in a multi-berth and multi-ship environment. The authors also provided a solution based on list scheduling and tabu search algorithms to tackle the computational complexity issue of the problem.

2.3 Heuristic techniques on solving the problems in container terminal operations

A heuristic method is a procedure for tacking problems by an intuitive approach in which the structure of a problem can be interpreted and exploited intelligently to obtain a reasonable solution. Heuristic methods are used when an exact solution is unknown or when an exact solution method has prohibitive computational requirements, e.g. NP-hard problem. The heuristic method is simpler to understand and implement than exact methods, and it can also provide insight into a complex problem. For the NP-hard problem, a good, near-optimal solution that is found in a reasonable computation time is preferred to an optimal solution that would take a very long time to calculate (Casey and Kozan, 2006). Therefore, a number of heuristic algorithms have been implemented in solving problems on container terminal operations. The most commonly used ones are

the genetic algorithm and the neighbourhood search algorithm (including simulated annealing and tabu search).

2.3.1 Genetic algorithm (GA)

Genetic algorithms (GAs) have been used extensively in solving sequencing and scheduling problems. GA is a well-known heuristic approach inspired by the natural evolution of the living organisms that works on a population of the solutions simultaneously. It was first proposed by Goldberg (1989). It combines the concept of survival of the fittest with structured, yet randomised, information exchange to undertake robust exploration and exploitation of the solution space. GA starts with a set of random solutions called a population. In the natural world, each individual named chromosome is assigned with a fitness value. The exploration process is performed by a genetic operator - namely crossover, and the exploitation process is performed by another operator - namely mutation. The new generation is selected based on the Darwinian theory of evolution in which individuals with better performances will have more probability of being chosen, and it is controlled by the parent selection and offspring acceptance strategies. The reason why we choose GA in this thesis is that, firstly GA is a well-known heuristic approach that its efficiency is verified by many problems in the literature (Bazzazi, Safaei and Javadian, 2009; Han, Lu and Xi, 2010; Lee *et al.*, 2010; Tavakkoli-Moghaddam *et al.*, 2009), and secondly, we need a population-based approach such as GA to better explore the solution space. It has been proven that GA-based approaches can solve large-sized problems with approximately optimal solutions.

There exist many studies which have developed GA for applications in container terminal operations. In the area of quay crane scheduling problems, Tavakkoli-Moghaddam *et al.* (2009) presented a novel MIP model for the QC scheduling and assignment problem in a container terminal. They proposed GA to solve this problem in the large-sized real-world situations. The efficiency of GA is compared against LINGO software in terms of objective function values and computational times. Han, Lu and Xi (2010) addressed the berth allocation and QC scheduling problem simultaneously, considering uncertainties in the vessel arrival time and container-handling time. They also proposed an MIP model for the problem. A simulation-based GA search procedure

was used to solve the problem and gave a good performance. Another study on integrated berth allocation and the QC assignment problem was undertaken by Yang, Wang and Li (2012); they developed a GA with nested loops to obtain solutions for the problem: two inner loops were used for solving berth allocation and quay crane assignment respectively; and an outer loop was used for finding an approximate solution based on the results of the two inner loops. Their results showed that the proposed approach was efficient in solving the proposed problem and thus improving the operational efficiency of container terminals. The scheduling of two QCs with non-interference constraints at Narvik Container Terminal in Norway has been studied by Hakam, Solvang and Hammervoll (2012), in which GA was adapted to the case of two cranes to provide practical solutions.

From the perspective of scheduling vehicles, Bose *et al.* (2000) studied the process of container transport by yard cranes and straddle carriers between the ship and the yard. They aimed to minimise the ship's berth time by maximising the productivity of yard cranes. GA showed a good performance in solving the problem by providing an improvement in the productivity of the gantry cranes. Lee, Cao and Shi (2008b) formulated the QC scheduling and yard truck scheduling as an MIP model and solved it with a GA because the optimal solutions for large-sized problems cannot be obtained by exact algorithm in a reasonable time duration. Lau and Zhao (2008) proposed a multi-layer genetic algorithm (MLGA) to obtain the near-optimal solution of the integrated scheduling of different types of handling equipment. They also investigated another heuristic method, called genetic algorithm plus maximum matching (GAPM), to reduce the computational complexity of the MLGA method. More recently, Lee *et al.* (2010) presented the vehicle dispatching problem in a transhipment hub in order to minimise the berth time of the ship. They developed two heuristics to tackle the computational difficulty: the first one was neighbourhood search, and the second one was GA and the minimum cost flow (MCF) network model. The results showed that the GA-based algorithm was better than the neighbourhood search algorithm. Choi *et al.* (2011) proposed GA for the efficient dispatching of container trucks to minimise the transportation cost of trucks. Fereidoonian and Mirzazadeh (2012) studied the GA for the integrated scheduling problem of container-handling equipment in the loading and unloading operations. The problem was formulated as a hybrid flow shop scheduling

with parallel machines and the proposed GA allocated containers to these machines at every stage through the evolution process.

In the field of yard operations, Kozan and Preston (1999) used GA techniques to reduce container-handling times and ships' berth time at container terminals by speeding up handling operations. The results showed that the schedule storage policy, in which containers were stored close to the berth, is better than the random storage policy. Nishimura, Imai and Papadimitriou (2005) addressed the dynamic assignment rules for yard trailers to QCs and the optimisation of yard trailer routing. The GA procedure was employed to obtain the near optimal solution to the problem in order to save yard operation time and costs. An efficient GA had also been applied to solve the extended storage space allocation problem (SSAP) in a container terminal in the work of Bazzazi, Safaei and Javadian (2009); The SSAP concerned how to allocate import/export containers to the storage blocks in order to balance the workload between blocks so as to minimise the handling times of containers at the yard. GA had proven to be able to quickly provide feasible solutions in the real-world cases. Huang, Liang and Yang (2009) presented a mathematical model for the problem of container unloading/loading by yard cranes. An optimal route method based on GA was developed to satisfy the length, smooth degree and safety distance requirements. Computational results showed the effectiveness of the proposed GA. He *et al.* (2010) studied a novel yard crane scheduling model based on the rolling-horizon approach. Parallel GA was developed to solve this NP-complete problem.

2.3.2 Neighbourhood search

Neighbourhood search methods offer a practical approach to tackling real-life combinatorial optimisation problems. The most commonly used methods considered in solving container terminal operations problems are simulated annealing and tabu search algorithms.

Simulated annealing algorithm (SA)

The simulated annealing (SA) algorithm models itself on the physical annealing process for solids which are cooled slowly to produce a low-energy state. It was first proposed

by Kirkpatrick and Vecchi (1983). A cooling schedule specifies the temperature at each stage of the algorithm and the initial temperature is gradually decreased. An initial solution, which is set to be the current best solution, is generated either randomly or using a constructive heuristic. A neighbourhood contains a group of solutions that are obtained from the current solution by a defined scheme. The best solution in the neighbourhood is compared against the current solution. If the new solution is better than the current solution, it will replace it; otherwise, it will still have a chance to replace it with some probability until the algorithm terminates. The main idea is to use a probabilistic approach to avoid getting stuck at a local minimum.

For example, SA was applied to the berth-scheduling problem to find near-optimal solutions as in the work of Kim and Moon (2003), in order to determine the berthing times and positions of container ships. Experiments demonstrated that the proposed algorithm obtained the solution similar to the optimal solutions found by the MIP model. Jung and Kim (2006) applied both GA and SA for the scheduling problem in the loading operations when multiple YCs are working in the same block. Numerical experiments showed that the SA outperformed GA in terms of average computational time and average objective function value. Kang, Ryu and Kim (2006) proposed a method based on SA to derive a good stacking strategy for containers with uncertain weight information. The implemented SA can find a good stacking strategy to reduce the number of container re-handlings during loading operations within a reasonable time. Lee, Cao and Meng (2007) applied SA to solve the YC scheduling problem, in which two YCs serving the loading operations for a QC at two blocks aimed to minimise the total loading time at a stack area. The computational results showed that SA performed well in all different-sized problems. Moussi, Ndiaye and Yassine (2012) modelled the seaport system as a container location model aiming to minimise the distance between the ship berthing locations and the container storage yard. A hybrid GA/SA was proposed to deal with the computational complexity of the problem. In terms of small-scale problems, the results from GA/SA were compared with optimal solutions obtained from the commercial software, ILOG CPLEX. Computational results on real dimensions taken from a French terminal showed the good quality of the solution provided by this hybrid method.

Tabu search algorithm (TS)

The tabu search (TS) algorithm is similar to simulated annealing in that it can avoid getting stuck at a local minimum; however, the decision of whether an alternative solution should replace the current is determined by the tabu list to overcome the problem of cycling. It was first proposed by Glover (1989). As with simulated annealing, TS starts with an initial solution which is known as the current best solution. A neighbourhood of solutions are obtained through the replacement of the current solution. The best solution found in the neighbourhood is accepted as the new current solution to replace the current solution, and then the attributes of the replacement are added to the tabu list that cannot be performed in the future search.

For example, Preston and Kozan (2001b) designed a scheduling model taking into account factors such as container-handling equipment, labour resources, storage capacities and terminal layout. TS was able to find better solutions and the computation time of TS is 10 times faster than that of GA. Cordeau *et al.* (2005) developed a TS algorithm on solving the berth allocation problem in both a discrete case and a continuous case, and the algorithm can achieve good results for large-size problems. Chen *et al.* (2007) presented an integrated model to schedule different types of equipment aiming to minimise the makespan. The problem was formulated as a hybrid flow shop scheduling problem and a TS algorithm was proposed to solve this problem. It can be observed that a good initial solution heuristic is important for the scheduling problem and helpful to further improve the solution. Casey and Kozan (2006) developed a container storage handling model in order to minimise transfer tardiness in the storage area. A hybrid of simulated annealing and tabu search (TS/SA) was developed in which the tabu list from TS was used in conjunction with the selection probability from simulated annealing to refine the solution. It was found that the TS/SA hybrid performed well in improving the feasible solutions. Han *et al.* (2008) studied a storage yard management problem in a trans-shipment hub in order to minimise the potential traffic congestion. A TS-based heuristic algorithm was used to solve the problem, and results showed that the proposed method can generate good results in a reasonable time period. Wong and Kozan (2010) developed a model to improve the operational efficiency of the seaports and investigated the relationship between QCs, vehicles and

container storage location. TS was used to solve this NP-hard scheduling problem. The results proved that this algorithm was generic and adaptive, and was applicable to many seaports. Chen *et al.* (2003) studied the general yard allocation problem (GYAP), in which space allocation for cargo was minimised for all incoming requests for space required in the yard. As GYAP is a NP-hard problem, they proposed several heuristic algorithms, including TA, SA and GA, among others. Comparisons of solutions among these heuristics developed showed that GA implementation achieved the best results.

2.4 Conclusions

The increasing number of studies in container terminal operations in the last decades indicates the importance of operations research methods in the field of optimising logistic operations at a container terminal. It is important for a terminal to fully utilise its resources, such as handling equipment and storage space, to achieve optimum productivity and customer satisfaction. To date, most of the studies have focused on optimising one of the logistics problems; however, recently there is an emerging tendency to focus on the integration of several optimisation problems. In this chapter, we have reviewed a few research fields that are directly related to our work.

First, we reviewed the work that has been devoted to the scheduling problems of container-handling equipment to determine the time and sequence for handling each container. A significant body of the research focuses on the QC scheduling, vehicle scheduling and YC scheduling separately, but these problems are in fact highly interrelated. For example, the QC schedule determines the vehicle's transportation, and the vehicle schedule determines the release/pickup time for containers by YCs. Therefore, optimising only one type of equipment cannot yield the overall optimisation. Coordination of different forms of handling equipment is vital to increase the performance of the terminal.

Second, the studies on container storage problems have been reviewed. The objectives are to minimise the handling times and utilise space efficiently. Storage space is a vital resource in the container terminal and the locations that are allocated to containers usually affect the operations of the yard. Also, some studies combined the problems of

equipment scheduling and storage allocation in order to obtain integrated solutions so that the overall performance can be improved.

Third, we reviewed previous studies that applied heuristic methods to solve the operational problems of container terminals. Because the system of a container terminal is highly complicated and exact solutions for these problems cannot always be obtained in a reasonable time, operational research techniques (modelling approach, heuristics) must be developed to shorten computation times for solving such planning activities. In particular, GA has been widely used for problems in different areas of container terminal operations; therefore, we use GA for solving the problems in this thesis.

To conclude, most studies in container terminal operations have only discussed one single operational problem, such as scheduling of vehicles or the container storage allocation problem. Far too little attention has been paid to the operational issues in the straddle-carrier system which is the typical direct transfer system, and to the automated container terminal represents the next generation of container terminals, indicating the importance of further study on the related issues. The type of containers considered in this research is the standard twenty feet equivalent units (TEUs); other containers are measured by means of TEUs, for example 40' and 45' containers can be considered as 2 TEUs, which could also be incorporated in our research. Containers which require special storage facilities, such as refrigerated, chemical goods are usually stored far away from normal containers, and thus are not scheduled together. Therefore, we seek to address the integration of scheduling container-handling equipment and container storage allocation problem in the above two types of container terminals. In addition, this research will also generate a heuristic algorithm based on genetic algorithm for each problem in order to solve the problems in practical sizes.

Chapter 3. Modelling of integrated vehicle scheduling and container storage problems at automated container terminals

This chapter presents detailed descriptions of the problems that have been identified in the operations of the automated container terminal, and develops mathematical programming models to address these. Section 3.1 investigates the problem of vehicle scheduling and storage allocation during the container unloading process. Section 3.2 studies the problem of scheduling different types of container-handling equipment during the container loading process. Section 3.3 discusses vehicle scheduling and storage allocation problems when taking both unloading and loading processes into consideration simultaneously.

All the problems are formulated in the form of mixed-integer programming (MIP) models. The goal of all three models is to minimise the ship's berth time, which is one of the most significant indicators to measure the efficiency of container terminals. In each section, we design a heuristic approach, e.g. genetic algorithm (GA), for the models in order to solve the problem in large sizes. By describing and analysing each model and its results in the following sections, the main findings of the models and the proposed algorithms are discussed.

3.1 Model 1: The integrated AGV scheduling and container storage allocation problem during the container unloading process

An important objective for the container terminal's operations is to minimise the berth time of ships. Most of the literature considered the scheduling problem of container-handling equipment and container storage allocation problem separately. However, as the schedules of vehicles/cranes and the yard locations of containers are highly interrelated, it is important to address these issues simultaneously to ensure an efficient terminal operation. In the literature, only a limited number of studies have focused on the integrated problem in the container unloading process; for example, the integrated

quay crane and yard truck scheduling problem as in the study by Lee, Cao and Shi (2008b). None of these studies, though, considered the integrated vehicle scheduling and container storage allocation problem in the container unloading process. The problem studied in this section is to decide how to dispatch vehicles (i.e. AGVs) to deliver containers and how to assign each container to a yard location in order to minimise the time needed to unload containers from the ship.

This section studies the integrated problem of AGV's scheduling and storage allocation for import containers. We implement the proposed model in two steps: Firstly, we use optimisation software AIMMS 3.11 for solving the problem in small sizes. An example is presented to illustrate the proposed problem and the integrated solution. However, obtaining an optimal solution for the large-sized (hundreds of containers) problem by the existing solver embedded in AIMMS 3.11 is difficult. We thus, in the second step, develop an efficient genetic algorithm (GA) for handling large-sized scenarios.

3.1.1 Problem description and formulation

We consider an automated container terminal, which consists of a berthing area at the quayside, an AGV's travelling area and a storage yard. The berthing area is equipped with quay cranes (QCs) for unloading containers; the storage yard is used for temporary storing of import containers before further delivery by trains or trucks; AGVs are used to deliver containers from the berthing area to the storage yard. Figure 3.1 illustrates a typical layout of an automated container terminal, which is used in the computational experiment of this section. In this study, 'yard locations' are used to refer to all the available locations in the yard. The places where AGVs transfer containers to YCs are referred to as 'transfer points'. 'Apron' indicates the area at the quayside where containers are loaded onto AGVs from QCs. In figure 3.1, transfer points, where YCs pick up containers from AGVs, are located in front of each block. Under each QC, there is one working point for this QC to drop off containers onto AGVs. As the AGV has a very simple command and control structure, it requires a crane to unload a container onto it. At the beginning of the unloading process, AGVs are at the quayside ready for handling containers from QCs. The AGVs travel along the guided paths to deliver containers between the ship and the storage yard. After having delivered an import container to the storage yard, the AGV returns to the quayside to handle the next

container. The delivery schedule of AGVs plays an important role in synchronising operations of different types of container-handling equipment, e.g. YCs and QCs. Moreover, it has an impact on the assignments of container storage locations.

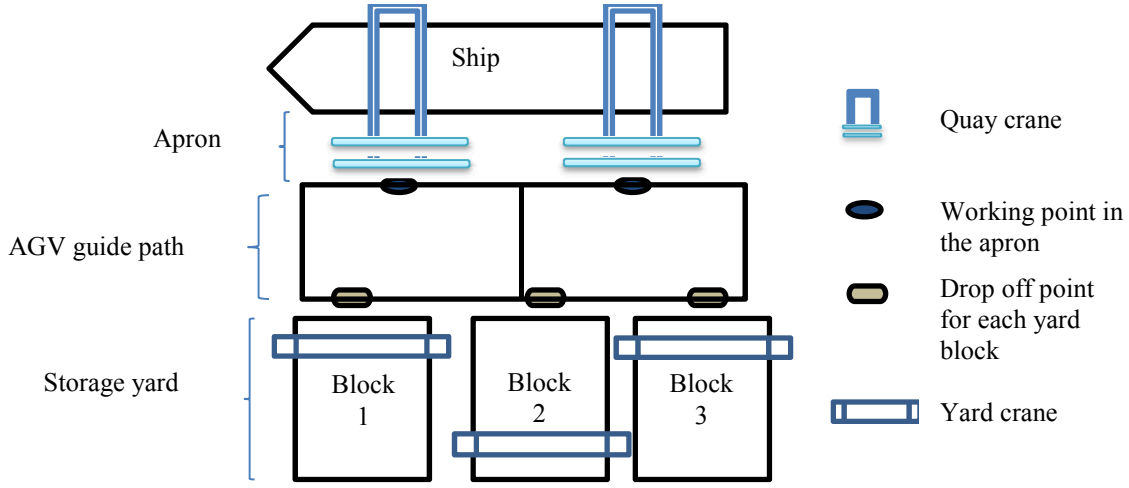
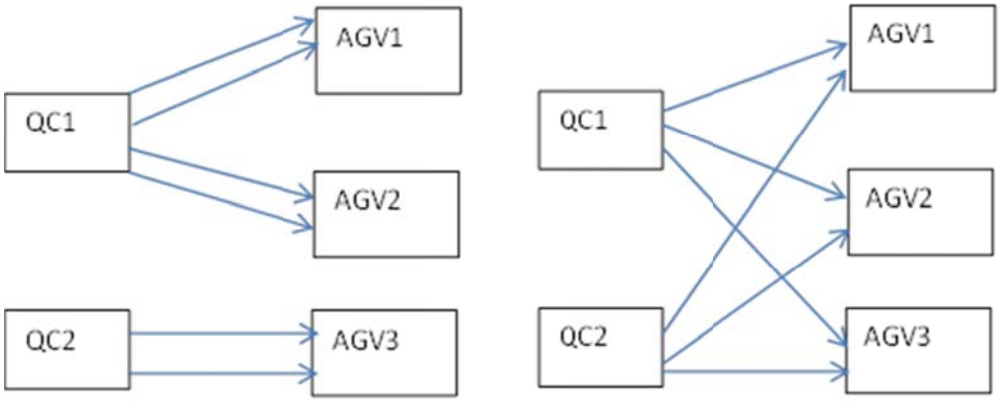


Figure 3.1: The layout of an automated container terminal

Before a ship arrives at the terminal, it has already been allocated a berth area, according to the tonnage, estimated arrival time and berth time (Lee, Cao and Shi, 2009), and QCs have also been allocated to work on this ship. The unloading sequences of the containers are known in advance for each QC (Grunow, Günther and Lehmann, 2004). When actual unloading operations begin, it is carried out in the same order as specified in the sequence list of QCs. The inside of a container ship is constructed in the form of compartments. Containers on the ship are stacked up to six levels high due to the limit in the weight that the bottom container can withstand from those on top of it (Wong and Kozan, 2010). Each vessel is divided into bays and each bay accommodates a row of container stacks. During the container unloading process, a container, which has been picked up from the ship by a QC, is loaded on to an AGV (which delivers this container to the storage yard). At the storage yard, this container is handled by an YC which locates it in the assigned location in the storage area.

When more than two QCs are working for a ship, either non-pooling or pooling policy for AGVs can be used. Figure 3.2 describes the difference in non-pooling and pooling policies. The arrows indicate how AGVs assign to QCs. In the non-pooling policy, each AGV can only serve one QC. For example, in figure 3.2 (a), AGV 1 can only serve QC

1. By contrast, in the pooling policy, each AGV can serve any of the QCs. For example, in figure 3.2 (b), AGV 1 can serve both QC 1 and QC 2. Although non-pooling is easy to implement because each AGV serves only one QC, here we adopt the pooling policy, which is most commonly used in container terminal operations. Congestion among AGVs on the path is not considered here because the study about the interference of vehicles involves more complex scheduling and control of detailed movements of vehicles, which is another important issue for the AGV system in automated container terminal (Evers and Koppers, 1996).



(a) (b)

Figure 3.2: (a) Non-pooling strategy and (b) Pooling strategy in container unloading process

The container storage yard is used to temporarily store containers until they are then picked up by vehicles/trains or loaded onto the ship. A yard may be divided into a number of blocks, where containers are stacked side by side and one on top of each other. Interferences (conflicts) among YCs are not considered here because this requires details on the YC control mechanisms, which is outside the scope of this study. The width of a block is usually divided into six rows of containers. In front of each block are the transfer points for AGVs to interact with YCs for collecting import containers. Retrieving and stacking containers within blocks are performed by YCs, which can move among blocks.

The objective is to minimise the unloading time element of the berth time of the ship, which is the time duration when all the containers have been discharged from the ship. The main operational decisions of the problem are to determine (1) how to dispatch

AGVs to deliver containers, (2) in which sequences YCs should move containers, and (3) where to locate the import containers in the storage yard.

The integrated model is developed based on the following assumptions:

(1) The handling sequences of containers for each QC are known and the unloading process must be carried out in the exact order appearing in the QC's sequence lists.
(2) Number of import containers, number of AGVs, number of QCs and YCs are all known.
(3) QCs, YCs and AGVs can only take one container at a time.
(4) Pooling policy is applied to AGVs.
(5) Travelling times of AGVs between any QC and any transfer point of the yard block are known. Similarly, travelling times of YCs among transfer points of blocks, and the travelling times of YCs from each transfer point to each available yard slot within that block are known.
(6) Traffic congestion of the AGVs on the path is not considered.
(7) The interference (conflict) among YCs and the interference (conflict) among QCs are also not considered, because the interferences are difficult to anticipate without scheduling and controlling detailed movements of the cranes.
(8) The time needs for QCs dropping off containers onto AGVs and YCs picking up containers from AGVs are negligible.
(9) The yard storage capacity is larger than the number of the import containers.

We introduce the notations related to the integrated AGV scheduling and storage allocation problem during container unloading process.

Sets and Parameters

D set of import containers

K set of QCs

P	set of yard locations (container storage locations)
B	set of blocks
V	set of AGVs
C	set of YCs
v	Total number of AGVs
c	Total number of YCs
k, l	index for QCs
b, a	index for blocks
$(i,k),(j,l)$	index for containers; container *(i, k)* means the *i*th container to be handled by QC *k*
N_k	index for the total number of containers handled by QC *k*. Therefore container (N_k, k) means the last container handled by QC k
(n,b)	index for yard locations, $n \in N^+$ positive integer, location *(n, b)* means slot *n* in block *b*
$h_{(i,k)}$	QC k's handling time of container (i,k)
$t_{(k,b)}$	AGV's travelling time from QC *k* to block *b* (same as travelling time from block *b* to QC *k*)
$w_{(a,b)}$	YC's travelling time from block *a* to block *b* (same as travelling time from block *b* to block *a*)
$\varphi_{(n,b)}$	YC's travelling time of a container from the transfer point of block *b* to the location *(n, b)*
M	a very large positive number

(S, I) dummy starting job (a job is defined as a container to be unloaded)

(F, I) dummy ending job

O_S The job set which contains all the jobs including the dummy starting job,

$$O_S = D \cup (S, I)$$

O_F The job set which contains all the jobs including the dummy ending job,

$$O_F = D \cup (F, I)$$

O The job set which contains all the jobs including dummy starting and ending jobs, $O = \{(S, I), (F, I)\} \cup D$

Decision variables

$u_{(i,k)}$ the time QC k starts to handle container (i, k) from the ship

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

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

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

We have an intermediate decision variable $y_{(i,k)}^{b}$, where $\sum_{n \in N^+} z_{(i,k)}^{(n,b)} = y_{(i,k)}^{b}, \forall (i,k) \in D, \forall b \in B$

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

$\sigma_{(i,k)}^{(j,l)} =$

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

Objective: minimise the berth time of the ship

$$\text{Min: Max}(u_{(N_k,k)} + h_{(N_k,k)})$$

Contraints

$$\sum_{(j,l)\in O_F} x_{(i,k)}^{(j,l)} = 1, \forall (i,k) \in D \tag{3.1.1}$$

$$\sum_{(i,k)\in O_S} x_{(i,k)}^{(j,l)} = 1, \forall (j,l) \in D \tag{3.1.2}$$

$$\sum_{(j,l)\in D} x_{(S,I)}^{(j,l)} = v \tag{3.1.3}$$

$$\sum_{(i,k)\in D} x_{(i,k)}^{(F,I)} = v \tag{3.1.4}$$

$$\sum_{b\in B} y_{(i,k)}^{b} = 1, \forall (i,k) \in D \tag{3.1.5}$$

$$\sum_{(n,b)\in P} z_{(i,k)}^{(n,b)} = 1, \forall (i,k) \in D \tag{3.1.6}$$

$$\sum_{(i,k)\in D} z_{(i,k)}^{(n,b)} \leq 1, \forall (n,b) \in P \tag{3.1.7}$$

$$\sum_{n\in N^+} z_{(i,k)}^{(n,b)} = y_{(i,k)}^{b}, \forall (i,k) \in D, \forall b \in B \tag{3.1.8}$$

$$\sum_{(j,l)\in O_F} \sigma_{(i,k)}^{(j,l)} = 1, \forall (i,k) \in D \tag{3.1.9}$$

$$\sum_{(i,k)\in O_S} \sigma_{(i,k)}^{(j,l)} = 1, \forall (j,l) \in D \tag{3.1.10}$$

$$\sum_{(j,l)\in D} \sigma_{(S,I)}^{(j,l)} = c \tag{3.1.11}$$

$$\sum_{(i,k)\in D} \sigma_{(i,k)}^{(F,I)} = c \tag{3.1.12}$$

$$u_{(i,k)} + h_{(i,k)} + \sum_{b\in B} t_{(k,b)} y_{(i,k)}^{b} \leq d_{(i,k)}, \forall (i,k) \in D \tag{3.1.13}$$

$$d_{(i,k)} + \sum_{b\in B} t_{(l,b)} y_{(i,k)}^{b} \leq u_{(j,l)} + h_{(j,l)} + M * \left(1 - x_{(i,k)}^{(j,l)}\right), \forall\ (i,k) \in O_S, \quad (j,l) \in O_F \tag{3.1.14}$$

$$d_{(i,k)} + \sum_{(n,b)\in P} \varphi_{(n,b)} * z_{(i,k)}^{(n,b)} + \sum_{a,b} w_{(a,b)} * y_{(i,k)}^{b} * y_{(j,l)}^{a} \leq d_{(j,l)} + M * \left(1 - \sigma_{(i,k)}^{(j,l)}\right), \forall\ (i,k) \in O_S, (j,l) \in O_F \tag{3.1.15}$$

$$u_{(i+1,k)} - u_{(i,k)} \geq h_{(i,k)}, \forall (i+1,k), (i,k) \in D, i = 1,2, \dots N_k - 1 \tag{3.1.16}$$

$$x_{(i,k)}^{(j,l)}, y_{(i,k)}^{b}, z_{(i,k)}^{(n,b)}, \sigma_{(i,k)}^{(j,l)} \in \{0,1\}, \forall (i,k), (j,l) \in O, \forall (n,b) \in P, \forall b \in B \tag{3.1.17}$$

$$u_{(i,k)}, d_{(i,k)} \geq 0, \forall (i,k) \in D, i = 1,2, \dots, N_k, \forall k \in K \tag{3.1.18}$$

The objective of the model above is to minimise the makespan of the unloading time of the QC's operations, i.e. unloading element of the ship's berth time.

Constraint (3.1.1) ensures that each container (i,k) in D has one successor container (j,l) in O_F and they are delivered by the same AGV.

Constraint (3.1.2) represents that every container (j,l) in D has one predecessor container (i,k) in O_S, and they are delivered by the same AGV.

Constraints (3.1.3) and (3.1.4) guarantee that the number of AGVs used to deliver the set of import containers is exactly *v*.

Constraint (3.1.5) guarantees that each import container (i, k) will be assigned to one of the blocks *b* during the unloading process.

Constraint (3.1.6) ensures that each import container (i, k) will be stored in one of the available locations (n, b) in the yard.

Constraint (3.1.7) represents that each available location (n, b) in the yard can store at most one container (some locations may not store any container).

Constraint (3.1.8) means that if a container is assigned to block *b*, it can only be assigned to one available location *(n, b)* within block *b*. This constraint gives the relationship between the two decision variables $y_{(i,k)}^{b}$ and $z_{(i,k)}^{(n,b)}$.

Constraint (3.1.9) implies that for any container (i, k) in D there is one container (j, l) succeeding it. Container (i, k) and container (j, l) are handled by the same YC.

Constraint (3.1.10) implies that for any container (j, l) in D, there is one predecessor container (i, k) and they are handled by the same YC.

Constraints (3.1.11) and (3.1.12) guarantee that the number of YCs deployed for handling containers is exactly *c*.

Constraint (3.1.13) means that the AGV needs a certain travelling time from QC *k* to block *b* (if container (i, k) is assigned to block *b*) before container (i, k) is handled by a YC.

Constraint (3.1.14) means that if the AGV, which just handled container (i, k), is scheduled to handle container (j, l), then this AGV needs to be allocated a certain travelling time from the yard to the quayside.

Constraint (3.1.15) denotes that if the YC, which just handled container (i, k), is scheduled to handle container (j, l), then this YC needs a certain time to travel from the

location (i.e. location (n, b)) of container (i, k) to the transfer point of the block that is assigned to store container (j, l) .

Constraint (3.1.16) is the time interval between QC k's starting times for two successive containers (i, k) and $(i + 1, k)$ must be at least the handling time of the container (i, k).

Constraints (3.1.17) and (3.1.18) are binary and non-negative restrictions.

3.1.2 An illustrative example

To better understand and verify the performance of the proposed model, in this section, we present an example to illustrate the MIP model proposed in section 3.1.1. Based on the objective function and constraints considered, an example is used to illustrate how the solutions of the MIP model can be used in analysing the unloading operations. This example is solved by AIMMS 3.11 which adopted the branch and bound (B&B) algorithm embedded in the solver of CPLEX 11.2 (Bisschop and Roelofs, 2011). B&B has been successfully applied to several optimisation problems: for example, quay cranes' scheduling problems at container terminals (Moccia *et al.*, 2006), and scheduling of rail crane and container deliveries (Jeong and Kim, 2011). It has also been applied to several routing problems (Ascheuer, Fischetti and Grötschel, 2001; Bard, Kontoravdis and Yu, 2002; Kenyon and Morton, 2003; Laporte, Riera-Ledesma and Salazar-González, 2003) and general cutting planes problems (Balas, Ceria and Cornuéjols, 1996).

Following is the operating environment of the container unloading process at the automated container terminal. At the ship's berth, two QCs are serving the ship for unloading containers; three blocks at the storage yard are used for storing containers. There are three YCs working in the yard and three AGVs travelling between the ship and storage yard. All the YCs and AGVs are available to handle their first jobs. We give an example with 10 containers to be unloaded from the ship. The parameter settings considered in our problem are listed in the following tables.

Firstly, in table 3.1, assuming there are five containers to be unloaded by each QC, we present the sequence list handled by each QC. The relating handling times $h_{(i,k)}$ are assumed to follow uniform distribution U(30, 180)s (Tavakkoli-Moghaddam *et al.*,

2009), i.e. $Uniform(mean - \delta * mean, mean + \delta * mean)$, where mean =105, $\delta = 0.72$. For example, the handling time for container (2, 1) by QC 1 is 50 seconds, and the handling time for container (1, 2) by QC 2 is 35 seconds.

Table 3.1: The QC's sequence list and the handling time $h_{(i,k)}$ of containers

Containers (QC 1)	$h_{(i,k)}$ (sec)	Containers (QC 2)	$h_{(i,k)}$ (sec)
(1, 1)	93	**(1, 2)**	35
(2, 1)	50	**(2, 2)**	150
(3, 1)	152	**(3, 2)**	86
(4, 1)	79	**(4, 2)**	147
(5, 1)	64	**(5, 2)**	137

Secondly, we list the AGV's travelling times between each QC k and block b ($t_{(k,b)}$) in table 3.2. For example, an AGV takes 62 seconds travelling between block 2 and QC 1; and it takes 82 seconds travelling between block 1 and QC 2.

Table 3.2: The travelling time of AGVs between QCs and blocks-the values of $t_{(k,b)}$

Travel time (sec)	QC 1	QC 2
Block 1	68	82
Block 2	62	20
Block 3	84	40

Thirdly, noting that YCs can travel between blocks, we list the following travelling times of YCs among the transfer points of blocks, i.e. the values of $w_{(a,b)}$, which are shown in table 3.3. In this example, a YC's travelling time between any two adjacent

blocks is 40 seconds; therefore, it takes 80 seconds to travel between block 1 and block 3.

Table 3.3: YC's travelling time between any two transfer points of blocks-the values of $w_{(a,b)}$

Travel time (sec)	Block 1	Block 2	Block 3
Block 1	0	40	80
Block 2	40	0	40
Block 3	80	40	0

Lastly, the YC's travelling time $\varphi_{(n,b)}$ from the transfer point of each block to each available location *(n, b)* is presented in table 3.4. The values are generated from a uniform distribution: U(40, 160)s. Assume that there are 12 available locations in the yard to accommodate 10 import containers and there are four locations available for each block. Specifically, location (2, 1) means slot 2 in block 1, which is different with container (2, 1)-2nd container handled by QC 1. $\varphi_{(2,1)} = 52$ means that it takes a YC 52 seconds to move a container from the transfer point of block 1 to location (2, 1).

Table 3.4: The travelling time of YCs from the transfer point of a block to each available location within the block

Block 1	$\varphi_{(n,b)}$ (sec)	Block 2	$\varphi_{(n,b)}$ (sec)	Block 3	$\varphi_{(n,b)}$ (sec)
(1, 1)	131	**(1, 2)**	121	**(1, 3)**	136
(2, 1)	52	**(2, 2)**	109	**(2, 3)**	121
(3, 1)	90	**(3, 2)**	86	**(3, 3)**	49
(4, 1)	126	**(4, 2)**	108	**(4, 3)**	127

In our example, by solving the MIP problem, we obtain the optimal berth time, $\text{Max}\left(u_{(N_k,k)} + h_{(N_k,k)}\right) = 582$ sec, and an integrated optimal solution which gives the detailed schedules, i.e. the time and sequence required to handle each container by each AGV and each YC, as well as the assigned locations for each unloading container.

Table 3.5: The assigned location of each container at the yard- the values of the optimal solutions $z_{(i,k)}^{(n,b)}$ and $y_{(i,k)}^{b}$

Containers (i, k)	Locations (n, b)	$z_{(i,k)}^{(n,b)}$	$y_{(i,k)}^{b}$
(1, 1)	(3, 1)	$z_{(1,1)}^{(3,1)} = 1$	$y_{(1,1)}^{1} = 1$
(2, 1)	(3, 2)	$z_{(2,1)}^{(3,2)} = 1$	$y_{(2,1)}^{2} = 1$
(3, 1)	(4, 1)	$z_{(3,1)}^{(4,1)} = 1$	$y_{(3,1)}^{1} = 1$
(4, 1)	(1, 2)	$z_{(4,1)}^{(1,2)} = 1$	$y_{(4,1)}^{2} = 1$
(5, 1)	(2, 2)	$z_{(5,1)}^{(2,2)} = 1$	$y_{(5,1)}^{2} = 1$
(1, 2)	(4, 2)	$z_{(1,2)}^{(4,2)} = 1$	$y_{(1,2)}^{2} = 1$
(2, 2)	(3, 3)	$z_{(2,2)}^{(3,3)} = 1$	$y_{(2,2)}^{3} = 1$
(3, 2)	(1, 3)	$z_{(3,2)}^{(1,3)} = 1$	$y_{(3,2)}^{3} = 1$
(4, 2)	(2, 1)	$z_{(4,2)}^{(2,1)} = 1$	$y_{(4,2)}^{1} = 1$
(5, 2)	(2, 3)	$z_{(5,2)}^{(2,3)} = 1$	$y_{(5,2)}^{3} = 1$

Firstly, the assigned location of each container at the storage yard is shown in table 3.5. These decisions are obtained from the values of $z_{(i,k)}^{(n,b)}$ and $y_{(i,k)}^{b}$. For example, container

(1, 1) will be stored in slot 3 of block 1, i.e. location (3, 1); container (1, 2) will be stored in slot 4 of block 2, i.e. location (4, 2) and so on.

Table 3.6: The yard crane handling sequences- the values of the optimal solutions $\sigma_{(i,k)}^{(j,l)}$

YCs	Containers	$\sigma_{(i,k)}^{(j,l)}$
YC 1	(1, 2)	$\sigma_{(1,2)}^{(2,2)} = 1$
	(2, 2)	$\sigma_{(2,2)}^{(3,2)} = 1$
	(3, 2)	$\sigma_{(3,2)}^{(4,2)} = 1$
	(4, 2)	$\sigma_{(4,2)}^{(5,1)} = 1$
	(5, 1)	$\sigma_{(5,1)}^{(5,2)} = 1$
	(5, 2)	
YC 2	(1, 1)	$\sigma_{(1,1)}^{(3,1)} = 1$
	(3, 1)	
YC 3	(2, 1)	$\sigma_{(2,1)}^{(4,1)} = 1$
	(4, 1)	

Secondly, the order of containers handled by each YC is given in table 3.6, which is obtained from the solutions of $\sigma_{(i,k)}^{(j,l)}$. According to this table, YC 1 will be assigned to handle six containers while the other YCs will only handle two containers. This is mainly because the problem size is small (i.e. only a small number of containers), for most of the time, YC 2 and YC 3 are idle and available to work when YC 1 is busy. Note that each YC is available to handle its first container at the yard when the unloading process starts. Each YC handles containers in the yard following the order as listed. For example, YC 2 first collects container (1, 1) from an AGV and moves it to its

location and then picks up container (3, 1) from an AGV; YC 3 first picks up container (2, 1) from an AGV and then after taking this container to its location, YC 3 collects container (4, 1) from an AGV at the transfer point of the block.

Thirdly, we look at the schedule of each AGV which is given in table 3.7. This decision is obtained from the optimal solutions $x_{(i,k)}^{(j,l)}$. For example, in this solution, AGV 1 is assigned to deliver container (1, 2) first, then deliver containers (2, 2), (3, 2), (5, 2) in order; AGV 2 delivers containers (1, 1), (4, 1) and (5, 1) in sequence from the quayside to the yard; AGV 3 has been dispatched to deliver container (2, 1) first, then containers (3, 1) and (4, 2).

Table 3.7: The delivery sequences of AGVs- the values of the optimal solution $x_{(i,k)}^{(j,l)}$

AGVs	Containers	$x_{(i,k)}^{(j,l)}$
AGV 1	(1, 2)	$x_{(1,2)}^{(2,2)} = 1$
	(2, 2)	$x_{(2,2)}^{(3,2)} = 1$
	(3, 2)	$x_{(3,2)}^{(5,2)} = 1$
	(5, 2)	
AGV 2	(1, 1)	$x_{(1,1)}^{(4,1)} = 1$
	(4, 1)	$x_{(4,1)}^{(5,1)} = 1$
	(5, 1)	
AGV 3	(2, 1)	$x_{(2,1)}^{(3,1)} = 1$
	(3, 1)	$x_{(3,1)}^{(4,2)} = 1$
	(4, 2)	

Lastly, we list the values of $u_{(i,k)}$ and $d_{(i,k)}$ in table 3.8, i.e. the time QCs start to handle each container from the ship, and the time YCs start to pick up each container from the AGVs. The schedules of all the handling equipment can thus be obtained. For example, when the unloading process starts, container (1, 1) and container (1, 2) are handled by QC 1 and QC 2 respectively at time 0; at time 161 seconds, container (1, 1) has been collected by a YC and at time 55 seconds, container (1, 2) has been picked up by another YC. The optimal berth time is achieved by container (5, 2), which is $u_{(5,2)} + h_{(5,2)}$ =445+137=582 seconds.

Table 3.8: The start handling time of QCs and YCs for each container- the values of the optimal solutions $u_{(i,k)}$ and $d_{(i,k)}$

Containers (i, k)	$u_{(i,k)}$ (sec)	$d_{(i,k)}$ (sec)
(1, 1)	0	161
(2, 1)	93	205
(3, 1)	143	363
(4, 1)	295	436
(5, 1)	434	560
(1, 2)	0	55
(2, 2)	35	225
(3, 2)	185	311
(4, 2)	298	500
(5, 2)	445	622

Due to the computational complexity (increase in number of constraints and increase in number of variables), it is unlikely that the optimal solutions for large-sized problems (hundreds of containers) of the proposed problem can be obtained in a reasonable time using the B&B algorithm embedded in AIMMS 3.11. For example, when the problem size increases to 15 (the number of containers), there are 3560 constraints and it takes almost 300 minutes to return a solution (refer to table 3.9). Thus, in the following section, we propose a heuristic solution method based on genetic algorithm for solving the problem in large sizes.

3.1.3 Solution method: genetic algorithm

Genetic algorithm (GA) is a well-known heuristic approach for finding solutions to optimisation problems (Holland, 1975; Goldberg, 1989). The GA procedure adopted in this research is shown in figure 3.3.

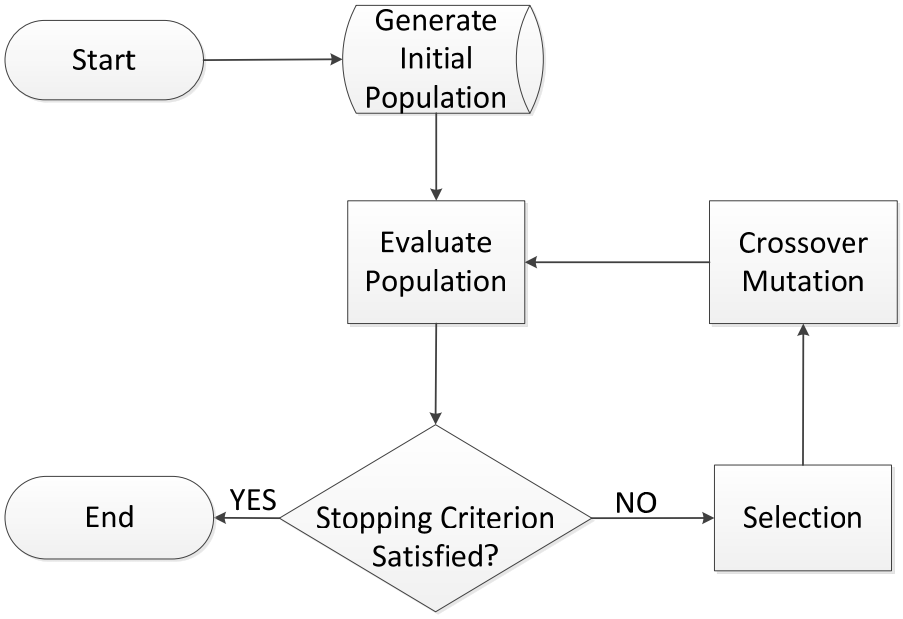


Figure 3.3: Flow chart of the genetic algorithm

For the AGV scheduling and storage allocation problem in the unloading process, however, little literature has explored adopting GA to solve this integrated problem. This might be due to the difficulty in presenting the problem as an expression of genetic chromosome and defining its evolving process. Therefore, in this section, we first propose, and then attempt, to represent the AGV scheduling and storage allocation in

the context of the GA-based evaluation process. Our GA-based approach adopts population-based evolving methods to efficiently explore the global solution space and search approximately optimal solutions quickly. Such a solution method would allow us to solve the computational complexity challenge of the integrated problem in a reasonable time, which cannot be solved by existing techniques (i.e. tree search, branch-and-bound). We introduce the design of our GA-based approach in the following steps.

Chromosome representation and initialisation: The initial step of the GA design is the chromosome/solution representation. It is known that the structure of the initial population plays an important role in the performance of the GA (Goldberg, 1989). By considering the decision problems $x_{(i,k)}^{(j,l)}$, $y_{(i,k)}^{b}$, $z_{(i,k)}^{(n,b)}$ and $\sigma_{(i,k)}^{(j,l)}$, we use the matrix structure to represent the solution of the proposed problem. The matrix indicates the allocated AGVs (column 1) and YCs (column 2), and the assigned yard locations associated with each container (column 3). More specifically, for containers handled by each QC, we have a matrix Ψ with three columns, which represent the AGV delivery sequences, YC handling sequences and yard location assignments (see the form of Ψ in figure 3.4). Each row in matrix Ψ is the chromosome representation for each container under QC k. Let $|D|$ denote the total number of import containers, v the total number of AGVs and c the total number of YCs.

The initial population is constructed by the following steps:

(1) Compare travelling times from each QC to each available yard location, which consists of the time AGV travels from QC to block and the time YC travels from the transfer point of the block to yard location. We choose the locations with shortest travelling times to be allocated to all the import containers, which satisfies constraint (3.1.7). Then we assign numbered labels from 1 to $|D|$ for all selected locations.

(2) Randomly assign these locations to each container and each location can only be assigned to one container (column 3 of chromosome) to satisfy constraints (3.1.5) and (3.1.6).

(3) Randomly choose an AGV from 1 to v(constraints (3.1.3) and (3.1.4)) i.e. assign one AGV to deliver a container (column 1 of chromosome) so that constraints (3.1.1) and (3.1.2) are met.

(4) Randomly choose an YC from 1 to c, i.e. assign one YC to handle a container (column 2 of chromosome) so that constraints (3.1.11) and (3.1.12) are met.

(5) Chromosomes are generated respectively by steps 1-4 until the population size *Pop* reaches a given number (e.g. 100) to ensure a large search space to start with.

(6) Evaluate each matrix Ψ in the initial population by calculating the values of $u_{(i,k)}$ and $d_{(i,k)}$ according to constraints (3.1.13)-(3.1.18). The objective function is obtained by $\text{Max}(u_{(N_k,k)} + h_{(N_k,k)})$, where N_k the last container is handled by QC k.

QC1			
Container	Dispatched AGV	Assigned YC	Assigned location
(1, 1)	1	1	1
(2, 1)	2	2	3
(3, 1)	2	2	8
(4, 1)	3	3	6
(5, 1)	1	1	10

QC2			
Container	Dispatched AGV	Assigned YC	Assigned location
(1, 2)	2	3	2
(2, 2)	3	1	4
(3, 2)	1	2	7
(4, 2)	3	1	5
(5, 2)	1	3	9

Figure 3.4: Chromosome representation example for 10 containers handled by two QCs

According to the sequence list of containers handled by each QC, we illustrate the corresponding chromosome representation for 10 containers in figure 3.4. For example, the matrix Ψ for the chromosome of QC 1 is the columns of 'dispatched AGV', 'assigned YC' and 'assigned location'. This figure gives a solution for the problem with 10 containers, two QCs, three AGVs, three YCs, ten selected locations and three blocks. Assuming locations 1-4 are in block 1, locations 5-7 are in block 2 and locations 8-10 are in block 3, then container (2, 2) will be delivered by AGV 3, and then handled by YC 1 to be stored in location 4 (in block 1); and container (3, 2) will be transported by AGV 1, then collected by YC 2 to be stored in location 7 (in block 2), etc.

Constructing the initial population by this method ensures that all the generated chromosomes are feasible and can be evaluated. This is because:

Firstly, according to steps (3) and (4), each container will be handled by one AGV and one YC. For those containers that have been assigned to the same AGV/YC, the sequences can be obtained according to each generated chromosome. For example, as in figure 3.4, the delivery sequence of AGV 1 is container (1, 1), (5, 1), (3, 2) and (5, 2); the sequences for all the AGVS and YCs can be obtained similarly; This ensures that

each container has one succeeding container and one preceding container delivered by the same AGV.

Secondly, we apply a simple heuristic to assign storage locations to containers. Once a location has been allocated to a container, this location will be removed from the 'selected locations' list (as obtained in step (1)) and thus cannot be assigned to another container. This ensures that each container will be assigned to one location and each selected location can only accommodate one container.

Thirdly, after the main decisions (the container handing sequences of AGVs and YCs, and the assigned container locations) have been made, the unloading process is carried out exactly according to the schedules. This process is simulated (calculated according to assumption (5) in section 3.1.1) and the berth time can thus be obtained.

Genetic operators design: In order to efficiently explore the solution space and maintain the feasibility of the newly generated offspring simultaneously, we propose the following crossover and mutation operations.

(1) *Two-point crossover:* the two-point crossover approach is proven to encourage the exploration of the global search space, rather than causing early convergence in the search (e.g. as a local optimal solution), thus making the search of optimal solutions more efficient (Spears and De Jong, 1991). This approach is proposed for the first and second columns of the chromosome matrix Ψ, because each container can be handled by any AGV and any YC. The new offspring is produced by randomly choosing two points along the length of chromosomes of parents and then exchanging the genes between the two points (as highlighted in figure 3.5). The two-point crossover guarantees that the generated children will remain feasible if the parents are feasible.

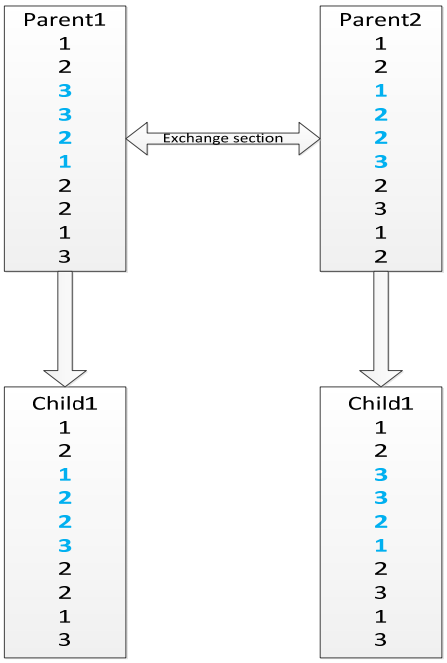


Figure 3.5: An illustration of two-point crossover for an example of 10 containers

(2) *Uniform order-based crossover:* The two-point crossover operator has been adopted for the first and second columns of the chromosome matrix Ψ; however, such an approach cannot apply to the third column of the chromosome due to constraints (3.1.6) and (3.1.7), i.e. one-to-one assignment between containers and locations to ensure one location is exactly assigned to only one container, otherwise there will be redundant and missing genes in the generated children by adopting the two-point crossover. In addition, the two-point crossover defines cross points as places to split and recombine the chromosome, while uniform crossover generalises this scheme to make every place a potential crossover point to improve global solution search space. Particularly, we construct the uniform order-based crossover operator in a similar way to the algorithm proposed in Cheng and Gen (1997), for the third column of the chromosome matrix Ψ. For example, figure 3.6 shows the uniform order-based crossover operator for creating one child. Such a crossover operator generates a template binary string (as shown in figure 3.6) with the same length as the chromosome (10 rows in this example because there are 10 containers) based on the uniform distributed "1"s and "0"s. The template string is then mapped to one of the selected parents, in which the genes that have the same positions with "1"s in the template string, i.e. 1, 2, 5, 7, 9, are given to a child, and the

remaining empty genes of this child are filled from another parent with no duplicates, i.e. 3, 4, 8, 10, 6.

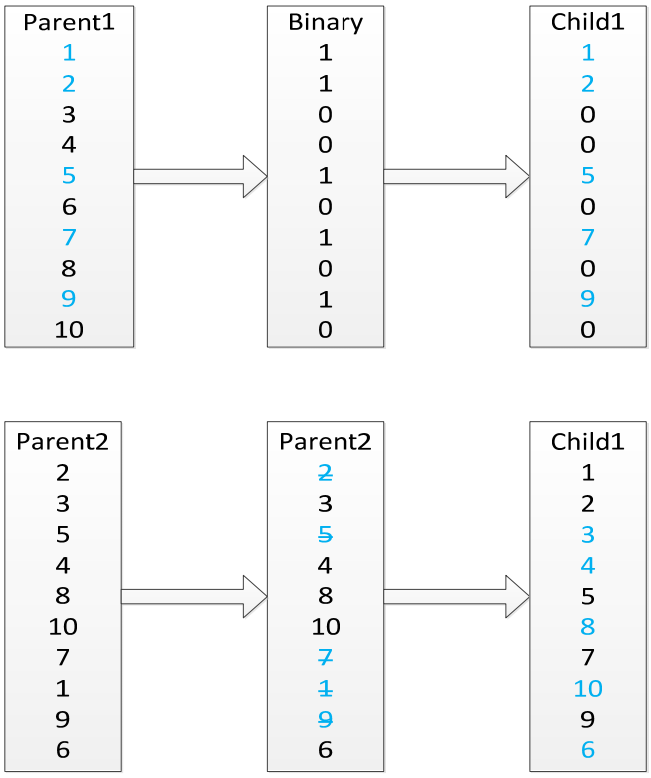


Figure 3.6: An illustration of uniform order-based crossover for an example of 10 locations

(3) *Swap mutation*: In addition to the crossover operators, mutation operation is adopted to maintain the diversity of the population in the successive generations and to maximise the exploitation of the solution space. To achieve the mutation operation, we specify a mutation probability P_m. As mutation happens rarely, P_m is set to be a small value, e.g. 0.1 (Yang, Wang and Li, 2012). For each individual Ψ in the population, we generate a random value uniformly distributed between 0 and 1, and compare this value with P_m. If the value is less than P_m, we perform the swap mutation on that individual, otherwise, there is no mutation operation, i.e. the individual remains the same. This mutation operation is carried out by choosing two positions (two rows) of that individual at random and then swapping the genes on these positions (see figure 3.7).

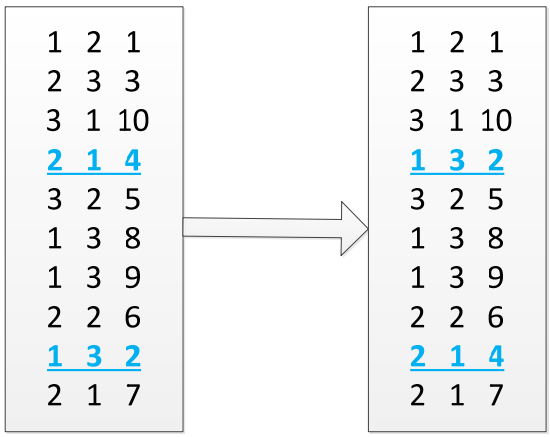


Figure 3.7: An illustration of swap mutation for an example of 10 containers

Offspring acceptance strategy: To achieve evolution between generations, in each generation, chromosomes are randomly paired to produce offspring for the next generation. We propose a semi-greedy strategy (first introduced by Hart and Shogan (1987) to accept offspring. In such a strategy, an offspring is accepted as the new generation only if its fitness is better than the average fitness of its parent(s). Using this strategy, we would guarantee that the offspring would always carry better than average genes (compared to its parents) to the next generation. Such an approach balances the needs for global solution space search as well as the evolution speed in terms of the number of evolving generations. It enables the whole GA to reduce the computation time of the optimisation process and guarantees a monotonous convergence (that is, the best OFV in any generation is equal to/less than the best OFV in the previous generation) toward evolving a near-optimal solution.

Parent selection strategy: Effective parent selection strategies (e.g. how to choose chromosomes in the current population as parents) are needed for creating offspring for the next generation and improving the speed of evolution. Generally, it is better to choose the best solutions in the current generation to create offspring so that the average OFV of the offspring is better than the average OFV of their parents. In the context of the AGV scheduling and storage allocation problem, the chromosomes with a smaller berth time, $\mathrm{Max}\left(u_{(N_k,k)} + h_{(N_k,k)}\right)$, are likely to be selected as parents for creating offspring. The most common method is the 'roulette wheel' sampling, in which each chromosome is assigned a slice of a circular roulette wheel and the size of the slice is

proportional to the chromosome's fitness. Here the fitness is represented by 1/OFV, which gives a bigger slice for the smaller berth time. The wheel spins population-size times. On each spin, the chromosome under the wheel's marker is selected to be in the pool of parents for the next generation. Here, as the objective is to minimise the berth time, we choose the ones with smaller objective function values (OFVs) as parents for the next generation. This parent selection strategy would always select a population of parents with smaller OFVs. To ensure that the best solution in the current generation always survives to the next generation, we use elitism strategy where the best individual is always kept in the population. Such a parent selection strategy will accelerate the entire evaluation procedure and search fast for an approximate optimal solution.

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

Initialise:

M_g-maximum number of generation

Pop-population size; ε-a small constant

P_c-crossover rate; P_m-mutation rate

Variables:

G-current number of generation

P- population

Pop_1-population for crossover

Pop_2-population for mutation

p_1, p_2-parents; c_1, c_2-children

Begin

$G \leftarrow 0$

$P \leftarrow initialise(Pop)$

Evaluate (P)

While $G \leq M_g \&\& \sigma_T > \varepsilon$

$\forall i \in P, \mu_i \in [0,1] \xrightarrow{assign} i$

if $u_i < P_c$, then add *i* toPop_1

$\forall p_1, p_2 \in Pop_1, (c_1, c_2) \leftarrow crossover(p_1, p_2)$

end

selection→ $update\ P$

$\forall i \in P, w_i \in [0,1] \xrightarrow{assign} i$

if $w_i < P_m$, then add *i* to Pop_2

$\forall p_1 \in Pop_2, c_1 \leftarrow mutation(p_1)$

end

selection→ $update\ P$

$P \leftarrow roulette\ wheel\ (P)$

G=G+1

End (Loop until terminated)

Figure 3.8: Pseudo code for our proposed genetic algorithm for the AGV scheduling and storage allocation problem

Figure 3.8 shows our GA for the proposed problem. First, we generated a number of population-sized initial solutions. Then for each individual solution, we randomly assign a number generated from 0 to 1 (based on uniform distribution) and compare this number with the crossover rate. If this number is less than the crossover rate, then the corresponding individual will be selected into the 'population for crossover'; if not, then the individual will be kept for the next step (mutation). The 'population for mutation' is also generated similarly through comparing the randomly generated value and the mutation rate. Each time a new offspring is generated after crossover/mutation, we apply the 'offspring acceptance strategy' to decide whether to accept this offspring or not. We always keep the same size of individuals during the GA operations. Therefore, after crossover and mutation, we have population-sized individuals, and each individual has a chance to be selected as a parent for the next generation. According to 'roulette wheel sampling', some chromosomes with smaller OFVs (higher fitness) may be selected more times while some chromosomes with bigger OFVs may not be selected. The wheel is spun population-sized times. This approach guarantees that those individuals with 'good genes' will pass on to offspring. After the parents selection step, we update the population and test whether to move to the next generation or stop the algorithm with our proposed GA, we aim to transform the proposed integrated model into a search problem which can be solved by adapting it to the evolutionary nature of GA. Such an approach would allow our proposed algorithm to search the solution space effectively and achieve fast optimal solution approximation.

3.1.4 Computational results

For model evaluation, we aim to obtain optimal solutions fast, which would satisfy the constraints introduced in Section 3.1.1, Eqs. (3.1.1)-(3.1.18) within a reasonable time duration (e.g. minutes). However, it is known that container scheduling is an NP-hard problem (Bish *et al.* (2001)); that is, the computation time required to optimally solve

such a problem increases exponentially as the problem size grows. To address this computational complexity challenge, we adopted two approaches. First, we evaluate a set of scenarios with different parameter settings by adopting a branch and bound (B&B) algorithm developed using AIMMS 3.11 (note as B&B). Small-sized problems can be solved by B&B with optimal solutions. However, it is not practical to obtain optimal solutions by using B&B when the problem size becomes larger (e.g. 100+ containers) due to the large number of constraints as well as the complexity of the model. Therefore, secondly, we adopted our proposed GA as introduced in Section 3.1.3 for obtaining approximate optimal solutions for large-sized problems fast. Furthermore, in order to evaluate the impact of GA initial parameter settings (such as initial population size Pop, crossover rate P_c, mutation rate P_m and maximum generation M_g), we also execute a set of parameter sweep evaluations for our proposed GA. This would allow us to fully understand the performance of GA and the effect of the parameter settings.

The experiment design and parameter values are chosen based on the values in the literature and also based on the problems considered here. For example, the classification of problem size (small size and large size) differs in previous work: some work considered less than eight containers as a small-sized problem (Lee, Cao and Shi, 2008b; Tavakkoli-Moghaddam *et al.*, 2009) and some work considered less than 30 as small-sized problem (Cao *et al.*, 2010; Kim, Lee and Hwang, 2003; Lee *et al.*, 2010); apart from this, QC handling times, YC handling times, vehicle travelling times are assumed to follow uniform distributions in many previous studies but with different values (Cao *et al.*, 2010; Kim, Lee and Hwang, 2003; Fereidoonian and Mirzazadeh, 2012; Lee *et al.*, 2009; Lee *et al.*, 2010; Tavakkoli-Moghaddam *et al.*, 2009); in addition, GA parameters were also taken with different values. Because our aim of running the computational experiments is to test the effectiveness of our proposed model and heuristic algorithm, for all experiments in section 3.1.4, we consider the following experimental and parameter settings:

(i) All experiments are based on the layout as illustrated in figure 3.1: QCs work at the quayside for unloading containers, AGVs travel between the quayside and storage yard, where YCs stack and store containers. Note that YCs are not one-to-one assigned to blocks.

(ii) The number of containers varies from 5 to 200, where 5-20 are considered as small-sized problems and 20-200 are considered as large-sized problems, similarly with the work of Lee *et al.* (2010). We also consider different number of AGVs from 3 to 10; the number of YCs varies from 2 to 5; and the number of blocks varies from 2 to 8.

(iii) The uniform distribution was assumed for all operations/travelling times (Lau and Zhao, 2008), because containers are evenly distributed and located on the ship and in the yard. The travelling times (in seconds) of YCs from the transfer point in front of each block to each available location within that block are generated from a uniform distribution U(60,140)s; the processing times of QCs unloading these containers follows a uniform distribution U(30,180)s.

(iv) AGV's travelling times from each QC to each block are known according to the terminal's layout. QCs have fixed positions along the ship because it takes a very long time for QCs to move; these moves are very unproductive and should be avoided. For simplicity, the values of these travelling times are generated from uniform distribution U(20, 120)s; YC's move times between blocks are also known by the locations of blocks. Here we assume the move times between transfer points of any two adjacent blocks are 40s, so the move times between any two blocks can be calculated similarly, as in table 3.3.

(v) To test the performance of our proposed algorithm, when evaluating the model, we set the following GA parameters: crossover rate $P_c = 0.8$, mutation rate $P_m = 0.02$, Population size $Pop = 100$, Maximum generation $M_g = 50$. In the literature, these GA control parameters had taken different values based on different problems. For example, crossover rate takes the value of 0.7 (Mak and Zhang, 2009), 0.8 (Han, Lu and Xi, 2010), 0.25 (Tavakkoli-Moghaddam *et al.*, 2009), or ranges from 0.7 to 0.9 (Choi *et al.*, 2011; Hakam, Solvang and Hammervoll, 2012); mutation rate is usually very small, such as 0.05 (Kozan and Preston, 2006), 0.01 (Casey and Kozan, 2006), or ranges from 0.01 to 0.03 (Hakam, Solvang and Hammervoll, 2012); similarly, population size can be 50 (Yang, Wang and Li, 2012), 100 (Choi *et al.*, 2011), 150 (Tavakkoli-Moghaddam *et al.*, 2009), 200 (Jung and Kim, 2006), 500 (Huang, Liang and Yang, 2009) and maximum generation takes the value of 30 (He *et al.*, 2010),

50 (Casey and Kozan, 2006), 150 (Lau and Zhao, 2008), 500 (Han, Lu and Xi, 2010). Here, we carried out some preliminary tests in order to select GA control parameters, and then chose the above settings for our experiments taking into account computation times and OFVs. We will do a GA parameter sweep evaluation later in this section, in which these parameters take several different values.

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

Results for small-sized problems

Ten small-size experiments are considered with the number of containers varies from 5 to 20. To reduce possible bias generated by the randomness of GA in a single experiment, each model is run 20 times by GA. For all the runs, we adopt the same setting of parameters to obtain more reliable results, and average values of OFVs (in seconds) and computation times (in seconds) are computed as the final results, which is similar to the approach taken by Bazzazi, Safaei and Javadian (2009).

Table 3.9 shows that for small-sized problems, our proposed GA can obtain approximate optimal solutions comparable to the B&B in a faster speed, ranging from 1.24 seconds to 17.51 seconds while B&B takes more computation time to return the results. Specifically, there is not much difference in the results between these two evaluation approaches: the OFV difference between B&B and GA ranges from 0% to 2.1%, and thus the average OFV difference is 0.7% in the first eight cases, which is a very impressive result as shown in table 3.9. For smaller-scale examples, from the cases 1, 2, and 3, the OFVs obtained by GA are the same as those solved by B&B in AIMMS 3.11. However, we observe that exact algorithm (B&B) cannot solve larger-sized problems in an acceptable time duration; AIMMS 3.11 could not return any results for

the problems with more than 15 containers, and the computation time of B&B grows exponentially as the problem size increases.

Table 3.9: Results of computational experiments in small sizes

No	Containers	AGVs/ QCs/ YCs	B&B (MIP)		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
1	5	2/2/2	34.49	386	4.12	386	0%
2	6	2/2/2	35.02	406	2.84	406	0%
3	7	2/2/2	10.97	426	3.15	426	0%
4	8	3/2/3	18.16	560	1.24	563	0.5%
5	9	3/2/3	12.81	793	2.22	798	0.7%
6	10	2/2/3	713.83	781	1.35	788	0.8%
7	10	2/2/2	502.41	825	1.26	833	0.9%
8	15	3/2/2	16070.26	988	4.11	1009	2.1%
9	20	3/2/3	/	/	10.06	1208	/
10	20	4/2/3	/	/	17.51	873	/

Results for large-sized problems

Due to the exponential increases of the solution search space, when the model size grows, it has been show (Tavakkoli-Moghaddam *et al.*, 2009) that it is not possible to compute optimal solutions for the models with large sizes by using existing mathematical programming approach (e.g. the introduced B&B approach). To evaluate the scalability of our proposed GA, we examined a number of large-sized scenarios, as shown in table 3.10.

Table 3.10 : Results of computational experiments in large sizes (in seconds)

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

28	150	8/3/4	282.27	9324
29	150	8/3/5	424.25	8363
30	200	8/3/5	503.01	11654
31	200	9/3/5	406.44	11490
32	200	10/3/5	761.10	11319

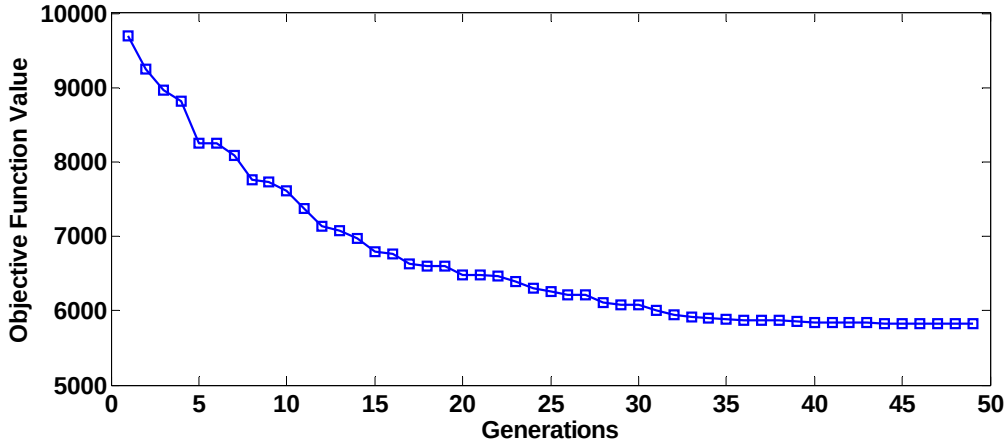


Figure 3.9: Typical convergence of GA for the case with 100 containers, five AGVs, three QCs and four YCs.

GA searches the solution space by evolving a population of solutions through crossover, mutation and selection strategies. The main advantage of GA over the neighbourhood search approaches (refer to section 2.3.2) is that it can explore a larger area of solution space in order to evolve a near optimal solution. The simplest practical rule for this problem is to dispatch available vehicles to containers on a First-Come-First-Served (FCFS) bases and to allocate container storage locations randomly or based on operators' experiences. However, this practical rule can only evolve a feasible solution, and our proposed approach can obtain the optimal or near-optimal solution. In addition, as stated in Kim and Bae (2004), in container terminals, AGVs must be dispatched in advance by considering future delivery tasks so as to optimise the shipside operations and FCFS rules cannot be used. Other previous works, such as Bierwirth and Meisel (2010) and Preston and Kozan (2001a), also found that heuristic method is able to obtain better solutions, comparing with the practical rules, such as FCFS, first come last

serve (FCLS) and others. From our experiments, we observe that (1) our proposed GA performs stably to provide approximately optimal solutions for large sizes even when the number of containers reaches 200; for all the cases, the computational running times can be completed within minutes; (2) The OFVs increase with the problem size (number of containers) as expected, which means it takes more time to unload more containers; and (3) the trend of performances of the number of AGVs/QCs/YCs is similar: when increasing the AGV/QC/YC numbers, the OFV reduces (for example, case 15 and case 16 for the effect of AGV numbers). The effects of the number of cranes (QCs and YCs) are more significant than the effect of the number of AGVs on the berth time: the proportion of reduction in the OFVs when increasing the number of cranes is larger than that with an increased number of AGVs (such as case 18 and case 19 for the effects of QCs numbers; case 11, 12 and 13 for the effects of YCs numbers). However, for each problem, it is also important to choose the appropriate number of vehicles and cranes; otherwise, the increasing in the number of equipment will cause the further traffic congestions or conflicts during the container handling process, which influences the container terminal's efficiency. In the perspective of computation time, it also takes longer time to get the solution with the increased number of equipment. Figure 3.9 shows the typical convergence of the GA for a case of 100 containers, five AGVs, three QCs and four YCs. It shows that GA converges steadily and fast to a fixed value, which demonstrates that our proposed GA is able to provide the approximately optimal solutions. Computation time is important in real-time operations at the container terminal, because when the ship arrives at the terminal, the handling operations of containers should start as soon as possible.

GA parameters sweep experiments

Now we look at the experiments on evaluating the impact of GA parameters, which would show the performance of the proposed GA with different initial parameters, and give the possible optimal parameter settings for GA. A combination of several different values is taken in order to find the values with best convergence performance curve, i.e. faster convergence and better (smaller) OFV. A similar approach to this is adopted by Fereidoonian and Mirzazadeh (2012), Han, Lu and Xi (2010), and Yang, Wang and Li

(2012). For the problem with 50 containers, five AGVs, three QCs and five YCs, GA parameter settings take the following values:

Crossover rate P_c = {0.6, 0.7, 0.8, 0.9};

Mutation rate P_m = {0.01, 0.02, 0.1, 0.2};

Population size Pop = {30, 50, 100, 150};

$M_g = 50$.

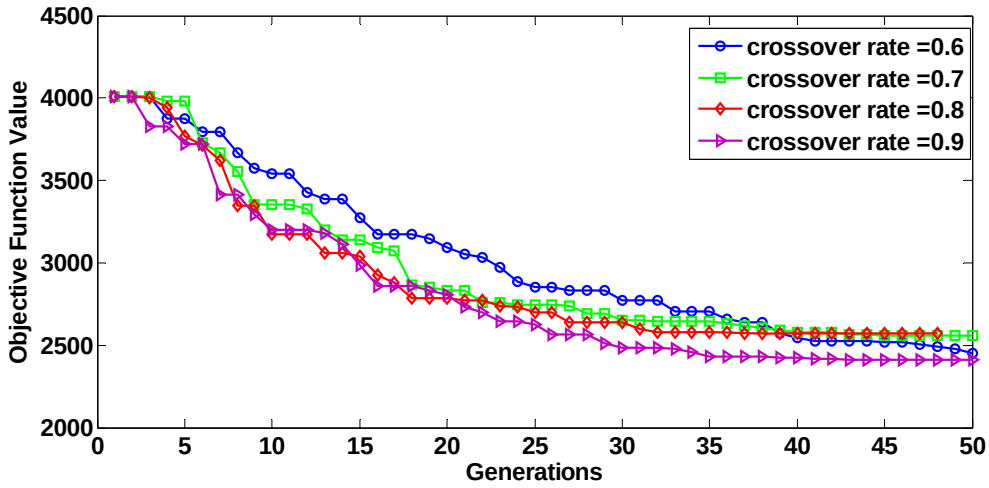


Figure 3.10: Performance comparison of different crossover rates for an example under the $Pop = 100$ and $P_m = 0.02$

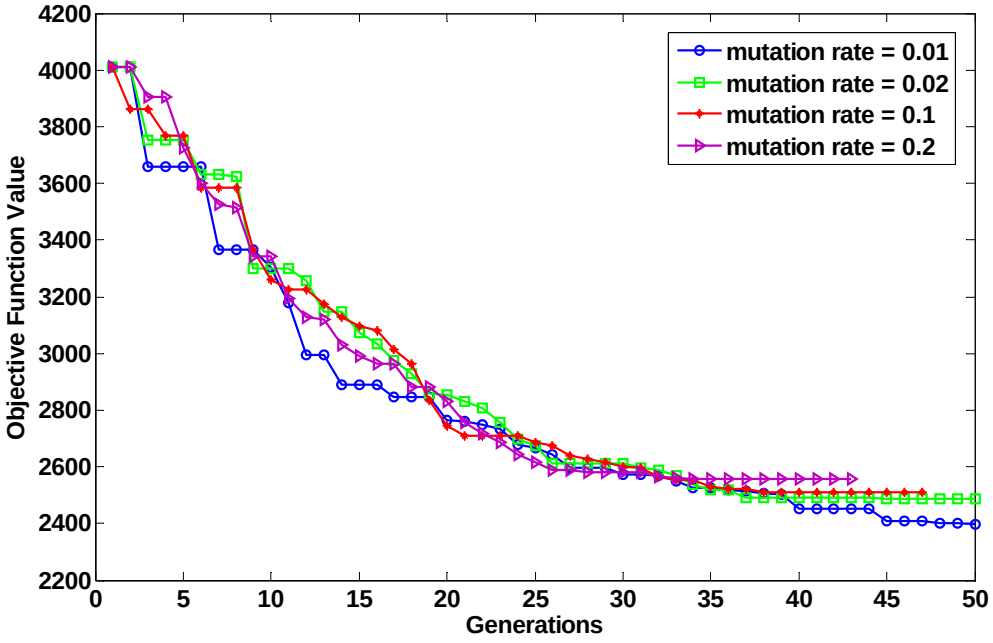


Figure 3.11: Performance comparison of different mutation rates for an example under the $Pop = 100$ and $P_c = 0.9$

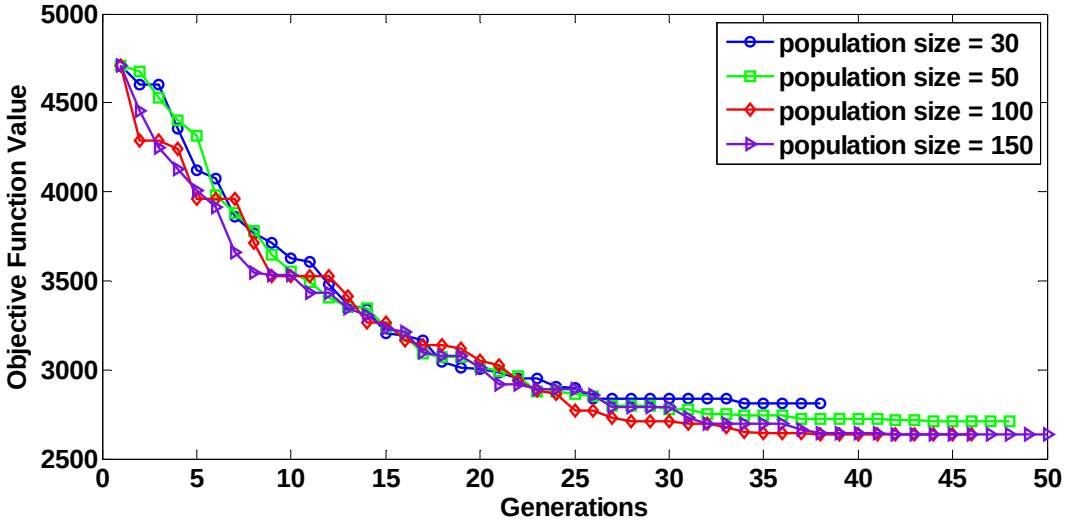


Figure 3.12: Performance comparison of different population size for an example under $P_c = 0.9$ and $P_m = 0.01$.

Figures 3.10 to 3.12 show the performance comparisons among different parameter settings. According to these convergence curves, we observe that the one of crossover rate P_c= 0.9, the one of mutation rate P_m= 0.01, and the one of population size *Pop* =100

outperform others for this particular problem. Specifically, in figure 3.10, the curve with $P_c = 0.9$ converges to a smaller value of OFV; in figure 3.11, the curve with $P_m = 0.01$ also converges to a smaller value of OFV, and in figure 3.12, the curve with $Pop = 100$ and $Pop = 150$ converge to the same value; however, the one with $Pop = 100$ stops earlier, which requires a shorter evolving time. These figures also show that for all the experiments, OFVs are not improved after 50 generations due to the fast convergence speed of our proposed GA . So the maximum generations of 50 will be sufficient to obtain near-optimal solutions, which is ideal and smaller than many other proposed GA -based approaches (He *et al.*, 2010; Kozan and Preston, 2006; Han, Lu and Xi, 2010).

At last, in order to test the stability of our proposed GA and analyse the possibility of random effect (e.g. occasional fast optimal solution), we repeated our proposed GA 10 times for the case of 60 containers, five AGVs, three QCs and four YCs with the same parameter settings. Figure 3.13 shows the box plot of results in 10 runs. Box plot is able to show the range of OFVs in each generation. Each box represents the OFVs in one generation. The central mark is the median (50% percentile), the bottom and top of the box are the 25^{th} and 75^{th} percentiles (the lower and upper quartiles, respectively) and the whiskers are the most extreme data points (1.5 times more than upper quartile or 1.5 times less than lower quartile). The ends of the box plot in each generation are the maximum and minimum values excluding extreme values. Therefore, this figure further proves that our proposed GA performs in a stable manner in all the experiments.

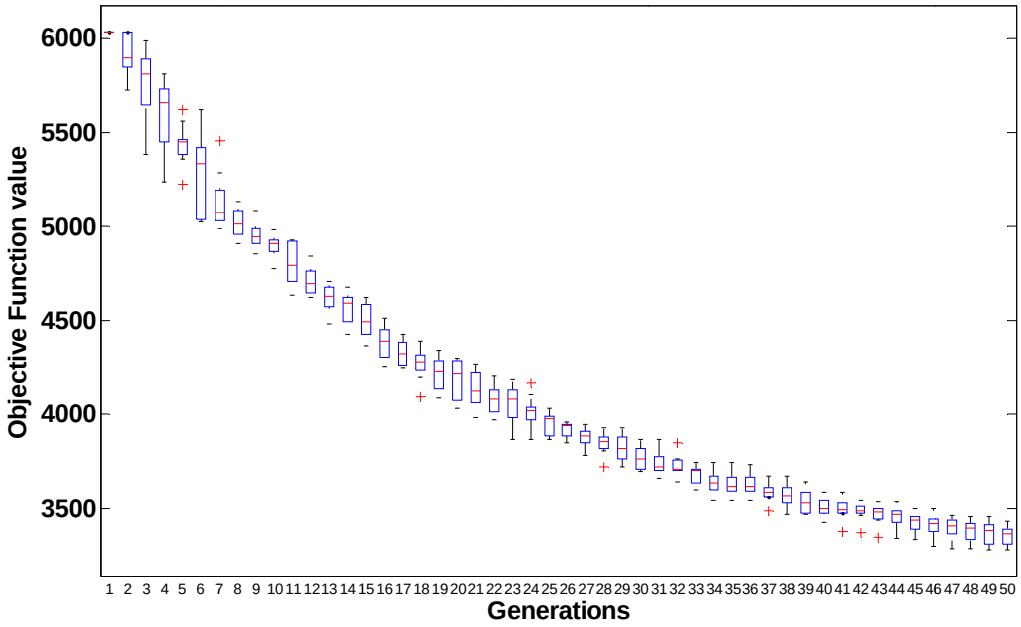


Figure 3.13: GA performance in 10 runs with $P_c = 0.9$, $P_m = 0.01$ and $Pop = 100$ for the case with 60 containers, five AGVs, three QCs and four YCs

3.2 Model 2: Scheduling of container-handling equipment during container loading process

The makespan of the container loading operation highly depends on the loading sequence of containers by AGVs/cranes as well as the number of containers to be moved (Kim and Kim, 1999b). Therefore the makespan for the loading operations can be improved significantly by efficiently scheduling the loading sequences. To improve the productivity of an automated container terminal, it is important to harmoniously synchronise operations of different types of container-handling equipment, e.g. automated guided vehicle (AGV), yard crane (YC) and quay crane (QC). Therefore, this section seeks to address the scheduling problem of all the handling equipment in an integrated way by utilising information about export containers' yard storage locations in an automated container terminal. In the problem considered in this section, the yard locations for export containers are known: this is because when the loading process starts, export containers are all located in the yard, which is different in the case of the import containers. A mixed-integer programming (MIP) model is developed to

formulate the problem with the objective to minimise the ship's berth time. As the existing solver cannot solve the MIP model for large sizes in a reasonable time, we propose a genetic algorithm (GA) to obtain approximate optimal solutions to tackle the computational difficulty in solving large-sized problems. Computational results are provided to demonstrate the benefits of integrated methods and the effectiveness of the proposed algorithm.

3.2.1 Problem description and formulation

During the loading process, a container picked up by an YC is put on an AGV that delivers the container to the quayside. At the quayside, a QC picks up the container from the AGV and puts it in its location on the ship. The layout of the automated container terminal considered in this section is the same as that in figure 3.1. In this section, we adopt the pooling strategy for AGVs where AGVs can serve any QC, as illustrated in figure 3.2. Congestion among AGVs on the path is not considered because the study about the interference of vehicles involves more complex scheduling of detailed movements of vehicles, which is another important issue for the AGV system in automated container terminals (Evers and Koppers, 1996).

Each export container will be loaded to a specified stack on the ship, and the time required to load an export container by a QC depends on its stack location on the ship. In addition, in the vessel, ship-bays are partitioned, each of which will be served by one QC (Lee, Cao and Shi, 2008b). Interferences among QCs are not considered in this study, because in the real-life case, any two adjacent QCs must be set apart from each other by at least one ship-bay to avoid interference. If a container is in the stack which is in the covered area of a QC, then this container will be handled by this QC. Therefore, the destinations of containers (which QC will handle which container) are known. Usually, containers in the same stack on the ship will go to the same destination port; so these containers (sharing the same stack) can be loaded in any order. Therefore, in the proposed problem, we assume that the containers can be handled by QCs in any order. The same assumption has been made in the work of Cao *et al.* (2010). We aim to model different problems, so this assumption is different with the assumption in model 1. The detailed QC schedule will then be determined based on the information of container locations on the ship and the synchronisation with other handling equipment.

The export containers usually arrive at the terminal over a period of more than a week before the scheduled loading time, and are assigned to particular slots in the storage yard (Kang, Ryu and Kim, 2006). Therefore, the locations of export containers in this study are assumed to be pre-determined.

The problem being investigated is to minimise the loading time element of berth time of the ship, which is the time duration when all export containers have been transferred and placed onto the ship. This loading time duration consists of (1) the retrieving time for containers by YCs at the storage yard; (2) the travelling time of containers from the storage yard to the ship by AGVs; and (3) the handling time for all containers to be placed on the ship by QCs. Because the transportation between the yard-side and quayside plays a crucial role in determining the productivity of the terminal (Lee *et al.*, 2010), attention is paid to the AGV scheduling problem, in which the sequence and time of containers handled by the AGVs will be determined. Additionally, the problem will also determine the container-handling sequences of the YCs during the loading operations, which is very important in improving the overall efficiency of the system (Chen *et al.*, 2007; Li *et al.*, 2009). An integrated model is developed in order to address the problems at the same time.

The objective of this section is to analyse the automated container terminal system and determine a detailed schedule for QCs, AGVs and YCs during the container loading process in order to improve the terminal's operational efficiency by reducing the berth time of the ship.

From the descriptions above, we summarise the assumptions regarded in this study:

(1) Only loading operations are studied in this model.
(2) The yard storage locations for export containers are given.
(3) The locations of QCs by which containers will be loaded (destinations of containers) are known.
(4) Number of export containers, number of QCs, YCs and AGVs are all known.
(5) QCs, YCs and AGVs can only take one container at a time.
(6) Pooling policy is applied to AGVs, which means AGVs are shared among all the QCs.

(7) AGV's travelling times between any two processing locations (transfer points of blocks and QCs) are known, e.g. travelling time between each QC and each block, travelling time among transfer points of blocks, etc.

(8) Traffic congestion of the AGVs on the path is not considered.

(9) The interference among YCs and the interference among QCs are also not considered.

(10) The time needs for YCs dropping off containers onto AGVs and QCs picking up containers from AGVs are negligible.

Now we develop a mixed-integer programming (MIP) model for the integrated scheduling problem of container-handling equipment during the loading process. As discussed, the objective is to minimise the berth time of the ship. The main operational decisions are to determine the schedules, i.e. times and sequences to handle containers, of AGVs, YCs and QCs. This is because export containers arrive at the terminal a few days before loading onto the ship; their storage locations are already known. We specify the yard location information by YC's handling time of each container, which is the time YC needs to handle each export container from its location to the transfer point in front of the block. Therefore, different with model 1, we do not have the storage-block-related notations. The following are the notations related to this problem. Each loading container is referred to as a job.

Sets and parameters

N set of export containers

Q Set of QCs

W set of YCs

K set of AGVs

k index for AGVs

w index for YCs

q	Total number of QCs
i, j	index for export containers
h_i	QC's handling time of container i
t_i	AGV's travelling time taking container *i* from storage yard to quayside
p_i	YC's handling time of container *i*
s_{ij}	AGV's travelling time from the destination QC of container *i* to the origin block (the block which stores container *j*) of container *j*
w_{ij}	YC's travelling time from the transfer point of the block which stores container *i* to the yard location of container *j*
M	a very large positive number
S	dummy starting job (container)
F	dummy ending job (container)
O_S	The job set which contains all the jobs including the dummy starting job, $O_S = N \cup \{S\}$
O_F	The job set which contains all the jobs including the dummy ending job, $O_F = N \cup \{F\}$
O	The job set which contains all the jobs including dummy starting and ending jobs, $O = \{S, F\} \cup N$

Decision variables

u_i	the time a QC picks up container *i* from an AGV

d_i the time a YC releases container *i* onto an AGV

In this study, the problems are to decide the sequences and times of containers for QCs, YCs and AGVs to operate. Let m and n be the indices of sequences (for AGVs and YCs), where $m, n \in N^+$ (positive integer). Then we introduce another three decision variables:

$$x_{imk} = \begin{cases} 1, if\ \ container\ i \text{ is the } m\text{th container delivered by AGV } k \\ 0\ , otherwise;\ \forall i \in N, \forall m \in N^+, \forall k \in K \end{cases}$$

$$y_{inw} = \begin{cases} 1, if\ container\ i \text{ is the } n\text{th container handled by YC } w \\ 0\ , otherwise;\ \forall i \in N, \forall n \in N^+, \forall w \in W \end{cases}$$

$$z_{ij} = \begin{cases} 1, if\ \ container\ j \text{ is handled after conainer } i \text{ by same QC} \\ 0\ , otherwise;\ \forall i \in O_s, \forall j \in O_F \end{cases}$$

Objective: minimise the ship's berth time

$$Min: Max(u_i + h_i)$$

Subject to:

$$\sum_{m \in N^+} \sum_{k \in K} x_{imk} = 1, \forall i \in N \tag{3.2.1}$$

$$\sum_{i \in N} x_{imk} \leq 1, \forall m \in N^+, \forall k \in K \tag{3.2.2}$$

$$\sum_{i \in N} x_{imk} \geq \sum_{i \in N} x_{im+1k}\ , \forall m \in N^+, \forall k \in K \tag{3.2.3}$$

$$\sum_{n \in N^+} \sum_{w \in W} y_{inw} = 1, \forall i \in N \tag{3.2.4}$$

$$\sum_{i \in N} y_{inw} \leq 1, \forall n \in N^+, \forall w \in W \tag{3.2.5}$$

$$\sum_{i \in N} y_{inw} \geq \sum_{i \in N} y_{in+1w}, \forall n \in N^+, \forall w \in W \tag{3.2.6}$$

$$\sum_{i \in O_S} z_{ij} = 1, \forall j \in N \tag{3.2.7}$$

$$\sum_{j \in O_F} z_{ij} = 1, \forall i \in N \tag{3.2.8}$$

$$\sum_{i \in N} z_{iF} = q \tag{3.2.9}$$

$$\sum_{j \in N} z_{sj} = q \tag{3.2.10}$$

$$d_j + M\left(2 - x_{imk} - x_{jm+1k}\right) \geq u_i + s_{ij}, \forall i \in O_S, j \in O_F, \forall m \in N^+, \forall k \in K \tag{3.2.11}$$

$$d_j + M\left(2 - y_{inw} - y_{jn+1w}\right) \geq d_i + w_{ij} + p_j, \forall i \in O_S, j \in O_F, \forall n \in N^+, \forall w \in W \tag{3.2.12}$$

$$u_j + M\left(1 - z_{ij}\right) \geq u_i + h_i, \forall i \in O_S, \forall j \in O_F \tag{3.2.13}$$

$$u_i \geq d_i + t_i, \forall i \in N \tag{3.2.14}$$

$$x_{imk}, y_{inw}, z_{ij} \in \{0,1\}, \forall i, j \in O, \forall m, n \in N^+, \forall k \in K, \forall w \in W \tag{3.2.15}$$

$$u_i, d_i, t_i \geq 0, \forall i \in N \tag{3.2.16}$$

The objective is to minimise the loading element of the ship's berth time, which is described as the time duration in which all the export containers have been loaded onto the ship.

Constraint (3.2.1) implies that for container i, it must be delivered once and only once by an AGV taking it from the storage yard to the quayside.

Constraint (3.2.2) means an AGV can only deliver at most one container at a time.

Constraint (3.2.3) insists that every container, $i \in N$, must be succeeded by another container delivered by the same AGV.

Constraint (3.2.4) ensures that each container $i \in N$ must be handled once and only once by an YC taking it to its assigned yard location.

Constraint (3.2.5) means that a YC can only carry at most one container at a time.

Constraint (3.2.6) demonstrates that every container $i \in N$ is succeeded by another container handled by the same YC.

Constraint (3.2.7) means that for every container $j \in N$ there is exactly one preceding container that is assigned to the same QC.

Constraint (3.2.8) means that for every container $i \in N$, there is exactly one succeeding container that is assigned to the same QC.

Constraints (3.2.9) and (3.2.10) guarantee that the number of QCs employed for handling these containers is exactly q.

Constraint (3.2.11) implies that if container j is delivered after container i by the same AGV, then the time container j had been picked up by this AGV at the storage yard must be at least after container i has been collected by the QC and a certain travelling time.

Constraint (3.2.12) represents that if container j is handled after container i by the same YC at the storage yard, then YC needs at least a certain (empty-loaded) travelling time from the location of container i to the location of container j, plus the handling time of container j before container j had been collected from an AGV.

Constraint (3.2.13) represents that if container j is handled after container i by the same QC, then the QC can only start handling container j after container i has been located on the ship.

Constraint (3.2.14) states that the time a QC picks up an export container *i* from an AGV must be at least equal to the time this AGV collects this container *i* from a YC plus a certain travelling time of AGV from the block of container *i* to its assigned QC.

Constraints (3.2.15)-(3.2.16) provide the restrictions of the decision variables which are binary and non-negative.

3.2.2 An illustrative example

In this section, we present an example to show the performance of the proposed MIP model. The benchmark method, the branch and bound (B&B) algorithm, is adopted for evaluating the model. We implement this algorithm in AIMMS 3.11 with solver CPLEX 12.2, which has been proven to be capable of finding good solutions very quickly.

The operating environment considered is as follows: The layout of the automated container terminal adopted in this section is the same as shown in figure 3.1 (at the beginning of the chapter). There are two QCs working at the quayside for loading containers onto the ship. In the yard, three YCs are working to retrieve containers within three blocks and to load them onto the AGVs. Between the quayside and yard-side, three AGVs are travelling for delivering containers. YCs are not one-to-one assigned to blocks and they travel at constant speed. Because the destinations of containers are known and the AGVs travel at constant speed, the time AGV 1 spends delivering one container from the yard to its destination QC is the same with the time spent by AGV 2. We give an example of nine containers to be loaded onto the ship. The destination QC and the yard storage block are given in table 3.11; for example, as in table 3.11, container 1 which is located in block 1 will be loaded by QC 1. Container 4 located in block 2 will be loaded onto the ship by QC 1. The AGV travel time from each block to each QC is given in table 3.12; for example, as in table 3.12, the AGV travel time from block 2 to QC 1 is 105 seconds, and the travel time from block 1 to QC 2 is 53 seconds.

Table 3.11: The destination QC and yard storage location information in an example of nine export containers

Container	Destination QC	Storage block
1	QC 1	Block 1
2	QC 1	Block 1
3	QC 1	Block 2
4	QC 1	Block 2
5	QC 1	Block 3
6	QC 2	Block 3
7	QC 2	Block 3
8	QC 2	Block 3
9	QC 2	Block 3

Table 3.12: AGV travel time between each block and each QC for the case of two QCs and three blocks (in seconds)

QCs	Block 1	Block 2	Block 3
QC 1	101	105	61
QC 2	53	49	50

According to tables 3.11 and 3.12, the values of t_i (AGV travel time for each container from the storage block, where it is located, to its destination QC) and s_{ij} (the empty-loaded travel time of AGVs between any two containers) can be calculated. Take container 4 as an example: t_4 will be the AGV travel time from block 2 (where container 4 is located) to QC 1 (container 4's destination QC), which is 105 seconds. If an AGV delivers container 3 and then goes to pick up container 5, then the empty

loaded travel from the destination of container 3 (QC 1) to the yard location of container 5 (block 3) s_{35} is 61 seconds. These values are calculated and shown in tables 3.13 and 3.14.

Table 3.13: The values of t_i-the AGV travel time for each container from storage block to QC

Containers	1	2	3	4	5
t_i (sec)	101	101	105	105	61
Containers	**6**	**7**	**8**	**9**	
t_i (sec)	50	50	50	50	

Table 3.14: The values of s_{ij}-every combination of the empty loaded travel time of AGV between any two containers (in seconds)

i \ j	1	2	3	4	5	6	7	8	9
1	N/A	101	105	105	61	61	61	61	61
2	101	N/A	105	105	61	61	61	61	61
3	101	101	N/A	105	61	61	61	61	61
4	101	101	105	N/A	61	61	61	61	61
5	101	101	105	105	N/A	61	61	61	61
6	53	53	49	49	50	N/A	50	50	50
7	53	53	49	49	50	50	N/A	50	50
8	53	53	49	49	50	50	50	N/A	50
9	53	53	49	49	50	50	50	50	N/A

The handling time of each container by a QC is the time duration from picking up the container from an AGV until it is placed on its location on the ship. The handling time of each container by the YC is determined by the container's location in the yard; it is the time duration between retrieving the container in the storage yard and placing it onto the AGV at the working points in front of the block. Assuming containers are evenly distributed both on the ship and in the yard, the handle time of QC for each container - h_i and handle time of YC for each container - p_i are generated from uniform distribution U(30, 180)s and U(60, 240)s respectively. These values are shown in table 3.15.

Table 3.15: The values of QC handling times h_i and YC serving times p_i (in seconds)

Containers	h_i	p_i
1	80	104
2	73	232
3	128	107
4	87	84
5	74	122
6	133	98
7	175	210
8	132	138
9	180	160

The handling time of each container by an YC is defined as the time from each container's yard location to the transfer point in front of the block. As YC travels at constant speed, there is no difference in the speed of loaded move and empty-loaded move. Let the YC travel time between the transfer points of any two adjacent blocks be

40 seconds, as in table 3.3. There are two conditions when calculating the values of a YC's empty-loaded travel time w_{ij} for any two successive containers handled by the same YC:

(1) Containers in the same block. For any two consecutive containers located within the same block and to be handled by the same YC, the empty-loaded travel time from dropping off the previous container onto an AGV at the transfer point to the target container is the same as the process time of the target container because of the same travel distance. For example, consider that container 6 and container 7 will be performed by the same YC. After the YC releases container 6 onto an AGV at the transfer point of block 3, this YC moves to the location of container 6 without carrying containers, i.e. empty-loaded. Thus this empty-loaded travel time is the process time of container 7 and from table 3.15, which is 210 seconds.

(2) Containers in different blocks. For calculating the empty-loaded travel times between any two consecutive containers in different blocks and to be handled by the same YC, additional YC travel time between blocks should be added in addition to the process time. Let the travel time of YCs between the transfer points at block 1 and block 2 be 40 seconds. For example, if container 2 and container 3 are processed by the same YC, then after an AGV collects container 2 at the transfer point of block 1, YC will take 40 seconds moving to the transfer point at block 2; then goes to pick up container 3. Thus the empty travel time is 40+107=147 seconds, where 107 is the process time of container 3, as in table 3.15.

Therefore, the values of w_{ij} could be calculated followed by the above discussion: we simply list the results in table 3.16.

Table 3.16: The empty-loaded travel time between any two successive containers by the same YC

i \ j	1	2	3	4	5	6	7	8	9
1	N/A	232	147	164	202	178	290	218	240
2	104	N/A	147	164	202	178	290	218	240
3	144	272	N/A	124	162	138	250	178	200
4	144	272	107	N/A	162	138	250	178	200
5	184	312	147	164	N/A	98	210	138	160
6	184	312	147	164	122	N/A	210	138	160
7	184	312	147	164	122	98	N/A	138	160
8	184	312	147	164	122	98	210	N/A	160
9	184	312	147	164	122	98	210	139	N/A

After setting all the sets and parameters, we run the proposed MIP model and get the optimal berth time of 783 seconds for loading nine export containers with three AGVs and three YCs. The computation time is 1577 seconds.

Table 3.17: The values of the time an YC releases each container onto an AGV at the yard d_i, the time a QC picks up each container u_i and the time when each container has been loaded onto the ship $u_i + h_i$

Containers	d_i	u_i	$u_i + h_i$
1	236	337	417
2	312	413	490
3	463	568	696
4	522	696	783
5	0	61	135
6	0	50	183
7	0	183	358
8	138	358	490
9	467	517	697

Table 3.17 shows the results for the values of u_i and d_i : the time when a QC picks up each container from an AGV and the time when an YC releases each container onto an AGV. The times when each container has been placed onto the ship is represented by $u_i + h_i$, the largest number of which is the optimal berth time $min: max(u_i + h_i)$. Here, this number is 783 seconds, which means that after container 4 has been loaded onto the ship, the loading operation for the set of containers is completed.

From table 3.17, when the loading process starts at the yard-side at time 0, container 5, container 6 and container 7 are collected by three AGVs separately, following which QC 1 will pick up container 5 at the time of 61 seconds; QC 2 will pick up container 6 at the time of 50 seconds and container 7 at the time of 183 seconds.

Here, waiting times at the apron under QCs are considered, which means that AGVs may have to wait at the quayside for QCs to pick up the containers if QCs are serving other containers. Take the operations of handling container 1 as another example: container 1 will be processed by QC 1 at time 337 seconds after being picked up by an AGV from the storage yard at 236 seconds, and finally placed onto the ship at 417 seconds.The handling sequences of QCs can be seen by the values of u_i from table 3.17, so here we do not list the values of z_{ij}. Now we look at the handling sequences of AGVs and YCs.

Table 3.18: The results of handling sequences of containers by AGVs and YCs - the values of x_{imk} and y_{inw}

Sequence m	AGV 1	x_{imk}	AGV 2	x_{imk}	AGV 3	x_{imk}
1st	Container 7	$x_{711} = 1$	Container 5	$x_{512} = 1$	Container 6	$x_{613} = 1$
2nd	Container 1	$x_{121} = 1$	Container 8	$x_{822} = 1$	Container 2	$x_{223} = 1$
3rd	Container 9	$x_{931} = 1$	Container 3	$x_{332} = 1$	Container 4	$x_{433} = 1$
Sequence n	**YC 1**	y_{inw}	**YC 2**	y_{inw}	**YC 3**	y_{inw}
1st	Container 7	$y_{711} = 1$	Container 5	$y_{512} = 1$	Container 6	$y_{613} = 1$
2nd	Container 1	$y_{121} = 1$	Container 2	$y_{222} = 1$	Container 8	$y_{823} = 1$
3rd	Container 9	$y_{931} = 1$	Container 3	$y_{332} = 1$	Container 4	$y_{433} = 1$

Table 3.18 gives the results of handling sequences of containers by AGVs and YCs and it can be explained as follows: for example, AGV 1 delivers container 7 first, and then delivers container 1 and container 9; AGV 2 first moves container 5, followed by container 8 and container 3 successively. For the schedules of YCs, YC 2 handles container 5 first, then moves to transfer container 2 and container 3 separately in sequence.

Due to the computational complexity of the increasing number of constraints and number of variables, it is unlikely that exact optimal solutions for large-sized problems (hundreds of containers) to the proposed problem can be obtained in a reasonable time using the B&B algorithm embedded in AIMMS 3.11. Thus, in the following section, we propose a heuristic solution method based on genetic algorithm for solving the problem in realistic sizes within a reasonable computation time.

3.2.3 Solution method: genetic algorithm

Chromosome representation and initialisation: The chromosome representation demonstrates three main decision variables: the handling sequences of containers for AGVs, YCs and QCs. Figure 3.14 (a) and (b) shows a feasible solution, namely chromosome, of the problem by two matrices that imply the values of the decision variables, i.e. the handling sequences of all the handling equipment. There are six containers considered in this table. In figure 3.14 (a), for example, container 3 will be handled by YC 2 and delivered by AGV 2; as in assumption 3, the destinations of containers are known, so we assume that containers 1, 3 and 4 will be handled by QC 1 and containers 2, 5 and 6 will be handled by QC 2. Note that the actually order of QCs for handling these containers is one of the decision variables. The initial population is generated by the same method as in section 3.1.3.

Figure 3.14 (b) shows one possible solution for the QC handling sequences. For example, QC 1 will handle container 4 first, then container 3 and then container 1 in sequence.

Container	*AGV*	*YC*
1	2	2
2	3	1
3	2	2
4	1	3
5	1	3
6	2	1

(a)

QC 1	*QC 2*
4	2
3	6
1	5

(b)

Figure 3.14: (a) An illustration of chromosome representation for the AGV and YC handling sequences (b) An illustration of chromosome representation for the QC handling sequences

Parents selection strategy: Parents selection strategy decides how to choose chromosomes in the current population as parents for creating offspring for the next generation. Here we adopted the most common method is the 'roulette wheel' sampling to select parents for next generation.

Genetic operators design: A genetic operator helps to improve the solution gradually in the evolving process while maintaining the feasibility of the newly generated offspring for the problem. Here we introduce the following uniform crossover and swap mutation used in this study: (1) uniform crossover: This type of crossover operator generates a template binary string of uniformly distributed "1"s and "0"s in the same length of parents. The template string is then mapped to one of the parents, in which the genes that have the same positions with "1"s in the template string are given to a child, and the remaining empty genes of this child are filled from another parent. This

crossover can be directly used for the chromosome of AGV and YC sequences part, as illustrated in figure 3.15. In our preliminary tests, the evolving results obtained through two-point crossover (as in section 3.1.3) and uniform crossover are very close, thus we adopt uniform crossover for this problem in order to investigate another type of GA crossover operator. For the QC sequences part, we need to delete the duplicate genes from another parent and then filled the remaining genes as shown in figure 3.6 in section 3.1.3 (uniform order-based crossover); (2) swap mutation: According to the mutation rate P_m, swap mutation is carried out by choosing two positions of the chromosomes, and then swapping the genes on these positions, as illustrated in figure 3.7 in section 3.1.3.

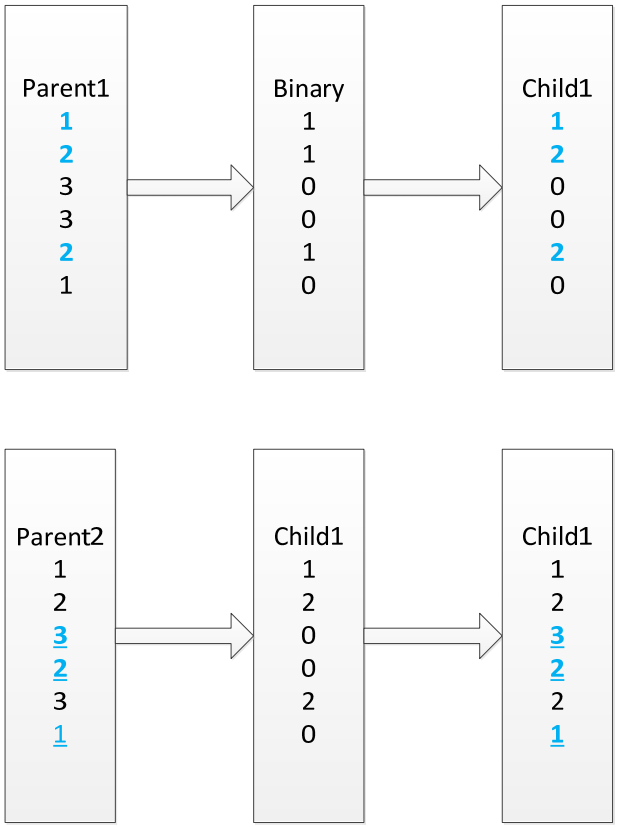


Figure 3.15: An illustration of uniform crossover for an example of six containers, three AGVs and three YCs

Offspring acceptance strategy: We use a semi-greedy strategy to accept the offspring created by the GA operators. In this strategy, an offspring is accepted as the new generation only if its OFV is less than the average of OFVs of its parent(s).

Stopping criterion: we use two criteria as stopping rules: (1) the maximum number of evolving generations M_g allowed for GA. This is a common criterion adopted by many GA-based optimisation problems (Kozan and Preston, 1999; Bazzazi, Safaei and Javadian, 2009; Huang, Liang and Yang, 2009); and (2) the standard deviation of the fitness values of chromosomes (σ_T) in the current generation T is below a small value (Tavakkoli-Moghaddam and Safaei, 2006).

3.2.4 Computational results

For small-sized problems, results obtained by B&B algorithm and GA are compared in terms of OFV and computation time. Due to the exponential increasing in the computation time by B&B algorithm as the problem size getting larger, the problem is infeasible to solve by B&B algorithm to get the exact solution. Therefore, GA is adopted for solving large-sized problems by providing approximately optimal solutions within reasonable time. GA is implemented in MATLAB 7.11. All the experiments are performed on a computer with Intel® Core™ i3 CPU M370@2.40GHz and 4GB RAM under the Windows 7 operating system. For each problem, we use the same setting of parameters in order to test the results. Each model is run 20 times by GA and the means of objective function values and computation times are listed.

Parameters settings

(1) The number of containers varies from 5 to 250, where 5-20 are considered as small-sized problems and 20-250 are considered as large-sized problems. We also consider that the number of AGVs varies from 2 to 15, the number of YCs varies from 2 to 8 and the number of QCs varies from 2 to 3.

(2) The uniform distribution was assumed for all the operation times. The processing times of each QC on these containers follows uniform distribution U(30, 180)s, and the handling times of each YC from each container's available location to the transfer point of the block in which this container is located follows uniform distribution U(60, 240)s.

(3) GA parameters take the following settings based on preliminary tests: Crossover rate P_c: 0.8; Mutation rate P_m: 0.01; Population size *Pop*: 100; Maximum generations M_g: 40.

Results for small-sized problems

Table 3.19: Results of computational experiments in small sizes

No	Containers	AGVs/QCs/ YCs	B&B (MIP)		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
1	5	2/2/2	2.82	407	1.97	407	0%
2	6	3/2/2	24.37	541	2.88	543	0.3%
3	7	3/2/2	172.02	604	3.06	612	1.3%
4	8	2/2/2	183.69	640	3.44	654	2.2%
5	9	3/2/3	1577.09	850	4.03	869	2.3%
6	10	2/2/2	16044.91	1867	3.98	1898	1.7%
7	10	3/2/2	/	/	3.25	1715	/
8	15	3/3/3	/	/	4.11	2108	/
9	15	2/3/3	/	/	4.32	2342	/
10	20	2/3/2	/	/	4.61	2985	/

Table 3.19 shows the results of B&B algorithm using AIMMS software and GA for small-sized problems. Generally, our proposed GA can obtain approximate optimal solutions in a faster computation speed. The computation time of GA for small-sized problems ranges from 1.97 seconds to 4.61 seconds. However, B&B embedded in AIMMS requires more computation time to run. It can also be observed that there is not much difference between these two evaluation algorithms in terms of OFVs.

Compared with AIMMS results, the average OFV difference of GA is only 1.5%, which is a promising result. This small gap is acceptable by container terminal operators due to the shorter computation time (planning time). As expected, when increasing the number of containers, i.e. problem size getting larger, the OFV increases accordingly and the computation time of AIMMS increases exponentially. We also find that AIMMS could not return results for this NP-hard problem with more than 10 containers. The overall performance of GA is considered satisfactory from the practical point of view.

Results of large-sized problems

Due to the complexity of the proposed problem, it is difficult to obtain exact solutions for the problem in large sizes by B&B algorithm using AIMMS 3.11: thus we adopt the proposed GA, which is able to obtain near-optimal solutions in the reasonable computation times for such practical-sized problems. We evaluate a series of large-sized problems by the proposed GA. The number of containers varies from 30 to 250 with different numbers of AGVs, QCs and YCs. As containers are handled by a series of equipment-types, it is therefore necessary to examine the effects of the number of different types of handling equipment on the objective function value. Figure 3.16 shows the typical convergence of the GA approach for a case of 50 containers, five AGVs, three QCs and four YCs. It has a monotonous convergence towards a fixed value. We can find that the OFV for this case reached convergence in 30 generations where the OFV is the best (smallest). Table 3.20 shows the evaluation results of the GA approach in large-sized problems. In most of the experiments, GA is able to get solutions within one minute. Even for the case with 250 containers, fifteen AGVs, three QCs and eight YCs, it only takes about two minutes to solve. This indicates that the proposed GA is reliable for different-size experiments. The OFVs improve at a faster

pace when more cranes are assigned (such as case 14 and case 15), as compared with the effects of AGVs (such as case 11 and case 12); the OFVs increase at a relative, constant speed when more containers are to be loaded. As discussed similarly, when the number of vehicles or cranes increases, traffic congestion and crane conflicts issues will be considered, which cause the inefficiency of terminal operation.

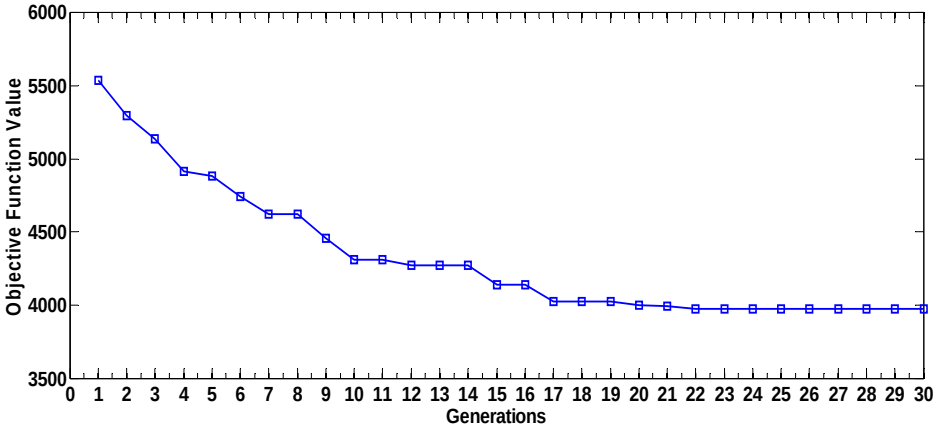


Figure 3.16: The typical convergence of GA in a single run

Table 3.20: Results of computational experiments in large sizes

No	Containers	AGVs/QCs/ YCs	Computation time (s)	OFV (s)
11	30	3/3/2	5.95	4773
12	30	4/3/2	7.72	4377
13	30	5/3/2	12.17	4444
14	40	4/3/2	8.28	5327
15	40	4/3/3	10.19	4515
16	40	4/3/4	15.14	3573
17	50	4/2/4	14.71	4089

18	50	5/2/4	7.91	4136
19	50	5/2/5	19.38	3423
20	60	6/3/5	14.70	3738
21	70	6/3/5	23.47	4379
22	80	6/3/5	26.35	4518
23	90	6/3/5	21.51	4863
24	100	6/3/6	27.33	6352
25	100	8/3/6	41.06	6099
26	100	10/3/6	45.05	6128
27	150	10/3/5	74.92	9794
28	150	10/3/6	54.05	8822
29	150	10/3/7	48.17	8338
30	200	10/3/6	89.81	11768
31	200	15/3/6	66.23	10196
32	200	10/3/8	60.73	9716
33	250	10/3/4	134.06	21402
34	250	10/3/6	107.56	15508
35	250	10/3/8	109.05	12284
36	250	15/3/8	131.31	11553

GA parameters sweep experiments

For the problem of 60 containers, five AGVs, three QCs and four YCs, GA parameter settings take the following values:

Crossover rate P_c = {0.6, 0.7, 0.8, 0.9};

Mutation rate P_m= {0.01, 0.02, 0.1, 0.2};

Population size Pop = {30, 50, 100, 150};

$$M_g = 40.$$

Figures 3.17 to 3.19 show the performance comparisons among different parameter settings. According to these convergence curves, we observe that the one of crossover rate=0.7, the one of mutation rate=0.01, and the one of population size=100 outperform others for this particular problem. Specifically, figure 3.17 shows the convergence curves on different crossover rates for a typical instance, and the curve with $P_c = 0.7$ converges to a smaller value; figure 3.18 shows the convergence curves on different mutation rates, and the curve with $P_m = 0.01$ has a smaller reached convergence; and figure 3.19 shows the convergence curves on different population sizes, and the solution of curve with $Pop = 100$ is better than the others. These figures also show that for all the experiments, OFVs are not improved after 40 generations due to the fast convergence speed of our proposed GA. So the maximum generations of 40 will be sufficient to obtain near-optimal solutions.

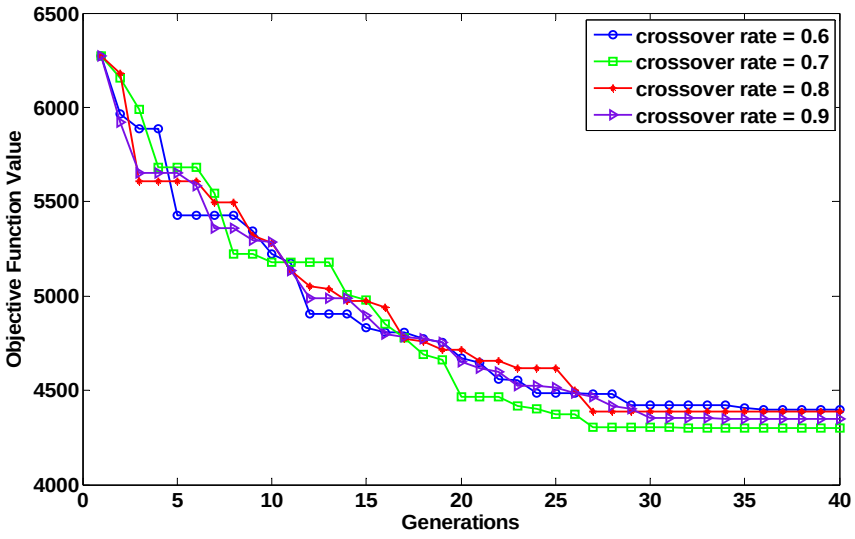


Figure 3.17: Performance comparison of different crossover rates for an example under $Pop = 100$ and $P_m = 0.02$

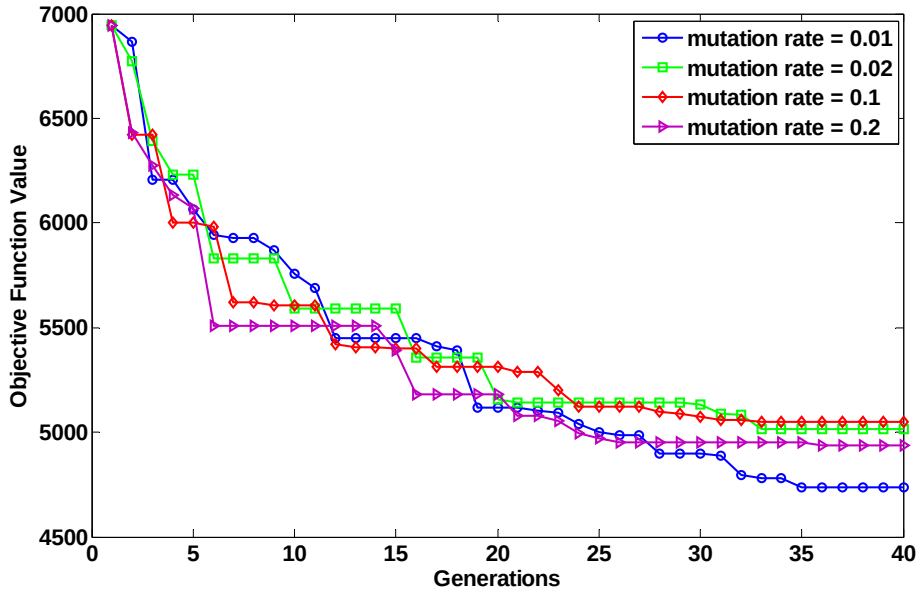


Figure 3.18: Performance comparison of different mutation rates for an example under $Pop = 100$ and $P_c = 0.7$

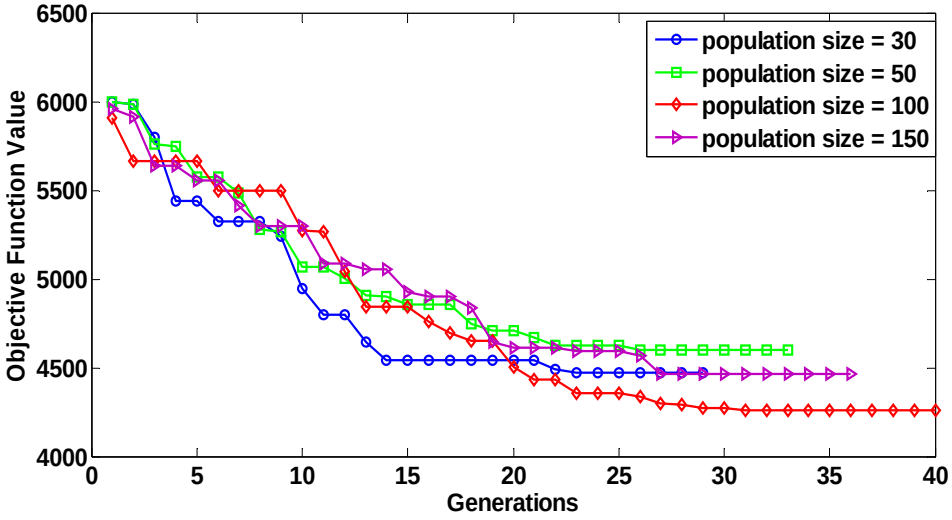


Figure 3.19: Performance comparison of different population sizes for an example under $P_c = 0.7$ and $P_m = 0.01$

Lastly, we apply our proposed GA to the problem with 50 containers, five AGVs, three QCs and five YCs with the same GA parameter settings, and the algorithm is run 10 times. The results are illustrated following the pattern shown by the box plot in figure 3.20. Each box represents the OFVs of the 10 runs in one generation. The central mark is the median of OFVs, the edges of the box are the 25th and 75th percentiles and the whiskers are the most extreme data points. We can find that our proposed GA performs a stable manner in all the experiments and the OFVs of the best solutions at 40 generations are close to each other.

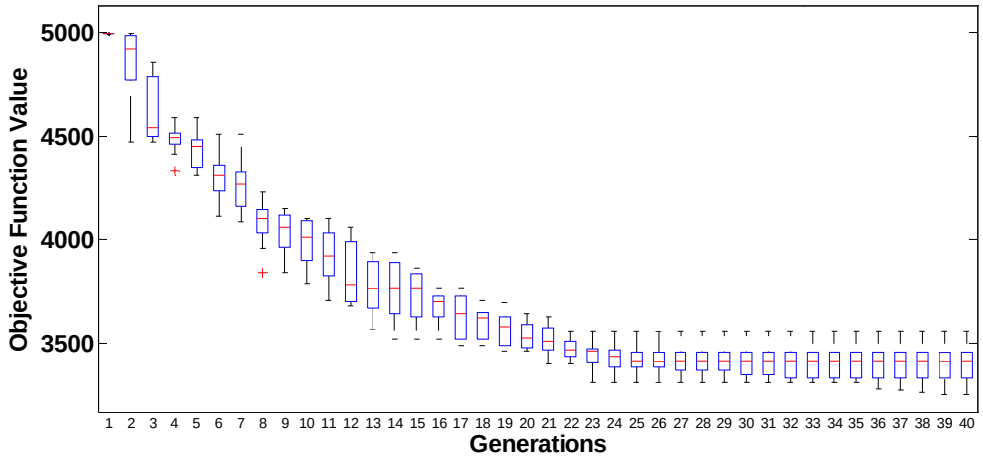


Figure 3.20: GA performance in 10 runs with $P_c = 0.7$, $P_m = 0.01$ and $Pop = 100$ for the case with 50 containers, five AGVs, three QCs and five YCs

3.3 Model 3: The integrated AGV scheduling and storage allocation problem in the dual-cycle mode

Most container terminals adopt the single-cycle method in which loading operations start after all the unloading operations have been finished. This method generates more empty-loaded movements of QCs, AGVs and YCs as discussed in model 1 and model 2. However, in some advanced container terminals, dual-cycle techniques have been used for reducing berth time of ships. It has been proven that this strategy can decrease the empty-loaded movements and thus increase the berth productivity (Zhang and Kim, 2009). Therefore, in this section, we study the problem which integrates the AGV scheduling problem and storage allocation problem in the dual-cycle mode. In our model, dual-cycle is defined as simultaneously considering unloading and loading operations of containers. Our aim is to find the optimal schedule of AGVs for handling a set of import and export containers and also to find the optimal locations for import containers. The model is formulated as a mixed-integer programming (MIP) model with the objective to minimise the ship's berth time, the same as with models 1 and 2. We also use two different approaches, i.e. B&B and GA for solving the problem in small and large sizes.

3.3.1 Problem description and formulation

It is an accepted convention that the most important objective of day-to-day container terminal operations is the minimisation of a ship's berth time. In this section, as stated in section 3.1, we consider a terminal using QCs, YCs and AGVs for handling containers. During the loading process, a container picked up by an YC is put onto an AGV that delivers the container to the quayside. At the quayside, a QC picks up the container from the AGV and locates it in its designated location on the ship. The unloading process performs in the reverse order of the loading process. Here the dual-cycle operation means that unloading and loading operations are considered simultaneously; it is not necessary that there are both unloading and loading operations taking place in the same cycle (figure 2.8 (b)).

In practice, there are a number of ships berthed along the quayside at any one time, and each ship is served by a set of QCs. A few hours before the arrival of the ship, the terminal receives detailed information about the containers to be discharged from and loaded onto this ship. This information allows the terminal operators to generate the QC schedule which specifies the handling sequence of containers by each QC and the estimated serving time of each container. Thus, at any time point in the operations, the QC operator has the information on which container to work on next, i.e. the handling sequence of each QC is known. The time required to handle each container is assumed to be deterministic (Bish *et al.*, 2001), and because each container has a specific location on the ship, we assume that the QC's handling times are different among containers (include both import and export containers) according to their ship locations. Because import containers have already been placed on the ship, their ship locations are known. For export containers considered in this problem, we assume their ship locations have been allocated before loading operation starts, which is a common assumption. Table 3.21 shows an example of the QC's sequence list and handling times. The first task for QC 1 is a loading container (L), i.e. export container, which will take 153 seconds of handling time by QC 1 to locate it to its ship location - bay 05 row 04 tier 04; the second task of QC 1 is also a loading operation, and it takes 125 seconds of handling time to load the container to the location of bay 04 row 06 tier 02 on the ship.

Table 3.21: An example of a QC's sequence list (L-loading; D-discharging)

Task list	Type	Ship location Bay-Row-Tier	Handle time (sec)
QC 1	L	05-04-04	153
	L	04-06-02	125
	D	07-03-03	190
QC 2	L	04-05-03	112
	D	06-02-02	155
	L	09-05-02	191

The location of a container in the storage block determines the time that the YC needs to handle it. Hence, container storage locations are very important to decide the YC schedule and thus have a great impact on the terminal's overall performance due to the high operational cost of YCs. We assume there are some potential slots for accommodating the incoming containers (import containers), and the number of these slots is always greater than the number of import containers. This indicates that the storage yard always has the ability to locate the import containers. For export containers, when the loading operations begin, all the export containers must have been located in the yard; therefore the locations of export containers are usually known. Moreover, the YC's travelling time between any two locations (slots) in the yard are known, which can be easily calculated according to the location of containers and the yard layout; for example, the YC's travelling times between blocks are determined by the layout (distance) of the blocks.

After describing the QC and YC operations, we now look at the AGV transportation phase. In this study, all the handling equipment - QCs, YCs and AGVs - can carry only one container at a time. We assume every AGV can serve any QC, i.e. pooling strategy

is applied for AGV dispatching. The traffic congestion is not considered here because the study of traffic control requires a more sophisticated engineering mechanism and thus is beyond the scope of our study. In the dual-cycle operations, there are four transportation conditions for any two consecutive containers delivered by the same AGV:

Condition 1: the AGV transfers an import container after another import container.

Condition 2: the AGV delivers an import container after an export container.

Condition 3: the AGV transports an export container after another export container.

Condition 4: the AGV moves an export container after an import container.

Generally, import containers and export containers are not mixed within the same block. Specifically, export containers are usually placed near the quayside, where they can easily be loaded onto the ship. Import containers are usually stored near the gatehouse, close to rail/road exchange area, where they can be quickly picked up by trucks/trains. Therefore, here we assume each YC can work for only one type of containers, i.e. either export containers or import containers, to reduce the unproductive moves between the blocks for import containers and the blocks for export containers. In this case, for YC's movements, there are only two conditions that need to be considered for any two consecutive containers handled by the same YC, i.e. a YC moves an import container after another import container and a YC moves an export container after another export container.

Summarising the descriptions above, this study considers a novel integrated optimisation problem in the dual-cycle mode in which, given the QC's sequence lists and handling times for each container, AGV's schedules and the assignment of yard locations to import containers then can be determined. In the meanwhile, YC's schedules can also be established. The following assumptions are made throughout this study:

(1) The container handling sequences of the QCs are known, which means that containers must be handled by the order exactly appeared in the QC's handling

sequence list. Container locations on the ship are known and thus the handling time of each container by the QCs is known.

(2) Number of containers, number of AGVs, QCs and YCs are all known.

(3) AGVs, QCs and YCs can only carry one container at a time.

(4) Travelling times of AGVs/YCs between any two processing locations are known. For example, the AGV's travelling time between each block to each QC, and the YC's travelling time between any two yard locations are known.

(5) Traffic congestion of AGVs on the path is not considered here.

(6) The storage yard has the ability to accommodate all the incoming containers.

(7) The time needs for picking up/dropping off containers from/onto AGVs by QCs and YCs are negligible.

(8) Interferences among QCs and YCs are not considered.

(9) Import and export containers are not mixed within one block and each YC can travel among blocks either for import containers or export containers.

Now we compare model 3 with model 1 and model 2, and identify some similarities and differences. First, model 3 considers both unloading and loading processes simultaneously, while model 1 and model 2 only deal with either unloading or loading process; second, model 3 combines the idea of integrating vehicle scheduling and storage allocation problem, which have been discussed in model 1 and model 2. Third, although the problems considered in model 1 and model 2 are covered in model 3, it is still necessary to develop these models, because most container terminals are adopting the single-cycling mode, in which unloading and loading processes are not taking place at the same time.

Now we introduce the notations related to the integrated problem. Each unloading/loading container action is referred to as a job.

Sets and parameters

D set of import containers (jobs)

L set of export containers (jobs)

N	Set of all the containers i.e. $N = D \cup L$
P	set of yard locations
B	set of blocks
K	Set of QCs
V	Set of AGVs
C	Set of YCs
k, l	index for QCs
b, a	index for blocks
$(i,k),(j,l)$	index for containers (jobs), job *(i, k)* means the *i*th container handled by QC *k*
N_k	the total number of containers handled by QC *k*
v	Total number of AGVs
c	Total number of YCs
(n,b)	index for yard locations, $n \in N^+$ positive integer, location *(n, b)* is slot *n* in block *b*
$h_{(i,k)}$	QC's handling time of container (i,k)
$w_{(i,k)}$	YC's handling time for export container (i,k) from the location of container (i,k) to the transfer point in front of the block that container (i,k) is placed. This is known because yard locations of export containers are known.
$\tau_{(i,k)}$	AGV's travelling time for each export container (i,k) from its located block to its assigned QC *k*

t_k^b AGV's travelling time between QC *k* and block *b*

$\pi_{(k,l)}$ AGV's travelling time from QC *k* to QC *l*

$v_{(a,b)}$ YC's travelling time between block *a* and block *b*

$v1_{(i,k)}^{(j,l)}$ YC's travelling time from the transfer point of the block which stores export container (i,k) to the yard location of export container (j,l)

$\varphi_{(n,b)}$ YC's travelling time from the transfer point in front of block *b* to the location *(n, b)*

$\rho_{(i,k)}^b$ AGV's travelling time from the transfer point of the block that an export container *(i, k)* locates to transfer point of block *b*

M a very large number

(S,I) dummy starting job

(F,I) dummy ending job

O_S The job set which contains all the jobs including the dummy starting job,

$$O_S = N \cup (S,I)$$

O_F The job set which contains all the jobs including the dummy ending job,

$$O_F = N \cup (F,I)$$

O The job set which contains all the jobs including dummy starting and ending jobs, $O = \{(S,I),(F,I)\} \cup N$

Decision variables

$u_{(i,k)}$ the time QC *k* starts handling container (*i, k*): for import container, it represents the time a QC picks it up from the ship; for export container, it means the time a QC picks it up from an AGV

$d_{(i,k)}$ the time a YC starts handling container (i, k): for import container, it represents the time a YC picks it up from an AGV at the transfer point in front of the block; for export container, it means the time a YC picks up this container from its yard location

Note that each QC's handling sequence is known. Now we introduce another four decision variables for AGV delivery sequences, import container yard location assignment and YC handling sequences:

$$x_{(i,k)}^{(j,l)} =$$

$$\begin{cases} 1, \text{if an AGV, which is scheduled to deliver container } (j,l), \text{has just delivered container } (i,k) \\ 0, \text{otherwise}, \forall (\mathrm{i,k}) \in O_S, \forall (j,l) \in O_F \end{cases}$$

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

We introduce the following intermediate variable $y_{(i,k)}^{b}$, where $\sum_{n \in N^+} z_{(i,k)}^{(n,b)} = y_{(i,k)}^{b}, \forall (i,k) \in D, \forall b \in B$

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

$$\sigma_{(i,k)}^{(j,l)} =$$

$$\begin{cases} 1, \text{if an YC, which is scheduled to handle container } (j,l), \text{ has just handled container } (i,k) \\ 0, \text{otherwise}, \forall (\mathrm{i,k}) \in \mathrm{D} \cup (\mathrm{S,I}), \forall (j,l) \in D \cup (F,I) \ \ or \ \ \forall (\mathrm{i,k}) \in \mathrm{L} \cup (\mathrm{S,I}), \forall (j,l) \in L \cup (F,I) \end{cases}$$

Objective: Minimise the berth time of the ship

$$\text{Min: Max}(u_{(N_k,k)} + h_{(N_k,k)})$$

Contraints:

$$\sum_{(j,l)\in O_F} x_{(i,k)}^{(j,l)} = 1, \forall (i,k) \in N \tag{3.3.1}$$

$$\sum_{(i,k)\in O_S} x_{(i,k)}^{(j,l)} = 1, \forall (j,l) \in N \tag{3.3.2}$$

$$\sum_{(j,l)\in N} x_{(S,I)}^{(j,l)} = v \tag{3.3.3}$$

$$\sum_{(i,k)\in N} x_{(i,k)}^{(F,I)} = v \tag{3.3.4}$$

$$\sum_{b\in B} y_{(i,k)}^{b} = 1, \forall (i,k) \in D \tag{3.3.5}$$

$$\sum_{(n,b)\in P} z_{(i,k)}^{(n,b)} = 1, \forall (i,k) \in D \tag{3.3.6}$$

$$\sum_{(i,k)\in D} z_{(i,k)}^{(n,b)} \leq 1, \forall (n,b) \in P \tag{3.3.7}$$

$$\sum_{n\in N^+} z_{(i,k)}^{(n,b)} = y_{(i,k)}^{b}, \forall (i,k) \in D, \forall b \in B \tag{3.3.8}$$

$$\sum_{(j,l)\in D\cup(F,I) or\ L\cup(F,I)} \sigma_{(i,k)}^{(j,l)} = 1, \forall (i,k) \in D\ or\ L \tag{3.3.9}$$

$$\sum_{(i,k)\in D\cup(S,I) or L\cup(S,I)} \sigma_{(i,k)}^{(j,l)} = 1, \forall (j,l) \in D\ or\ L \tag{3.3.10}$$

$$\sum_{(j,l)\in D or L} \sigma_{(S,I)}^{(j,l)} = c \tag{3.3.11}$$

$$\sum_{(i,k)\in DorL} \sigma_{(i,k)}^{(F,I)} = c \tag{3.3.12}$$

$$u_{(i,k)} + h_{(i,k)} + \sum_{b\in B} t_k^b y_{(i,k)}^b \le d_{(i,k)}, \forall (i,k) \in D \tag{3.3.13}$$

$$d_{(i,k)} + w_{(i,k)} + \tau_{(i,k)} \le u_{(i,k)}, \forall (i,k) \in L \tag{3.3.14}$$

$$d_{(i,k)} + \sum_{(n,b)\in P} \varphi_{(n,b)} * z_{(i,k)}^{(n,b)} + \sum_{a,b} v_{(a,b)} * y_{(i,k)}^b * y_{(j,l)}^a \le d_{(j,l)} + M * \left(1 - \sigma_{(i,k)}^{(j,l)}\right), \forall (i,k),(j,l) \in D \tag{3.3.15}$$

$$d_{(i,k)} + w_{(i,k)} + v1_{(i,k)}^{(j,l)} \le d_{(j,l)} + M * \left(1 - \sigma_{(i.k)}^{(j,l)}\right), \forall (i,k),(j,l) \in L \tag{3.3.16}$$

$$u_{(i+1,k)} - u_{(i,k)} \ge h_{(i,k)}, \forall (i+1,k),(i,k) \in N, i = 1,2,\dots,N_k - 1 \tag{3.3.17}$$

$$d_{(i,k)} + \sum_{b\in B} t_l^b y_{(i,k)}^b \le u_{(j,l)} + h_{(j,l)} + M * \left(1 - x_{(i,k)}^{(j,l)}\right), \forall (i,k),(j,l) \in D \tag{3.3.18}$$

$$u_{(i,k)} + \pi_{(k,l)} + \tau_{(j,l)} \le d_{(j,l)} + w_{(j,l)} + M * \left(1 - x_{(i,k)}^{(j,l)}\right), \forall (i,k),(j,l) \in L \tag{3.3.19}$$

$$u_{(i,k)} + \pi_{(k,l)} \le u_{(j,l)} + h_{(j,l)} + M * \left(1 - x_{(i,k)}^{(j,l)}\right), \forall (i,k) \in L,(j,l) \in D \tag{3.3.20}$$

$$d_{(i,k)} + \sum_{b\in B} \rho_{(j,l)}^b y_{(i,k)}^b \le d_{(j,l)} + w_{(j,l)} + M * \left(1 - x_{(i,k)}^{(j,l)}\right), \forall (i,k) \in D,(j,l) \in L \tag{3.3.21}$$

$$x_{(i,k)}^{(j,l)}, y_{(i,k)}^b, z_{(i,k)}^{(n,b)}, \sigma_{(i,k)}^{(j,l)} \in \{0,1\}, \forall (i,k),(j,l) \in O, \forall (n,b) \in P, \forall b \in B \tag{3.3.22}$$

$$u_{(i,k)}, d_{(i,k)} \ge 0, \forall (i,k) \in N, i = 1,2,\dots,N_k, \forall k \in K \tag{3.3.23}$$

The objective is to minimise the ship's berth time for unloading and loading a set of containers. Constraint (3.3.1) implies that for every container $(i,k) \in N$, there is one container $(j,l) \in O_F$ delivered after it by the same AGV.

Constraint (3.3.2) represents that for every container $(j, l) \in N$, there is one container $(i, k) \in O_S$ delivered before it by the same AGV.

Constraints (3.3.3) and (3.3.4) guarantee that the number of AGVs employed for delivering these containers is exactly *v*.

Constraint (3.3.5) guarantees that each import container $(i, k) \in D$ will be assigned to one block *b* after the unloading process.

Constraint (3.3.6) ensures that each import container $(i, k) \in D$ will be located into one of the available slots in the yard.

Constraint (3.3.7) represents that each available slot *(n, b)* in the yard can locate, at most, one container.

Constraint (3.3.8) means that if a container is assigned to a block *b*, it can only be assigned to one available slot *(n, b)* within that block *b*. This constraint gives the relationship between two decision variables $y^b_{(i,k)}$ and $z^{(n,b)}_{(i,k)}$.

Constraint (3.3.9) implies that for every container $(i, k) \in D \; or \; L$, there is one container $(j, l) \in D \cup (F, I) \; or \; L \cup (F, I)$ handled after it by the same YC.

Constraint (3.3.10) represents that for every container $(j, l) \in D \; or \; L$, there is one container $(i, k) \in D \cup (S, I) \; or \; L \cup (S, I)$ handled before it by the same YC.

Constraints (3.3.11) and (3.3.12) guarantee that the number of YCs employed to handle containers in the storage yard is exactly *c*.

Constraint (3.3.13) means that for each import container (i, k), the time between an YC picking it up from an AGV and a QC picking it up from the ship must be at least a certain QC's handling time $h_{(i,k)}$ plus the AGV's travelling time between QC *k* and the assigned block *b* for container (i, k).

Constraint (3.3.14) means that for each export container (i, k), the time between a QC picking it up from an AGV and an YC picking it up from its yard location must be at

least a certain YC's handling time $w_{(i,k)}$ plus AGV's travelling time $\tau_{(i,k)}$ between the yard and QC.

Constraints (3.3.15) and (3.3.16) are the time constraints for two successive containers handled by the same YC in the two handling conditions - import-import and export-export, because YCs can only handle one type of container, either import or export container.

Specifically, constraint (3.3.15) means that if import container (j,l) is handled immediately after import container (i,k) by the same YC, then the time between this YC picking up container (j,l) and picking up container (i,k) must be set at least by the handling time to move container (i,k) to its assigned location (n,b) plus the YC's travelling time between the assigned blocks of the two containers.

Constraint (3.3.16) means that if export container (j,l) is handled immediately after export container (i,k) by the same YC, then the time between this YC picking up container (j,l) and picking up container (i,k) must be set at least by a certain YC's handling time and travelling time.

Constraint (3.3.17) states that all the containers are handled according to the QC schedule and the time interval between handling any two successive containers (i,k) and $(i+1,k)$ must be at least the time this QC takes to handle container (i,k).

Constraints (3.3.18)-(3.3.31) are the time constraints for two successive containers delivered by the same AGV in the four transportation conditions**:** import-import, export-export, export-import and import-export.

Specifically, constraint (3.3.18) implies that if both container (i,k) and container (j,l) are import containers, then the time between picking up container (j,l) from the QC l and transferring container (i,k) to an YC must be at least the AGV's travelling time from the assigned block b of container (i,k) to QC l.

Constraint (3.3.19) means that if both container (i,k) and container (j,l) are export containers, then the time between delivering container (i,k) to QC k and picking up container $(j,l)$from an YC must be at least the AGV's travelling time between QC k

and QC l plus the AGV's travelling time from quayside to yard-side for picking up container (j, l).

Constraint (3.3.20) represents that if container (i, k) is an export container and container (j, l) is an import container, then the time between picking up container (j, l) from QC l and transferring container (i, k) to QC k must be at least the AGV's travelling time between QC k and QC l.

Constraint (3.3.21) indicates that if container (i, k) is an import container and container (j, l) is an export container, then the time between picking up container (j, l) from the yard and transferring container (i, k) to an YC must be at least the AGV's travelling time between the blocks which locate container (i, k) and container (j, l).

Constraints (3.3.22) and (3.3.23) are binary and non-negative constraints.

3.3.2 An illustrative example

In this section, computational experiments are designed to evaluate the performance of the proposed model. An example is presented to illustrate the MIP model and show how the optimal solutions characterise the coordination of different types of container-handling equipment. We use the B&B algorithm and implement it by AIMMS 3.11 with CPLEX 12.3.

Consider the loading and unloading processes simultaneously at the container terminal: QCs are serving for a ship to unload and load containers. There are four blocks at the storage yard, where blocks 1 and 2 are used for storing import containers and blocks 3 and 4 are used for storing export containers. Two YCs are travelling in the yard and will handle either import containers or export containers. There are three AGVs travelling between the ship and yard to move containers. We present an example of 10 containers with five import containers and five export containers. The values of parameters at the terminal considered in our problem are listed in the following tables.

First, we present table 3.22 for the handling times of QCs for all the containers, which are drawn from uniform distribution U(30, 180)s. The handling time for each container is known since each container has a specific location on the ship. Let containers (1, 1),

(3, 1), (5, 1), (2, 2) and (4, 2) be import containers and (2, 1), (4, 1), (1, 2), (3, 2) and (5, 2) be export containers. For example, QC 1 first takes 156 seconds moving import container (1, 1) from the ship to an AGV, and it takes 164 seconds to move export container (2, 1) from the AGV to the ship.

Table 3.22: The handling time $h_{(i,k)}$ of each container by QCs

Containers (QC 1)	$h_{(i,k)}$ (sec)	Containers (QC 2)	$h_{(i,k)}$ (sec)
(1, 1)	156	**(1, 2)**	65
(2, 1)	164	**(2, 2)**	71
(3, 1)	149	**(3, 2)**	124
(4, 1)	95	**(4, 2)**	45
(5, 1)	83	**(5, 2)**	154

Now we list the AGV's travel times between each QC and each block, i.e. the values of t_k^b in table 3.23. For example, the AGV's travel time from block 1 to QC 2 is 82 seconds, and the AGV's travel time from block 3 to QC 1 is 84 seconds, and so on. Because AGVs are identical and travel at constant speed, the travel times listed in this table are the same for all the AGVs.

Table 3.23: The travelling time of AGVs between QCs and blocks: the values of t_k^b

Travel time (sec)	QC 1	QC 2
Block 1	68	82
Block 2	62	20
Block 3	84	40
Block 4	98	56

Table 3.24: The travel time of YCs to each available slot within blocks for import containers: the values of $\varphi_{(n,b)}$

Block 1	$\varphi_{(n,b)}$ (sec)	Block 2	$\varphi_{(n,b)}$ (sec)
(1, 1)	131	**(1, 2)**	121
(2, 1)	136	**(2, 2)**	50
(3, 1)	109	**(3, 2)**	121
(4, 1)	90	**(4, 2)**	81

Note that each YC can handle either import containers or export containers. YC's travelling time between the two adjacent blocks is 40 seconds, similar as in table 3.3. YC's travelling time $\varphi_{(n,b)}$ from the transfer point of block *b* to each available slot *(n, b)* is presented in table 3.24. Because these available slots are known, the time YC takes from the transfer point to each slot is known. The values are generated from a uniform distribution U(40, 160)s, assuming there are eight available slots to accommodate the five import containers in blocks 1 and 2, each of which has four slots available. For example, within block 1, location (2, 1) means location 2 in block 1 and it takes 136 seconds for a YC to move a container from the transfer point in block 1 to location (2, 1); in block 2, a YC spends 121 seconds moving a container from the transfer point of block 2 to location (1, 2). All the other values can be explained based on the above.

Let export containers (2, 1), (4, 1) and (1, 2) be in block 3 and export containers (3, 2) and (5, 2) be in block 4. Then the AGV travel time from the transfer point of container *(i, k)* to each block b, i.e. the values of $\rho^{b}_{(i,k)}$ (as shown in table 3.25) can be calculated as follows: for example, $\rho^{1}_{(2,1)}$ is the AGV travel time from block 3 where container (2, 1) locates, to block 1, which is 80 seconds.

Table 3.25: The values of $\rho^{b}_{(i,k)}$-AGV travel time from the block that *(i, k)* locates to other blocks *b*

	(2, 1)	(4, 1)	(1, 2)	(3, 2)	(5, 2)
$\boldsymbol{\rho^{1}_{(i,k)}}$	80	80	80	120	120
$\boldsymbol{\rho^{2}_{(i,k)}}$	40	40	40	80	80
$\boldsymbol{\rho^{3}_{(i,k)}}$	0	0	0	40	40
$\boldsymbol{\rho^{4}_{(i,k)}}$	40	40	40	0	0

Lastly, we list the YC's travel time for each export container from its location to the transfer point in front of its block $w_{(i,k)}$, and the AGV travel time for each export container from its located blocks to its assigned QC $\tau_{(i,k)}$ in table 3.26. For example, for export container (2, 1), YC takes 101 seconds to move it from its yard location to the transfer point of its block, and AGV takes 84 seconds to deliver container (2, 1) from block 3 to QC 1.

Table 3.26: The values of $w_{(i,k)}$ and $\tau_{(i,k)}$

Containers (i, k)	$w_{(i,k)}$	$\tau_{(i,k)}$
(2, 1)	101	84
(4, 1)	76	84
(1, 2)	46	40
(3, 2)	75	56
(5, 2)	86	56

By solving this example, the optimal berth time for loading five export containers and unloading five import containers is 966 seconds. AIMMS took 79.54 seconds of computation time to arrive at this solution. Now we look at the time schedule of each QC for handling each container and the time when AGVs pick up/ release each container at the storage yard. Table 3.27 consists of two parts: one for the time schedule of import containers and the other for the time schedule of export containers. For an import container, u_i is the time QC picks up the container from the ship, $u_i + h_i$ is the time this container is ready to be collected by an AGV, and d_i is the time this container has been picked up by an YC in the yard. For an export container, d_i is the time an YC starts to handle this container, u_i is the time QC collects this container from an AGV and $u_i + h_i$ is the time this container has been placed onto the ship. Therefore, the largest number of $u_i + h_i$ for both import and export containers is the value of our objective function, which is 966 seconds achieved by container (5, 2), which means that after container (5, 2) has been loaded onto the ship, the operations of handling the set of containers are finished.

Table 3.27: The time schedules of QCs and the times when AGVs pickup/drop-off containers at the yard-side

Import containers				Export containers			
	u_i	$u_i + h_i$	d_i		d_i	u_i	$u_i + h_i$
(1, 1)	0	156	218	(1, 2)	0	86	151
(2, 2)	151	192	380	(2, 1)	147	231	315
(3, 1)	315	464	638	(3, 2)	363	494	618
(4, 2)	618	663	920	(4, 1)	554	638	733
(5, 1)	733	816	1182	(5, 2)	756	812	966

Table 3.28: The AGV delivery sequences of containers-the values of $x_{(i,k)}^{(j,l)}$

AGVs	Containers	$x_{(i,k)}^{(j,l)}$
AGV 1	(1, 2)	$\boldsymbol{x_{(1,2)}^{(2,2)}} = 1$
	(2, 2)	$\boldsymbol{x_{(2,2)}^{(3,2)}} = 1$
	(3, 2)	$\boldsymbol{x_{(3,2)}^{(3,1)}} = 1$
	(3, 1)	$\boldsymbol{x_{(3,1)}^{(5,1)}} = 1$
	(5, 1)	
AGV 2	(2, 1)	$\boldsymbol{x_{(2,1)}^{(5,2)}} = 1$
	(5, 2)	
AGV 3	(1, 1)	$\boldsymbol{x_{(1,1)}^{(4,1)}} = 1$
	(4, 1)	$\boldsymbol{x_{(4,1)}^{(4,2)}} = 1$
	(4, 2)	

Next, we have the delivery sequences of containers by each AGV as shown in table 3.28. For example, AGV 2 first collects export container (2, 1) from the yard and delivers it to QC 1; after that, AGV 2 returns to the yard for picking up another export container (5, 2) and then delivers it to QC 2.

Now we look at the handling sequences by each YC as given in table 3.29. In this example, YC 1 is assigned to handle import containers and YC 2 is assigned to handle export containers. For example, in the solution, we have $\boldsymbol{\sigma_{(2,2)}^{(3,1)}} = 1$ which means that YC 1 is assigned to handle container (3, 1) after it finishes its delivery for container (2, 2).

Table 3.29: The handle sequences of containers by each YC-the values of $\sigma_{(i,k)}^{(j,l)}$

YCs	Containers	$\sigma_{(i,k)}^{(j,l)}$
YC 1	(1, 1)	$\boldsymbol{\sigma_{(1,1)}^{(2,2)}} = 1$
	(2, 2)	$\sigma_{(2,2)}^{(3,1)} = 1$
	(3, 1)	$\boldsymbol{\sigma_{(3,1)}^{(4,2)}} = 1$
	(4, 2)	$\boldsymbol{\sigma_{(4,2)}^{(5,1)}} = 1$
	(5, 1)	
YC 2	(1, 2)	$\boldsymbol{\sigma_{(1,2)}^{(2,1)}} = 1$
	(2, 1)	$\boldsymbol{\sigma_{(2,1)}^{(3,2)}} = 1$
	(3, 2)	$\boldsymbol{\sigma_{(3.2)}^{(4,1)}} = 1$
	(4, 1)	$\boldsymbol{\sigma_{(4,1)}^{(5,2)}} = 1$
	(5, 2)	

Lastly, let us look at the assigned location for each import container at the storage yard as in table 3.30, where import containers (2, 2), (4, 2) and (5, 1) are stored in block 1 and import containers (1, 1) and (3, 1) are stored in block 2. Moreover, container (1, 1) will be stored in location 4 of block 2; container (2, 2) will be located in location 3 of block 1, and so on.

Table 3.30: The assigned locations and blocks for import containers-the values of $z_{(i,k)}^{(n,b)}$ and $y_{(i,k)}^{b}$

Containers (i, k)	Locations (n, b)	$z_{(i,k)}^{(n,b)}$	$y_{(i,k)}^{b}$
(1, 1)	(4, 2)	$z_{(1,1)}^{(4,2)} = 1$	$y_{(1,1)}^{2} = 1$
(2, 2)	(3, 1)	$z_{(2,2)}^{(3,1)} = 1$	$y_{(2,2)}^{1} = 1$
(3, 1)	(1, 2)	$z_{(3,1)}^{(1,2)} = 1$	$y_{(3,1)}^{2} = 1$
(4, 2)	(1, 1)	$z_{(4,2)}^{(1,1)} = 1$	$y_{(4,2)}^{1} = 1$
(5, 1)	(2, 1)	$z_{(5,1)}^{(2,1)} = 1$	$y_{(5,1)}^{1} = 1$

3.3.3 Solution method: genetic algorithm

Chromosome representation and initialisation: By considering the main decision variables $x_{(i,k)}^{(j,l)}$, $y_{(i,k)}^{b}$, $z_{(i,k)}^{(n,b)}$ and $\sigma_{(i,k)}^{(j,l)}$, we use a matrix structure to represent the solution of the proposed problem. Here we differentiate between the chromosomes for import containers and those for export containers because the decision variables are not the same for these two types of container:

- For import containers, the decisions indicate the handling sequences of AGVs, YCs and assigned yard locations associated with each import container. More specifically, we have a matrix *Ψ1* with three columns, which represent the AGV delivery sequences, YC handling sequences and yard location assignments.
- For export containers, since the locations of export containers are known, the chromosome representation demonstrates two main decision variables: the handling sequences of containers for AGVs and YCs. We have a matrix *Ψ2* with two columns.

Import Containers			
Container	***Dispatched AGV***	***Assigned YC***	***Assigned Location***
(1, 1)	1	1	1
(3, 1)	2	2	3
(5, 1)	2	2	4
(2, 2)	3	1	2
(4, 2)	1	1	5

(a) The chromosome representation for import containers-the matrix *Ψ1*

Export Containers		
Container	***Dispatched AGV***	***Assigned YC***
(2, 1)	2	3
(4, 1)	1	3
(1, 2)	3	4
(3, 2)	1	4
(5, 2)	2	3

(b) The chromosome representation for export containers- the matrix of *Ψ2*

Figure 3.21: An illustration of chromosome representation for import and export containers

Figure 3.21 shows an example of the chromosome representation for the example of 10 containers same as in section 3.2.2. Three AGVs can carry both import and export containers. There are four YCs working in the storage yard, with YC 1 and YC 2 handling import containers, and YC 3 and YC 4 handling export containers. Since there

are five import containers, we select five available locations and assign them to store these containers (see column 3 of matrix $\Psi 1$).

Let us denote $|D|$ as the total number of import containers, v as the total number of AGVs and c as the total number of YCs, in which YCs from 1 to $|C1|$ ($|C1| < c$) are used for handling import containers and YCs from $|C1| + 1$ to c are used for handling export containers. The initial population is constructed by the following steps:

(1) Comparing travelling times from each QC to each available yard location, including the time AGV travels from QC to block and the YC moves from the transfer point of the block to the assigned location for import containers. We choose the locations with shortest distances for all the import containers, which satisfies constraint (3.3.7). Then we assign numbered labels from 1 to $|D|$ for all selected locations.

(2) Randomly assign these locations to each import container and each location can only be assigned to one container (column 3 of matrix $\Psi 1$) to satisfy constraints (3.3.5) and (3.3.6).

(3) Randomly choose an AGV from 1 to v (constraints (3.3.3) and (3.3.4)), i.e. assign one AGV to deliver a container (column 1 of matrix $\Psi 1$ and matrix $\Psi 2$) so that constraints (3.3.1) and (3.3.2) are met. This applies to both import and export containers.

(4) Randomly choose an YC from 1 to $|C1|$, i.e. assign one YC to handle an import container (column 2 of matrix $\Psi 1$) and also choose an YC from $|C1| + 1$ to c (constraints (3.3.11) and (3.3.12)) to handle an export container (column 2 of matrix $\Psi 2$), so that constraints (3.3.9) and (3.3.10) are met.

(5) Chromosomes are generated respectively by steps (1)-(4) until the population size *Pop* reaches a given number (e.g. 100) to ensure a large search space to start with.

(6) Evaluate all the matrices $\Psi 1$ and $\Psi 2$ in the initial population by calculating the values of $u_{(i,k)}$ according to constraints (3.3.13)-(3.3.23). The objective function value is obtained by $\mathrm{Max}(u_{(N_k,k)} + h_{(N_k,k)})$.

Parents selection strategy: We use 'roulette wheel' sampling for this problem. The fitness of each solution is obtained by calculating its OFV. Here, as the objective is to

minimise the berth time, it is preferable that the individuals with smaller objective function values (OFVs) are chosen as parents for the next generation.

Genetic operators design: we use two-point crossover for the AGV and YC assignment parts of chromosomes and use uniform order-based crossover for the location assignment part of the chromosomes. Two-point crossover is an extended version of single-point crossover in which two selected parents are recombined by two randomly chosen points, as illustrated in figure 3.5. This crossover applies to import and export container chromosomes (columns 1 and 2 of matrix $\Psi 1$ and the matrix $\Psi 2$). Uniform order-based crossover is an extended version of classical uniform crossover in which, after recombination, conflict genes are deleted and missing genes are replaced to prevent generating infeasible solutions, as shown in figure 3.6. This crossover works for column 3 of import container chromosome matrix $\Psi 1$. Again, swap mutation is applied to exploit neighbour solutions, in which the two randomly selected genes are exchanged, as illustrated in figure 3.7. The mutation operator works separately for the chromosomes of import containers and export containers.

Each operator is performed with a certain probability that is known in advance by GA parameter settings. The crossover rate P_c and mutation rate P_m determine the performance of GA; therefore, proper value settings are needed in order to ensure the convergence of GA to the global optimal neighbourhood in a reasonable time. The population size is kept unchanged during the crossover and mutation operations.

Offspring acceptance strategy: We use a semi-greedy strategy to accept the offspring created by the GA operators. In this strategy, an offspring is accepted as the new generation only if its OFV is less than the average of the OFVs of its parent(s). This approach enables GA to reduce the computation time and results in a fast convergence toward an optimal solution.

Stopping criterion: In order to balance the searching computation time as well as evolving an approximate optimal solution, we use two criteria as stopping rules: (1) the maximum number of evolving generations M_g allowed for GA, and (2) the standard deviation of the fitness values of chromosomes (σ_T) in the current generation T is below a small value.

3.3.4 Computational results

For small-sized problems, results obtained by the B&B algorithm and GA are compared in terms of OFV and computation time. Due to the exponential increase in the computation time by the B&B algorithm as the problem size is getting larger, it is infeasible to try to solve the problem by the B&B algorithm to get exact solution. Therefore, GA is adopted for solving large-sized problems by providing approximately optimal solutions within a reasonable time. GA is implemented in MATLAB 7.11. All the experiments are performed on a computer with Intel® Core™ i3 CPU M370@2.40GHz and 4GB RAM under the Windows 7 operating system. For each problem, we run it by GA for 20 times using the same parameter settings and the means of objective function values and computation times are reported to justify the effectiveness and reliability of the proposed approach.

Parameters settings

(1) The number of containers varies from 5 to 200, where 5-25 are considered as small-sized problems and 25-200 are considered as large-sized problems in this experiment. We also consider the number of AGVs varies from 2 to 10, the number of YCs varies from 2 to 7 and the number of QCs varies from 2 to 3.

(2) The uniform distribution was assumed for all the operation times. The processing times of each QC on these containers follows uniform distribution U(30, 180)s, and the handling times of each YC from each container's available location to the transfer point of the block in which this container locates follows uniform distribution U(40, 160)s.

(3) GA parameters take the following settings based on preliminary tests: Crossover rate P_c: 0.8; Mutation rate P_m: 0.01; Population size *Pop*: 100; and Maximum generations M_g: 60.

Results for small-sized problems

Table 3.31: Results of computational experiments in small sizes

No	Containers	AGVs/ QCs/ YCs	B&B (MIP)		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
1	5	2/2/2	0.02	355	0.01	355	0%
2	6	2/2/2	0.26	420	0.12	424	0.9%
3	7	2/2/2	0.44	496	0.22	499	0.6%
4	8	2/2/2	0.56	547	1.56	557	2%
5	9	2/2/2	0.62	608	2.06	615	1.2%
6	10	3/2/2	79.54	966	2.38	979	1.4%
7	10	4/2/3	169.82	926	4.62	948	2.4%
8	15	3/2/2	187.01	2029	6.46	2059	1.5%
9	20	3/2/2	1108.60	2563	9.05	2619	2.1%
10	25	5/2/3	/	/	12.14	3731	/

Ten random instances in small sizes are examined in this section. The problem parameters and GA parameters are set as above. Table 3.31 shows the comparison of results between B&B and GA. As shown in table 3.31, the computation time of AIMMS 3.11 grows exponentially as the problem size increases, in which the proposed problem is known as NP-hard. For the problem with over 20 containers, AIMMS took over three

hours without providing a solution. Moreover, it is found that the proposed GA can obtain the optimal/near-optimal solutions in all the small-sized cases in reasonable computation times. Compared with the B&B results from AIMMS 3.11, the average of the relative OFV gap rate is 1.24%, which is a promising outcome. In terms of the computation times, GA outperforms B&B to give solutions at a faster speed. It demonstrates that our proposed GA is able to obtain the near optimal solution fast.

Results for large-sized problems

Table 3.32: Results of computational experiments in large sizes

No	Containers	AGVs/QCs/ YCs	Computation time (s)	OFV (s)
11	30	3/2/3	14.53	4867
12	30	4/2/3	8.01	4228
13	30	5/2/3	15.90	4123
14	40	4/2/2	42.26	6372
15	40	4/2/3	20.97	5206
16	40	4/2/4	46.43	3169
17	50	4/2/4	60.23	4341
18	50	4/3/4	70.97	3884
19	50	5/3/4	111.36	3382
20	60	4/2/3	187.61	6487
21	60	5/2/3	245.88	5862

22	60	6/2/3	143.81	5128
23	70	6/2/3	127.44	6400
24	80	5/3/4	223.05	7589
25	80	5/3/5	418.58	5024
26	80	6/3/5	211.33	4782
27	90	6/3/5	285.35	7119
28	100	6/3/5	216.68	7200
29	100	8/3/5	171.53	6655
30	100	10/3/5	275.59	5685
31	120	10/3/5	549.71	8677
32	150	8/3/5	701.37	11182
33	150	10/3/5	260.03	9034
34	150	12/3/5	169.59	10822
35	180	10/3/5	264.53	12907
36	200	10/3/5	612.37	13811
37	200	10/3/6	225.62	11917
38	200	10/3/7	203.70	11992

Table 3.32 shows the average computation times and OFVs in large-sized instances. Firstly, we examine the effect of the total number of containers on the optimal berth

time and computation time. A general tendency is that the optimal berth time is increasing with the increasing number of containers.

Secondly, it is also noted that the decrease in the optimal berth time with respect to the number of QCs (such as case 17 and case 18) is more obvious than in other cases with different numbers of AGVS (such as case 12 and case 13) and YCs (such as case 24 and case 25), which means that the optimal berth time is affected most significantly by the number of QCs in container terminals. However, in reality, there are usually two to three QCs serving a medium-sized ship. This is because, when assigning more QCs to work for the ship, these QCs may have to operate together in a narrow berth space which may cause conflict. Also, the effect of the number of YCs is more significant on the optimal berth time than the effect of the number of AGVs, which further proves that cranes are not only the more expensive equipment, but also that they are always the bottleneck resources in the terminal operations. However, the number of cranes and vehicles must take reasonable values to avoid traffic congestions and conflicts.Thirdly, there is no apparent relationship between the problem size and the computation time for particular problems. However, generally, the computation time increases with the problem size.

Generally, these experiments indicate that the proposed GA is reliable in solving problems in different sizes and can be used in the real-life context to find the best combination of equipment for handling a set of containers.

From figure 3.22, we observe that our proposed GA reached convergence quickly before 60 generations where the OFV gives the best value for an instance of 80 containers, five AGVs, three QCs and five YCs.

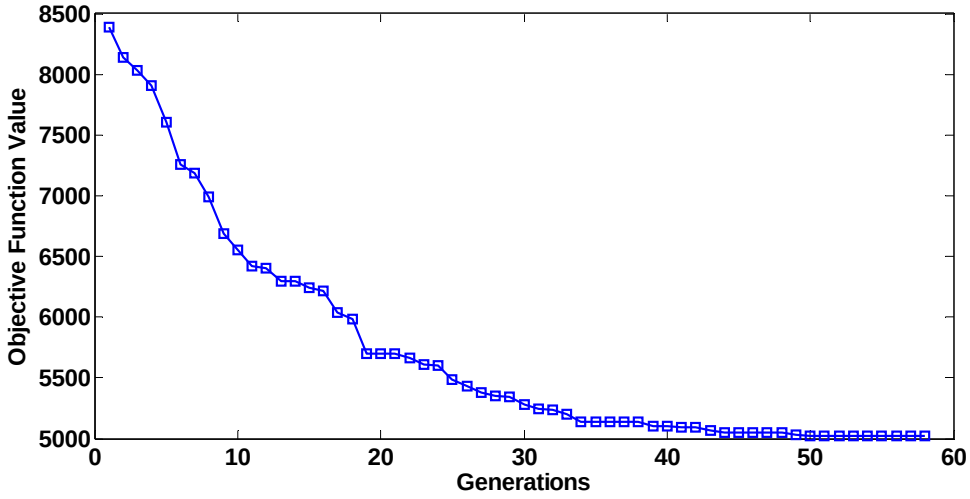


Figure 3.22: The typical convergence process of GA in a single run with the case of 80 containers, five AGVs, three QCs and five YCs

GA parameter sweep experiments

GA searches with different parameter settings are investigated for a particular case in order to select an effective parameter setting combination for this problem. Here, we consider the instance of 60 containers, six AGVs, two QCs and four YCs. GA parameter settings take the following values:

Crossover rate P_C = {0.6, 0.7, 0.8, 0.9};

Mutation rate P_M = {0.01, 0.02, 0.1, 0.2};

Population size Pop = {30, 50, 100, 150};

$$M_g = 50.$$

Figures 3.23-3.25 show the performance comparisons among different parameter settings. Figure 3.23 shows the convergence curves on different crossover rates for this particular case with mutation rate of 0.01 and population size of 100. The results in figure 3.23 show that the OFVs with the best solutions after 25 generations are close to each other with the crossover rates of 0.8 and 0.9, but the curve with crossover rate of 0.9 terminates earlier; figure 3.24 shows the convergence curves on different mutation rates for this particular case with crossover rate of 0.9 and population size of 100. The

results in figure 3.24 show that the curve with mutation rate of 0.01 converges to a smaller OFV; figure 3.25 shows the convergence curves on different population sizes for this particular case with crossover rate of 0.9 and mutation rate of 0.01. The results in figure 3.25 show that the OFVs with best solutions are close to each other after 20 generations with population size of 100 and 150, but the curve with population size of 100 terminates earlier. Therefore, according to these convergence curves, the crossover rate = 0.9, mutation rate = 0.01, and population size =100 is the best set for this typical problem. These figures also show that for all the experiments, OFVs are not improved after 45 generations due to the fast convergence of our proposed GA. So, the maximum generations of 50 will be sufficient to acquire the near-optimal solutions. Besides, in figures 3.23 and 3.24, the OFVs of the best convergence solutions are close to each other with different settings of crossover rates and mutation rates; however, in figure 3.25, the OFVs of the best convergence solutions vary with different population sizes, and the curve with larger population size gives a smaller OFV. This further proves that the scope of the initial search space is important to the performance of the GA.

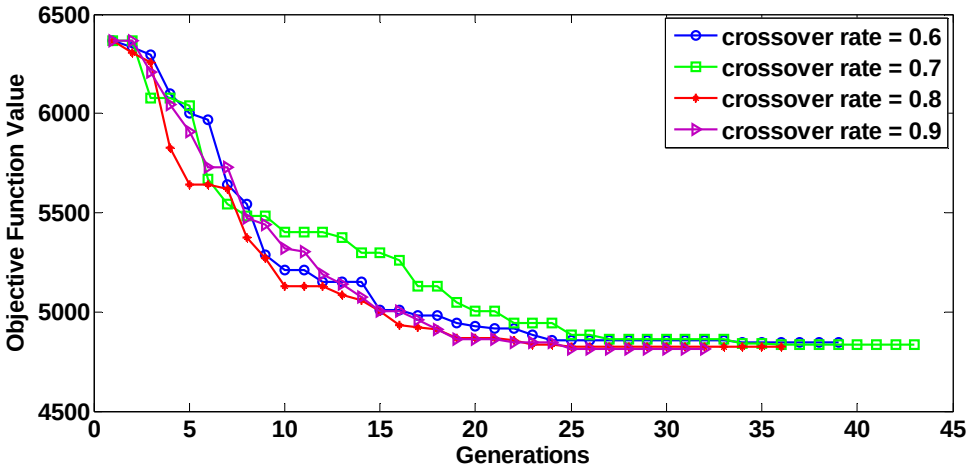


Figure 3.23: Performance comparison of different crossover rates for an example under the $Pop = 100$ and $P_m = 0.01$

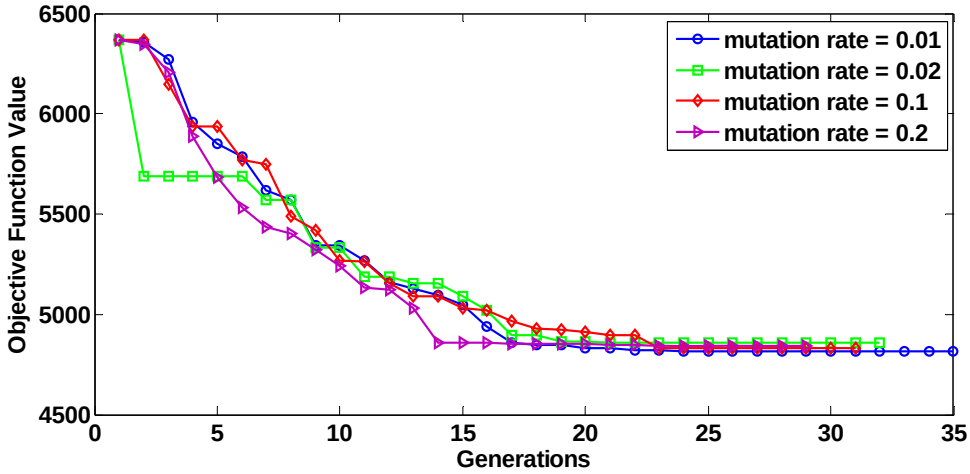


Figure 3.24: Performance comparison of different mutation rates for an example under the $Pop = 100$ and $P_c = 0.9$

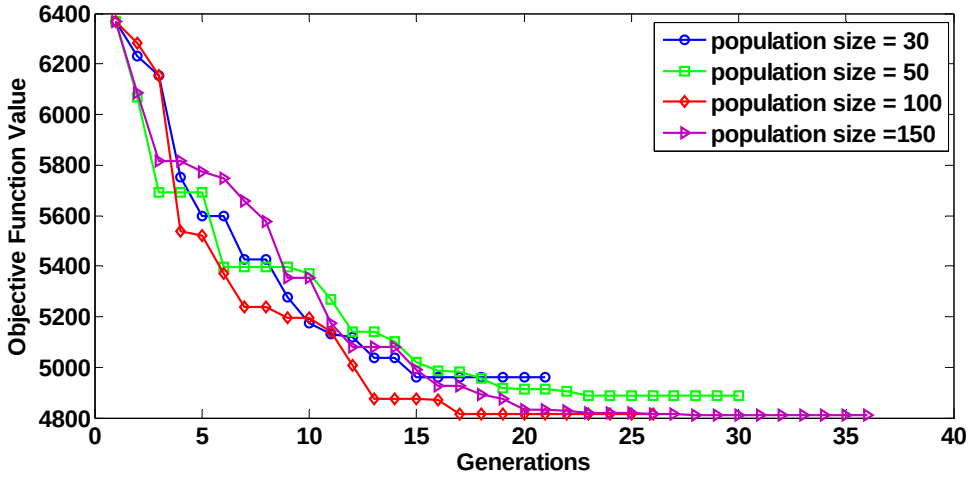


Figure 3.25: Performance comparison of different population sizes for an example under the $P_c = 0.9$ and $P_m = 0.01$

Lastly, we apply our proposed GA to the problem with 60 containers, six AGVs, two QCs and four YCs with the above optimal GA parameter settings, and the algorithm was run 10 times. The results are illustrated following the pattern shown by the box plot in figure 3.26. Each box represents the OFVs of the 10 runs in one generation. The central mark is the median of the OFVs, the edges of the box are the 25th and 75th percentiles and the whiskers are the most extreme data points. We can find that from the same starting point, each of the experiments improves the initial solution gradually, i.e. our proposed GA performs a stable manner in all the experiments; the deviation of

OFVs in each generation is becoming stable during the evolving progress, and all the OFVs converge to a small neighbourhood of optimal solution at about 32 generations.

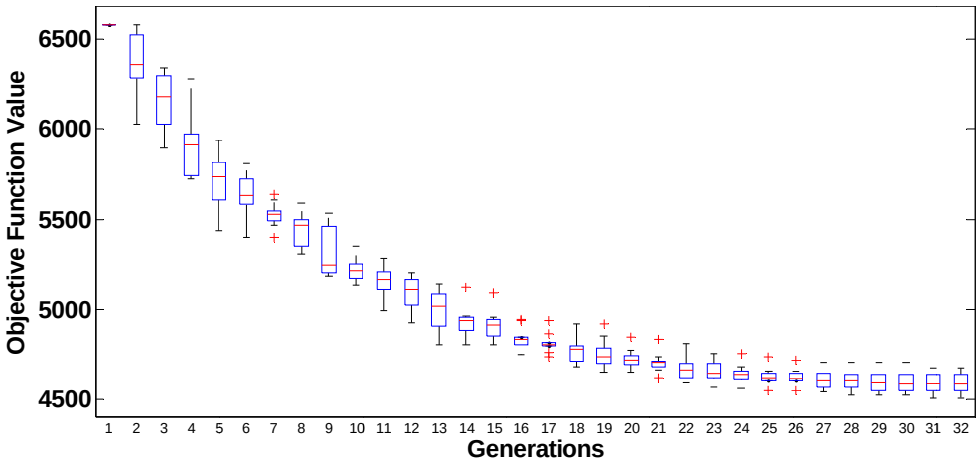


Figure 3.26: GA performance in 10 runs with $P_c = 0.9$, $P_m = 0.01$ and $Pop = 100$ for the case with 60 containers, six AGVs, two QCs and four YCs

3.4 Conclusion

The efficiency of container terminals depends extensively on the effectiveness of allocation of the terminal resources (vehicles/cranes/locations). In this respect, we considered the integrated vehicle scheduling and container storage problems in automated container terminals in this chapter. In particular, we studied these related logistics problems in the container unloading process (model 1), container loading process (model 2) and dual-cycle process (model 3). We aim to model different problems, so the assumptions in each model are not the same. From the container-handling point of view, the automated container terminal has its own specific characteristic, i.e. automated vehicles/equipment, which aims to reduce the cost of labour and thus requires more advanced planning of this complex environment. The motivation for these models comes from the fact that vehicle/crane scheduling and container storage allocation problems are linked, a factor which is not considered in previous studies. The objective is to increase the productivity of automated container terminals by reducing the berth time of ships.

The techniques used are to formulate these problems as mixed-integer programming (MIP) models. The models can easily be solved using available optimisation software AIMMS 3.11 in small-sized problems. However, the computation time increases exponentially as the problem size gets larger. Therefore, to solve the complicated NP-hard problem, we suggested an evolutionary heuristics, GA, for each model. In order to investigate different GA crossover operators in their abilities for solving the proposed problems, we carry out some preliminary tests and find that the evolving results with different crossover methods (two point crossover and uniform crossover) are very close. Therefore, we use these two crossover types for the developed GA. Sufficient computational experiments are carried out to test the performance of the models and the proposed solution methods. By solving the models, detailed schedules of AGVs and YCs as well as the container yard storage locations (for import containers) are determined. Our analysis shows that the effects of crane numbers are more significant than the effects of vehicles, which proves that cranes are the most critical resources in container terminals. It has also been shown that compared with the B&B algorithm embedded in AIMMS 3.11, our proposed GA for each model performs in a stable manner in solving the problem of different sizes. The average gap between B&B and GA in terms of the objective function values for small-sized problems is very small for all the models with the gap ranging from 0% to 2.3%. For each problem, we ran a series of GA parameter sweep experiments for a particular large-sized problem to further investigate the performance of GA on the proposed problem.

From an academic standpoint, this study is the first in the field to investigate these problems, and to provide the heuristic methods and suggest solutions. From a practical standpoint, all the models incorporate the real-life operational issues, such as how to dispatch vehicles/cranes, where to locate containers, and the findings will benefit terminal managers in their day-to-day operations.

However, as the sizes of containerships are continuously increasing and the capacity of containerships will by implication also increase, more efficient operational research techniques and decision-making algorithms are needed for solving problems in real-time situations. Specifically, adapting other efficient heuristic algorithms to these problems and comparing the performances with the GAs proposed here are a needed

direction of future research, to see if these techniques are able to achieve better results more efficiently than GA and thus improve the solutions to dispatch AGVs and cranes, and assign container locations. In addition, more practical considerations may be included. For example, precedence relationships because of the physical locations among tasks on the ship can be added in the model, and stochastic factors on the container information may also influence the implementation of the dual-cycle strategy.

Chapter 4. Modelling of integrated vehicle scheduling and container storage problems at straddle-carrier systems

This chapter considers another type of container terminal, straddle-carrier system, which utilises straddle carriers (SCs) and quay cranes (QCs) to handle containers between the quayside and the yard-side. With the aim of reducing the completion time at the quayside (known as berth time), we investigate the QC handling, SC scheduling and container storage locations by addressing them simultaneously during container unloading and loading processes. Due to the inherent complexity of the problems, we propose an evolutionary heuristic algorithm based on GA for each problem. In order to prove the efficiency of the proposed GA, the results are compared with the optimal solutions for small-scale problems by B&B using the commercial software, AIMMS 3.11. Computational results on large scales show the good quality of the solutions provided by GA and the performance of the proposed models.

Section 4.1 studies the integrated problem of SC scheduling and storage allocation problems during the container unloading process, section 4.2 investigates the integrated problem of scheduling QCs and SCs simultaneously during the container loading process, and section 4.3 discusses the integrated scheduling of QCs and SCs and the container storage problems taking both unloading and loading processes into consideration.

4.1 Model 4: The integrated SC scheduling and storage allocation problem during the container unloading process

In this section, we consider the integrated SC scheduling and storage allocation problem during the container unloading process and formulate it as a mixed-integer programming (MIP) model. SC scheduling and container storage problems are in fact interrelated. We aim to determine how the SCs can be dispatched to deliver containers to their optimal storage locations. The objective is to minimise the berth time of the ship, which is the completion time for unloading all the containers from the ship. The main

operational decisions are to determine the schedule for SCs to deliver containers and to assign the locations to store the set of import containers.

4.1.1 Problem description and formulation

In the straddle-carrier system, only QCs and SCs are used for handling containers, as shown in figure 4.1. For the unloading process, the buffer area is where QCs drop off containers from the ship. A container is picked up by a QC and then set down on the buffer area under this QC if the buffer is available; otherwise, if the buffer is not available, this QC has to wait for SCs picking up the containers on the buffer before it can set down any more containers on the buffer. A SC then picks up the container and delivers it to the storage yard. There are wide aisles between each yard bay to allow enough space for the SCs to pass through (see figure 4.1). The SC then travels back to the quayside with empty load to deliver another container.

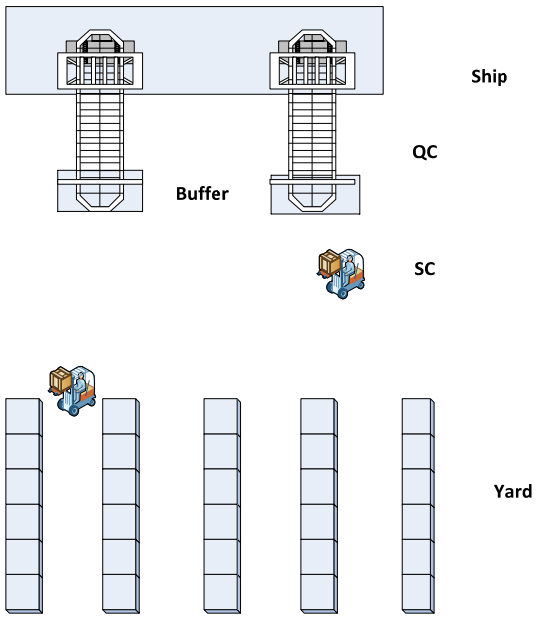


Figure 4.1:The typical layout of the straddle-carrier system

First, we look at the operational environment on the ship side. In general, a few hours before the ship's arrival, the terminal receives detailed information on those containers to be unloaded from this ship, which gives guidelines to the terminal operators for deciding the handling sequences for the QCs. Therefore, the QCs' handling sequences are known in this problem. In the vessel, ship bays are partitioned and each of them will

be covered by one QC (Lee *et al.*, 2009). The unloading process is carried out in the same order as in the sequence list of each QC (Nguyen and Kim, 2009). The unloading sequence is mainly determined by the containers' current positions on the ship, and as pointed out by Bish *et al.* (2005), the time required to move each container also depends on its ship position. As each container has a specific position on the ship, the QCs' unloading times are known and vary for different containers. Specifically, each unloading container is referred to as a job. For example, let D be the set of all the jobs (import containers) and k, l denote QCs (the index for QCs). Containers are indexed as (i,k) which denotes as the *ith* job in the sequence list of the QC k. Each QC k has number of N_k containers to handle, which means container (N_k,k) is the last container handled by QC k in the sequence list. Then, we define the time required to unload container (i,k) by the QC k is denoted by $h_{(i,k)}$, which is assumed to be deterministic as introduced by Bish *et al.* (2001).

Now we introduce the operations related to SCs. Since SCs cannot directly collect containers from QCs due to the function of SCs, the pickup time of a container by an SC may be different from the release time of this container by a QC. The reason for this might be that SCs are not always at the quayside. To model such a situation, there are two events considered at the quayside: (1) the event $E_{(i,k)}$, represents the event of QC k starting to handle its *ith* container, i.e. container(i,k) from its ship location; and (2) the event $e_{(i,k)}$, represents the event of a SC picking up container (i,k) from the buffer area under QC k. The event time of $E_{(i,k)}$is denoted by $u_{(i,k)}$, which corresponds to the time that container (i,k) has been picked up by QC k from the ship; the event time of $e_{(i,k)}$ is denoted by $w_{(i,k)}$, which represents the time of a SC picking up container (i,k) from the buffer area under QC k.

When modelling the problems in the SC system, we need to consider the buffer area at the quayside. To reduce the complexity of our model, we adopt a similar approach to that in Wong and Kozan (2010), which assumes that the capacity of the buffer area under each QC is considered to be one. It means that at any time during the unloading process, there will be at most one container in the buffer area under the QC. Although the capacity of buffer space is considered to be sufficient in the work of Nguyen and

Kim (2009), their study did not consider the integrated problem. We assume there are always enough SCs serving the QCs so as to avoid containers piling up in the buffer because there is not enough space to reshuffle containers at the quayside. Therefore, assuming capacity of one container in the buffer is a reasonable and efficient approach for terminal operations. For example, the current container in the buffer is container (i,k) and the next container to be released by a QC is container $(i+1,k)$ as in the sequence list of QC k. The time a SC picks up container (i,k), $w_{(i,k)}$ must be between the times that QC is able to release containers (i,k) and $(i+1,k)$, i.e. $u_{(i,k)}+h_{(i,k)}$ and $u_{(i+1,k)}+h_{(i+1,k)}$, thus ensuring there is at most only one container at the buffer at any time.

To decide the yard location, we define each yard location by using index $m \in N^+$ and denote the set of all locations as $M = \{1,2,...,m\}$, where m is a positive integer. There are a finite number of available slots in the storage yard, and typically this number is bigger than the number of import containers. Because SCs are able to pick up and dropoff containers by themselves without the aid of cranes, they could function as the yard cranes for stacking containers in the storage yard, then YCs are not necessary for straddle carrier system. In practical container terminals adopting pure straddle carrier system, such as DP World Southampton, there is no YCs employed for the container handling process. Containers are usually not stacked higher than four tiers in order to reduce the reshuffles. Reshuffle is a process to reach the target container which is not currently located on the top. Therefore, reshuffles are the non-productive moves in the yard operations. Usually, the process of container reshuffle is carried out during the quiet times in the yard to ensure the smoothness of future container-handling. The SC travelling time from QC k to each available location m in the yard is denoted as τ_k^m, which is known according to the distance between each QC and each yard location. However, the serve time of any import container from QCs to its yard location by a SC is determined by its assigned storage location in the yard, as stated in Lee *et al.* (2009). Then, such a storage allocation problem is transformed into assignment decisions between all containers in D and locations in M, where D is the set of all the import containers and M is the set of all available locations.

The model is developed based on the following assumptions:

(1) The handling sequences for each QC is given and the unloading process must be carried out in the exact order appearing in the QC's' sequence lists.

(2) Number of import containers, number of SCs and number of QCs are all known.

(3) QCs and SCs can only handle one container at a time.

(4) Pooling policy is applied to dispatch SCs, which means SCs are shared among all the QCs and can serve any QC during the unloading operation.

(5) Travelling/handling times between any two processing locations (QCs and yard locations) are known. For example, the SCs' travelling times between each QC and each available yard location are known; the SCs' travelling times among QCs are known.

(6) Traffic congestion of the SCs on the path is not considered.

(7) The conflicts among QCs are not considered. This is because QCs are positioned far enough away from each other at the quayside to avoid interference.

(8) The times for picking up/dropping off containers by QCs/SCs are negligible, which is a common assumption in the literature.

Now we list the mathematical notations related to the integrated model.

Sets and Parameters

D	set of import containers
K	set of QCs
C	Set of SCs
M	set of yard locations
k, l	index for QCs
$(i,k),(j,l)$	index for containers, container *(i, k)* means the *i*th container handled by QC *k*, container *(j, l)* is the *j*th container handled by QC *l*
m	index for yard locations, $m \in N^+$ where $N^+$is positive integer

c Total number of SCs

$h_{(i,k)}$ QC k's handling time of container (i,k)

τ_k^m SCs' travelling time between the buffer area of QC k and yard location m

M' a very large number

N_k The number of jobs in the working list of QC k

(S,I) the dummy starting job

(F,I) the dummy ending job

O_S The job set which contains all the jobs including the dummy starting job,

$$O_S = D \cup (S,I)$$

O_F The job set which contains all the jobs including the dummy ending job,

$$O_F = D \cup (F,I)$$

O The job set which contains all the jobs including dummy starting and ending jobs, $O = \{(S,I),(F,I)\} \cup D$

Decision variables

$u_{(i,k)}$ the time a QC k starts handling container (i, k) from its ship location

$w_{(i,k)}$ the time a SC picks up container (i, k) from the buffer under QC k

$v_{(i,k)}$ the time container (i, k) has been placed into its assigned yard location

The decisions of the SCs' schedules (delivery sequences) and storage allocation can be represented using the following two decision variables:

$$x_{(i,k)}^{(j,l)} = \begin{cases} 1, \text{if a SC which is scheduled to deliver container } (j,l), \text{has just delivered container (i, k)} \\ 0, \text{otherwise } \forall (i,k) \in O_S, (j,l) \in O_F \end{cases}$$

$$y_{(i,k)}^{m} = \begin{cases} 1, \text{if container } (i,k) \text{ will be located in location } m \text{ in the yard} \\ 0, \text{otherwise } \forall (i,k) \in D, \forall m \in M \end{cases}$$

Objective: Minimise the unloading element of berth time of the ship

$$\text{Min: Max}(u_{(N_k,k)} + h_{(N_k,k)})$$

Because the QCs' handling sequences are fixed, the unloading operation finishes when all the QCs have finished their jobs.

Contraints

$$\sum_{(j,l)\in O_F} x_{(i,k)}^{(j,l)} = 1, \forall (i,k) \in D \tag{4.1.1}$$

$$\sum_{(i,k)\in O_S} x_{(i,k)}^{(j,l)} = 1, \forall (j,l) \in D \tag{4.1.2}$$

$$\sum_{(j,l)\in D} x_{(S,l)}^{(j,l)} = c \tag{4.1.3}$$

$$\sum_{(i,k)\in D} x_{(i,k)}^{(F,l)} = c \tag{4.1.4}$$

$$\sum_{m\in M} y_{(i,k)}^{m} = 1, \forall (i,k) \in D \tag{4.1.5}$$

$$\sum_{(i,k)\in D} y_{(i,k)}^{m} \leq 1, \forall m \in M \tag{4.1.6}$$

$$w_{(i,k)} + \sum_{m\in M} \tau_k^m \, y_{(i,k)}^{m} = v_{(i,k)}, \forall (i,k) \in D, \forall k \in K \tag{4.1.7}$$

$$u_{(i+1,k)} - u_{(i,k)} \geq h_{(i,k)}, \forall (i+1,k), (i,k) \in D, \quad i = 1,2,\dots,N_k - 1, \forall k \in K \tag{4.1.8}$$

$$v_{(i,k)} + \sum_{m} \tau_l^m y_{(i,k)}^m \leq w_{(j,l)} + M'\left(1 - x_{(i,k)}^{(j,l)}\right), \forall (i,k) \in O_S, (j,l) \in O_F, \forall l \in K \tag{4.1.9}$$

$$u_{(i,k)} + h_{(i,k)} \leq w_{(i,k)} \leq u_{(i+1,k)} + h_{(i+1,k)}, \forall (i+1,k), (i,k) \in D, \quad i = 1,2,\dots,N_k - 1, \forall k \in K \tag{4.1.10}$$

$$x_{(i,k)}^{(j,l)}, y_{(i,k)}^m \in \{0,1\}, \forall (i,k), (j,l) \in O, \forall m \in M \tag{4.1.11}$$

$$u_{(i,k)}, w_{(i,k)}, v_{(i,k)} \geq 0, \forall (i,k) \in D \tag{4.1.12}$$

The objective function is to minimise the ship's berth time, which is the makespan of all the unloading jobs that are done.

Constraint (4.1.1) represents that for each container, there is one container delivered after it by the same SC.

Constraint (4.1.2) means that for each container, there is one container delivered before it by the same SC.

Constraints (4.1.3) and (4.1.4) guarantee that the total number of SCs deployed for handling containers is exactly *c*.

Constraint (4.1.5) implies that each import container must be stored in one location after the unloading process.

Constraint (4.1.6) implies that there are enough locations in the yard to accommodate the import containers.

Constraint (4.1.7) gives SC's travelling time between container *(i, k)* being collected by a SC from the buffer under QC *k* and this container has been placed in one storage location *m*, which must be at least a certain travelling time between QC *k* and location *m*, i.e. τ_k^m .

Constraint (4.1.8) represents the time between QC handling two consecutive containers *(i, k)* and container *(i+1, k)* must be at least the handling time of container *(i, k)*.

Constraint (4.1.9) is the time constraint, which requires that the empty-loaded travel time of SCs between handling two consecutive jobs *(i, k)* and *(j, l)* must be set at least by the travelling time from the yard location *m* of container *(i, k)* to the buffer under QC l, i.e. τ_l^m.

Constraint (4.1.10) ensures that there is at most one container at the buffer area during the whole unloading process, which means the pickup time of container *(i, k)* by a SC from the buffer must be after the time when this container *(i, k)* has been released by the QC, and also before the time when the QC releases the next container *(i+1, k)*.

Constraints (4.1.11) and (4.1.12) state that the restrictions of the decision variables are binary and non-negative.

4.1.2 An illustrative example

In this section, a computational example is given to evaluate the performance of the proposed MIP model and to assess the quality of solution. As the benchmark method, branch and bound (B&B) algorithm (embedded in AIMMS 3.11) is adopted for evaluating the model.

The operating environment considered in this computational example is as follows: the layout of the container terminal adopted in this analysis is the same as the one shown in figure 4.1. At the quayside, two QCs are assigned for unloading a set of eight import containers for a ship; and three SCs are travelling between the ship and the storage yard for delivering and stacking of containers. The parameters considered in this example are listed in the following tables.

First, in table 4.1, assuming there are four containers to be unloaded by each QC, we present the sequence list handled by each QC and the related handling time $h_{(i,k)}$, which follows uniform distribution U(30, 180)s. In this example, the time needed for retrieving container (1, 1) from its ship location to the buffer under QC 1 is 125 seconds, and the handling time for container (2, 2) by QC 2 is 85 seconds, and so on.

Table 4.1: The sequence list of each QC *k* and handle time $h_{(i,k)}$ of each container *(i, k)*

Containers (i, k) QC 1 (k=1)	$h_{(i,k)}$ (sec)	Containers (i, k) QC 2 (k=2)	$h_{(i,k)}$ (sec)
(1, 1)	125	**(1, 2)**	162
(2, 1)	87	**(2, 2)**	85
(3, 1)	143	**(3, 2)**	137
(4, 1)	158	**(4, 2)**	148

Table 4.2: The travel time of SCs between QCs and yard locations - the values of τ_k^m

Travel time (sec)	QC 1	QC 2
Location 1	271	253
Location 2	90	207
Location 3	234	275
Location 4	119	68
Location 5	173	217
Location 6	170	213
Location 7	65	249
Location 8	221	230
Location 9	136	116
Location 10	287	219

Assume there are ten locations in the yard available for accommodating this set of eight import containers. Now we list the SCs' travel times between each QC k and each available yard location m, τ_k^m in table 4.2. For example, a SC takes 271 seconds from QC 1 (k=1) to location 1 (m=1) in the yard, i.e. $\tau_1^1 = 271$; and takes 275 seconds from QC 2 (k=2) to location 3 (m=3) in the yard, i.e. $\tau_2^3 = 275$.

In our example, by solving the MIP problem proposed in section 4.1.1, we obtain the optimal berth time $\text{Max}(u_{(N_k,k)} + h_{(N_k,k)}) = 532$ sec, and we also have an optimal solution which gives the detailed schedule of each SC, as well as the assigned locations for each unloading container. Now we analyse the solutions in detail as follows.

Table 4.3: The assigned location of each container at the yard - the values of the optimal solutions $y_{(i,k)}^m$

Containers (i, k)	$y_{(i,k)}^m$
(1, 1)	$y_{(1,1)}^7 = 1$
(2, 1)	$y_{(2,1)}^2 = 1$
(3, 1)	$y_{(3,1)}^6 = 1$
(4, 1)	$y_{(4,1)}^1 = 1$
(1, 2)	$y_{(1,2)}^4 = 1$
(2, 2)	$y_{(2,2)}^9 = 1$
(3, 2)	$y_{(3,2)}^8 = 1$
(4, 2)	$y_{(4,2)}^3 = 1$

Firstly, the assigned location of each container at the storage yard is shown in table 4.3. These decisions are obtained from the values of $y_{(i,k)}^m$. For example, container (1, 2) will

be located in location 4 in the yard after the unloading operation; and container (2, 1) will be stored in location 2 in the yard.

Secondly, we look at the schedule of each SC which is given by table 4.4. This decision is obtained from the optimal solutions $x_{(i,k)}^{(j,l)}$. For example, in this solution, SC 1 delivers containers (1, 2), (2, 2), (3, 2) and (4, 2) consecutively; SC 2 transports container (1, 1) first from the quayside to the yard, then travels back to the quayside to pick up container (2, 1) and delivers container (2, 1) to the yard before collecting container (4, 1) from QC 1. Note that SC 3 only has to deliver container (3, 1). For consistency, we have dummy starting and ending jobs (S, I) and (F, I) in this solution.

Table 4.4: The delivery sequences of SCs - the values of the optimal solutions $x_{(i,k)}^{(j,l)}$

SCs	Containers	$x_{(i,k)}^{(j,l)}$
SC 1	(1, 2)	$x_{(1,2)}^{(2,2)} = 1$
	(2, 2)	$x_{(2,2)}^{(3,2)} = 1$
	(3, 2)	$x_{(3,2)}^{(4,2)} = 1$
	(4, 2)	
SC 2	(1, 1)	$x_{(1,1)}^{(2,1)} = 1$
	(2, 1)	$x_{(2,1)}^{(4,1)} = 1$
	(4, 1)	
SC 3		$x_{(S.I)}^{(3,1)} = 1$
	(3, 1)	$x_{(3,1)}^{(F,I)} = 1$

Lastly, we look at the values of $u_{(i,k)}$ $w_{(i,k)}$ and $v_{(i,k)}$ which are presented in table 4.5, i.e. the time QCs start to handle each container from the ship, the time SCs pick up each

container from the buffer, and the time each container has been placed into its yard location. Therefore, the schedules of all the handling equipment, QCs and SCs, can be obtained. For example, when the unloading operation starts, container (1, 1) and container (2, 1) are handled by QC 1 and QC 2 separately at time 0; at time 125 seconds, a SC comes to collect container (1, 1) from the buffer under QC 1 and delivers it to its assigned location at 190 seconds; similarly, container (1, 2) is picked up by a SC at time 162 seconds and stored in the yard at 230 seconds. The optimal objective function value is achieved by container (4, 2), which has the largest value of 532 seconds, i.e. $u_{(4,2)} + h_{(4,2)} = 532$.

Table 4.5: The optimal solutions of QCs' start handling time, SCs' start delivery time and SCs' arrival time at the yard location - the values of $u_{(i,k)}$, $w_{(i,k)}$ and $v_{(i,k)}$

Containers (i, k)	$u_{(i,k)}$ (sec)	$w_{(i,k)}$ (sec)	$v_{(i,k)}$ (sec)	$u_{(i,k)} + h_{(i,k)}$ (sec)
(1, 1)	0	125	190	125
(2, 1)	125	255	345	212
(3, 1)	212	355	525	355
(4, 1)	355	513	784	513
(1, 2)	0	162	230	162
(2, 2)	162	298	415	247
(3, 2)	247	532	762	384
(4, 2)	384	992	1268	532

From this example, we have found that our proposed model is able to provide solutions for the SC scheduling and container storage allocation problem during container

unloading operations. However, as an NP-hard problem, the results of this problem obtained by commercial software AIMMS 3.11 have some limitations: for instance, the computation time increases exponentially as the problem size (i.e. number of containers) becomes larger. Therefore, in the next section, we develop an evolutionary heuristic method-genetic algorithm for dealing with the large-sized problems.

4.1.3 Solution method: genetic algorithm

GA-based approaches have been proven to solve large-sized practical problems efficiently by providing approximately optimal solutions. However, for the integrated straddle carrier scheduling and storage allocation problem, little literature has explored adopting GA to solve the problem. Therefore, in this section, we attempt to represent the SC schedules (sequences) and container storage locations as expressions of chromosomes and propose a GA-based evaluation process towards the near-optimal solution. We introduce the design of our GA-based approach in the following stages.

Chromosome representation and initialisation: For our approach, the chromosome representation demonstrates two main decision variables, $x_{(i,k)}^{(j,l)}$ and $y_{(i,k)}^{m}$, as introduced in section 4.1.1; i.e. in which order to assign SC to deliver containers and where to store containers in the yard. Chromosomes are presented by using a two-dimensional matrix ψ with two columns (n^* *2*), where *n* is the number of containers, the first column represents $x_{(i,k)}^{(j,l)}$, and the second column represents $y_{(i,k)}^{m}$. Each row in ψ is the chromosome for each container.

Figure 4.2 demonstrates a chromosome representation with eight containers (eight rows), three SCs and eight locations. For instance, the matrix (denotes as ψ) showed in figure 4.2 specifies that container (4, 1) will be delivered by SC 2 to location 6 in the yard i.e. $y_{(4,1)}^{6} = 1$.

Containers	*Dispatched SCs*	*Assigned locations*
(1, 1)	**1**	**3**
(2, 1)	**2**	**2**
(3, 1)	**3**	**4**
(4, 1)	**2**	**6**
(1, 2)	**3**	**8**
(2, 2)	**2**	**5**
(3, 2)	**1**	**7**
(4, 2)	**3**	**1**

Figure 4.2: Two-dimensional matrix ψ of the chromosome representation for an example of eight containers, three SCs and eight locations

Let us denote $|D|$ as the total number of import containers, and c as the total number of SCs. We construct our initial population in the following steps:

Step 1: Shortest distance-based location selection: comparing distances (travelling times) between each available yard location to each QC, which is deterministic as introduced in section 4.1.1, and then choosing the locations with shortest distances for the incoming containers, which satisfies constraint (4.1.6), because we assume that there is enough space in the yard to store import containers.

Step 2: Location assignment: assigning the selected locations randomly to each container and each location can only be assigned to one container, which addresses constraint (4.1.5).

Step 3: SC assignment: randomly choose only one SC from 1 to c (constraints (4.1.3) and (4.1.4)), i.e. assign SCs to deliver all containers so that constraints (4.1.1) and (4.1.2) are satisfied.

Step 4: Population size (*Pop*): chromosomes are generated respectively by adopting Steps 1-3 until the population size Pop reaches a given number, e.g. 100 initial solutions (Choi *et al.* (2011); (He *et al.*, 2010)) to ensure starting from a large search space. Therefore, after this process (step 4), we have a set of 100 population-sized matrices of ψ.

Step 5: Estimating objective function value: evaluate each matrix ψ in the initial population by calculating the values of $u_{(i,k)}$, $w_{(i,k)}$ and $v_{(i,k)}$, i.e. the time a QC starts to handle each container, the time a SC starts to handle each container from the buffer and the time each container has been stored in the yard, subject to the constraints (4.1.7)-(4.1.12). The value of objective function (OFV) is obtained by $\max(u_{(N_k,k)} + h_{(N_k,k)})$, where $h_{(N_k,k)}$ is the handling time of the last container in the QC k 's handling sequence, i.e. N_k *th* container by QC k where N_k is the total number of containers handled by QC k.

After introducing the chromosome representation and initial solution generation, in the next stage of the GA design, we discuss how to carry out crossover and mutation operations, particularly for the chromosomes in this problem.

Genetic operators design: We use a method similar to that adopted in chapter 3 for this problem. One of the crossover operators we employed is the so-called uniform crossover (illustrated in figure 3.14), which applies to the first column of the chromosome (column for 'despatched SCs' as in figure 4.2). However, when applying uniform crossover directly for the second column of the chromosome (column for 'assigned locations' as in figure 4.2), infeasible children may be generated with missing and/or repeated values in which some locations may not be assigned to any container, or some locations are assigned twice for two different containers. So constraint (3) is not satisfied. Thus, we use uniform order-based crossover (illustrated in figure 3.6) for the second column of the chromosome in order to correct the infeasibility of the generated offspring. Both crossover methods operate according to the crossover rate P_c . Then we

could evaluate the fitness values (1/OFV) of the two offspring and make the selection (as in the next stage) to ensure that, on average, the fitness of population is improving in the evolving process.

For each individual offspring, there exists the possibility to experience a mutation operation. The main task of the mutation operation is to maintain the diversity of the population in the successive generations, thus forcing GA to search new spaces. Chromosomes are selected at random in terms of the probability of mutation, P_m, two randomly chosen positions of the chromosome swaps, as illustrated in figure 3.17. This mutation operator applies to both columns of chromosomes.

Offspring acceptance strategy: The semi-greedy strategy is used to accept the offspring generated by genetic operators. An offspring is accepted for the next generation if its fitness is larger than the average fitness of its parent(s).

Parents selection strategy: The selection mechanism used here is binary tournament selection (denoted as B), in which two randomly chosen individuals are compared in terms of their fitness values, and the one with higher fitness will be selected as one parent for generating offspring in the next generation. The reason we choose 'binary tournament selection' instead of 'roulette wheel sampling' (as in the last chapter) is that through our preliminary tests, on average binary tournament selection gives a better evolving result.In addition, the best individual (with smallest objective function value - OFV) in the current generation will always be selected to be the parent of the next generation to ensure a monotonous behaviour of the GA during the evolving process.

Stopping criterion: We use two criteria as stopping conditions of GA: (1) maximum of evolving generations M_g, which is a common rule; and (2) the standard deviation σ_T of the fitness values of chromosomes in the current generation is below a small value ε.

Figure 4.3 shows the pseudo code of our proposed genetic algorithm designed as above. Furthermore, G denotes the number of current generations, Pop_1 is the population needed for crossover, Pop_2 is the population needed for mutation, Pop_c is the population after crossover, Pop_m is the population after mutation and Pop_{new} is the parents for the next generation.

1. Initialise Pop, P_c, P_m, M_g, G
2. Generate Pop feasible solutions and denote as Pop_{old}
3. While $G \leq M_g$ && $\sigma_T > \varepsilon$
4. $\forall\, i \in Pop_{old}$, $assign\ a\ random\ number\ \mu_i \in [0,1] \rightarrow i$, if $\mu_i < P_c$, then add i in Pop_1 ; $\forall\, k_1 \in Pop_1, \forall k_2 \in Pop_1$, do crossover and select offspring to Pop_c.
5. $\forall i \in Pop_c$, $assign\ a\ random\ number\ \omega_i \in [0,1] \rightarrow i$, if $\omega_i < P_m$, add $i\ in\ Pop_2$; $\forall k_3 \in Pop_2$, do mutation and select offspring to Pop_m.
6. Apply binary tournament selection: B(Pop_m) → Pop_{new}.
7. $G = G + 1$; Loop until terminated.

Figure 4.3: Pseudo code of our proposed genetic algorithm for the SC scheduling and storage allocation problem

4.1.4 Computational results

In this section, the performance of the proposed model and the designed GA are examined by a series of numerical examples with different sizes. Branch and bound (B&B) algorithm in AIMMS 3.11 software is used to solve small-sized examples and provide optimal solutions. However, it is impossible to get an optimal solution for a large-sized problem (with over 60 containers for this problem) in a reasonable computation time by AIMMS 3.11. Therefore, all the small-sized examples are also solved by GA and the results from GA are compared with those from B&B in terms of OFVs and computation times. We use GA to solve large-sized examples. GA is implemented in MATLAB 7.11. All the experiments are performed on a computer with Intel® Core™ i3 CPU M370@2.40GHz and 4GB RAM under the Windows 7 operating system. For each problem, we run it 20 times by GA using the same parameter settings, and the means of OFVs and computation times are listed.

Parameters settings

(1) The number of containers varies from 5 to 300, where 5-20 are considered as small-sized problems and 20-300 are considered as large-sized problems. We also consider that the number of SCs varies from 2 to 10, and the number of QCs

varies from 2 to 3, because for a single ship, usually two or three QCs are serving it.

(2) The uniform distribution was assumed for all the operation times. For example, the processing times of each QC on these containers follows uniform distribution U(30, 180)s, and the travelling times of each SC from each QC to each available yard location follows uniform distribution U(40, 300)s.

(3) GA parameters take the following typical settings: Crossover rate P_c: 0.8; Mutation rate P_m : 0.01; Population size *Pop*: 100; and Maximum generations M_g: 50.

Results for small-sized problems

After having developed the model and calibrated the algorithm, now firstly, we test the effectiveness of the proposed model and algorithm by the following small-sized examples. The results for small-sized examples are summarised in table 4.6, which includes the OFVs and computation times obtained by both B&B (in AIMMS 3.11) and GA. The results from the AIMMS software are optimal solutions, which are used to compare the results from GA. In all these instances, the GA stopped when the stopping criterion is satisfied, as introduced in section 4.1.3. Therefore, the computation time of GA depends on the quality of solutions (diversity) as well as the number of evolving generations. The results show that, as the problem size increases (with more containers), the computation times and OFVs of both B&B and GA increase. Although GA sacrifices more computation time than B&B for small sizes, it can obtain near-optimal solution for all the cases with an average OFV gap of 0.68%; that is a promising outcome, which can be accepted by container terminals.

Table 4.6: Comparison results of computational examples in small sizes by B&B and GA

No	No. of containers	No. of QCs/ SCs	B&B		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
1	5	2/2	0.01	253	3.84	253	0%
2	6	2/2	0.03	306	4.16	306	0%
3	7	2/2	0.06	384	3.62	384	0%
4	8	2/3	0.25	532	2.15	532	0%
5	9	2/3	0.21	578	4.18	581	0.34%
6	10	2/3	0.31	602	4.64	607	0.83%
7	10	2/4	0.26	556	3.97	562	1.07%
8	15	2/4	0.23	1048	7.64	1051	0.28%
9	20	2/3	2.85	1627	6.57	1650	1.41%
10	20	2/4	2.78	1203	6.31	1258	2.90%

Results of large-sized problems

Secondly, we look at the comparison between B&B and GA conducted in large-sized problems. In order to verify the effectiveness and reliability of the proposed model and algorithms, the following experiments in different large sizes are carried out. The details, i.e. problem sizes, OFVs and computation times are shown in table 4.7. Referring to this table, the OFVs obtained from B&B tend to be lower than those obtained from GA,

with an average gap of 3.73% from case No.11 to case No. 22. However, the computation time of B&B increases exponentially as the problem size increases, from which aspect GA performs better than B&B. Generally, increasing in the number of QCs (for example case 20 and case 21) and number of SCs (for example case11 and case 12) could lead to a decrease in the OFV. However, more QCs and SCs used in the container handling operations may cause conflicts; and may need more planning time (computation time). Due to the complexity of the problem, B&B cannot solve this problem with more than 60 containers, but GA may still be able to solve the problem with even larger sizes. Although the computation time of GA for solving the problem with more than 300 containers is not very short, it is still within minutes of time and thus acceptable in practice. These results indicate that the proposed GA is reliable for different-sized examples and can be adapted to the real-life situation where hundreds of containers are handled at the same time.

Table 4.7: Comparison results of large-sized examples by B&B and GA

No	No. of containers	QCs/ SCs	B&B		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
11	30	2/3	6.23	2247	7.41	2309	2.75%
12	30	2/4	7.8	1935	5.42	2007	3.72%
13	30	2/5	8.12	1821	4.19	1908	4.77%
14	40	2/3	20.89	2797	9.77	2878	2.89%
15	40	2/4	26.66	2582	9.79	2668	3.3%
16	40	2/5	31.04	2673	12.28	2801	4.7%

17	50	2/5	2026.50	2962	25.23	3046	2.83%
18	50	2/6	2253.09	2997	20.52	3110	3.77%
19	50	2/7	2108.56	2848	20.06	2972	4.35%
20	60	2/6	2348.69	4097	17.01	4236	3.39%
21	60	3/6	3219.04	3063	32.96	3227	5.35%
22	60	3/7	3374.56	2891	8.96	2978	3.01%
23	80	2/6	/	/	31.79	5064	/
24	80	2/7	/	/	63.19	4693	/
25	80	3/7	/	/	64.39	3642	/
26	100	2/6	/	/	67.35	6155	/
27	100	2/7	/	/	65.04	5772	/
28	100	2/8	/	/	107.21	5908	/
29	150	2/7	/	/	185.41	9161	/
30	150	3/7	/	/	156.08	8035	/
31	150	3/8	/	/	94.67	7815	/
32	200	3/8	/	/	305.70	10585	/
33	200	3/9	/	/	151.89	9952	/
34	200	3/10	/	/	154.43	9558	/
35	300	3/8	/	/	1066.21	16632	/

36	300	3/9	/	/	928.10	16056	/
37	300	3/10	/	/	740.68	14674	/

A typical evolutionary process of GA for an example with 80 containers, three QCs and eight SCs is shown in figure 4.4. Each generation includes a population size of 100 solutions and the best one with smallest OFV is plotted. For this case, GA stopped after 43 generations by providing the solution of 3500 seconds (OFV) for the problem. It is also shown that the OFV is improving quickly over the whole evolving process until the algorithm stops.

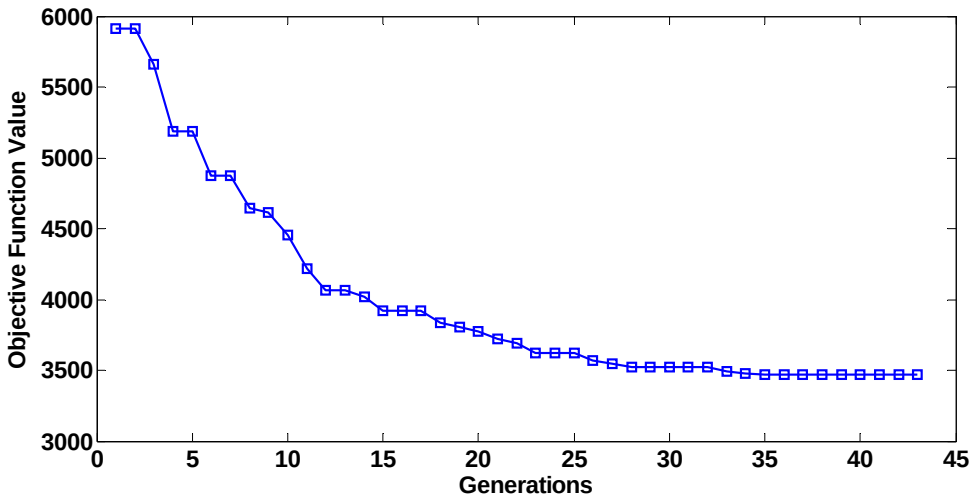


Figure 4.4: Typical convergence of GA in a single run for the problem with 80 containers, three QCs and eight SCs

GA parameter sweep experiments

In order to evaluate the performance of GA more deeply, another set of experiments is adopted, i.e. GA parameter sweep experiments, to test the performances of different parameters. This performance measure is often used to test the effectiveness of the proposed heuristics, and to determine the best parameter setting combination for a particular problem (Han, Lu and Xi, 2010; Yang, Wang and Li, 2012). We apply GA to the problem with 80 containers, three QCs and eight SCs, with different GA parameter settings. The crossover rates P_c take the values of 0.6, 0.7, 0.8 and 0.9; the mutation

rates P_m take the values of 0.01, 0.02, 0.1 and 0.2; and the population sizes Pop take the values of 30, 50, 100 and 150 respectively. Figure 4.5 shows how different values of P_c affect the OFVs for the considered problem with $P_m = 0.01$ and $Pop = 100$. According to these convergence curves, the one of $P_c = 0.7$ outperforms other values in reaching a lower OFV in the end. Figure 4.6 shows the convergence curves on different P_m for the case with $P_c = 0.7$ and $Pop = 100$. It can be found that the one of $P_m = 0.02$ outperforms other values in getting a better OFV and requiring a relatively shorter evolving process. Figure 4.7 shows how different population sizes affect the OFVs with $P_c = 0.7$ and $P_m = 0.02$ for this case. The curves show that the one of $Pop = 100$ and the one of $Pop = 150$ both return the best OFVs at the same generation, which means the initial solution with $Pop = 100$ is enough to provide a good performance of GA. To reduce the computation time, we choose $Pop = 100$ as the best value for population size because it usually takes more evolving time for a large population of solutions.

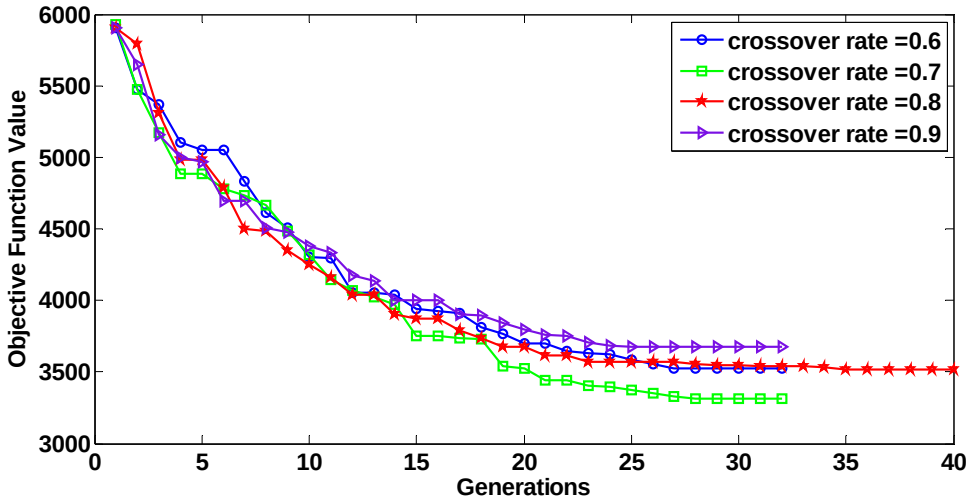


Figure 4.5: Performance comparison of different crossover rates for an example with $P_m = 0.01$ and $Pop = 100$

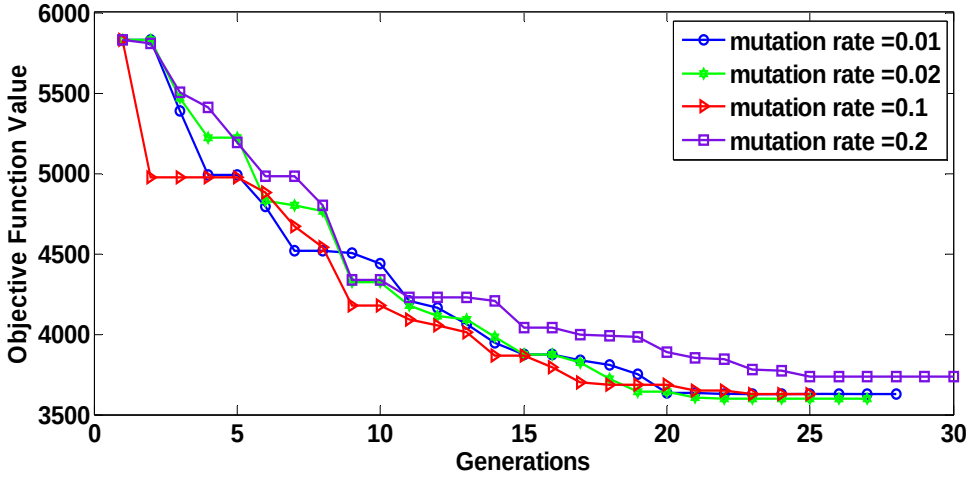


Figure 4.6: Performance comparison of different mutation rates for an example with $P_c = 0.7$ and $Pop = 100$

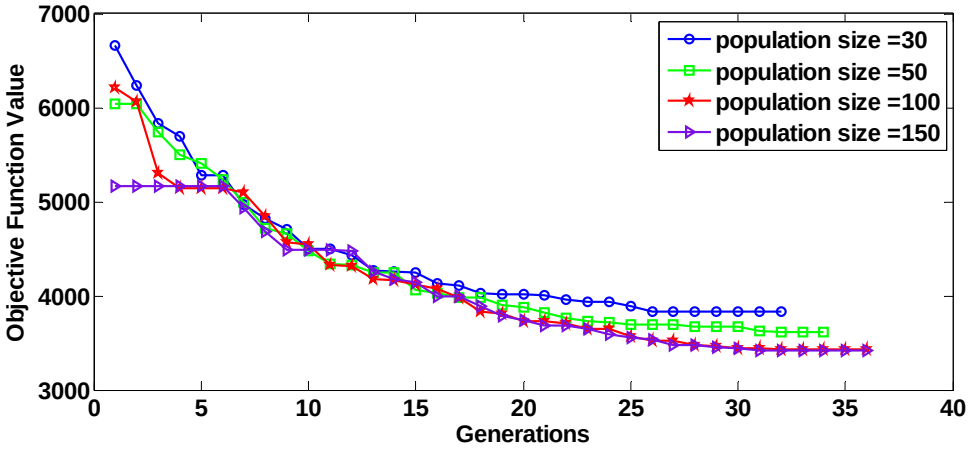


Figure 4.7: Performance comparison of different population sizes for an example with $P_c = 0.7$ and $P_m = 0.02$

Note that in figures 4.5 and 4.6, all the curves start from the same initial solution. In figure 4.7, here we randomly generate solutions in different population sizes. Therefore, the starting values are different. It can be found that solutions with larger population sizes can usually provide better initial solutions. To sum up, the best combination of GA parameters are $P_c = 0.7$, $P_m = 0.02$ and $Pop = 100$ for the example with 80 containers, three QCs and eight SCs. At last, we run this problem 10 times to further test the performance of GA. The convergence processes are shown in the box plot in figure 4.8. The central mark is the median, the edges of the box are the 25th and 75th percentiles and

the whiskers are the most extreme data points. We can find that for all the cases, GA reached convergence within 40 generations where the fitness is the best. It also demonstrates that our proposed GA is good at refining the initial solutions quickly. The performances of all the 10 runs are quite close in each generation and have similar convergence behaviour. This further proves the reliability of our proposed GA.

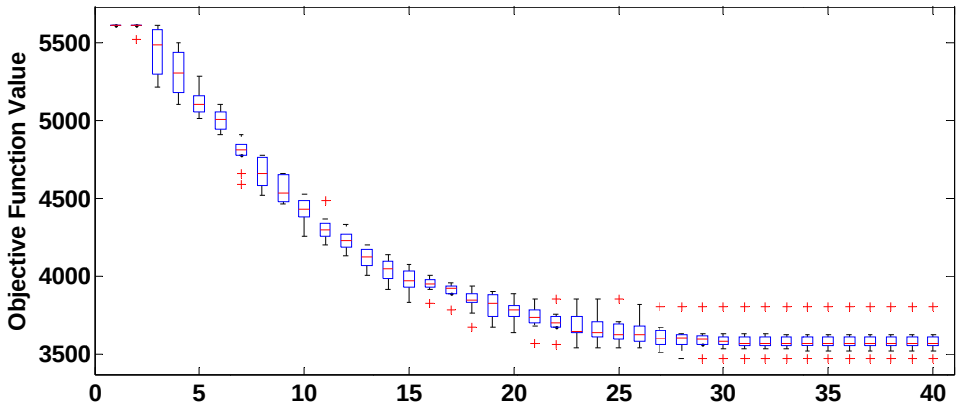


Figure 4.8: GA performance in 10 runs with $P_c = 0.7$, $P_m = 0.02$ and $Pop = 100$ for the case with 80 containers, three QCs and eight SCs

4.2 Model 5: The integrated QC and SC scheduling problems during the container loading process

In the straddle-carrier system, export containers are transported from the storage yard to the ship by straddle carriers. In such a system, the coordination of quay cranes (QCs) and straddle carriers (SCs) has a great impact on the container terminal's operational efficiency. In the problem proposed in this section, the loading schedules of QCs and SCs (decision variables) are determined simultaneously to specify the orders in which QCs load containers onto the container vessel and the order in which the SCs deliver containers from the yard to the vessel. This provides a guideline for the terminal manager on how to most efficiently carry out an actual loading operation employing QCs and SCs. Most previous studies have focused on the optimisation problems of scheduling one single piece of handling equipment, and for those studies on the SC scheduling problem, containers are delivered by a combination of yard trucks and SCs,

in which SCs function as the flexible moving YCs, which is different from the focus of our work. We model such a constrained optimisation problem as a mixed-integer programming model, aiming to minimise the berth time of the vessel, which is the completion time that all the export containers have been loaded onto the ship. A series of numerical experiments are reported to evaluate the effectiveness of the proposed integration approach and algorithm.

4.2.1 Problem description and formulation

In the straddle-carrier system, only two types of equipment are employed. Figure 4.9 shows the layout of a container terminal in the straddle-carrier system. Usually there are no more than two containers in the buffer under each QC to avoid container reshuffles and traffic congestion at the quayside, which are inefficient operations in the port; therefore, we consider that the capacity at the buffer under each QC is one (for example, Wong and Kozan, 2010).This means that at any time during the loading process, there will be at most one container in each buffer. If there is one container in the buffer, the SC cannot drop any more containers until this container has been picked up by a QC.

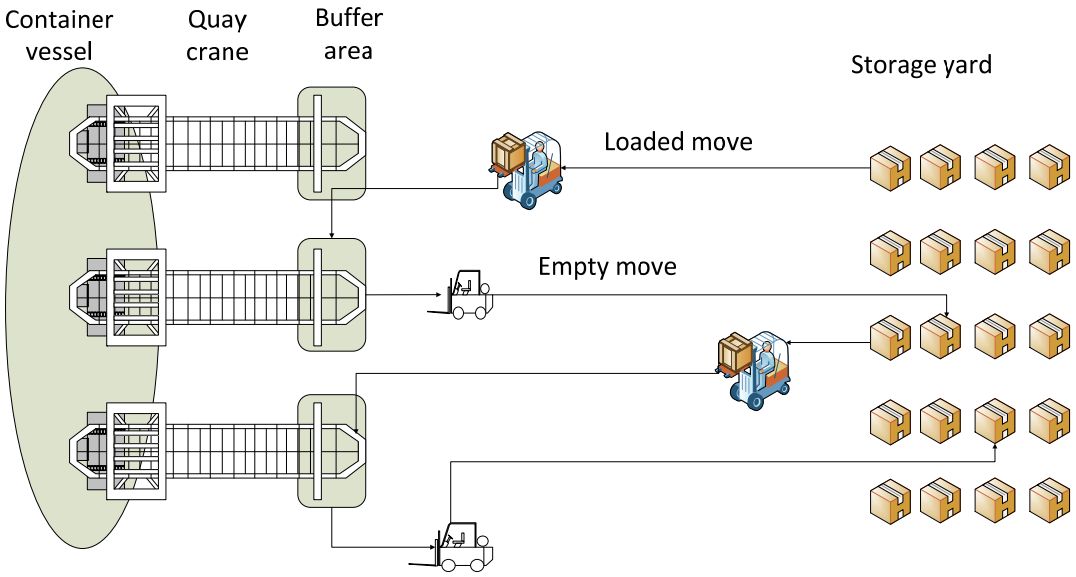


Figure 4.9: The layout of the container terminal in the straddle-carrier system

Secondly, we examine the loading operation in such a system. In the loading operation, a SC picks up an export container and delivers it to the quayside (loaded move); then

the container is set down in the buffer area (when the buffer space is available) under a QC, which loads this container onto the ship. The SC returns to the yard to collect the next container without carrying any containers (empty-loaded move). The makespan of the quayside operation depends not only on the performance of QCs but also on the delivery of containers by SCs. For example, if the delivery operation of a SC slows down, then the operation of a QC must slow down accordingly due to the buffer capacity; on the other hand, if the buffer area under a QC is full (reached the buffer capacity), then SCs have to wait to put down any more containers until this buffer is available again.

The berth time of the vessel, i.e. the latest completion time among all handling tasks of the vessel, is a critical factor for measuring the success of a terminal. Therefore, in our study, we aim to minimise the loading element of the berth time which corresponds to the time duration when all the export containers have been loaded onto the vessel. On the sea side, after a container vessel arrives at its berth location, several QCs are assigned to serve this vessel. We assume that the vessel and QCs are available at the start of the loading operations and the QCs' positions are usually set a safe distance apart in order to avoid conflict between each other, which means that interference among QCs are not considered in this study. In this problem, we determine (1) the QCs' handling sequences and time schedules to load containers onto the ship, and (2) the sequences and time schedules of SCs picking up containers from the storage yard, simultaneously.

Thirdly, the QC scheduling and the SC scheduling problems are analysed as follows. On one hand, we consider the QC scheduling problem. Generally, a few days before the vessel arrives at the port, the shipping company provides the load profile of this vessel to the container terminal. The load profile shows the pre-determined locations for loading containers from the corresponding terminal onto the container vessel. Due to the fact that containers are allocated fixed locations on the ship, in our study we assume that the handling time for each container by QCs is known. The terminal planner determines the handling sequences of each QC based on the load profile to minimise the completion time of QCs' operations. On the vessel, containers are stacked on top of one

another and arranged in rows. A container vessel is partitioned into a number of bays, which consist of stacks of containers.

On the other hand, let us examine the straddle carrier (SC) scheduling problem, which considers how to dispatch SCs to deliver containers, as mentioned in section 4.1. A SC is a large eight-wheeled vehicle which can lift the containers directly from the ground, can transport the containers to specified locations and can also stack containers up to four tiers high. However, in this study, we assume a SC can carry only one container at a time. Since the operational costs of SCs and the associated equipment are very high, efficient scheduling of SCs is one of the main goals for the container terminal to achieve, in order to ensure a fast and economical movement of containers. In this study, the SC functions as the vehicle to transport containers between the vessel and container yard, and also functions as a flexible moving yard crane for stacking and retrieving containers in the yard.

Lastly, we examine the storage yard and corresponding operations. At the storage yard, export containers are normally stored separately from import containers, and are stacked up to four tiers high. The storage locations of specific containers are pre-determined by the shipping agent when the containers arrive at the terminal. In addition, each storage location has a unique address represented by bay, row and tier numbers. Export containers' locations in the yard and QCs' locations (QCs have fixed positions along the ship) are known. Therefore, the travelling times of SCs from each export container's yard location to each QC are known.

The assumptions are as follows:

(1) The yard location for each export container is known. Therefore, the travelling time of SCs to deliver any container from its yard location to any of the QCs is known.

(2) The ship locations for export containers are known, i.e. where to put these export containers on the ship, which are provided by the shipping agent. Moreover, QCs are able to load containers to their predetermined ship locations. This is a different situation with the loading problem in model 2.

(3) The number of export containers, number of QCs and number of SCs are all known.

(4) Each SC can only carry one container at a time; similarly, each QC can only handle one container at a time.

(5) The handling time of each container by any QC is known, which is determined by the container's location on the ship and also by the position of the considered QC.

(6) Pooling strategy applies to the SC dispatching problem, which means that each SC can deliver containers to any of the QCs.

(7) Congestion along the SCs' travelling paths is not considered.

(8) The buffer capacity is considered to be one under each QC.

Now we introduce the notations related to this problem.

Sets and parameters

N	set of export containers
K	set of SCs
Q	set of QCs
s	index for SCs
q	index for QCs
i, j	index for export containers
m,n	index for operational sequences, $m, n \in N^+$ (positive integer)
h_{iq}	QC q's handling time of container i
z_{iq}	SC's travelling time from the yard location of container i to the buffer under QC q

M — a very large number

S — the dummy starting job (container); a job refers to the operation of an export container

F — the dummy ending job (container)

O_S — The job set which contains all the jobs including the dummy starting job,

$$O_S = N \cup \{S\}$$

O_F — The job set which contains all the jobs including the dummy ending job,

$$O_F = N \cup \{F\}$$

O — The job set which contains all the jobs including dummy starting and dummy ending jobs, $O = \{S, F\} \cup N$

Decision variables

d_i — the time a SC collects container *i* from the container's yard location

w_i — the time container *i* has been placed in the buffer by a SC

u_i — the time a QC picks up container *i* from the QC's buffer

In this study, the problems are to decide the sequences and time schedules of containers for QCs and SCs to process. Thus we introduce the related decision variables, α_{ims} and β_{inq}:

$$\alpha_{ims} = \begin{cases} 1, if\ \ i\ is\ the\ mth\ container\ delivered\ by\ SC\ s \\ 0\ , otherwise;\ \forall i \in N, \forall m \in N^+, \forall s \in K \end{cases}$$

$$\beta_{inq} = \begin{cases} 1, if\ \ i\ is\ the\ nth\ container\ handled\ by\ QC\ q \\ 0\ , otherwise;\ \forall i \in N, \forall n \in N^+, \forall q \in Q \end{cases}$$

We introduce another decision variable as follows:

$$\gamma_{iq} = \begin{cases} 1, if\ \ i\ is\ handled\ by\ QC\ q \\ 0\ , otherwise;\ \forall i \in N, \forall q \in Q \end{cases}$$

Where

$$\sum_{n \in N^+} \beta_{inq} = \gamma_{iq}, \forall i \in N, \forall q \in Q$$

Objective: Minimise the loading element of ship's berth time

$$Max\left(u_i + \sum_{q} h_{iq} r_{iq}\right)$$

Subject to:

$$\sum_{m \in N^+} \sum_{s \in K} \alpha_{ims} = 1, \forall i \in N \tag{4.2.1}$$

$$\sum_{i \in N} \alpha_{ims} \leq 1, \forall m \in N^+, \forall s \in K \tag{4.2.2}$$

$$\sum_{i \in N} \alpha_{ims} \geq \sum_{i \in N} \alpha_{im+1s}\ , \forall m \in N^+, \forall s \in K \tag{4.2.3}$$

$$\sum_{n \in N^+} \sum_{q \in Q} \beta_{inq} = 1, \forall i \in N \tag{4.2.4}$$

$$\sum_{i \in N} \beta_{inq} \leq 1, \forall n \in N^+, \forall q \in Q \tag{4.2.5}$$

$$\sum_{i \in N} \beta_{inq} \geq \sum_{i \in N} \beta_{in+1q}\ , \forall n \in N^+, \forall q \in Q \tag{4.2.6}$$

$$\sum_{q \in Q} \gamma_{iq} = 1, \forall i \in N \tag{4.2.7}$$

$$\sum_{n\in N^+} \beta_{inq} = \gamma_{iq}, \forall i \in N, \forall q \in Q \tag{4.2.8}$$

$$w_i \geq d_i + \sum_{q\in Q} z_{iq}\gamma_{iq}\,, \forall i \in N \tag{4.2.9}$$

$$d_j + M\big(2 - \alpha_{ims} - \alpha_{jm+1s}\big) \geq w_i + \sum_{q\in Q} z_{jq}\gamma_{iq}\,, \forall i \in O_S, j \in O_F, \forall m \in N^+, \forall s \in K \tag{4.2.10}$$

$$u_j + M\big(2 - \beta_{inq} - \beta_{jn+1q}\big) \geq u_i + h_{iq}\gamma_{iq}, \forall i \in O_S, j \in O_F, \forall n \in N^+, \forall q \in Q \tag{4.2.11}$$

$$w_j + M\big(2 - \beta_{inq} - \beta_{jn+1q}\big) \geq u_i, \forall i \in O_S, j \in O_F, \forall n \in N^+, \forall q \in Q \tag{4.2.12}$$

$$u_i \geq w_i, \forall i \in N \tag{4.2.13}$$

$$\alpha_{ims}, \beta_{inq}, \gamma_{iq} \in \{0,1\}, \forall i, j \in O, \forall m, n \in N^+, \forall s \in K, \forall q \in Q \tag{4.2.14}$$

$$d_i, w_i, u_i \geq 0, \forall i \in N \tag{4.2.15}$$

The objective is to minimise the loading part of the berth time of the ship, which is described as the time duration during which all the export containers have been loaded onto the ship.

Constraint (4.2.1) implies that for every container $i \in N$, it must be handled once by a SC which delivers it from the storage yard to the quayside.

Constraint (4.2.2) means that for every possible delivery of SC $s \in K$, if there is one container $i \in N$ transported in this delivery, no other container can be assigned to the same delivery, because SC can only handle one container at a time.

Constraint (4.2.3) requires that every container $i \in N$ handled by the same SC $s \in K$ is succeeded by another container.

Constraint (4.2.4) ensures that for each container $i \in N$, it must be handled once by a QC which is assigned to collect it from the buffer onto the ship.

Constraint (4.2.5) demonstrates that for each possible handling of QC $q \in Q$, if there is one container $i \in N$ handled in this operation, no other container can be handled at the same time.

Constraint (4.2.6) means that every container $i \in N$ handled by the same QC q is succeeded by another container.

Constraint (4.2.7) ensures that each container should be handled by a QC once only.

Constraint (4.2.8) gives the relationship between the two decision variables β_{inq} and γ_{iq}.

Constraint (4.2.9) means that SCs need at least a certain travelling time to take a container from the yard location to the buffer.

Constraint (4.2.10) means that the SC need at least a certain travelling time from the buffer of the QC that handles container *i* to the yard location of container *j* if container *j* is delivered after container *i*.

Constraint (4.2.11) means that if container *j* is handled after container *i* by the same QC, then the time this QC starts to handle container *j* must be after container *i* has been placed onto the ship, i.e. set at least by the handling time of container *i*.

Constraints (4.2.12) and (4.2.13) ensure that there is at most one container at the buffer during the loading process. Specifically, constraint (4.2.12) means that if *j* is handled immediately after container *i* by the same QC, then container *j* can only be placed on the buffer under the QC after container *i* has been picked up by this QC.

Constraint (4.2.13) implies that container *i* can be picked up by a QC after it has been placed in the buffer.

Constraints (4.2.14) and (4.2.15) are binary and non-negative restrictions.

4.2.2 An illustrative example

In this section, we present an example to illustrate the MIP model discussed in section 4.2.1, and show how the optimal solutions demonstrate the integrated operations of QCs and SCs. Based on the objective function and constraints considered in the last section, we present a small-sized problem to illustrate how the MIP model can be used in the real loading operations. The problem is solved by the B&B algorithm in AIMMS 3.12.

Table 4.8: The handling time of containers by each QC - the values of h_{iq}

Container *i*	QC 1 (q=1)	QC 2 (q=2)
1	145	144
2	35	120
3	115	173
4	70	102
5	144	132
6	150	43
7	116	132
8	92	81

Consider a loading problem at the straddle-carrier system. Two QCs are assigned for loading eight export containers for a ship. Before the loading operation begins, all the export containers have been stored in the yard. There are four SCs travelling between the ship and the yard for delivering containers. The layout of the operational environment is similar as in figure 4.9. The parameters at the terminal considered in our problem take the values as follows: the values of QCs' handling time h_{iq} follow

uniform distribution U (30, 180)s (see table 4.8) and the values of SCs' travelling time z_{iq} follow uniform distribution U (20, 120)s (see table 4.9).

Take an example from table 4.8: if QC 1 will be assigned to handle container 2, then it takes 35 seconds by QC 1 from its buffer to container 2's ship location; if container 2 is handled by QC 2, then it takes 120 seconds by QC 2 from its buffer to container 2's ship location.

Table 4.9: The travelling time of SCs from yard location of each container to any of the QCs - the values of z_{iq}

Container *i*	QC 1 (q=1)	QC 2 (q=2)
1	27	92
2	77	93
3	79	94
4	116	59
5	43	43
6	29	78
7	56	92
8	70	89

Let us refer to an example from table 4.9: if container 2 will be handled by QC 1, i.e. container 2 is assigned to be handled by QC 1, then it takes 27 seconds for SC to move it from its yard location to the buffer area under QC 1; however, it will take 92 seconds if container 2 is handled by QC 2.

In our example, by solving the MIP model proposed in section 4.2.1, an integrated optimal solution gives (1) the sequences of containers delivered by each SC - that is, if

the decision $\alpha_{ims} = 1$, then container *i* is the *m*th container handled by SC *s*; (2) the handling sequences of containers by each QC, i.e. if the decision $\beta_{inq} = 1$, then container *i* is the *n*th container performed by QC *q*; and (3) the handling list of each QC, i.e. if the decision $\gamma_{iq} = 1$, then container *i* is handled by QC *q*. All these values of the three decision variables are presented in table 4.10. The optimal solution for loading eight containers onto the ship is 431 seconds and the computation time is 356.45 seconds by the B&B method.

Table 4.10: Delivery sequences of containers by SCs, the handling sequences by QCs and the handling list of each QC - the values of the optimal solutions α_{ims}, β_{inq} and γ_{iq}

container	α_{ims}	β_{inq}	r_{iq}
1	$\alpha_{114} = 1$	$\beta_{111} = 1$	$\gamma_{11} = 1$
2	$\alpha_{211} = 1$	$\beta_{231} = 1$	$\gamma_{21} = 1$
3	$\alpha_{313} = 1$	$\beta_{321} = 1$	$\gamma_{31} = 1$
4	$\alpha_{423} = 1$	$\beta_{441} = 1$	$\gamma_{41} = 1$
5	$\alpha_{512} = 1$	$\beta_{512} = 1$	$\gamma_{52} = 1$
6	$\alpha_{621} = 1$	$\beta_{632} = 1$	$\gamma_{62} = 1$
7	$\alpha_{724} = 1$	$\beta_{722} = 1$	$\gamma_{72} = 1$
8	$\alpha_{834} = 1$	$\beta_{842} = 1$	$\gamma_{82} = 1$

From table 4.10, we find that SC 1 (s=1) will move container 2 first and then container 6 from the yard to the quayside; SC 2 (s=2) will only handle container 5, SC 3 will transport containers 3 and 4 in order; and SC 4 will deliver containers 1, 7 and 8 in sequence. Similarly, for the obtained handling sequences of QCs, QC 1 will handle containers 1, 3, 2 and 4 in order, and QC 2 will handle containers 5, 7, 6 and 8 in order.

Now we look at the pickup time by SCs from the yard and by QCs from the buffer, and the release time by SCs down to the buffer for each container, and the results are presented in table 4.11. The objective function value is achieved by container 8, which has been picked up by a SC from the yard at 264 seconds, then arrives at the buffer at 350 seconds; a QC then loads it onto the ship at 431 seconds.

Table 4.11: The pickup time by QCs and SCs and the release time by SCs of each container - the values of the optimal solutions d_i, $w_i$and u_i

Containers i	d_i	w_i	u_i	$u_i + \sum_q h_{iq} r_{iq}$
1	0	27	27	172
2	95	172	287	322
3	0	79	172	287
4	195	311	322	392
5	0	43	43	175
6	201	279	307	350
7	83	175	175	307
8	264	350	350	431

The above example shows that our proposed model is efficient in finding integrated solutions for handling sequences of QCs and SCs in the loading process. However, finding an exact solution is impossible for large-scale complex problems in a reasonable time with the existing software, AIMMS 3.12. Therefore, we develop a heuristic technique in the next section based on genetic algorithm to obtain the near-optimal solution within reasonable time duration.

4.2.3 Solution method: Genetic algorithm

We introduce the designed GA for this problem as in the following stages.

Container	*Dispatched SC*	*Assigned QC*
1	2	1
2	2	2
3	3	1
4	1	2
5	3	2
6	2	1
7	1	1
8	3	2

Figure 4.10: A chromosome representation for the problem with eight containers, two QCs and three SCs

Chromosome representation and initialisation: This is the initial step of the GA design. By considering the main decision variables α_{ims}, β_{inq} and γ_{iq}, we use matrix structure to present the initial solution of the problem. Chromosomes are presented by two columns: the first column represents $\alpha_{ims}$and the second column represents β_{inq}. There is no need to present γ_{iq} because the values of γ_{iq} can be obtained from the values of β_{inq}. Each row of the matrix is the chromosome for each container. Figure 4.10 shows an example of the chromosome representation with eight containers, two QCs and three SCs. In this example, container 1 will be delivered by SC 2 and handled by QC 1; container 2 will be delivered by SC 2 and loaded by QC 2 onto the ship, and so on. One possible delivery sequence for SC 2 is container 1, container 2 and container 6.

Let us denote $|N|$ is the total number of export containers, $|K|$ is the number of SCs, and $|Q|$ is the number of QCs. we construct our initial population in the following steps:

(1) SC dispatching: randomly choose a SC from 1 to $|K|$ and dispatch this number to one container; repeat it $|N|$ times until a string of length $|N|$ is generated. This satisfies constraints (4.2.1)-(4.2.3).

(2) QC assignment: randomly choose a QC from 1 to $|Q|$ and assign this number to one container, repeat it $|N|$ times until a string of length $|N|$ is generated. This satisfies constraints (4.2.4)-(4.2.8).

(3) Chromosomes are generated respectively following steps 1-2 until the population size reaches a given number (say 100) to ensure a large search space to start.

(4) Evaluate each individual by calculating the values of d_i, w_i and u_i according to constraints (4.2.9)-(4.2.15). The objective function value of the model is then obtained by the largest number of $u_i + \sum_q h_{iq} r_{iq}$.

Parents selection strategy: The binary tournament selection is used here, in which two randomly chosen individual chromosomes are compared according to their OFVs and the ones with smaller OFVs are chosen to be in the pool of parents for creating the next generation. We also adopt the elitism strategy to ensure that the best solutions are always retained in the population.

Genetic operator design: Multi-point crossover is designed here for both columns, i.e. column for 'dispatched SC' and column for 'assigned QC' in the chromosome. This crossover operator chooses multiple points (three points are used here) as the interchange points. The genes in the first and third segments are exchanged between two parents. The second operator, mutation, introduces random changes to the chromosomes by altering values in the gene according to a probability which is called the mutation rate. In our application, we create a vector with the same length of chromosomes consisting of uniformly generated values within 0 and 1. If the value is less than the specified mutation rate, we randomly change two positions in the individual chromosome.

Offspring acceptance: The semi-greedy strategy is used here to accept the offspring generated by the genetic operators: an offspring is accepted for the next generation if its OFV is less than the OFV of its parent(s).

Stopping criterion: When the number of evolving generations reaches the maximum number of generations or the standard deviation of the OFVs in the current generation is below a small value, the algorithm stops. These two criteria can reduce the computation time of the algorithm.

4.2.4 Computational results

To access the solution quality and the efficiency of the proposed algorithm, we conduct a number of experiments in this section. All the experiments were performed on an Intel® Core™ i3 CPU M370@2.40GHz and 4GB RAM under the Windows 7 operating system. Our proposed GA is implemented by using MATLAB 7.14. Each problem is run 20 times by GA with the same parameter setting, and the means of OFVs and computation times are presented. For solving the MIP formulations in small sizes, the solver, CPLEX 12.5 in AIMMS 3.12 is used, which applies B&B by default. There are three sets of experiments in total: the first set of experiments is on small-sized problems. The purpose of the experiments is to compare the differences between the results obtained from B&B and GA. The second set of experiments is on large-sized problems, for which only GA will be used to evaluate the performance of the model, because B&B cannot return any results for large sizes. The third set of experiments comprises GA parameter sweep tests, in which we test the effects of GA parameter values on the performance of the algorithm.

Parameters settings

(1) The number of containers varies from 5 to 800, where 5-20 are considered as small-sized problems and 20-800 are considered as large-sized problems. We also consider that the number of SCs varies from 2 to 20, and the number of QCs varies from 2 to 3.

(2) The uniform distribution was assumed for all the operation times. The processing times of each container by QCs follow uniform distribution U(30,

180)s, and the travelling times by each SC from each container's available yard location to each QC follows uniform distribution U(20, 120)s.

(3) GA parameters take the following settings based on preliminary tests: Crossover rate P_c: 0.8; Mutation rate P_m: 0.01; Population size *Pop*: 100; and Maximum generations M_g: 40.

Small-sized problems

Table 4.12: Comparison results of small-sized problems

No	No. of containers	QCs/ SCs	B&B		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
1	5	2/2	4.94	157	2.47	157	0%
2	6	2/2	13.57	302	2.05	302	0%
3	7	2/2	50.76	399	4.61	399	0%
4	7	2/3	65.38	309	4.78	309	0%
5	8	2/3	63.85	478	5.19	481	0.62%
6	8	2/4	260.77	395	5.92	398	0.75%
7	9	2/3	6368.58	547	5.42	552	0.91%
8	10	2/4	/	/	4.59	582	/
9	15	2/4	/	/	5.54	801	/
10	20	2/4	/	/	6.37	1054	/

Table 4.12 presents the comparison results in terms of OFVs and computation times. GA is an efficient approach in that it achieves the optimal/near-optimal results in a much shorter time. Although B&B can find better solutions, the performance of B&B shows an exponential increase regarding computation times. It can be noted that for cases from 1 to 4, both algorithms obtained the same results of OFVs. Although the results from GA have a less than 1% difference with B&B, in other cases GA is able to solve larger-sized problems and outperforms B&B with respect to computation times.

Large-sized problems

To investigate the solution approach - i.e. GA - to the problem, we test the following large-sized examples. For these examples as, in table 4.13, we set different numbers of containers, numbers of QCs and numbers of SCs in order to validate the scalability of the proposed GA. For loading a certain number of containers, generally increasing the number of QCs or SCs will lead to a positive effect on the OFVs (reduced OFVs), such as case 20 and 21 for the effect of number of QCs, case 11 and 12 for the effect of number of SCs. However, the number of equipment should be set appropriate according to the number of containers so as to minimise the traffic congestions/conflicts. Meanwhile with fixed numbers of QCs and SCs, increasing the number of containers will cause an increase in the computation times and OFVs as expected. It can be found that for the largest size in our experiments (case 42), the total computation time by GA is still within minutes. We next look at the convergence behaviour of GA; we test an example with 200 containers, three QCs and ten SCs, and the searching process of the GA is shown in figure 4. 11, which indicates the fast convergence of the proposed GA.

Table 4.13: Computational results for large-sized problems by GA

No	Containers	QCs/SCs	Computation time (s)	OFV (s)
11	30	2/3	6.92	1578
12	30	2/4	4.71	1446

13	30	2/5	6.04	1272
14	40	2/3	4.62	2043
15	40	2/4	4.32	1652
16	40	2/5	5.69	1763
17	50	2/5	6.26	2204
18	50	2/6	4.90	2260
19	50	2/7	13.65	1969
20	60	2/6	5.42	2604
21	60	3/6	10.27	1866
22	80	3/6	8.17	2530
23	80	2/6	22.15	3776
24	80	2/7	6.67	3690
25	80	2/8	7.44	3232
26	100	2/8	6.98	4438
27	100	3/8	10.16	3078
28	100	3/9	9.79	3109
29	150	3/8	19.35	4856
30	150	3/9	17.46	4674
31	150	3/10	24.86	4624

32	200	3/8	25.64	6615
33	200	3/9	48.77	6840
34	200	3/10	24.75	6641
35	300	3/10	45.42	9520
36	400	3/10	73.92	13690
37	500	3/10	87.09	17695
38	500	3/12	86.53	17306
39	500	3/15	105.62	16386
40	600	3/20	108.83	20371
41	700	3/20	176.28	23400
42	800	3/20	249.14	26982

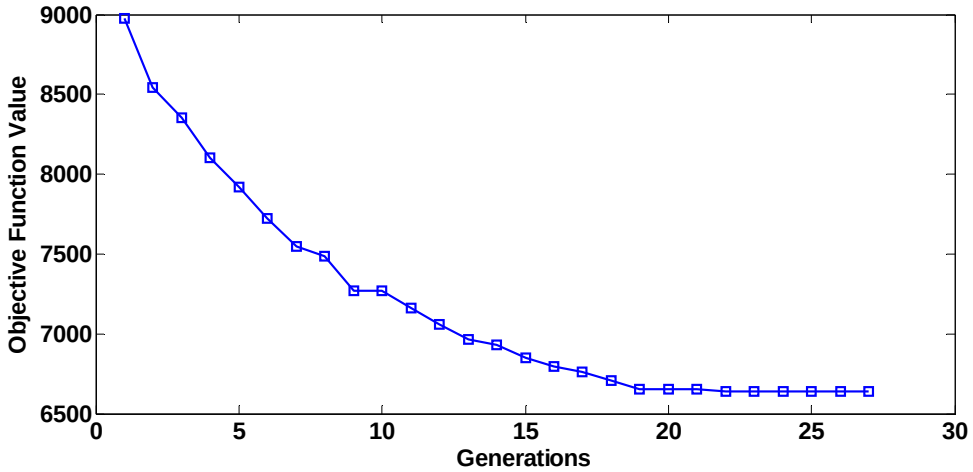


Figure 4.11: Convergence process of GA for an example with 200 containers, three QCs and 10 SCs

GA parameter sweep experiments

Crossover rates, mutation rates and population sizes are parameters which can control the performance of GA. From all the experiments above, the maximum number of

generations, 40, is sufficient for evolving a good solution. Therefore, we do not change the values of M_g in the experiments which test GA parameters. For this series of experiments, we use an example of 100 containers, three QCs and eight SCs. The crossover rates P_c take the values of 0.6, 0.7, 0.8 and 0.9; the mutation rates P_m take the values of 0.01, 0.02, 0.1 and 0.2; and the population sizes Pop take the values of 30, 50, 100, and 150 respectively. Figures 4.12-4.14 show the GA performance comparison with different values of parameters. In figure 4.12, all the curves started at the same initial solution, and the solutions found in each generation improved over the evolving process. When the algorithm stops, the curve with crossover rate of 0.9 gives a smaller OFV, which indicates that $P_c = 0.9$ is the best value for this particular case. Similarly, in figure 4.13, the curve with mutation rate of 0.2 was able to find more improvement for the initial solution in that it converged to a smaller OFV. Now we look at the effects of different population sizes. In figure 4.14, the curves start with different initial solutions. This is because this time we aim to find how randomly generated populations in different sizes can affect the performance of the proposed GA. We can note that for the case with large population sizes, it has a greater chance of starting with an initial solution with smaller OFV, since larger population size gives a range of solutions. Besides, it is noted that the curve with population size of 150 gave a smaller OFV. The differences between the best solutions with population sizes of 150 and 100 are very small; also, considering it takes a longer computation time for GA starting with a larger population of solutions, we do not test GA on a population size of more than 150.

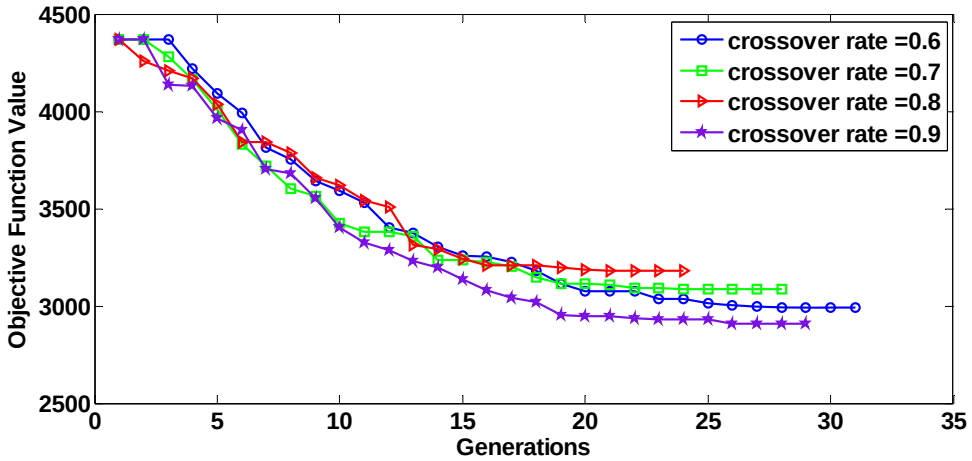


Figure 4.12: Performance comparison of different crossover rates for an example with $P_m = 0.01$ and $Pop = 100$

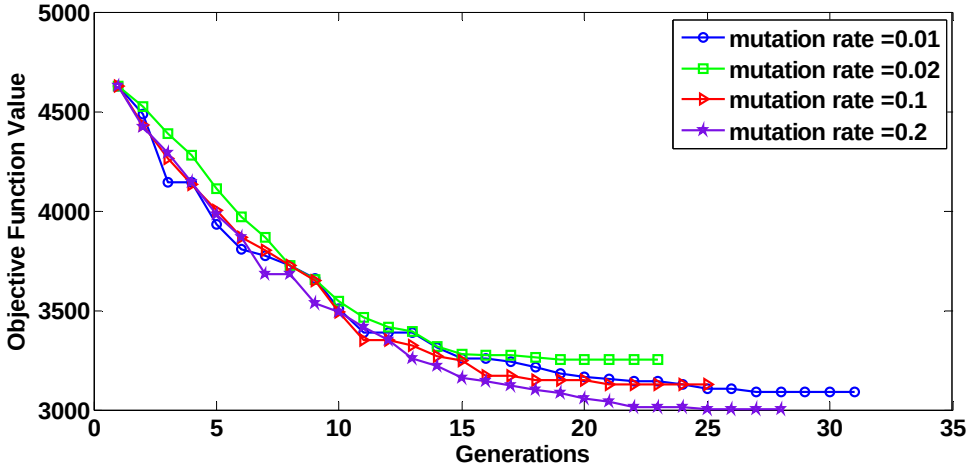


Figure 4.13: Performance comparison of different mutation rates for an example with $P_c = 0.9$ and $Pop = 100$

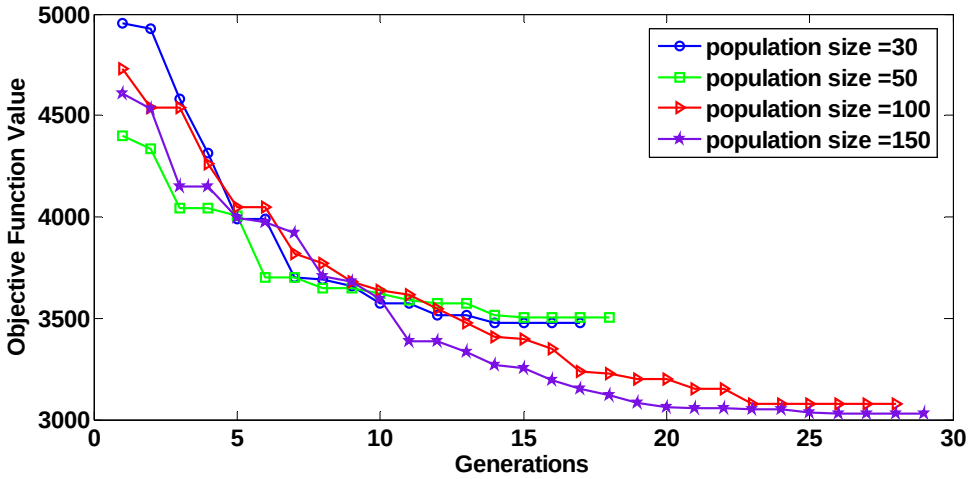


Figure 4.14: Performance comparison of different population sizes for an example with $P_c = 0.9$ and $P_m = 0.2$

Lastly, we use the best combination of GA parameters in this particular case, e.g. $P_c = 0.9$, $P_m = 0.2$ and $Pop = 150$, and run GA 10 times. The convergence processes are shown in figure 4.15. The central mark is the median, the edges of the box are the 25^{th} and 75^{th} percentiles and the whiskers are the most extreme data points. It is observed that in all the 10 runs, the algorithm converges fast and stops before 40

generations; also we could find that the variance of the OFVs in each generation is very small. This offers further proof that our proposed GA is stable and robust in all runs.

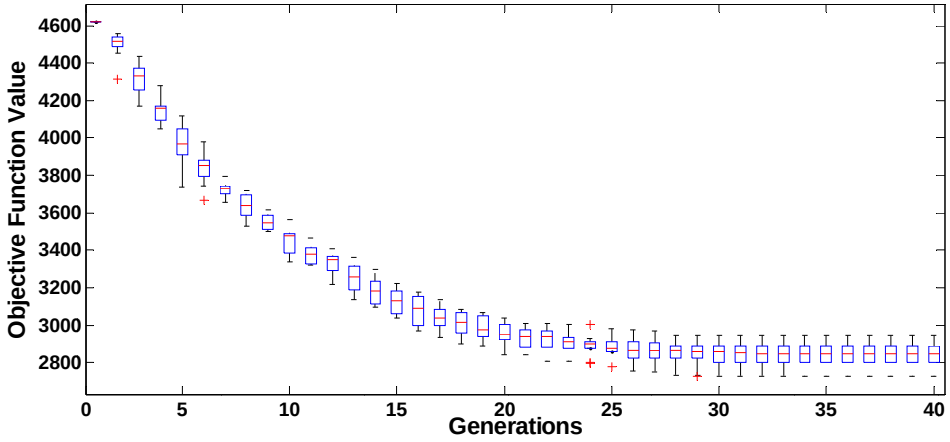


Figure 4.15: GA performance comparison in 10 runs for the case of 100 containers, three QCs and eight SCs with $P_c = 0.9$, $P_m = 0.2$ and $Pop = 150$

4.3 Model 6: Straddle carrier scheduling and storage allocation problem in dual-cycle mode

After studying the container unloading and loading processes separately in sections 4.1 and 4.2, we now consider both two processes simultaneously in this section. In contrast to traditional methods, where containers are often unloaded from the ship before any are loaded, in this section, we consider the case in which containers are loaded and unloaded simultaneously, and this has been proven to improve the port productivity. Instead of increasing terminal capacity and developing advanced information technology, this method (dual-cycle) does not require new investment in technology and infrastructure; also it is a low-cost method which can be easily implemented. Specifically, we attempt to determine the optimal schedules for SCs and to assign optimal yard locations for import containers in order to minimise the ship's turnaround time. A formulation of this integrated problem is proposed as a mixed-integer programming model. A heuristic approach based on genetic algorithm is proposed to solve this model in practical sizes.

4.3.1 Problem description and formulation

A ship's turn-around time is one of the most important values used to measure the terminal's efficiency and it consists of the times needed for unloading and loading containers. As explained earlier, in the container terminal with the straddle-carrier system, two main types of equipment are involved in the container unloading and loading processes: QCs and SCs.

Firstly we introduce the special characteristics of the straddle-carrier system and the dual-cycle operational strategy in the container terminal. As mentioned in earlier sections, there is a special characteristic in the layout of the terminal with the straddle-carrier system: a buffer area under each QC for accommodating containers must be considered. In fact, during the unloading process, QCs must put containers down on the ground for SCs to pick up, which is similar with the loading process. However, the capacity of this buffer area is limited due to container re-handling and safety issues. In this section, again, we assume that, at most, one container is allowed at the buffer area for each QC during the handling process. The same assumption has been made in the work of Wong and Kozan (2010).

Secondly, we refer to the dual-cycle mode considered in our study. Similar to the case in section 3.3, we consider unloading and loading processes simultaneously which is defined as the dual-cycle mode in our problem. Currently most of the terminals adopt the single-cycle operation strategy, in which the loading process starts after the unloading process has been finished. However, dual-cycle operations could greatly improve the performance of the container terminal because single-cycle operations generate more empty moves of both QCs and SCs. Figure 4.16 compares the differences between these two operational strategies. A solid line means the loaded moves of QCs and SCs while a dashed line refers to empty-loaded moves of QCs and SCs. For the unloading process in the single-cycle method, a QC unloads an import container, and places it down in the buffer area under this QC before this container has been collected by a SC. This QC then moves on to handle the next import container which is to be discharged from the ship. For the transportation process of a SC, it takes a container to the yard and stacks it in its location before returning to the quayside without carrying any other containers. The loading process from the storage yard to the quayside in the

single-cycle operation is in the reverse order of the unloading process (see figure 4.16 (1)). However, the dual-cycle operations allow both QCs and SCs to perform the unloading of containers in the same cycle as loading containers (see figure 4.16 (2)). Therefore, dual-cycle operations can increase the efficiency of both SCs' transportation and QCs' operation and observably reduce the empty-loaded moves of both types of handling equipment.

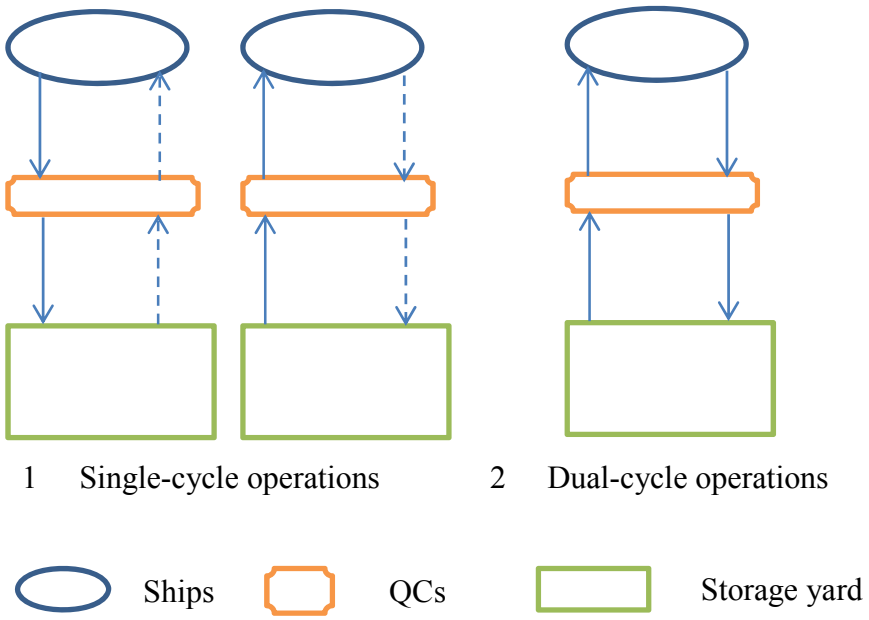


Figure 4.16: Single-cycle and dual-cycle operational strategies in container terminals

In such a situation that considers both unloading and loading operations, there are four transportation conditions for any two consecutive containers delivered by the same SC:

Condition 1: the SC transfers an import container after another import container.

Condition 2: the SC delivers an import container after an export container.

Condition 3: the SC transports an export container after another export container.

Condition 4: the SC moves an export container after an import container.

Thirdly, we introduce the environment of the storage yard. The storage yard consists of blocks, which are represented by bays, rows and tiers. Figure 4.17 shows the structure of a container yard in the straddle-carrier system. In this study, the reshuffle problem is not considered: we assume all the export containers are located on the top of the stacks and all the import containers will be placed on the top of the stacks.

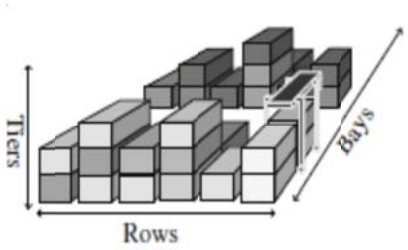


Figure 4.17: Yard block structure under the straddle-carrier system (source: Wiese, Suhl and Kliewer (2010))

The problems considered in this section are: (1) the SC scheduling problem. Specifically, there are a set of containers and a set of SCs. Each container must be processed by a SC once and each SC can handle one container at a time. The solution to this problem, which is the SC's schedule, defines the sequences and the start times for every container handled by the set of SCs; and (2) the storage allocation problem for import containers. We do not consider the storage problems for export containers. This is because the information of export containers in the yard is provided by the yard map, which shows the locations for containers as well as the information on the type, destination, weight and size of each export container. However, the locations for import containers are unknown and taken as a decision variable. The decision is to determine where to locate these import containers in the yard. Therefore, one of the main contributions of this study is that it is the first to formulate the integration of SC scheduling and container storage problems by considering both unloading and loading processes, and to provide solution methods to solve it.

We summarise the assumptions made in this study:

- Both loading and unloading processes are considered simultaneously.
- Travelling time between any two QCs are known, assuming QCs are not moving along the quayside during the unloading and loading processes.

- There is at most one container at the buffer area at any time during the operations.
- The sequences for each QC to handle both import and export containers are known.
- Number of containers, number of SCs and number of QCs are all known.
- A SC can only deliver one container at a time, and a QC can only handle one container at a time.
- SCs are shared among all the QCs. In other words, SCs can deliver containers to any of the QCs.
- The travelling times between export containers' yard locations to any of the QCs are known. This is because the locations of export containers and the locations of QCs are known.
- Yard locations for all the export containers are known; therefore the SCs' travelling times between any two export containers in the yard are known.
- Container reshuffles in the yard are not included in this study. All the containers to be processed are either on the top or will be placed on the top of the stacks.
- The pickup/drop-off times of containers by QCs and SCs are negligible.
- Traffic congestion on the road of SCs is not considered.

Similar with model 3, this model (model 6) considers the integration of vehicle scheduling and storage allocation problem with the unloading and loading processes simultaneously. By contrast, model 4 only investigated the integrated problem in the unloading process; and model 5 studied the loading process with different assumptions. In the mathematical model the following sets and parameters are defined:

K	set of QCs
C	Set of SCs

N	set of containers
L	set of export containers
D	set of import containers
M'	a very large number
M	the set of locations for import containers
(S, I)	the dummy starting job (container)
(F, I)	the dummy ending job (container)
O_S	The job set which contains all the jobs including the dummy starting job, i.e. $$O_S = N \cup (S, I)$$
O_F	The job set which contains all the jobs including the dummy ending job, i.e. $$O_F = N \cup (F, I)$$
O	The job set which contains all the jobs including dummy starting and ending jobs, i.e. $$O = \{(S, I), (F, I)\} \cup N$$
k, l	index for QCs
$(i, k), (j, l)$	index for containers, container *(i, k)* means the *i*th container handled by QC *k*
m	index for yard locations, $m \in N^+$ positive integer
c	The total number of SCs
$h_{(i,k)}$	QC *k*'s handling time of container (i, k)
$t_{(i,k)}$	The SC travel time for each export container *(i, k)* from its yard

location to its QC k

τ_k^m The SC travel time between any QC k to any available yard location m

$\theta_{(i,k)}^l$ The SC travel time for each export container *(i, k)* from its yard location to any QC l

$\sigma_{(i,k)}^m$ The SC travel time for each export container (i,k)between its yard location and any of the available location m

π_k^l The SC travel time between any two QCs, i.e. QC k and QC l

The decision variables are as follows:

$u_{(i,k)}$ the time QC k starts handling container (*i, k*); for import containers, this corresponds to the time QC k picks it up from the ship; for export containers, it corresponds to the time QC k picks it up from the buffer

$w_{(i,k)}$ the time container (i,k) is at the buffer. For import containers, it corresponds to the time a SC picks up container *(i, k)* from the buffer; for export containers, it corresponds to the time a SC sets container *(i, k)* down on the buffer

$v_{(i,k)}$ the time container (i,k) is in the yard. For import containers, it corresponds to the time container *(i, k)* has been placed into its yard location by a SC; for export containers, it corresponds to the time container *(i, k)* has been picked up from its yard location by a SC

The decisions of SC schedules and storage allocation can be represented using the following two decision variables:

$$x_{(i,k)}^{(j,l)} = \begin{cases} 1, \text{if container } (j,l) \text{ is delivered immediatly after container (i, k) by} \\ \text{the same SC} \\ 0, \text{otherwise } \forall (i,k) \in O_S, \forall (j,l) \in O_F \end{cases}$$

$$y_{(i,k)}^{m} = \begin{cases} 1, \text{if import container } (i,k) \text{will be located in location } m \\ 0, \text{otherwise } \forall (i,k) \in D, \forall m \in M \end{cases}$$

Objective: Minimise berth time $\max(u_{(N_k,k)} + h_{(N_k,k)})$

Subject to:

$$\sum_{(j,l)\in O_F} x_{(i,k)}^{(j,l)} = 1, \forall (i,k) \in N \tag{4.3.1}$$

$$\sum_{(i,k)\in O_S} x_{(i,k)}^{(j,l)} = 1, \forall (j,l) \in N \tag{4.3.2}$$

$$\sum_{(j,l)\in D} x_{(S,I)}^{(j,l)} = c \tag{4.3.3}$$

$$\sum_{(i,k)\in D} x_{(i,k)}^{(F,I)} = c \tag{4.3.4}$$

$$\sum_{m\in M} y_{(i,k)}^{m} = 1, \forall (i,k) \in D \tag{4.3.5}$$

$$\sum_{(i,k)\in D} y_{(i,k)}^{m} \leq 1, \forall m \in M \tag{4.3.6}$$

$$v_{(i,k)} + t_{(i,k)} \leq w_{(i,k)}, \forall (i,k) \in L \tag{4.3.7}$$

$$w_{(i,k)} \leq u_{(i,k)}, \forall (i,k) \in L \tag{4.3.8}$$

$$w_{(i,k)} + \sum_{m\in M} \tau_k^m \, y_{(i,k)}^{m} = v_{(i,k)}, \forall (i,k) \in D, \forall k \in K \tag{4.3.9}$$

$$u_{(i,k)} + h_{(i,k)} \leq w_{(i,k)}, \forall (i,k) \in D \tag{4.3.10}$$

$$u_{(i+1,k)} - u_{(i,k)} \geq h_{(i,k)}, \forall (i+1,k), (i,k) \in N, \tag{4.3.11}$$

$$i = 1,2, \dots, N_k - 1, \forall k \in K$$

$$w_{(i,k)} + \theta^{k}_{(j,l)} \leq v_{(j,l)} + M'(1 - x^{(j,l)}_{(i,k)}), \forall (i,k) \in L \cup (S,I), \forall (j,l) \in L \cup (F,I), \forall k \in K \quad (4.3.12)$$

$$v_{(i,k)} + \sum_{m} \tau^{m}_{l} y^{m}_{(i,k)} \leq w_{(j,l)} + M'\left(1 - x^{(j,l)}_{(i,k)}\right), \forall (i,k) \in D \cup (S,I), \forall (j,l) \in D \cup (F,I), \forall l \in K \quad (4.3.13)$$

$$w_{(i,k)} + \pi^{l}_{k} \leq w_{(j,l)} + M'\left(1 - x^{(j,l)}_{(i,k)}\right), \forall (i,k) \in L \cup (S,I), \forall (j,l) \in D \cup (F,I), \forall k, l \in K \quad (4.3.14)$$

$$v_{(i,k)} + \sum_{m \in M} \sigma^{m}_{(j,l)} y^{m}_{(i,k)} \leq v_{(j,l)} + M'\left(1 - x^{(j,l)}_{(i,k)}\right), \forall (i,k) \in D \cup (S,I), \forall (j,l) \in L \cup (F,I) \quad (4.3.15)$$

$$u_{(i,k)} \leq w_{(i+1,k)}, \forall (i+1,k), (i,k) \in L,\ i = 1,2,\dots,N_k - 1, \forall k \in K \quad (4.3.16)$$

$$w_{(i,k)} \leq u_{(i+1,k)} + h_{(i+1,k)}, \forall (i+1,k), (i,k) \in D, i = 1,2,\dots,N_k - 1, \forall k \in K \quad (4.3.17)$$

$$u_{(i,k)} + h_{(i,k)} \leq u_{(i+1,k)}, \forall (i+1,k) \in D, (i,k) \in L, i = 1,2,\dots,N_k - 1, \forall k \in K \quad (4.3.18)$$

$$w_{(i,k)} \leq w_{(i+1,k)}, \forall (i+1,k) \in L, (i,k) \in D \quad , \quad i = 1,2,\dots,N_k - 1, \forall k \in K \quad (4.3.19)$$

$$x^{(j,l)}_{(i,k)}, y^{m}_{(i,k)} \in \{0,1\}, \forall (i,k), (j,l) \in O, \forall m \in M \quad (4.3.20)$$

$$u_{(i,k)}, w_{(i,k)}, v_{(i,k)} \geq 0, \forall (i,k) \in N \quad (4.3.21)$$

The objective is to minimise the makespan of unloading and loading containers at the quayside, i.e. ship's berth time. It provides a way to calculate the completion time for all the QCs.

Constraints (4.3.1)-(4.3.6) are resource constraints.

Constraint (4.3.1) implies that for every container $(i,k) \in N$ there is one container $(j,l) \in N$ handled after it by the same SC.

Constraint (4.3.2) represents that for every container $(j,l) \in O_F$, there is one container $(i,k) \in O_S$ delivered before it by the same SC.

Constraint (4.3.3) and (4.3.4) guarantee that the total number of SCs that are employed for unloading and loading containers is exactly c.

Constraint (4.3.5) ensures that every import container $(i,k) \in D$ will be located in exactly one location m after the unloading process.

Constraint (4.3.6) means that every available location $m \in M$ can only accommodate, at most, one container.

Constraint (4.3.7) states that the time every export container has been placed in the buffer and the time it has been picked up from the yard by a SC must be set at least by a certain travel time between the yard and the quayside.

Constraint (4.3.8) guarantees that each export container will be loaded by a QC after it has been placed onto the buffer.

Constraint (4.3.9) means that the time of QC k between picking up any import containers by a SC at the buffer and the time SC placing it onto the container's assigned location must be set at least by a certain travelling time between QC k and location m.

Constraint (4.3.10) means that an import container can only be picked up from the buffer after the QC has released it to the buffer.

Constraint (4.3.11) ensures that the times for two containers *(i+1, k)* and *(i, k)* handled by the same QC k must be set at least by a certain handling time of container *(i, k)*.

Constraints (4.3.12)- (4.3.15) give the time relationships for delivering two successive containers by the same SC in the four transportation conditions.

Specifically, if container (j,l) is delivered immediately after container $(i,k)$by the same SC, constraint (4.3.12) means that if both $(j,l)$and $(i,k)$are export containers, then the

time between picking up container (j, l)in the yard and putting container(i, k) in the buffer must be set at least a certain travelling time of SC from QC k to the yard location of (j, l).

Constraint (4.3.13) means that if both containers $(j, l)$and $(i, k)$are import containers, then the time between releasing container (i, k) in the storage yard and picking up container (j, l) in the buffer must be set by at least a certain travelling time from container(i, k)'s assigned location to QC l.

Constraint (4.3.14) means that if container (i, k) is an export container and container (j, l) is an import container, then time between taking container (i, k) down to the buffer and picking up container (j, l) must be set at least by a certain travelling time from QC k to QC l.

Constraint (4.3.15) means that if container (i, k) is an import container and container (j, l) is an export container, then the time between placing container (i, k) into its assigned location and picking up container (j, l) must be set at least by a certain travelling time from the assigned location m of container (i, k) to the yard location of (j, l).

Constraints (4.3.16)- (4.3.19) are for the time relationship for handling two successive containers by the same QC for the four conditions to ensure there is, at most, one container at the buffer at any time. Constraint (4.3.16) implies that if container $(i + 1, k)$ and container (i, k) are both export containers, then container $(i + 1, k)$ can be placed on the buffer by a SC after container (i, k) has been picked up by QC k.

Constraint (4.3.17) implies that if container $(i + 1, k)$ and container (i, k) are both import containers, then container $(i + 1, k)$ can be placed on the buffer after container (i, k) has been picked up by a SC.

Constraint (4.3.18) implies that if container $(i + 1, k)$ is an import container and container (i, k) is an export container, then the starting time to handle container $(i + 1, k)$ must be after container (i, k) has been place in its ship location.

Constraint (4.3.19) implies that if container $(i+1,k)$ is an export container and container (i,k) is an import container, then the time in which container $(i+1,k)$ can be placed on the buffer must be set after the time that container (i,k) has been picked up by a SC.

Constraint (4.3.20) and (4.3.21) are binary and non-negative constraints.

4.3.2 An illustrative example

In this section, we provide a computational example for evaluating the model introduced in section 4.3.1. It is solved by AIMMS 3.12, combined with CPLEX12.5 as its optimisation tool, which in turn uses the branch and bound (B&B) method to solve the problem.

The operating environment considered in the computational experiments is as follows: in our computational analysis, the layout of the container terminal adopted in this analysis is similar to the one shown in figure 4.1. At the quayside, two QCs are assigned for unloading and loading containers for a ship. At the storage yard, the locations of all the export containers are known because export containers usually arrive at the terminal seven days before they will be loaded onto the ship. There are three straddle carriers (SCs) carrying containers between the quayside and storage yard. Noted that the QCs and SCs are not one-to-one assigned due to the pooling strategy which allows SCs to serve any QCs during the unloading and loading processes. To efficiently illustrate how the optimal solution works, we present an example with eight containers, from which containers (1, 1), (2, 1), (3, 1) and (4, 1) are handled by QC 1 and containers (1, 2), (2, 2), (3, 2) and (4, 2) are handled by QC 2. Particularly, containers (1, 1), (4, 1), (2, 2) and (3, 2) are export containers, and the others are import containers. The values of parameters used for the problem reflect the real container terminal.

First, in table 4.14 we present the QCs' handling times for all the containers. The handling time of each container highly depends on its ship location. For example, it takes QC 1 130 seconds to load container (1, 1) onto the ship and then it takes 152 seconds to unload container (2, 1) from its ship location onto the buffer.

Table 4.14: The handling sequences and handling times of each container by QCs - the values of $h_{(i,k)}$

Containers (QC 1)	$h_{(i,k)}$ (sec)	Containers (QC 2)	$h_{(i,k)}$ (sec)
(1, 1)	130	**(1, 2)**	133
(2, 1)	152	**(2, 2)**	124
(3, 1)	139	**(3, 2)**	88
(4, 1)	72	**(4, 2)**	142

Secondly, we list the SCs' travelling time for each export container from its yard location to its destination QC in table 4.15. For example, SC takes 66 seconds to deliver export container (1, 1) to its destination QC 1, and it takes 39 seconds to transport export container (2, 2) from its yard location to QC 2.

Table 4.15: The SC travelling time for each export container from yard location to its QC - the values of $t_{(i,k)}$

Container *(i, k)*	$t_{(i,k)}$ (sec)
(1, 1)	66
(4, 1)	109
(2, 2)	39
(3, 2)	94

Table 4.16: The SCs' travelling time between each export container's yard location and each available location *m* - the values of $\sigma^{m}_{(i,k)}$

Location *m*	(1, 1)	(4, 1)	(2, 2)	(3, 2)
1	68	57	54	41
2	39	66	41	58
3	63	42	38	36
4	38	61	67	49
5	63	57	46	60
6	42	74	68	79

Thirdly, we consider the SCs' travelling time between each export container's yard location and each available location for import containers, assuming there are six available locations in the yard for storing four import containers, i.e. $m \in \{1,2,3,4,5,6\}$. These are the values of $\sigma^{m}_{(i,k)}, \forall (i,k) \in L, \forall m \in M,$ which are shown in table 4.16. For example, if an import container is to be located in location 3, then it takes 63 seconds between location 3 and the location of export container (1, 1).

The SCs' travelling time from any QC to each of these available yard locations for import containers are known and these values τ^{m}_{k} are presented in table 4.17. For example, it takes a SC 71 seconds to travel from location 2 in the yard to QC 1.

Table 4.17: The SCs' travelling time between any QC to any of the available yard locations - the values of τ_k^m

Locations *m*	QC 1 (*k=1*)	QC 2 (*k=2*)
1	50	115
2	71	102
3	70	84
4	29	110
5	89	30
6	57	88

Table 4.18: The SCs' travelling time between any export container's yard location and any QCs - the values of $\theta_{(i,k)}^l$

Containers *(i, k)*	QC 1 (*l=1*)	QC 2 (*l=2*)
(1, 1)	66	174
(4, 1)	109	179
(2, 2)	86	39
(3, 2)	148	94

Now we look at the SCs' travelling time for each export container between its yard location and any of the QCs. These values of $\theta_{(i,k)}^l$ are given in table 4.18. For example, if a SC delivers a container to QC 2 and then moves to pick up export container (4, 1), the travelling time is 179 seconds. Lastly, we set SCs' travelling time between these two QCs as 20 seconds, i.e. $\pi_1^2 = 20$.

By solving this example, we obtain the berth time for loading four export containers and unloading four import containers as 676 seconds by two QCs and three SCs. First, we have the assigned locations for the four import containers, which are shown in table 4.19. For example, import container (2, 1) will be assigned to location 4 in the yard after the unloading process, due to $y^4_{(2,1)} = 1$. Then, let us look at the schedules of SCs, which are shown in table 4.20. For example, SC 3 takes import container (1, 2) to the yard, then picks up export container (2, 2) from the yard; after delivering container (2, 2) to QC 2, SC 3 returns to the yard to collect export container (3, 2).

Table 4.19: The assigned locations for import containers - the optimal decisions of $y^m_{(i,k)}$

Containers (i, k)	$y^m_{(i,k)}$
(2, 1)	$y^4_{(2,1)} = 1$
(3, 1)	$y^1_{(3,1)} = 1$
(1, 2)	$y^5_{(1,2)} = 1$
(4, 2)	$y^3_{(4,2)} = 1$

Table 4.20: The delivery sequences of SCs - the optimal decisions of $x^{(j,l)}_{(i,k)}$

SCs	Containers	$x^{(j,l)}_{(i,k)}$
SC 1	(1, 1)	$x^{(2,1)}_{(1,1)} = 1$
	(2, 1)	$x^{(3,1)}_{(2,1)} = 1$
	(3, 1)	$x^{(4,2)}_{(3,1)} = 1$
	(4, 2)	

SC 2	(4, 1)	$x_{(S,I)}^{(4,1)} = 1$ $x_{(4,1)}^{(F,I)} = 1$
SC 3	(1, 2) (2, 2) (3, 2)	$x_{(1,2)}^{(2,2)} = 1$ $x_{(2,2)}^{(3,2)} = 1$

Table 4.21: The values of the decision variables: (1) the starting and finishing times of each container delivered by SCs-values of $v_{(i,k)}$ and $w_{(i,k)}$, (2) the starting time to handle each container by QCs- the values of $u_{(i,k)}$ and (3) the finish time of each container at the quayside $u_{(i,k)} + h_{(i,k)}$

Containers (i, k)	$u_{(i,k)}$ (sec)	$w_{(i,k)}$ (sec)	$v_{(i,k)}$ (sec)	$u_{(i,k)} + h_{(i,k)}$ (sec)
(1, 1)	66	66	0	196
(2, 1)	196	348	377	348
(3, 1)	348	487	538	487
(4, 1)	602	487	387	674
(1, 2)	0	133	163	133
(2, 2)	203	203	163	327
(3, 2)	446	446	351	534
(4, 2)	534	676	761	676

We can also obtain the times of each container at the yard and quayside which are listed in table 4.21. Thus the schedules of SCs are determined. For example, export container

(4, 1) will be picked up by a SC from its yard location at 387 seconds, then it will be set down at the buffer at 487 seconds; lastly QC 1 will pick it up at 602 seconds

This problem is known to be NP-hard; thus its computational complexity increases exponentially with the problem size getting larger, i.e. the number of containers increases. This makes it difficult to solve the problem within a reasonable time using the exact solution technique (i.e. B&B algorithm); therefore we develop a heuristic method in the next section to deal with this computational difficulty.

4.3.3 Solution method: genetic algorithm

The container terminal operation scheduling problem is the NP-hard problem, and a heuristic algorithm can be used in order to obtain an (near) optimal solution of the proposed mathematical model in a reasonable computation time. we design GA for this problem. By this means, we want to obtain the best delivery sequences of SCs and the best assigned locations for import containers such that the berth time is minimised.

Chromosome representation and initialisation: Considering the decision variables $x_{(i,k)}^{(j,l)}$ and $y_{(i,k)}^{m}$, i.e. the travelling sequences of SCs and the assigned locations for import containers, we construct our initial solutions as follows: Let us denote $|D|$ as the total number of import containers, $|N|$ as the total number of containers, $|M|$ as the total number of available locations and c as the total number of SCs.

(1) SC assignment: randomly choose a SC from 1 to c (constraints (4.3.3) and (4.3.4) and assign this number to one container; repeat this $|N|$ times until a string of length of $|N|$ is generated. So that constraints (4.3.1) and (4.3.2) are met.

(2) Location assignment: since the locations are uniformly distributed, we first randomly choose $|D|$ locations from 1 to $|M|$; then assign a series of numbered labels from 1 to$|D|$ to these selected locations; now we can assign these locations to $|D|$ import containers. For each import container, choose a location from 1 to $|D|$.So that constraints (4.3.5) and (4.3.6) are met.

(3) Chromosomes are generated respectively following steps 1-2 until the population size reaches a given number (say 100) to ensure a large search space.

(4) Evaluate each individual in the initial population by calculating the objective function value (OFV), according to constraint (4.3.9)-(4.3.21).

Let us use an example as in section 4.3.2, where there are eight containers, from which containers (1, 1), (2, 1), (3, 1) and (4, 1) are handled by QC 1 and containers (1, 2), (2, 2), (3, 2) and (4, 2) are handled by QC 2. Particularly, containers (2, 1), (3, 1), (1, 2) and (4, 2) are import containers; the others are export containers. Figures 4.18 and 4.19 show a chromosome representation of this example; for example, in figure 4.18, import container (2, 1) and export container (2, 2) will be handled by SC 1 in sequence; and in figure 4.19, import container (2, 1) will be assigned to location 2 and container (4, 2) will be assigned to location 4.

Containers	*Dispatched SCs*
(1, 1)	**2**
(2, 1)	**1**
(3, 1)	**3**
(4, 1)	**2**
(1, 2)	**3**
(2, 2)	**1**
(3, 2)	**2**
(4, 2)	**2**

Figure 4.18: An initial solution of the dispatched SCs for an example with eight containers

Containers	*Assigned locations*
(2, 1)	2
(3, 1)	3
(1, 2)	1
(4, 2)	4

Figure 4.19: An initial solution of the assigned locations for an example with four import containers

Genetic operators design: (1) crossover: similar to section 4.1.3, we use uniform crossover for the chromosome of 'dispatched SCs' and use uniform order-based crossover for the chromosome of 'assigned locations' to avoid missing and conflict genes. (2) Mutation: it provides individual diversity during the search process, and it is controlled by a mutation rate. Swap mutation is adopted here in which two randomly chosen genes are exchanged for both two parts of the chromosome.

Offspring acceptance: The semi-greedy strategy is used here to accept the offspring generated by the genetic operators: an offspring is accepted for the next generation if its fitness is better than the average fitness of its parent(s). This strategy is able to reduce the computation time of the proposed algorithm and makes a monotonous convergence toward the best solution.

Parents selection strategy: The selection mechanism used here is binary tournament selection, in which two randomly chosen individuals are compared in terms of their fitness values, and the one with higher fitness will be selected as one parent for generating offspring in the next generation. In addition, the best individual in each generation is always chosen as parent for next generation.

Stopping criterion: When the number of evolving generations reaches the defined maximum number of generations or the standard deviation of the OFVs in the current generation is below a small value, the algorithm stops. These two criteria can decrease the computational time.

4.3.4 Computational results

To assess the efficiency of the genetic algorithm and the proposed integrated approach, we conduct a series of experiments through a comparison study between the B&B and GA to determine the best result for different sizes of problems. All the experiments were performed on an Intel® Core™ i3 CPU M370@2.40GHz and 4GB RAM under the Windows 7 operating system. Our proposed GA is implemented using MATLAB 7.14; and for solving the MIP formulations in small sizes, the solver CPLEX 12.5 by AIMMS 3.12 is used. For each example, we examined it in GA by 20 runs using the same set of parameters, and the means of OFVs and computation times are presented.

Parameters settings

(1) The number of containers varies from 5 to 300, where 5-20 are considered as small-sized problems and 20-300 are considered as large-sized problems. We also consider the number of SCs varies from 2 to 10, and the number of QCs varies from 2 to 3.

(2) The uniform distribution was assumed for all the operational times. The handling times of each QC on these containers follow uniform distribution U(30, 180)s, and the travelling times of SCs between each QC and each container's available location follow uniform distribution U(40, 300)s.

(3) GA parameters take the following settings based on preliminary tests: Crossover rate P_c: 0.8; Mutation rate P_m: 0.01; Population size Pop: 100; and Maximum number of generations M_g: 40.

Small-sized problems

We examine 10 problems to compare the performances of the proposed GA with the B&B algorithm in small sizes. Table 4.22 provides comparison results for B&B and GA for the various small-sized problems.

Table 4.22: Comparison results of B&B and GA in small sizes

No	No. of containers	QCs/ SCs	B&B		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
1	5	2/2	0.03	286	0.12	286	0%
2	6	2/2	0.17	502	0.21	502	0%
3	7	2/2	0.03	519	0.27	519	0%
4	8	2/3	0.02	538	0.36	544	0%
5	9	2/3	1.08	620	1.74	629	1.45%
6	10	2/3	1.03	742	1.08	754	1.62%
7	10	2/4	1.08	715	1.55	730	2.09%
8	15	2/4	2.47	1143	1.92	1162	1.66%
9	20	2/3	2.72	1647	2.64	1694	2.85%
10	20	2/4	4.77	1754	2.58	1797	2.45%

In order to compare the two algorithms, we consider the computation time and objective function value (OFV) as the measures of efficiency and effectiveness. In each problem, the related values are listed. In order to determine whether there is a significant difference among the performance of the two algorithms, an OFV gap (in percentage for each example) is calculated. These results indicate that generally, for small-sized problems, B&B outperforms GA in obtaining better OFV in a shorter computation time. However, the average gap between GA and B&B is quite small with the average gap of 1.21%, from which we are convinced about the validity of our proposed GA.

Large-sized problems

To further demonstrate the benefits of our proposed GA, we compare GA with B&B in solving the problems in large sizes; we conduct the following experiments and present the results in table 4.23. It is observed that GA provides much more stable solutions in

that it is able to solve problems with more than 100 containers which B&B cannot achieve. In addition, the convergence time of GA is less than that of B&B, which indicates that the proposed integrated problem is difficult to solve within a limited time by exact algorithms. However, when container number is small, computation time by B&B is less than that of GA; it shows an exponential increase in the computation time by B&B when the container number increases. As shown in table 4.23, B&B could only obtain the optimal solutions from cases 11 to 25. However, the average of the relative gap between GA and the best solution obtained by B&B in terms of OFVs for these 15 cases is about 2.6 %, which is a promising result. It is observed from this table that the OFV is reduced as the number of QCs is increased (for example case 20 and case 21). This trend is the same when the number of SCs is increased (for example case 21 and case 22). As discussed in previous sections, there is a trade-off between the improvement of OFV and the increased number of equipment. Further analysis has been performed to see the evolving process of our proposed GA as shown in figure 4.20. This figure shows the typical GA convergence performance for an example with 100 containers, two QCs and eight SCs. It can be seen that the evolving process converges fast and achieves the near-optimal solution before 30 generations. These results clearly indicate that our proposed GA can obtain better-quality solutions with much shorter computing time compared with B&B.

Table 4.23: Comparison results of B&B and GA in large sizes

No	No. of containers	QCs/ SCs	B&B		GA		OFV Gap rate (%)
			Computation time (s)	OFV (s)	Computation time (s)	OFV (s)	
11	30	2/3	8.63	2591	21.19	2629	1.46%
12	30	2/4	13.04	1963	40.71	1988	1.27%
13	30	2/5	12.03	1872	20.97	1898	1.38%
14	40	2/3	9.61	3309	72.51	3374	1.96%
15	40	2/4	10.33	2891	51.32	2988	3.35%
16	40	2/5	12.42	2413	112.11	2441	1.16%
17	50	2/5	19.50	3243	78.18	3316	2.25%

18	50	2/6	165.27	3027	104.56	3093	2.18%
19	50	2/7	113.15	2898	135.23	2938	1.38%
20	60	2/6	124.76	3607	179.82	3721	3.16%
21	60	3/6	227.35	3013	122.90	3105	3.05%
22	60	3/7	334.78	2871	98.63	2939	2.36%
23	80	2/6	1163.52	5106	245.14	5182	1.48%
24	80	2/7	1245.81	4992	192.74	5102	2.20%
25	80	3/7	2667.93	3897	272.01	3991	2.41%
26	100	2/6	/	/	414.68	6553	/
27	100	2/7	/	/	255.04	6491	/
28	100	2/8	/	/	301.70	5503	/
29	150	2/7	/	/	966.14	9383	/
30	150	3/7	/	/	970.27	7280	/
31	150	3/8	/	/	810.10	7253	/
32	200	3/8	/	/	1004.10	9925	/
33	200	3/9	/	/	881.22	9501	/
34	200	3/10	/	/	1574.70	9209	/
35	300	3/8	/	/	2013.73	16003	/
36	300	3/9	/	/	1964.01	15852	/
37	300	3/10	/	/	2316.28	15074	/

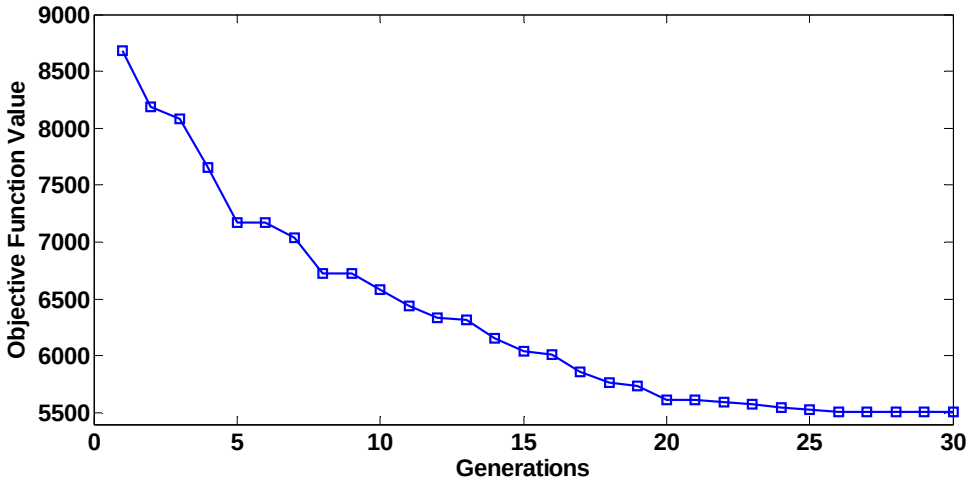


Figure 4.20: GA convergence process for an example with 100 containers, two QCs and eight SCs

GA parameter sweep experiments

Experiments in this section concentrate on testing GA parameters in order to verify and validate the GA performance. We consider the problem with 60 containers, three QCs and seven SCs. The crossover rates P_c take the values of 0.6, 0.7, 0.8 and 0.9; the mutation rates P_m take the values of 0.01, 0.02, 0.1, and 0.2; and the population sizes Pop take the values of 30, 50, 100, and 150, respectively. According to previous tests, the maximum number of generations of 40 is sufficient for GA to converge a best solution. Therefore, we do not change the values of M_G. Figures 4.21-4.23 show the GA convergence behaviours with respect to different crossover rates, mutation rates and population sizes. In consideration of these results, we can conclude that in figure 4.21, the curve with $P_c = 0.7$ outperforms others in achieving a smaller OFV; in figure 4.22, the one with $P_m = 0.02$ outperforms others as it requires less evolving generations and reaches a smaller OFV; and in figure 4.23, the one with $Pop = 100$ outperforms others due to its fast convergence and achieved lower OFV.

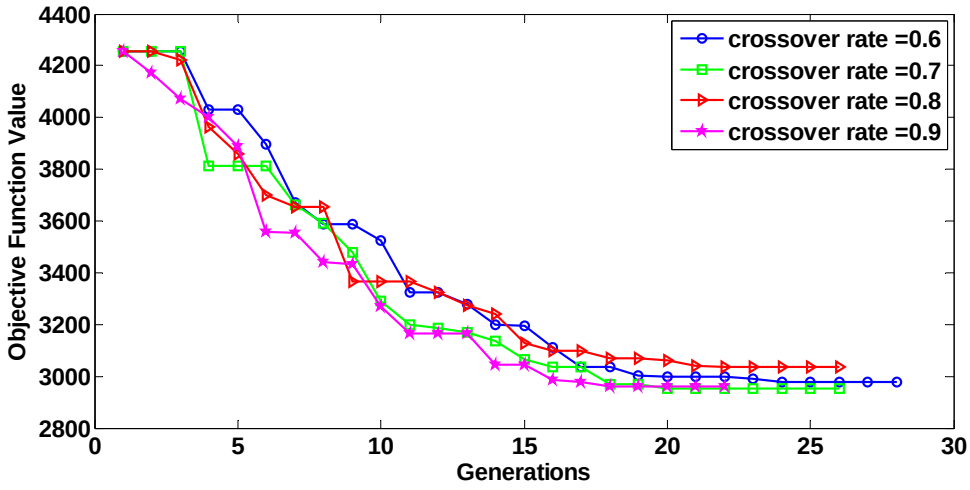


Figure 4.21: GA performances with different crossover rates for an example when $P_m = 0.01$ and $Pop = 100$

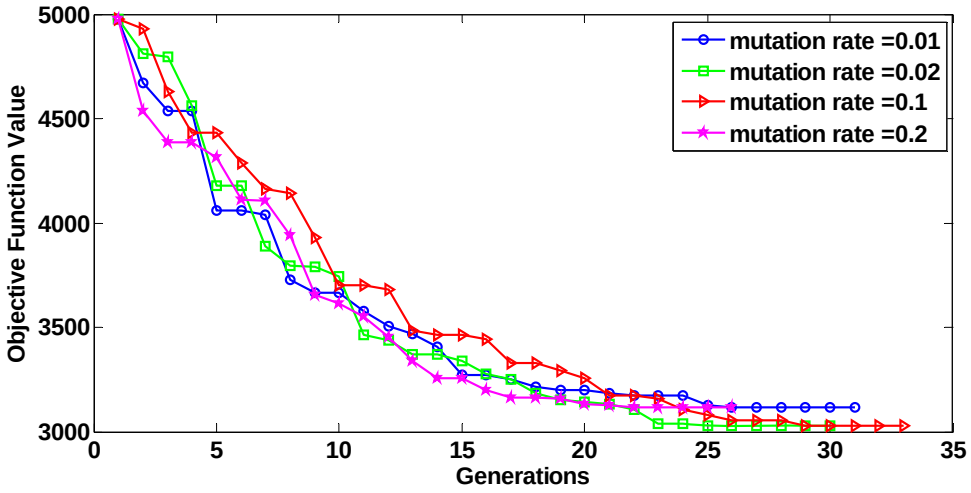


Figure 4.22: GA performance with different mutation rates for an example when $P_c = 0.7$ and $Pop = 100$

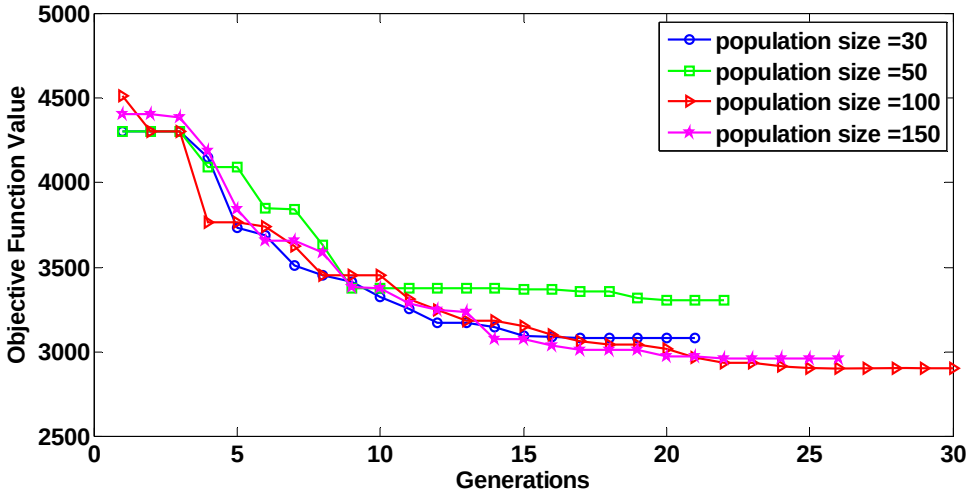


Figure 4.23: GA performance with different population sizes for an example when $P_c = 0.7$ and $P_m = 0.02$

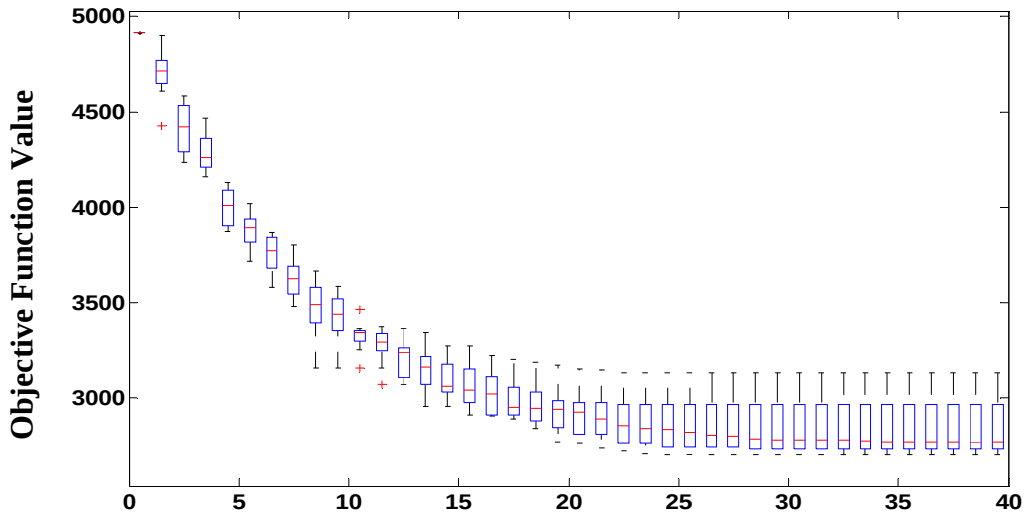


Figure 4.24: GA performance in 10 runs for an example when $P_c = 0.7$, $P_m = 0.02$ and $Pop = 100$

Lastly, we use the best combination of GA parameters in this particular case, e.g. $P_c = 0.7$, $P_m = 0.02$ and $Pop = 100$, and run GA 10 times. The convergence processes are shown in figure 4.24. The central mark is the median, the edges of the box are the 25th and 75th percentiles and the whiskers are the most extreme data points. This figure shows that there is only a little difference in the OFVs in each generation, which highlights the closeness of the different runs and the stability of GA.

4.4 Conclusions

The novelty of this chapter lies in the formulation of the new models and the heuristic methods of these models for solving these problems. This chapter considers the SC scheduling, container storage and the related integrated problems in the straddle-carrier system. These problems of searching optimal schedules of SCs and optimal locations of containers are very important in practical port logistics. Container unloading, loading and dual-cycle processes are studied in three sections. All the three models are formulated as the mixed-integer programming problems aiming to minimise the berth

time of the ship. As the NP-hard problem, the computational complexity increases exponentially with the increasing problem size. This makes it difficult to solve within a reasonable time with an exact method (B&B for example). This requires the use of heuristics methods (GA) to find approximately optimal solutions for large sizes. The crossover and mutation operators adopted in this chapter are similar with those in chapter 3, however, parents selection operator is different; here we adopt binary tournament selection, which gives a better result comparing with adopting roulette wheel sampling method.

The adoption of GA provides a strong potential to solve real-world problems. In this chapter, its efficiency is proved first by the quality of solutions obtained in small-sized problems compared with the solutions from B&B, and the average gap between B&B and GA in terms of OFVs for the small-sized problems is very small ranging from 0% to 2.9%; then the superiority of the proposed GA can be verified through large-scale problems in providing good solutions in a short computation time; and finally we are looking for the solutions with both efficiency and stability, which are illustrated by performance comparisons among different crossover rates, mutation rates and population sizes. Thus, it was concluded that GA outperforms B&B in the average computational time and the ability of solving large-sized problems.

Our proposed models and solution methods can serve terminal operators in two ways: firstly the models can be applied in daily operations of container terminals by providing an insight of schedule plans for vehicles and locations management; and secondly, the solutions can be assessed by applying the algorithm under the real configuration of the terminal and evaluating the berth time.

There still are, however, some possible directions for further research. The proposed models can be extended for uncertain environments. In practice, uncertainty exists in a number of areas, such as vehicle breakdown, incorrect information of container, or ship delay, among others. To incorporate such uncertainties into the problem may require other approaches; for example, the stochastic programming approach. Besides, implementing other heuristics or developing hybrid heuristics for these problems in order to find out whether the solutions can be further improved also suggests a promising piece of work.

Chapter 5. Conclusions and further research

In this thesis, we have studied the scheduling of different container-handling equipment and container storage allocation problems in two types of container terminal. Our main objective is to develop new models and algorithms to improve the operational efficiencies of container terminals. In this chapter, section 5.1 gives a brief overview of the results and discusses the main findings of this thesis. Section 5.2 makes some recommendations for further research in the field.

5.1 Conclusions

Container terminals serve as the interface between marine and land transportation systems. Currently, most of container terminals in the world are working to maximum capacity to serve the increasing volume of container traffic to and from the terminals. Therefore, the efficiency of the stacking and the transportation of large numbers of containers to and from the quayside is critical to the optimum functioning of container terminal (Stahlbock and Voß, 2008). This problem has motivated our study on developing operational research approaches in order to help the terminals to improve the container-handling efficiency and thus to reduce the ship's berth time in the terminals.

In chapter 2, we reviewed the literature and found that most previous research work decomposed the terminal operations into several sub-problems and studied them separately; for example these sub-problems include berth allocation, crane scheduling, storage allocation, and vehicle scheduling. This is due to the difficulty in integrated modelling of several sub-problems simultaneously and providing the corresponding integrated solutions. However, optimising one sub-problem of the container-handling operations cannot guarantee the overall optimal performance of the terminals because these sub-problems are interrelated. Therefore, to fill the gap, this research has focused on the integrated operational problems in container terminals. We aimed to develop novel mathematical optimisation models and explore heuristic algorithms for solving the practical-sized problems in reasonable computational time. Six new integrated problems are proposed in this thesis. In particular, the integration of vehicle/crane

scheduling and container storage allocation problems has been investigated in container unloading and loading processes in two types of container terminal: the automated container terminal and the straddle-carrier system.

In chapter 3, we considered the integration of scheduling container-handling equipment (cranes and vehicles) and container storage allocation problems at an automated container terminal. Because the vehicles considered in such a terminal are automated and there is a lack of drivers' experience, efficient scheduling is very important to improve the productivity of the terminal and thus to achieve the terminal's overall efficiency. Among the range of automated vehicles, the traditional automated guided vehicles (AGVs) are the most representative, and are used in the terminal considered in our study. We developed three models concerning the integrated problems in container unloading/loading operations; specifically we considered the integration of AGV scheduling and the container storage allocation problem in the unloading process, the integration of scheduling all the handling equipment in the container loading process, and the integrated AGV scheduling and storage allocation problem by addressing both unloading and loading processes simultaneously. The models developed in this chapter can be realised in the automated terminals such as ECT Rotterdam, CTA Hamburg, Thames port UK, PPT Singapore and others. The key feature of our proposed models is to provide the integrated solutions for vehicles'/cranes' schedules and containers' locations. The addressed models take into account all types of handling processes in order to support the terminal in different situations: unloading process (model 1), loading process (model 2) and dual-cycle process (model 3). In chapter 4, another three models are developed in the straddle-carrier (SC) system concerning the similar problems investigated in chapter 3. The major difference between AGVs and SCs lies in the way they transfer containers to and from the quay cranes (QCs). An AGV must wait for a QC to load a container onto its top or to pick up a container from its top; however a SC can drop off a container on the ground for a QC to pick it up later, or pick up one container which has been put down on the ground by a QC. Such a special feature allows a better utilisation of equipment in the straddle-carrier system. Therefore, when modelling the problems, we took into consideration the use of the ground as buffer. The models developed in this chapter can be applied in the terminals such as DP world Southampton, Le Havre Port France, Wilhelshaven Germany, Melbourne Australia and

others. Specifically, these models also help the terminal operators of the straddle carrier systems in the decision makings for different situations, when unloading (model 4), loading (model 5) and dual-cycle (model 6) processes take place.

Modelling techniques and solution approaches of the models are summarised as follows. The objective of all the problems is to minimise the ship's berth time. The problems are formulated as mix-integer programming models and solved by two methods. Exact algorithm B&B in AIMMS can only solve small-sized problems because the considered scheduling problems in container terminals are NP-hard and the computational time by B&B increases exponentially as the problem size becomes larger. However, as real-life problems often involve large numbers of containers, we developed heuristic algorithms to solve the practical-sized problems and to reduce the computational time of solving the considered problems. Specifically, we designed different genetic algorithms (GAs) which are suitable for the proposed problems. We have also developed different GA operators according to our preliminary tests in order to investigate their different effects on the evolving results. For small-sized examples, the results obtained from B&B are compared with GA, and it can be found that GA is good in achieving the (near) optimal solutions in shorter computation time for small sizes. The numerical experiments in practical sizes further proves the stability of our proposed GA, which performs stable in dealing with large-sized problems in providing good solutions in short computational times. Moreover, the computational times of GAs are not sensitive to the problem size, which implies that the proposed GAs can be applied to even larger-sized problems. Based on our findings, GA is a very effective approach to be applied in practice. The experimental results also demonstrated that the proposed integrated approaches are able to provide integrated solutions in all the cases which prove the effectiveness of the proposed models. Thus, we conclude that the models developed can be used for increasing the efficiency of container terminals, and hence reducing the daily operating costs of terminals.

In brief, this research makes the following contributions to the literature:

(1) We provided the integrated modelling approaches for the vehicle scheduling and container storage problems in container unloading process;

(2) The integrated models are developed for scheduling all the container-handling equipment in container loading process;
(3) The integrations of vehicle scheduling and container allocation problems are proposed in the dual-cycle process;
(4) The container operations have been analysed in both automated container terminal and straddle carrier system;
(5) Small-sized problems have been solved optimally by exact methods;
(6) Heuristic algorithms-genetic algorithms are developed for solving the problems in large-sizes.

Particularly, GA outperformed the exact solution approaches when considering applications in large-sized problems, in which exact method cannot provide any solutions. GA is able to get the results in a shorter computation time, and the OFV gap between the exact method and GA ranges from 0% to 2.9% for small-sized problems. Moreover, large-sized problems can be solved by GA in a few minutes; for example, in model 5, GA evolves to a solution for the problem with 800 containers, 3 QCs and 20 SCs in about 4 minutes.

Despite of these contributions, our research has the following limitations: (1) the data used in all the computational experiments are drawn from the literature; it would be better to use the real data so as to compare the results obtained from our approaches and the practical operations. (2) Throughout our models, we do not consider the traffic congestions within the terminal, which may cause the considered problems more complex. However, incorporating this issue into our models will be helpful for terminal managers in their decision makings to determine the schedules of vehicles, to assign different number of vehicles, etc. (3) similarly, the conflicts issues for quay cranes and yard cranes may happen in practice, and can be considered into our models. Both traffic congestions and cranes conflicts are unproductive events and have negative effects on the terminal's efficiency. (4) Buffer area capacity at straddle carrier system is assumed to be one in this research. It is worth investigating what the most appropriate capacities are for different situations. (5) The proposed integrated approaches certainly represent the current research challenge in this field. The integrated solutions provided a better control of terminal operations. However, there is a lack of experiments for comparing

different handling approaches, such as the results of single-cycle and dual-cycle, and the results of GA and practical rules.

5.2 Directions for future research

The following areas are worthy of further investigation.

Currently, most of the world's container terminals still adopt the single-cycle method, in which the loading process can only start after the unloading process has been completed. In contrast, the dual-cycle strategy has just begun to be adopted by some container terminals. Therefore, regarding the problems considering both unloading and loading processes simultaneously, it is also considered helpful to investigate to what extent container terminals could benefit from adopting this dual-cycle container-handling strategy. This could be investigated through comparative studies to find out the extent of the differences in the berth time of the ship by handling the same number of containers during the application of both strategies. Future research could also look at the effect of the dual-cycle method on the container traffic to see whether it can ease traffic congestion in the container terminals. Although the issues related to traffic were beyond the scope of this study, indeed, traffic congestion issues are becoming more and more important with the increase in the volume of container traffic. Container terminal operations are usually forced to slow down because of such congestion. The analysis of congestion can contribute a great deal to the optimisation solutions and thus improve the terminal's efficiency.

Some practical considerations can be additionally included in the models. We have not explicitly dealt with container reshuffles in all the models as we assume that reshuffle moves are carried out during the non-busy times of the container terminal. For example, during the night, containers in the yard are usually reshuffled according to their loading sequences by the quay cranes or the delivery sequences by external vehicles for the next day, so that when the loading/unloading process starts, there is no need to waste time in reshuffling containers. However, reshuffles can be incorporated in our proposed models; for example, by introducing empty stacks for temporarily accommodating reshuffled containers and providing the handling times for moving a container to the empty stacks. Moreover, the precedence relationship that is present due to the physical constraints of

containers on the ship can also be taken into account. This is because containers on the ship are separated into two locations: in the hold (covered by hatch covers) and on the deck, and containers in the hold must be loaded before containers on the deck are loaded; likewise, containers on the deck must be unloaded before containers in the hold are unloaded.

All the problems considered in this thesis are based on deterministic cases: however, risks exist in maritime logistics. For example, severe weather, machine break-down, mechanical problems, strikes and incorrect information relating to containers can all cause delays in the container-handling process. The resulting stochastic problems are typically very complex, but can yield significant improvements in container terminals in terms of efficiency, productivity and the scheduling of terminal operations.

The development of better solution algorithms will still have an emphasis on solving optimisation problems in container terminal operations, particularly if considering an industrial application aspect. One perspective is the further development of current GAs, which includes designing new crossover and mutation operators. Another alternative approach may concern combining current GAs with other solution methods, such as simulated annealing (SA), tabu search (TS), simulation or some exact methods. The hybrid algorithms can draw the strengths of each solution method in solving large-scale problems and identifying good solutions.

To conclude, it could prove a promising perspective to consider various complicated and practical constraints, such as interferences, reshuffles, and stochastic factors in our current models, particularly for the industrial aspect. Moreover, further investigating the feasibility of other solution methods or the hybrid algorithms is also an interesting research direction with much to recommend it.

References

Angeloudis, P. and Bell, M.G.H. (2010), 'An uncertainty-aware AGV assignment algorithm for automated container terminals', *Transportation Research Part E: Logistics and Transportation Review,* Vol. 46, No. 3, pp. 354-366.

Ascheuer, N., Fischetti, M. and Grötschel, M. (2001), 'Solving the asymmetric travelling salesman problem with time windows by branch-and-cut', *Mathematical Programming,* Vol. 90, No. 3, pp. 475-506.

Bae, H.Y., Choe, R., Park, T. and Ryu, K.R. (2011), 'Comparison of operations of AGVs and ALVs in an automated container terminal', *Journal of Intelligent Manufacturing,* Vol. 22, No. 3, pp. 413-426.

Balas, E., Ceria, S. and Cornuéjols, G. (1996), 'Mixed 0-1 programming by lift-and-project in a branch-and-cut framework', *Management Science,* Vol. 42, No. 9, pp. 1229-1246.

Bard, J.F., Kontoravdis, G. and Yu, G. (2002), 'A branch-and-cut procedure for the vehicle routing problem with time windows', *Transportation Science,* Vol. 36, No. 2, pp. 250-269.

Bazzazi, M., Safaei, N. and Javadian, N. (2009), 'A genetic algorithm to solve the storage space allocation problem in a container terminal', *Computers & Industrial Engineering,* Vol. 56, No. 1, pp. 44-52.

Bierwirth, C. and Meisel, F. (2010), 'A survey of berth allocation and quay crane scheduling problems in container terminals', *European Journal of Operational Research,* Vol. 202, No. 3, pp. 615-627.

Bilge, Ü. and Ulusoy, G. (1995), 'A time window approach to simultaneous scheduling of machines and material handling system in an FMS', *Operations Research,* Vol. 43, No. 6, pp. 1058-1070.

Bish, E.K. (2003), 'A multiple-crane-constrained scheduling problem in a container terminal', *European Journal of Operational Research,* Vol. 144, No. 1, pp. 83-107.

Bish, E.K., Chen, F.Y., Leong, Y.T., Nelson, B.L., Ng, J.W.C. and Simchi-Levi, D. (2005), 'Dispatching vehicles in a mega container terminal', *OR Spectrum,* Vol. 27, No. 4, pp. 491-506.

Bish, E.K., Leong, T.Y., Li, C.L., Ng, J.W.C. and Simchi Levi, D. (2001), 'Analysis of a new vehicle scheduling and location problem', *Naval Research Logistics,* Vol. 48, No. 5, pp. 363-385.

Bisschop, J. and Roelofs, M. (2011), *AIMMS Language reference, Version 3.11*, Haarlem, Netherland: Paragon Decision Technology Inc.

Bose, J., Reiners, T., Steenken, D. and Voß, S. (2000), 'Vehicle dispatching at seaport container terminals using evolutionary algorithms', *33rd Annual Hawaii International Conference on System Sciences*, 4-7 Jan 2000, Piscataway: IEEE, pp. 1-10.

Briskorn, D., Drexl, A. and Hartmann, S. (2007), 'Inventory-based dispatching of automated guided vehicles on container terminals', IN, Kim, K. and Guenther, H.O. (eds.), *Container Terminals and Cargo Systems*: Springer Berlin Heidelberg, pp. 195-214.

Cao, J.X., Lee, D.H., Chen, J.H. and Shi, Q. (2010), 'The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods', *Transportation Research Part E: Logistics and Transportation Review,* Vol. 46, No. 3, pp. 344-353.

Casey, B. and Kozan, E. (2006), 'A Container Storage Handling Model for High Tech Automated Multimodal Terminals', *the Second International Intelligent Logistics Systems Conference*, 22-23 Feb 2006, Port of Brisbane, Australia: Australian Society for Operations Research, Queensland Branch, pp. 10.1-10.20.

Chen, L., Bostel, N., Dejax, P., Cai, J. and Xi, L. (2007), 'A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal', *European Journal of Operational Research,* Vol. 181, No. 1, pp. 40-58.

Chen, P., Fu, Z., Lim, A. and Rodrigues, B. (2003), 'The general yard allocation problem', *Genetic and Evolutionary Computation-GECCO 2003*: Springer Berlin Heidelberg, pp. 209-209.

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

Cheng, R. and Gen, M. (1997), 'Parallel machine scheduling problems using memetic algorithms', *Computers & Industrial Engineering,* Vol. 33, No. 3-4, pp. 761-764.

Choi, H., Park, B., Lee, J. and Park, C. (2011), 'Dispatching of container trucks using genetic algorithm', *Interaction Sciences (ICIS) 4th International Conference*, 16-18 Aug 2011, Busan, South Korea: IEEE, pp. 146-151.

Christiansen, M., Fagerholt, K., Nygreen, B. and Ronen, D. (2007), 'Maritime transportation', IN, Barnhart C, L. (ed.) *Tranportation. Handbooks in operations research and management science*, Amsterdam: Elsevier, pp. 189-284.

Cordeau, J.F., Laporte, G., Legato, P. and Moccia, L. (2005), 'Models and tabu search heuristics for the berth-allocation problem', *Transportation Science,* Vol. 39, No. 4, pp. 526-538.

Daganzo, C.F. (1989), 'The crane scheduling problem', *Transportation Research Part B: Methodological,* Vol. 23, No. 3, pp. 159-175.

Das, S.K. and Spasovic, L. (2003), 'Scheduling material handling vehicles in a container terminal', *Production Planning & Control,* Vol. 14, No. 7, pp. 623-633.

Evers, J.J.M. and Koppers, S.a.J. (1996), 'Automated guided vehicle traffic control at a container terminal', *Transportation Research Part A: Policy and Practice,* Vol. 30, No. 1, pp. 21-34.

Fereidoonian, F. and Mirzazadeh, A. (2012), 'A genetic algorithm for the integrated scheduling model of a container-handling system in a maritime terminal', *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment,* Vol. 226, No. 1, pp. 62-77.

Glover, F. (1989), 'Tabu search—part I', *ORSA Journal on computing,* Vol. 1, No. 3, pp. 190-206.

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

Goodchild, A.V. (2005), *Crane double cycling in container ports: algorithms, evaluation, and planning*, Ph.D thesis, Department of Civil and Environmental Engineering, University of California, Berkeley, California.

Goodchild, A.V. and Daganzo, C.F. (2006), 'Double-cycling strategies for container ships and their effect on ship loading and unloading operations', *Transportation Science,* Vol. 40, No. 4, pp. 473-483.

Goodchild, A.V. and Daganzo, C.F. (2007), 'Crane double cycling in container ports: planning methods and evaluation', *Transportation Research Part B: Methodological,* Vol. 41, No. 8, pp. 875-891.

Grunow, M., Günther, H.O. and Lehmann, M. (2004), 'Dispatching multi-load AGVs in highly automated seaport container terminals', *OR Spectrum,* Vol. 26, No. 2, pp. 211-235.

Günther, H.O. and Kim, K.H. (2006), 'Container terminals and terminal operations', *OR Spectrum,* Vol. 28, No. 4, pp. 437-445.

Hakam, M., Solvang, W. and Hammervoll, T. (2012), 'A genetic algorithm approach for quay crane scheduling with non-interference constraints at Narvik container terminal', *International Journal of Logistics Research and Applications,* Vol. 15, No. 4, pp. 269-281.

Han, X., Lu, Z. and Xi, L. (2010), 'A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time', *European Journal of Operational Research,* Vol. 207, No. 3, pp. 1327-1340.

Han, Y., Lee, L.H., Chew, E.P. and Tan, K.C. (2008), 'A yard storage strategy for minimizing traffic congestion in a marine container transshipment hub', *OR Spectrum,* Vol. 30, No. 4, pp. 697-720.

Hart, J.P. and Shogan, A.W. (1987), 'Semi-greedy heuristics: An empirical study', *Operations Research Letters,* Vol. 6, No. 3, pp. 107-114.

He, J., Chang, D., Mi, W. and Yan, W. (2010), 'A hybrid parallel genetic algorithm for yard crane scheduling', *Transportation Research Part E: Logistics and Transportation Review,* Vol. 46, No. 1, pp. 136-155.

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

Huang, Y., Liang, C. and Yang, Y. (2009), 'The optimum route problem by genetic algorithm for loading/unloading of yard crane', *Computers & Industrial Engineering,* Vol. 56, No. 3, pp. 993-1001.

Jeong, B.J. and Kim, K.H. (2011), 'Scheduling operations of a rail crane and container deliveries between rail and port terminals', *Engineering Optimization,* Vol. 43, No. 6, pp. 597-613.

Jung, S.H. and Kim, K.H. (2006), 'Load scheduling for multiple quay cranes in port container terminals', *Journal of Intelligent Manufacturing,* Vol. 17, No. 4, pp. 479-492.

Kang, J., Ryu, K.R. and Kim, K.H. (2006), 'Deriving stacking strategies for export containers with uncertain weight information', *Journal of Intelligent Manufacturing,* Vol. 17, No. 4, pp. 399-410.

Kenyon, A.S. and Morton, D.P. (2003), 'Stochastic vehicle routing with random travel times', *Transportation Science,* Vol. 37, No. 1, pp. 69-82.

Kim, K.H. (2005), 'Models and methods for operations in port container terminals', IN, Langevin, A. and Riopel, D. (eds.), *Logistics systems: Design and optimization*, US: Springer, pp. 213-243.

Kim, K.H. and Bae, J.W. (2004), 'A look-ahead dispatching method for automated guided vehicles in automated port container terminals', *Transportation science,* Vol. 38, No. 2, pp. 224-234.

Kim, K.H. and Kim, H.B. (1999a), 'Segregating space allocation models for container inventories in port container terminals', *International Journal of Production Economics,* Vol. 59, No. 1-3, pp. 415-423.

Kim, K.H. and Kim, K.Y. (1999b), 'Routing straddle carriers for the loading operation of containers using a beam search algorithm', *Computers & Industrial Engineering,* Vol. 36, No. 1, pp. 109-136.

Kim, K.H., Lee, K.M. and Hwang, H. (2003), 'Sequencing delivery and receiving operations for yard cranes in port container terminals', *International Journal of Production Economics,* Vol. 84, No. 3, pp. 283-292.

Kim, K.H. and Moon, K.C. (2003), 'Berth scheduling by simulated annealing', *Transportation Research Part B: Methodological,* Vol. 37, No. 6, pp. 541-560.

Kim, K.H. and Park, Y.M. (2004), 'A crane scheduling method for port container terminals', *European Journal of Operational Research,* Vol. 156, No. 3, pp. 752-768.

Kim, K.Y. and Kim, K.H. (1999c), 'A routing algorithm for a single straddle carrier to load export containers onto a containership', *International Journal of Production Economics,* Vol. 59, No. 1-3, pp. 425-433.

Kirkpatrick, S. and Vecchi, M. (1983), 'Optimization by simmulated annealing', *Science,* Vol. 220, No. 4598, pp. 671-680.

Klabjan, D. (2005), 'Large-scale models in the airline industry', IN, Desaulniers, G., Desrosiers, J. and Solomon, M.M. (eds.), *Column generation*, US: Springer, pp. 163-195.

Kozan, E. and Preston, P. (1999), 'Genetic algorithms to schedule container transfers at multimodal terminals', *International Transactions in Operational Research,* Vol. 6, No. 3, pp. 311-329.

Kozan, E. and Preston, P. (2006), 'Mathematical modelling of container transfers and storage locations at seaport terminals', *OR Spectrum,* Vol. 28, No. 4, pp. 519-537.

Laporte, G., Riera-Ledesma, J. and Salazar-González, J.J. (2003), 'A branch-and-cut algorithm for the undirected traveling purchaser problem', *Operations Research,* Vol. 51, No. 6, pp. 940-951.

Lau, H.Y.K. and Zhao, Y. (2008), 'Integrated scheduling of handling equipment at automated container terminals', *International Journal of Production Economics,* Vol. 112, No. 2, pp. 665-682.

Lee, B.K. and Kim, K.H. (2010), 'Optimizing the block size in container yards', *Transportation Research Part E: Logistics and Transportation Review,* Vol. 46, No. 1, pp. 120-135.

Lee, D.H., Cao, J.X. and Shi, Q. (2008a), 'Integrated Model for Truck Scheduling and Storage Allocation Problem at Container Terminals', *Transportation Research Board 87th Annual Meeting*, 13-17 Jan 2008, Washington, D.C, USA: IEEE

Lee, D.H., Cao, J.X. and Shi, Q. (2008b), 'Integrated quay crane and yard truck schedule for inbound containers', *IEEE International Conference on Industrial Engineering and Engineering Management*, 2008, Singapore: IEEE, pp. 1219-1223.

Lee, D.H., Cao, J.X., Shi, Q. and Chen, J.H. (2009), 'A heuristic algorithm for yard truck scheduling and storage allocation problems', *Transportation Research Part E: Logistics and Transportation Review,* Vol. 45, No. 5, pp. 810-820.

Lee, D.H., Cao, J.X. and Shi, Q.X. (2009), 'Synchronization of yard truck scheduling and storage allocation in container terminals', *Engineering Optimization,* Vol. 41, No. 7, pp. 659-672.

Lee, D.H., Cao, Z. and Meng, Q. (2007), 'Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm', *International Journal of Production Economics,* Vol. 107, No. 1, pp. 115-124.

Lee, D.H., Wang, H.Q. and Miao, L. (2008), 'Quay crane scheduling with non-interference constraints in port container terminals', *Transportation Research Part E: Logistics and Transportation Review,* Vol. 44, No. 1, pp. 124-135.

Lee, L.H., Chew, E.P., Tan, K.C. and Wang, Y. (2010), 'Vehicle dispatching algorithms for container transshipment hubs', *OR Spectrum,* Vol. 32, No. 3, pp. 663-685.

Li, W., Wu, Y., Petering, M., Goh, M. and Souza, R. (2009), 'Discrete time model and algorithms for container yard crane scheduling', *European Journal of Operational Research,* Vol. 198, No. 1, pp. 165-172.

Lim, J.K., Kim, K.H., Yoshimoto, K., Lee, J.H. and Takahashi, T. (2003), 'A dispatching method for automated guided vehicles by using a bidding concept', *OR Spectrum,* Vol. 25, No. 1, pp. 25-44.

Linn, R., Liu, J., Wan, Y., Zhang, C. and Murty, K.G. (2003), 'Rubber tired gantry crane deployment for container yard operation', *Computers & Industrial Engineering,* Vol. 45, No. 3, pp. 429-442.

Liu, Q. (2010), *Efficiency analysis of container ports and terminals*, PhD Thesis, Centre for Transport Studies, Department of Civil, Environmental and Geomatic Engineering, University College London (UCL), London, UK.

Mak, K.L. and Zhang, L. (2009), 'Simultaneous scheduling of import and export containers handling in container terminals', *World Congress on Engineering, WCE 2009*.

Meisel, F. and Wichmann, M. (2010), 'Container sequencing for quay cranes with internal reshuffles', *OR Spectrum,* Vol. 32, No. 3, pp. 569-591.

Moccia, L., Cordeau, J.F., Gaudioso, M. and Laporte, G. (2006), 'A branch and cut algorithm for the quay crane scheduling problem in a container terminal', *Naval Research Logistics (NRL),* Vol. 53, No. 1, pp. 45-59.

Moussi, R., Ndiaye, N. and Yassine, A. (2012), 'Hybrid genetic simulated annealing algorithm (HGSAA) to solve storage container problem in port', IN, Pan, J.S., Chen, S.M. and Nguyen, N.T. (eds.), *Intelligent Information and Database Systems*, Berlin Heidelberg: Springer, pp. 301-310.

Moussi, R., Yassine, A., Kansou, A. and Galinho, T. (2011), 'Scheduling of lifting vehicles with time windows in an automated port container terminal', *4th International Conference on Logistics (LOGISTIQUA)*, May 31-June 3 2011, Le Havre, France: IEEE, pp. 55-61.

Murty, K.G., Liu, J., Wan, Y. and Linn, R. (2005), 'A decision support system for operations in a container terminal', *Decision Support Systems,* Vol. 39, No. 3, pp. 309-332.

Ng, W. and Mak, K. (2005), 'Yard crane scheduling in port container terminals', *Applied Mathematical Modelling,* Vol. 29, No. 3, pp. 263-276.

Ng, W. and Mak, K. (2006), 'Quay crane scheduling in container terminals', *Engineering Optimization,* Vol. 38, No. 6, pp. 723-737.

Nguyen, V.D. and Kim, K.H. (2009), 'A dispatching method for automated lifting vehicles in automated port container terminals', *Computers & Industrial Engineering,* Vol. 56, No. 3, pp. 1002-1020.

Nishimura, E., Imai, A., Janssens, G.K. and Papadimitriou, S. (2009), 'Container storage and transshipment marine terminals', *Transportation Research Part E: Logistics and Transportation Review,* Vol. 45, No. 5, pp. 771-786.

Nishimura, E., Imai, A. and Papadimitriou, S. (2005), 'Yard trailer routing at a maritime container terminal', *Transportation Research Part E: Logistics and Transportation Review,* Vol. 41, No. 1, pp. 53-76.

Peterkofsky, R.I. and Daganzo, C.F. (1990), 'A branch and bound solution method for the crane scheduling problem', *Transportation Research Part B: Methodological,* Vol. 24, No. 3, pp. 159-172.

Preston, P. and Kozan, E. (2001a), 'An approach to determine storage locations of containers at seaport terminals', *Computers & Operations Research,* Vol. 28, No. 10, pp. 983-995.

Preston, P. and Kozan, E. (2001b), 'A tabu search technique applied to scheduling container transfers', *Transportation planning and technology,* Vol. 24, No. 2, pp. 135-153.

Spears, W.M. and De Jong, K.A. (1991), 'An Analysis of Multi-Point Crossover', IN, Rawlins, G.J.E. (ed.) *Foundations of genetic algorithms*, San Mateo, CA: Morgan Kaufmann, pp. 301-315.

Stahlbock, R. and Voß, S. (2008), 'Operations research at container terminals: a literature update', *OR Spectrum,* Vol. 30, No. 1, pp. 1-52.

Steenken, D., Voß, S. and Stahlbock, R. (2004), 'Container terminal operation and operations research-a classification and literature review', *OR Spectrum,* Vol. 26, No. 1, pp. 3-49.

Tavakkoli-Moghaddam, R., Makui, A., Salahi, S., Bazzazi, M. and Taheri, F. (2009), 'An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports', *Computers & Industrial Engineering,* Vol. 56, No. 1, pp. 241-248.

Tavakkoli-Moghaddam, R. and Safaei, N. (2006), 'An evolutionary algorithm for a single-item resource-constrained aggregate production planning problem', *IEEE Conference on Evolutionary Computation (CEC)* 16-21 Jul 2006, Vancouver, B.C., Canada: IEEE, pp. 2851-2858.

United Nations Conference on Trade and Development, S. (2012), 'Review of maritime transport', Available: http://www.unctad.org/en/PublicationsLibrary/rmt2012_en.pdf [Accessed: 25 June 2012].

Vacca, I., Bierlaire, M. and Salani, M. (2007), 'Optimization at container terminals: status, trends and perspectives', *the Swiss Transport Research Conference (STRC)*, 2007, Monte Verita, Ascona, pp. 1-21.

Van Der Heijden, M., Ebben, M., Gademann, N. and Van Harten, A. (2002), 'Scheduling vehicles in automated transportation systems Algorithms and case study', *OR Spectrum,* Vol. 24, No. 1, pp. 31-58.

Van Der Meer, J.R. (2000), *Operational control of internal transport*, Erasmus Research Institute of Management (ERIM) Ph.D Series Research in Management 1, Rotterdam: Erasmus University.

Vis, I.F.A. and Carlo, H.J. (2010), 'Sequencing Two Cooperating Automated Stacking Cranes in a Container Terminal', *Transportation Science,* Vol. 44, No. 2, pp. 169-182.

Wiese, J., Kliewer, N. and Suhl, L. (2009), 'A survey of container terminal characteristics and equipment types', *Technical Report 0901*, DS&OR Lab, University of Paderborn.

Wiese, J., Suhl, L. and Kliewer, N. (2010), 'Mathematical models and solution methods for optimal container terminal yard layouts', *OR Spectrum,* Vol. 32, No. 3, pp. 427-452.

Wong, A. and Kozan, E. (2010), 'Optimization of container process at seaport terminals', *Journal of the Operational Research Society,* Vol. 61, No. 4, pp. 658-665.

Yang, C., Wang, X. and Li, Z. (2012), 'An optimization approach for coupling problem of berth allocation and quay crane assignment in container terminal', *Computers & Industrial Engineering,* Vol. 63, No. 1, pp. 243-253.

Yuan, S., Skinner, B.T., Huang, S., Liu, D.K., Dissanayake, G., Lau, H. and Pagac, D. (2011), 'A job grouping approach for planning container transfers at automated seaport container terminals', *Advanced Engineering Informatics,* Vol. 25, No. 3, pp. 413-426.

Zhang, C., Liu, J., Wan, Y., Murty, K.G. and Linn, R.J. (2003), 'Storage space allocation in container terminals', *Transportation Research Part B: Methodological,* Vol. 37, No. 10, pp. 883-903.

Zhang, C.Q., Wan, Y.W., Liu, J.Y. and Linn, R.J. (2002), 'Dynamic crane deployment in container storage yards', *Transportation Research Part B-Methodological,* Vol. 36, No. 6, pp. 537-555.

Zhang, H. and Kim, K.H. (2009), 'Maximizing the number of dual-cycle operations of quay cranes in container terminals', *Computers & Industrial Engineering,* Vol. 56, No. 3, pp. 979-992.