EVOLUTIONARY ALGORITHMS AND SIMULATION FOR INTELLIGENT AUTONOMOUS VEHICLES IN CONTAINER TERMINALS

SHAYAN KAVAKEB

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

March 2015

Abstract

The study of applying soft computing techniques, such as evolutionary computation and simulation, to the deployment of intelligent autonomous vehicles (IAVs) in container terminals is the focus of this thesis. IAVs are a new type of intelligent vehicles designed for transportation of containers in container terminals. This thesis for the first time investigates how IAVs can be effectively accommodated in container terminals and how much the performance of container terminals can be improved when IAVs are being used. In an attempt to answer the above research questions, the thesis makes the following contributions: First, the thesis studies the fleet sizing problem in container terminals, an important design problem in container terminals. The contributions include proposing a novel evolutionary algorithm (with superior results to the state-of-the-art CPLEX solver), combining the proposed evolutionary algorithm with Monte Carlo simulation to take into account uncertainties, validating results of the uncertain case with a high fidelity simulation, proposing different robustness measures, comparing different robust solutions and proposing a dynamic sampling technique to improve the performance of the proposed evolutionary algorithm. Second, the thesis studies the impact of IAVs on container terminals' performance and total cost, which are very important criteria in port equipment. The contributions include developing simulation models using realistic data (it is for the first time that the impact of IAVs on containers terminals is investigated using simulation models) and applying a cost model to the results of the simulation to estimate and compare the total cost of the case study with IAVs against existing trucks. Third, the thesis proposes a new framework for the simulations of container terminals. The contributions include developing a flexible simulation framework, providing a user library for users to create 3D simulation models using drag-and-drop features, and allowing users to easily incorporate their optimisation algorithms into their simulations.

Acknowledgements

This thesis would certainly not have been possible without the help and support of many people. I would like to take this opportunity to express my special debt of gratitude to them.

First of all, I would like to express my gratitude to my director of studies, Dr. Trung Thanh Nguyen. There are too many reasons for this, and I am incapable of expressing them all. He cared a lot about my research and he put as much effort into my PhD as he could. His promptness and enthusiasm to help me was exceptional; whenever I needed him for my research he was there for me, saying: "For you I always have time". He guided me through my research while letting me be myself and grow. His desire to tackle challenging and interesting problems is inspirational. His advice on both my research and my career has been priceless. I have also had the great honour to be his research assistant for a research project supported by the Chartered Institute of Logistics and Transport. I deeply appreciate what he has done for me.

I also owe special thanks to my second supervisor, Professor Zaili Yang. I could have not finished my PhD without his help and support. He also offered me a great opportunity to be part of the ENRICH project to advance my knowledge in transportation and supply chain management and also extend my network in the research community. He was also kind enough to offer me a wonderful opportunity to work on the Voyage Optimal Routing in the Cloud (VoyagOR) project. This gave me the chance to apply my research to other real-world applications.

Also, I would like to thank the other members of my supervisory board, Professor Ian Jenkinson and Professor Jin Wang, for their support and comments on the thesis.

This thesis was supported by a European project, Intelligent Transport for Dynamic Environment (InTraDE); a Seed-corn funding grant from the Chartered Institute of

Logistics and Transport; a research grant from RCUK NEMODE – New Economic Models in the Digital Economy, Network +; and a project from InnovativeUK, VoyagOR. I wish to thank them for their generous support.

I am grateful to Mrs. Kay McGinley and Mrs. Roisin Murray for providing their cost model to use in my thesis, by which Chapter 5 was made possible.

I would like to express my gratitude to the people of the InTraDE project, especially Professor Rochdi Merzouki and Dr. Mohamed Benmerikhi for their invaluable comments on my research at InTraDE's technical meetings and workshops. Also, I would like to thank the industrial partners of InTraDE for their prompt responses to my questionnaires and also for providing the required data.

I would like to thank internship students from Ecole Polytechnique de Lille for their contributions to the simulation study in the thesis during their short stay at Liverpool Logistics Offshore and Marine (LOOM) research institute. Notably, I am grateful to Mrs. Celine Ly for her help in coding of the simulation framework in Chapter 6.

I am grateful to Dr. Ramin Riahi for his priceless comments on my talks at LOOM. He has deep knowledge on maritime transport and container terminal operations due to his long industrial experience in the maritime industry.

My special thanks should go to Professor Edward Tsang for furthering my knowledge on optimisation techniques and also for his invaluable module about constraint satisfaction techniques, which I studied during my stay at University of Essex before starting my PhD at Liverpool John Moores University.

I would like to thank my wonderful friends and colleagues at LOOM, School of Technology, Engineering and Maritime Operations for their support and encouragement, in particular, Ariyan, Mohammad, Minho, Allan, Osumanu, Chaoyu, Haqimin, Saleh, Aymen, Fayas, Dimitrios and Ahmed, to name just a few.

Last (but not least), I am very grateful to my family. My mom, Ashraf, my siblings, Shabnam and Ashkan, you have always encouraged me to pursue my goals. During this PhD journey, I have been constantly missing you. The memory of my father, who was always encouraging me in my life, is an ongoing inspiration. Niyousha, my loving wife, you are the greatest love of my life. Your patience and continuous support during my research is admirable. With you this PhD journey was very enjoyable and pleasant.

This thesis is dedicated to my family.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AGV	Automated Guided Vehicle
ALV	Automatic Lifting Vehicle
CTSC	FlexSim CT's Straddle Carrier
EA	Evolutionary Algorithm
eMCS	extension on MC simulation
ESTT	Environment with Shuttle Transportation Tasks
FEU	40-foot Equivalent Unit
FJSP	Flexible Job-shop Scheduling Problem
FSEA	Fleet Sizing Evolutionary Algorithm
FSP	Fleet Sizing Problem
FS-VRP	Fleet Sizing and Mixed Vehicle Routing Problem
GA	Genetic Algorithm
IAV	Intelligent Autonomous Vehicle
iFSEA	improved FSEA
InTraDE	Intelligent Transportation for Dynamic Environment
IP	Integer Programming
JSP	Job-shop Scheduling Problem
MADM	Multi Attribute Decision Making Method
MC	Monte Carlo

MTTR	Mean Time To Repair
PDP	Pickup-and-Delivery Point
QC	Quay Crane
SC	Stack Crane
TEU	20-foot Equivalent Unit
TRMG	Triple Rail Mounted Gantry Crane
VRP	Vehicle Routing Problem

Chapter 1

Introduction

Container terminals play a vital role in international supply chains, since container terminals are major interfaces to transfer/distribute containers (carrying 90% of non-bulk world trade goods as of 2009 (Ebeling, 2009)). How container terminals handle goods greatly influences emissions and final cost, because up to 50% of cost could be due to handling and logistics (Rodrigue et al., 2013). Therefore, improving container terminal efficiency is an important/practical issue. This is even more important considering the steady growth of the global container market (the growth rate was 201% during the 2003-2012 period (UNCTAD, 2013)). In order to improve their efficiency, container terminal operators need to evaluate the possibility to optimise/modernise their transport and logistics operations. One of the possible ways to do so is to utilise automation technologies such as automated vehicles to transfer containers between the vessels and storage areas. It is believed that automated vehicles can significantly reduce safety risks, emissions and costs. For example an experiment by Saanen et al. (2003) showed that automation can reduce yearly expenses on transfer vehicles by up to 56%. Currently automated vehicles in container terminals, however, have two major limitations. First, they need fixed pre-paved paths, which require expensive infras-

tructure investment. Second, they cannot pick up/drop off containers by themselves. This in turn increases crane/vessel waiting time, one of the most expensive factors in container terminals. These limitations can be alleviated by a new generation of automated vehicles named intelligent autonomous vehicles (IAVs), recently developed in a European project entitled Intelligent Transportation for Dynamic Environment (InTraDE)[1]. However, the IAV system is a new concept in container terminals and there are many research questions about the applicability of IAVs in container terminals that need to be addressed before their deployment. Thus, the primary purpose of this thesis is to answer some of these questions by investigating the impact of utilising IAVs on the performance and cost of container terminals using soft computing approaches such as evolutionary algorithms (EAs) and simulation.

1.1 IAVs in container terminals

In container terminals (Figure 1.1 [2]), vehicles transport containers between quay-side and stack-side areas.

A quay-side area is a place where vessels are berthed and a stack-side area is a place where containers are stacked temporarily. Stack-side areas consist of a number of blocks to stack containers. Each block is served by a number of stacking cranes (SCs) to stack/unstack containers. Once a vessel is berthed, a number of quay cranes (QCs) would be assigned to that vessel. QCs discharge containers from the vessel. Vehicles would then come to collect containers and transport them to the stack-side area. If vehicles can pick up containers by themselves, QCs can place containers in a "buffer" where vehicles can come and collect them; otherwise QCs must wait until vehicles arrive. The IAV is one of the few vehicles that are able to pick up/drop off

[1] See, wwww.intrade-nwe.eu

[2] This figure is a screenshot of the developed simulation framework in the thesis.

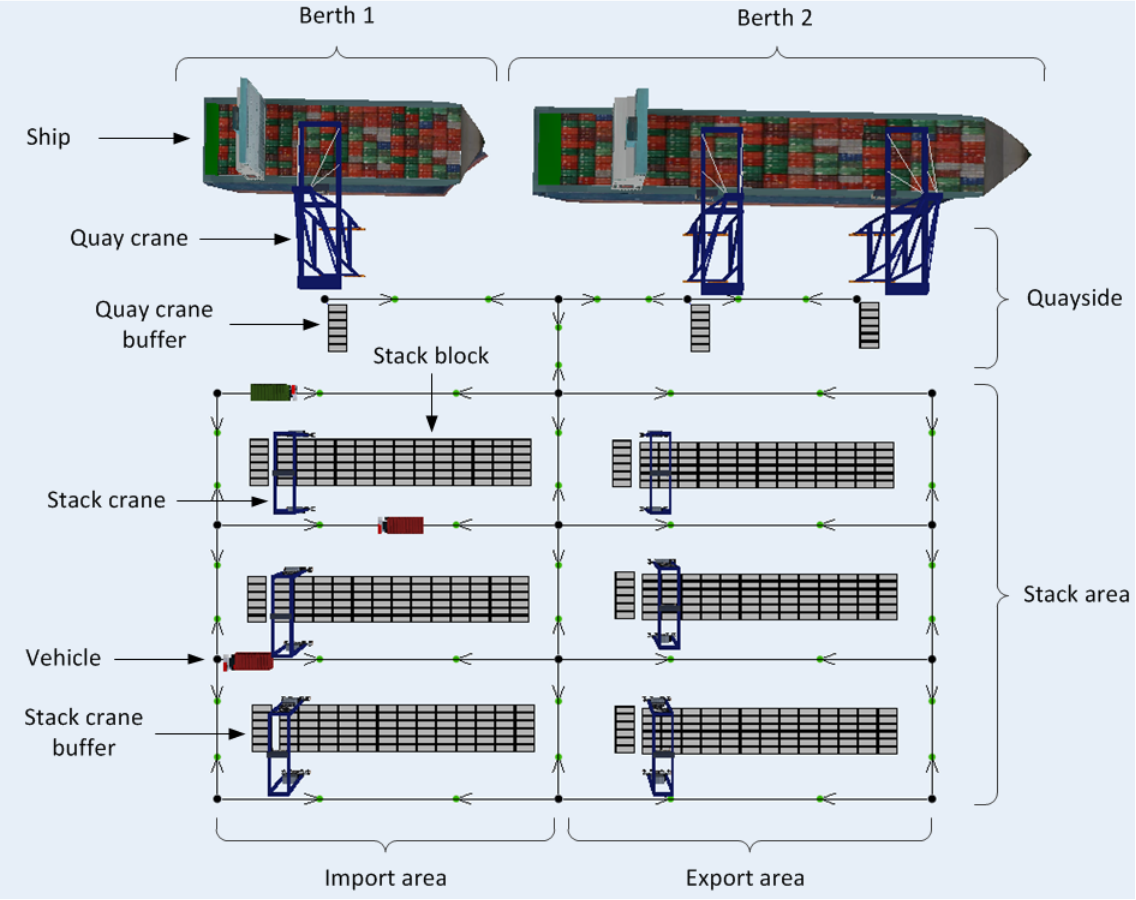

Figure 1.1: The general layout and components of a container terminal

containers by themselves thanks to a special table-like structure named the "cassette" (Figure 1.2). Therefore, IAVs can be used in combination with cassettes in container terminals so that cranes and IAVs do not have to wait for each other. This way, the waiting time of both cranes and IAVs can be minimised. Vehicles then transport containers to the stack-side area and drop them off in a buffer for SCs or in the case without buffers vehicles should wait for SCs to unload the containers from them. Once all the containers are discharged from the vessel the loading tasks start. The loading tasks are similar to the discharging tasks, but in the opposite direction, transporting containers from the stack-side area into vessels.

IAVs belong to a new type of intelligent vehicles to transport containers inside

(a) IAV prototype

(b) Cassette prototype

(c) IAV with empty cassette

(d) IAV with loaded cassette

Figure 1.2: IAV and cassette prototypes. Plots c and d shows the IAV with empty and loaded cassettes, respectively.

container terminals. IAVs are superior to existing automated guided vehicles (AGVs) that are currently being used in European container ports. Some novel features of IAVs are listed below:

1. As mentioned above, IAVs can pick up and drop off containers by themselves if they are combined with a table-like object named a "cassette". Using this feature the buffer of containers under cranes can be utilised to decrease the waiting time of cranes and IAVs.

2. IAVs' wheels have a 180-degree rotation capability by which IAVs can travel in considerably small spaces. Moreover, IAVs can travel in all directions (i.e. forward, backwards and sideways) without the need for turning.

3. Unlike AGVs, IAVs do not need to follow specific paved tracks. Thanks to the geographical positions systems feature, IAVs can navigate to the defined destination and when any obstacle appears on their way thanks to their sensor technology they can detect it and prevent from any collision by taking the proper decision.

4. IAVs can make platoons in which each IAV follows the IAV in the front and leads the IAV in the back. The platoon can be led by a leader (usually a man driver vehicle). This will be used for moving IAVs to places where due to some reasons the vehicles are not allowed to travel automatically.

Given that an IAV is a new concept in container terminals, there are many tactical and operational decision making problems (e.g. the optimal fleet size, schedule, travel routes etc) related to applicability of IAVs in container terminals that have not been tackled yet. Addressing these problems is not trivial, it requires applying advanced operational research techniques. Soft computing techniques are well known techniques in operational research that have been used to deal with complicated decision making problems. Examples of these techniques are meta-heuristic algorithms (e.g. EAs, ant colony optimisation etc (Boussaïd et al., 2013)) and simulation (e.g. discrete event, agent based etc (Angeloudis and Bell, 2011)). This thesis for the first time will investigate the impact of deployment of IAVs in container terminals in terms of performance and cost using soft computing (e.g. EAs and simulation) techniques.

1.2 Scope of the thesis

A container terminal is a very complex system that contains different optimisation and decision making problems. Chapters 3 and 4 focus on the very important fleet sizing problem (FSP) in container terminals. In these chapters, an EA is developed combined

with the Monte Carlo simulation to not only identify the optimal number of IAVs in static environments, but also the proposed algorithm can identify the optimal fleet size that is robust to the changes due to uncertainties. Among all possible sources of uncertainty that may affect the optimal fleet size, uncertainty in the travel time of vehicles and process time of quay cranes are considered to be the main source of uncertainty that can have a significant impact on the optimal number of vehicles. Chapters 5 and 6 study the performance and cost of the case study with IAVs using simulation of the quay-side and stack-side operations in container terminals. The gate side activities (i.e. picking up and dropping off containers by external trucks), are not considered in the developed simulation models, given that IAVs are supposed to only transport containers between the quay and stack areas.

1.3 General research questions

The approach of this thesis is first to look at an important tactical problem that container terminals will face due to utilisation of IAVs: identifying the optimal number of IAVs to perform the transportation tasks between the quay-side and stack-side areas. In this regard, the general questions that the thesis is interested in investigating and finding the answer to are:

What would be the optimal number of IAVs in container terminals? Is the optimal fleet size robust to the changes in the environment (e.g. vehicle breakdowns, deadlocks, collisions etc)? Can the optimal fleet size tolerate the changes in the process time of quay cranes?

Answering the questions above requires using soft computing techniques (i.e. EAs and Monte Carlo sampling) to develop optimisation algorithms to identify the optimal number of IAVs that can perform all the required transportation tasks. In addition,

these algorithms should be able to take into account uncertainty in the environment to ensure that the optimal fleet size is robust to the changes. Chapters 3 and 4 will be dedicated to answer these questions.

How much the performance of container terminals can be improved using IAVs compared with the exiting vehicle systems? How IAVs can be accommodated in container terminals? How much the IAV system can be financially beneficial for container terminals in order to reduce the cost?

To answer the above questions, IAVs should be deployed in container terminal environments to evaluate their actual performance accurately. However, such experiments are not possible given that IAVs have not been manufactured commercially yet and it can also be very expensive and time consuming to do so. Thus, for such cases simulation is a very good alternative tool to simulate the virtual environment of container terminals by considering the detailed transportation tasks and operations in container terminals. For this purpose, carrying out a simulation study on a specific container terminal as the case study is necessary. Chapters 5 and 6 are dedicated to the simulation study on the case study container terminal.

The questions above show the research directions to be followed in the following chapters.

1.4 Summary of contributions

This subsection provides a summary of contributions in the thesis. The major contributions are as follows.

Firstly, an extensive literature review on the FSP in environments with shuttle transportation tasks (ESTTs) and simulation approaches in container terminals is provided. Secondly, an EA is developed to identify the optimal number of vehicles

in ESTTs. To evaluate the developed EA, test cases are generated for the FSP in container terminals using realistic data from the case study container terminals. The results of the EA are then compared with those of the state-of-the-art CPLEX software on the generated test cases. The EA is then extended to provide the robust fleet size under uncertainty in travel time of vehicles and process time of machines. Thirdly, the efficiency of the EA is improved by developing a dynamic sampling technique. In addition, different robustness measures are incorporated to provide different robust solutions based on the requirement of users.

Fourthly, a simulation model is developed to compare the performance of the case study container terminal with IAVs versus the existing vehicle system (i.e. trucks). A cost model is then developed to estimate the cost of the case study container terminal with IAVs and trucks. Lastly, a simulation framework is developed for simulation of container terminals. This framework offers more flexibility in simulation of container terminals compared with the FlexSim CT software. This framework also provides an object library that users can use to develop their simulation models using the drag-and-drop feature.

1.5 Outline of the thesis

This thesis is an attempt to answer the research questions above. In Chapter 2, a literature review will be conducted to study important research influencing the current study on: firstly the FSP in ESTTs. In these environments there are a number of pickup and delivery points where vehicles transport goods between these points. At each point there is a machine to process the goods. Container terminals, manufacturing shop floors and warehouses are examples of ESTTs. Secondly, robust optimisation will be explained in this chapter and related research to robust fleet sizing will be reviewed.

Thirdly, this chapter will review simulation studies in container terminals. This part is not limited to only the FSP. It considers simulation studies in container terminals for various purposes.

Chapter 3 presents a new EA to identify the optimal number of vehicles in ESTTs with static and uncertain environments. For the static case, the results of the developed algorithm will be compared with the state-of-the-art CPLEX using an integer programming (IP) model from literature (Vis et al., 2005). For the uncertain case, the travel time of vehicles and process time of machines (i.e. quay cranes in container terminals) will be considered to be the main sources of uncertainty that have a significant impact on the optimal fleet size. To identify the robust fleet size against these sources of uncertainty, the developed EA will be combined with a Monte Carlo simulation technique to evaluate fitness of solutions in the presence of uncertainty in the system. Results of the EA for the uncertain environment will be validated with those of a high fidelity simulation study. To conduct the simulation study, a simulation framework will be developed for container terminals and details of this framework are explained in Chapter 6.

Chapter 4 presents two approaches to improving the performance of the EA developed in Chapter 3. New dynamic sampling techniques will be developed to achieve high quality solutions using fewer samples. This chapter also incorporates different robustness measures to evaluate the robustness of solutions based on different measures. The differences between the robust solutions will then be evaluated using a statistical test.

Chapter 5 describes the development of a simulation study to compare the performance of the case study container terminal with IAVs versus the existing vehicle system (i.e. trucks). The simulation will be developed using the FlexSim CT software. The quay crane net moves per hour and vessel staying time at berth will be considered

the two performance measures to compare IAVs with trucks. The quay crane net moves per hour are the net number of containers that are moved by a quay crane from/into vessels in one hour in discharging/loading cases. The optimal number of vehicles using the provided performance target will then be identified. This chapter also explains the details of the cost model that is developed to compare the cost of the case study with the optimal number of IAVs against that of trucks.

Chapter 6 explains the developed framework for simulation of container terminals. This framework will be developed based on the FlexSim CT software to address the limitations in this software and also to fill the gap of available simulation tools for container terminals. The framework is then used to simulate realistic scenarios of the case study which is not possible by the current version of the FlexSim CT software. This framework will also be used in Chapter 3 to validate the results of the developed EA for uncertain environments.

Chapter 7 concludes this thesis. It gives a summary of the thesis, it lists the main contributions and points out possible directions for further work.

The organisation of the thesis is summarised in Figure 1.3:

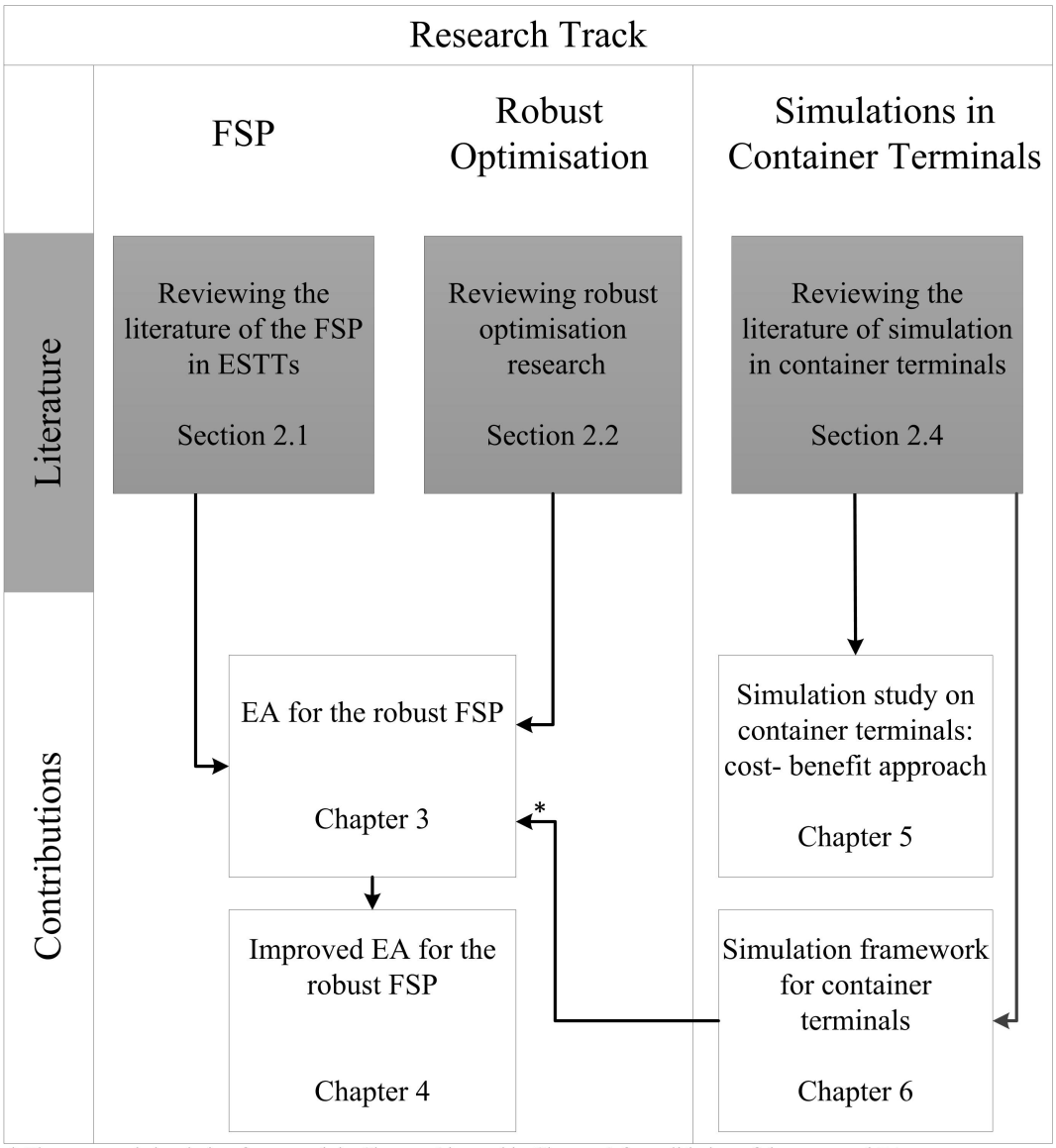

Figure 1.3: The thesis structure.

1.6 Publication resulting from this thesis

Refereed or submitted journal papers

1. S. Kavakeb, T. T. Nguyen, Z. Yang and I. Jenkinson (2015). Evolutionary fleet sizing in static and uncertain environments with shuttle transportation tasks - the case studies of container terminals, to appear in IEEE Computational Intelligent Magazine.

2. S. Kavakeb, T. T. Nguyen, K. McGinley, Z. Yang, I. Jenkinson and R. Murray (2015). Green vehicle technology to enhance the performance of a European port: a simulation model with a cost-benefit approach, submitted to Transportation Research Part C: Emerging Technologies. **(given the option to revise for acceptance)**.

In-preparation journal paper

3. S. Kavakeb, T. T. Nguyen, C. Ly, Z. Yang and I. Jenkinson (2015). A discrete-event simulation framework for container terminals. To be submitted to Advanced Engineering Informatics in July, 2015.

Refereed conference papers

4. S. Kavakeb, T. T. Nguyen, Z. Yang and I. Jenkinson (2014). Identifying the robust number of intelligent autonomous vehicles in container terminals. Evolutionary Algorithms and Meta-heuristics in Stochastic and Dynamic Environment. Applications of Evolutionary Computation, Springer, pp.829-840.

5. S. Kavakeb, T. T. Nguyen, M. Benmerikhi, Z. Yang and I. Jenkinson (2014). An improved memetic algorithm to enhance the sustainability and reliability of transport in container terminals. IEEE Symposium on Computational Intelligence for Security and Defense Applications, Hanoi, Vietnam, vol. 5. 2014.

6. T. T. Nguyen, S. Kavakeb, Z. Yang, I. Jenkinson and J. Wang (2014) Identifying the optimal type and number of transfer vehicles to improve productivity in ports – A simulation approach. 19th Annual Logistics Research Network Conference, 3-5 September, Huddersfield, UK.

7. S. Kavakeb, T. T. Nguyen, Z. Yang and I. Jenkinson (2015) An improved memetic algorithm to identify the robust fleet size of automatic vehicles in container terminals. 20th Annual Logistics Research Network Conference, 9-11 September, Derby, UK.

The following lists materials (or part) of the publications presented in the thesis:

- Chapter 2: publications [1-7]

- Chapter 3: publications [1, 4, 5, 7]
- Chapter 4: publications [4, 5, 7]
- Chapter 5: publications [2, 6]
- Chapter 6: publication [3]

Chapter 2

Literature Review on Fleet Sizing and Simulation in Container Terminals

This chapter will focus on the FSP and simulation in container terminals. The first part (Section 2.1) discusses the FSP in ESTTs which container terminals belong to. It will then explain robust optimisation and its application to the FSP to provide the robust fleet size in ESTTs (Section 2.2). The second part (Section 2.4) focuses on simulation in container terminals. In this part, various simulation studies on container terminals will be reviewed.

2.1 FSP in ESTTs

2.1.1 General view

ESTTs are industrial settings where goods are transferred repeatedly between multiple pickup-and-delivery points (PDPs) by a fleet of vehicles. At each PDP there is a

machine to process the goods. Once goods have been processed, they will be picked up by vehicles and be transferred to another machine for further processing. Next to each machine, there might be a buffer, which is a limited space designed to temporarily store goods in a queue. The purpose of the buffer is to reduce waiting time. Vehicles can drop off goods in the buffer without having to wait for the machines to be available. Machines can also place the goods in the buffer for vehicles to collect later. ESTTs are very common in industrial applications. Typical examples are manufacturing factories, material handling systems, warehouses, container terminals and distribution centres. Readers are referred to Vis (2006) and Le-Anh and De Koster (2006) for more details.

One very important problem in an ESTT is the FSP - identifying the optimal number of vehicles to transfer goods. Having too few vehicles may decrease performance while having too many vehicles is expensive and may introduce deadlocks. This problem is not trivial. In real-world cases, it is highly complex and the optimal fleet size depends on many factors such as the uncertainty in travel time of vehicles; the dynamics of machines' process time; and the size of the buffer. The formal representation of the FSP is provided in Section 3.4.

In the literature, different approaches were proposed for the FSP in ESTTs. These approaches can be categorised into: analytical and simulation based approaches (Steenken et al., 2004; Stahlbock and Voss, 2008; Rashidi and Tsang, 2013; Carlo et al., 2014*b*; Gharehgozli et al., 2014). The analytical approaches use: 1) calculus and queuing network methods and 2) meta-heuristics and exact optimisation methods to tackle the FSP in ESTTs. These approaches will be reviewed in Subsections 2.1.2 and 2.1.3. Section 2.4 will review the simulation-based approaches.

It should be noted that the FSP in ESTTs is very different from the fleet sizing and mixed vehicle routing problem (FS-VRP) and its variants, and hence requires a different solving approach. In the FS-VRP, the objective is to find the optimal routes

for vehicles to minimise the total cost, which usually comprises the cost of routes and cost of vehicles. In contrast, the objective of the FSP in ESTTs is to assign the transportation tasks to an optimal number of vehicles to increase the total throughput of the system. In addition, the ESTTs in the FSP are different to the environments in the FS-VRP in that vehicles have to shuttle among the PDPs. Therefore, existing algorithms used for the FS-VRP may not be suitable for solving the FSP in ESTTs. In fact, so far none of the algorithms for the vehicle routing problem (VRP) has been applied to the FSP in ESTTs. For a recent review of VRP and its extensions, readers are referred to Vidal et al. (2013).

2.1.2 Related work: Calculus-based and queuing network methods

The calculus-based methods use a set of straightforward computations to identify the optimal fleet size. These approaches usually have some static assumptions on the arrival time of goods and speed of vehicles to transport goods between pickup and delivery points. Approaches based on the queuing theory addressed the limitations in calculus-based methods by introducing more variations using the queuing techniques (Choobineh et al., 2012). These approaches, however, are not the focus of this thesis. Below are some attempts on using these methods to tackle the FSP in ESTTs.

Egbelu (1987) provided four calculus-based analytical procedures to tackle the problem of identifying the optimal fleet size of AGVs. Simulation experiments were conducted by incorporating different dispatching strategies to evaluate the performance of the four analytical models. Mahadevan and Narendran (1993) developed an analytical model to determine the minimum number of AGVs in the system. The required working time of AGVs was calculated based on the flows between different

machines and the average rate of each machine. Using the total required time of AGVs and available time of each vehicle, the minimum number of AGVs was calculated. Ji and Xia (2010) proposed an approximate analytical method to estimate the minimum number of vehicles that are required to maintain the system stability. In this approach, the lower and upper bounds of the fleet size were produced within which the stability of the system is guaranteed. The authors then used a binary search to identify the accurate optimal fleet size. To elaborate the proposed method, a numerical example was provided.

Arifin and Egbelu (2000) proposed a regression technique to determine the optimal number of AGVs for automated material handling systems. The main task to develop this model was identifying the independent variables that have significant impact on the number of required vehicles. The layout configuration and travel time were considered two important parameters to determine the optimal fleet size. The required data to create the model was generated using a simulation model which was developed in this study. In similar studies (Fitzgerald, 1985; Muller, 1983; Mahadevan and Narendran, 1990; Sinriech and Tanchoco, 1992; Ilić, 1994) regression methods were used to identify the optimal fleet size based on characteristics of AGVs.

Kuhn (1983) identified the optimal fleet size by calculating the loaded travel time and estimating the empty travel time of AGVs. The author argued that the loaded travel time of AGVs can be calculated directly based on the number and flows of jobs and hence cannot be minimised. However, to identify the minimum number of AGVs, the empty travel time and trips should be minimised. The authors used the first-come, first-served approach and the proportion of load of the pickup point to send empty AGVs to the pickup point. Following this approach the optimal fleet size was identified. This work was then enhanced by Yim and Linnt (1993) showing that the approach in Kuhn (1983) overestimated the fleet size. This was also shown in other research

such as Bartholdi and Platzman (1989) and Asef-Vaziri et al. (2007). Johnson (2001) investigated the impact of empty travel time on the optimal fleet size by developing analytical models to estimate the empty trip distribution under different dispatching strategies. Results showed that the empty travel time has a significant impact on the flow-path design (i.e. the routing path of AGVs) and fleet size of AGVs.

Solberg (1977), Yao and Buzacott (1985), Wysk et al. (1987) and Tanchoco et al. (1987) developed a queuing network to identify the optimal fleet size of AGVs by considering a material handling system as a central resource with aggregated type of customers. Turnquist and Jordan (1986) provided a stochastic model for fleet sizing of containers which transport parts from a manufacturing point to assembly points. Stochastic travel times for containers were considered while the parts production rate was assumed to be deterministic. Trade-off between the number of required fleet size and probability of running out of containers was provided by the analytical model. A two-step analytical method was presented in Raman et al. (2009). Time required for loading, unloading, empty travel, loaded travel and breakdown of material handling equipment was used to produce a preliminary solution in the first step. The second step ranked the solutions generated from the preliminary solutions found in the first step. In this step, performance factors such as utilisation, work-in-process, and life cycle cost were applied to rank the generated solutions. Mantel and Landeweerd (1995) used a hierarchical queuing network approach to determine the minimum number of the vehicle requirement. In Malmborg (1990), an analytical model was proposed to design a control zone of AGV systems. This model was used to measure the impact of decision variables in an AGV system such as the fleet size of vehicles, guide path layout, workstation storage capacity, and vehicle dispatching on the performance of the system. Choobineh et al. (2012) modelled operations of AGVs as a multi-class closed queuing network. The steady-state behaviour of this network was then modelled as a

linear programming problem with the objective of minimising the fleet size. Results of the analytical model were validated by simulation experiments. Comparison results showed that in a majority of cases the analytical model estimated correctly the fleet size.

2.1.3 Related work: Meta-heuristic and exact optimisation methods

This subsection will review some attempts on the FSP in ESTTs that used meta-heuristics or exact methods to tackle this problem. In the exact optimisation approach, a mathematical formulation for the problem is defined. The provided formulation would then be fed into an optimisation solver (e.g. CPLEX[1] and Gurobi[2]) to solve the problem and determine the optimal solution. This approach, however, may suffer computationally when the size of problems increases (i.e. the problems become hard to solve) and hence the solvers might not be able to solve the problems within a reasonable time. In contrast, meta-heuristics (e.g. EAs, guided local search and simulated annealing) can be used in such cases to identify the optimal or near optimal solutions. Thus, a meta-heuristic algorithm is used as an alternative to exact methods to solve approximately a hard optimisation problem without the need to deeply adapt to the problem characteristics to identify the global optima (Boussaïd et al., 2013).

An IP model was proposed in Kasilingam (1991) to determine the optimal number and type of automatic vehicles including assignments of vehicles to the workstations in AGV systems. The objective function in this model was the summation of the annualised operating cost of vehicles and the transportation cost of parts between the workstations. In Koo et al. (2004), a two-phase algorithm was provided for the

[1]See http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
[2]See http://www.gurobi.com/

problem of fleet sizing and routing of vehicles to carry containers in the Busan port. In the first phase, a lower bound for the number of required vehicles was produced by an optimisation algorithm. In the second phase, a tabu search was applied to find the optimal routes of vehicles for the given fleet size in phase one. If all the transportation tasks can be done within the given makespan then this problem is solved otherwise the number of vehicles must be increased by one and the routing algorithm must be run again. Results of the two-phase algorithm were compared to one heuristic algorithm and one greedy procedure.

Rajotia et al. (1998) provided an analytical and a simulation model to determine the optimal fleet size in flexible manufacturing systems. Load handling, empty travelling, and waiting/blocking were considered to be the three different states in which the vehicles spend their time. The load handling time was calculated based on the system parameters and waiting/blocking time was estimated using existing approaches. Empty travel time was estimated using a mixed IP. By estimating the time that vehicles spend on these three states, the optimal fleet size of vehicles can be obtained. A simulation study was conducted to validate the results of the analytical model. This validation showed that the analytical model under-estimated the fleet size, but its results are still close to the simulation results. A bee algorithm was proposed in Sayarshad (2010) to determine the optimal amount of material handling equipment in manufacturing systems. This problem was modelled as a multi-periodic optimisation problem in which the objective function is to maximise profits (calculated as the difference between the revenue and operational cost). Sinriech and Tanchoco (1992) provided a bi-objective goal programming to determine the optimal number of AGVs. A costs versus total throughput decision table was provided to show different solutions with the trade-off ratios between the two goals.

In Vis et al. (2001), a FSP of AGVs in ports was modelled as a minimum flow

problem. A strongly polynomial time algorithm was proposed to tackle this problem. The algorithm, however, follows some assumptions that may not be totally realistic. First, it is assumed that the release time of containers and travel time of vehicles are fixed and deterministic. Second, it is assumed that there is no buffer under the cranes. This may not be true for some new types of AGVs, like IAVs, which are able to work with a buffer under the cranes. This research was followed by Vis et al. (2005) where an IP model was proposed for the fleet sizing of automatic lifting vehicles (ALV) - a special type of automatic vehicles which has the ability to lift. Since ALV has the lifting ability, it is possible to use buffer areas underneath the cranes to reduce the waiting time of cranes and vehicles. The authors solved this IP problem using the CPLEX solver and validated results of the analytical model by results of a simulation study. The results showed that the use of buffers can significantly increase the capacity of the system by minimising the waiting time of machines (cranes in this case).

2.2 Optimisation

2.2.1 Static optimisation

An optimisation problem is usually formulated as in Equation 2.1.

$$\begin{aligned} min \quad & f(x) \\ s.t. \quad & g_i(x) \leq 0, \quad i = 1, ..., I \\ & h_j(x) = 0, \quad j = 1, ..., J \end{aligned} \tag{2.1}$$

where $f(.)$ is the objective function, g_i and h_j are the inequalities and equalities constraints and x is the design variables (i.e. decision variables). Note that here without

the loss of generality it is assumed that this is a minimisation problem. In static optimisation it is assumed that all the parameters in the problem formulation are deterministic and are known in advance. Thus in such cases where all the parameters are available Equation 2.1 can be used to formulate the problem and be solved using conventional methods in literature such as exact and meta-heuristic algorithms. For more details on the problem formulation and approaches to tackle static optimisation problems, readers are referred to text books on operations research such as Hillier and Lieberman (2010); Bazaraa et al. (2011); Murty (1994).

2.2.2 Robust[3] optimisation

Optimisation problems are subject to different uncertainties (Abbass et al., 2009; Bui et al., 2012). Robust optimisation is a methodology to consider uncertainties in optimisation problems in which the performance of results remains relatively unchanged by the presence of uncertainty. The pioneer attempts on robust optimisation were made by Taguchi (1986, 1989), the "father of robust design". Since Mulvey et al. (1995), robust optimisation attracted much attention from the operational research community (Beyer and Sendhoff, 2007). Any real-world optimisation problem may encounter different types of uncertainty and hence depending on the type of uncertainty, the formulation of the optimisation problem should be treated differently. Generally, uncertainty in optimisation problems can be categorised into three groups:

1. The first type of uncertainty is due to changes in general conditions of systems such as changes in environment temperatures, humidity, or material properties (Beyer and Sendhoff, 2007; Chen et al., 1996). Equation 2.2 is the modified form

[3]The term "Robust" in different disciplines might have different meanings. In the thesis, this term has the same meaning as appears in the operations research literature. Readers are referred to the survey paper Beyer and Sendhoff (2007) for detailed information on the definition of the term "Robust".

of Equation 2.1 with an additional input quantity α provided by the environment.

$$\begin{aligned} min \quad & f(x, \alpha) \\ s.t. \quad & g_i(x, \alpha) \leq 0, \quad i = 1, ..., I \\ & h_j(x, \alpha) = 0, \quad j = 1, ..., J \end{aligned} \tag{2.2}$$

2. The second type of uncertainty enters the system due to inaccuracy or malfunction of system components (Beyer and Sendhoff, 2007; Chen et al., 1996). This type of uncertainties was categorised as *Robustness* in Jin and Branke (2005). In this type of uncertainty, the decision variables are under the possibility of being changed after the optimal solution has been determined. Thus, it is of interest that a solution still works satisfactorily after the changes in design variables (Xiong et al., 2013). A good example of this type of uncertainty is changes that may occur due to failures in machines parts or vehicle breakdowns in a manufacturing shop floor. In such uncertain situations it is usually assumed that the possible disturbance δ may happen with the probability of $p(\delta)$ (Gaspar-Cunha and Covas, 2008; Jin and Branke, 2005; Jin and Sendhoff, 2003; Wiesmann et al., 1998) and this will be used to modify Equation 2.1 for robust optimisation. In this type of uncertainty, the expected objective function considering the possible disturbance δ will be used instead of the actual objective function. The expected objective is calculated as in the following:

$$\mathbb{E}[f(x + \delta)] = \int_{-\infty}^{+\infty} f(x + \delta)p(\delta)d\delta \tag{2.3}$$

However, calculating the accurate expected objective function (Equation 2.3) is not usually possible. Thus, alternative approaches should be followed to approximate the expected objective function. Monte Carlo sampling is an alternative

approach that is usually used to approximate the expected objective function. In this approach, different samples of the problem by incorporating the possible uncertainty are generated and then the value of the objective function for each sample is calculated separately. The average of the objective values on all the samples is considered the expected objective function. Thus, the expected objective function can be approximated by Equation 2.4 (Jin and Branke, 2005):

$$\widetilde{f} = \frac{1}{N}\sum_{i=1}^{N} f(x+\delta) \tag{2.4}$$

In addition to the approximation of the expected objective function, the variance of objective function is approximated to show the stability of solutions under uncertainty (Delage and Ye, 2010). This is shown in Equation 2.5.

$$Var(f(x+\delta)) = \frac{1}{N}\sum_{k=1}^{N}(f(x+\delta_k) - \widetilde{f})^2 \tag{2.5}$$

Given the nature of the FSP in ESTTs, this type of uncertainty may have a significant impact on the optimal fleet size. This type of uncertainty is the focus of this thesis (Chapters 3 and 4).

3. Inaccuracy in the objective function introduces uncertainty to the problem. This can happen due to using models instead of the physical objects (Beyer and Sendhoff, 2007). In addition, in some cases where calculating the exact objective function is very expensive or not available, the objective function is estimated using the generated data from experiments or simulation (Jin and Branke, 2005). The modified objective function for this type of uncertainty is shown in Equation 2.6.

$$\widetilde{f}(x) = f(x) + E(x) \tag{2.6}$$

where $E(x)$ is the approximation of the objective function error.

For further information on robust optimisation and approaches to take into account uncertainty, readers are referred to survey papers on this topic (Beyer and Sendhoff, 2007; Jin and Branke, 2005; Ben-Tal and Nemirovski, 2002; Bianchi et al., 2009).

2.2.3 Robust solutions

In real-world problems (e.g. from scheduling, fleet sizing to design optimisation problems) identifying robust solutions is very important. The robustness of solutions is evaluated based on the insensitivity of solutions in response to the small changes of design variables or environmental parameters. To search for robust solutions two approaches are usually followed: 1) optimising the expected fitness based on a given probability distribution of disturbance and 2) multi-objective approaches to search for non-dominated solutions to optimising different objectives such as expected fitness, fitness variance, fitness objectives etc (Jin and Branke, 2005).

1. **Optimising Expected Fitness:** As explained in Subsection 2.2.2, Equation 2.4 is used to approximate the expected fitness. To do so, the average fitness of a solution on a number of randomly generated samples including the uncertainty is calculated and considered the expected fitness. Thus, the higher the number of samples, the more accurate the estimated fitness. However, the process of sampling is sometimes very time consuming (i.e. computationally expensive). Therefore, in literature different approaches were proposed to alleviate this issue by using the lower number of samples to reach the required accuracy. For example, in Marseguerra and Zio (2000), Marseguerra et al. (2002), Marseguerra et al. (2007) and Cantoni et al. (2000) a drop-by-drop approach for EAs was used to incorporate the results of sampling from previous generations of indi-

viduals to calculate the accumulative results of sampling over a certain number of generations. There were also some attempts to use different approaches in generating samples such as Latin hypercube sampling (Helton and Davis, 2003; Dehghan et al., 2014) to enhance the sampling process. In this approach, the sample space is divided into different regions and samples are generated almost evenly from these regions. Thus, with the lower number of samples, the broader area of the sample space can be covered. This will also prevent the generation of biased samples focusing on specific areas of the samples space. In the thesis, the sampling technique is used to develop a robust EA (Chapter 3). In Chapter 4, a new approach is proposed to reduce the number of the required samples to enhance the performance of the proposed EA in Chapter 3.

2. **Multi-objective Approach:** For some problems considering only the expected fitness may not be enough (Jin and Sendhoff, 2003; Deb and Gupta, 2006) to ensure that the robust solution has a stable fitness. Thus, considering the fitness variance in addition to the expected fitness value may fill this gap by searching for non-dominated solutions that optimise both expected mean and variance of fitness value (Deb and Gupta, 2006; Ray, 2002). For example, Lee et al. (2011) used expected mean and variance fitness as the two objectives of an aerospace engineering design problem.

The two approaches above were widely used in the literature to search for robust solutions. The next subsection will review some relevant papers that dealt with robust optimisation.

2.2.4 Robust scheduling and planning

To the best of our knowledge, there is no existing research that takes into account uncertainties in the FSP in ESTTs to offer a robust optimisation solution. However, in the literature the impact of uncertainties on other problems was considered to provide robust solutions (Gabrel et al., 2014). Among the robust optimisation research, the studies on robust job-shop scheduling and planning are the most relevant research to the robust FSP in ESTTs because in both classes of problems, the optimisation algorithms search in the space of schedules of jobs to optimise an objective such as makespan, cost, fleet size, etc. In job-shop scheduling problems (JSPs), there are a number of operations that need to be assigned to a machine with specific starting and ending times to minimise the makespan. In JSPs, the process time of operations and breakdown of machines have been considered to be the main sources of uncertainties that can have a significant impact on the makespan. Different robustness measures were used in robust JSPs to evaluate the baseline schedule. These measures are usually based on the float time, tardiness and expected values of makespans under uncertainty. Due to the similarity between the robust FSEA and JSP, this subsection reviews some research on robust optimisation for scheduling and planning as related work to the robust FSP. Note that there are also some studies to tackle the FSP under uncertainty in other applications (e.g. military (Wojtaszek and Wesolkowski, 2012; Abbass et al., 2009, 2011, 2008), maritime (Alvarez et al., 2011; Meng and Wang, 2010; Meng et al., 2012; Pantuso et al., 2014) and rail (Milenković and Bojović, 2013; Sayarshad and Tavakkoli-Moghaddam, 2010)), however, due to the differences between the nature of those problems and that of the FSP in ESTTs, these studies are not reviewed here.

Xiong et al. (2013) dealt with the flexible job-shop scheduling problem (FJSP) under uncertainty of machines breakdown. The authors considered this problem to be a bi-objective problem in which makespan and expected delay of the schedule due

to uncertainties are the two objectives. To take into account the uncertainties, two robustness measures to evaluate the robustness of schedules (solutions) were proposed. In the first measure, the float time of machines with heavier workload was given higher weights, to prefer solutions with higher float time for heavier workload machines. The second proposed measure evaluates solutions based on the degree of overlap between the machines float times with their possible breakdown duration. The higher the degree of the overlap, the better machines can use their float times to cover their failures. The float time is the difference between the earliest start time and latest start time of an activity. An EA from literature was used to compare the proposed robustness measures with three measures from the literature. Zuo et al. (2009) developed a multi-objective immune scheduling algorithm with the optimality and robustness of schedules as the objectives. The authors used simulation to evaluate the robustness of solutions. Shadrokh and Kianfar (2007) proposed a genetic algorithm (GA) for the resource investment project scheduling problem. In this algorithm, delayed completion of the project was allowed. The objective of this algorithm was to optimise the sum of resource availability costs and tardiness penalty. Kılıç et al. (2008) developed a bi-objective GA to minimise the makespan and total cost for a FJSP. Some papers (Al-Fawzan and Haouari, 2005; Kobylański and Kuchta, 2007; Chtourou and Haouari, 2008) used the free slack as the robustness measure to identify the robust schedules.

Nguyen et al. (2014) developed four genetic programming-based hyper-heuristic algorithms to identify non-dominated scheduling policies based on the makespan, normalized total weighted tardiness and mean absolute percentage error. In these algorithms, dispatching rules and due-date assignment rules in job shop environments were considered simultaneously. The fitness of individuals in these algorithms was evaluated using a simulation model. The authors also proposed a diversified multi-objective cooperative evolution to handle multiple scheduling decisions simultaneously. Mahdavi

et al. (2010) developed a real-time simulation-based decision support system for production control of the stochastic FJSP under stochastic processing time of machines. Wang and Yu (2010) proposed a beam search-based heuristic algorithm to tackle the FJSP with machine availability constraints due to maintenance activities. Moradi et al. (2011) developed a bi-objective GA to minimise simultaneously the makespan and system unavailability. Al-Hinai and ElMekkawy (2011) proposed a two-stage hybrid GA for the FJSP problem under random machine breakdowns. The authors defined a number of bi-objective measures combining the robustness and stability of schedules. In Xiong et al. (2012), a knowledge-based multi-objective EA was proposed to tackle planning problems. The authors modelled planning problems as a resource investment project scheduling problem. Three sources of uncertainty were considered for this problem: duration of perturbation, resource breakdown and precedence alteration. A new robustness measure that takes into account the above uncertainties was proposed. The robustness of solutions was evaluated using the generated scenarios. Gu et al. (2010) proposed a co-evolutionary quantum GA for the stochastic FJSP to optimise the expected makespan. Sevaux and Sörensen (2004) modified the GA to compute robust machine schedules given uncertainties. New robustness measures were defined to evaluate solutions based on the robustness and distance to the baseline solutions.

2.3 Discussion on the gap of knowledge on the FSP in ESTTs

The literature review above reveals three possible gaps in current research of the FSP in ESTTs. First, despite the supposed usefulness of buffers in PDPs in reducing waiting time in ESTTs, very few existing methods actually consider buffers in the

FSP[4]. Second, most existing deterministic/meta-heuristic optimisation methods do not consider the impact of uncertainties on the optimal fleet size. Third, there is a clear lack of computational intelligence techniques, particularly EAs, in solving this problem. Given that EAs have been widely used to find approximated solutions for complex, non-linear, large-scale problems in both static and uncertain/dynamic cases (Nguyen et al., 2012; Cruz et al., 2011; Nguyen, January 2011), it is of interest to investigate how EAs can be used to address the aforementioned gaps in this thesis. The attempt to bridge the above gaps will be presented in Chapters 3 and 4.

2.4 Simulation in container terminals

2.4.1 What is simulation?

Simulation is a scientific methodology to study the behaviour of a system without imposing any interruption to the real system. Simulation is usually developed to experiment different scenarios in the system without any direct capital investment (Demirci, 2003) and hence it is a money and time saving approach. Using simulation, the effect of possible changes to system can be predicted before they actually happen to the system. Simulation is also used to evaluate the performance of system components and impact of introducing new elements to the system (Ha et al., 2007). Moreover, simulation is a very proper tool for dealing with complex systems when it is not possible to model a system due to the complexity or when it is very expensive to model the system (Yun and Choi, 1999). Thanks to the power of problem decomposition and parallelism of simulation, any complex system can be modelled and simulated using simulation. Operations of trans-shipment of containers in a container terminal are an

[4]Only Vis et al. (2005) took into account buffers but the considered buffer size is very small (up to two).

example of such a complex system and hence simulation has been widely used to deal with this type of complex system (Henesey, Aslam and Khurum, 2006).

Simulation models can be categorised based on their characteristics in three categories (adapted from Angeloudis and Bell (2011)):

1. State (static vs dynamic): Static simulations only consider the state of a system at a specific time. This type of simulation contains a number of inputs with specific equations to calculate the output. In contrast, in dynamic simulation, the behaviour of a system over a certain duration of time is considered. The dynamic simulation is more realistic and accurate compared with the static simulation, given that any real world environment is subject to dynamic changes. In container terminals simulation research, in a majority of cases the dynamic simulation was used.

2. Fidelity to details (microscopic vs macroscopic): Depending on the purpose of simulation, the fidelity to the details of real systems can be different. For strategic objectives, simulations are developed in a macro level with high abstraction, given that the aggregations and global dependencies are important in strategic decision making problems. In contrast, in a micro level, individual objects, exact sizes and distances are very important parameters that have a significant impact on the simulation results (Borshchev and Filippov, 2004). In the literature usually high fidelity simulations were developed to study container terminals.

3. Timing (continuous vs discrete): In physical sciences and finance usually continuous simulation is used whereas discrete simulation is widely used in logistics, manufacturing and container terminals. If any state of a system at any point in time is required, continuous simulation should be developed. In this type of simulation, usually by the help of differential equations, the continuous changes

in the system will be captured and simulated. In contrast, discrete simulation assumes that the changes in systems are based on discrete manners. However, discrete simulation can be categorised into two types: *discrete time* and *discrete event*. In discrete time simulation, the simulation time is divided into fixed steps and at the start of each step the state of objects in the simulation model will be calculated. In this type of simulation, the higher the number of the steps, the more accurate the simulation. However, in this approach by increasing the steps the complexity of the simulation will be increased. In the discrete event simulation, there is an event manager to list all the upcoming events to perform each event based on their order in the list. In this type of simulation, it is assumed that between the events no other events would happen and the simulation clock will skip the time between the events. The discrete event simulation is the focus of the thesis.

The following sections discuss the simulation studies in container terminals.

2.4.2 Simulation in container terminals

Due to advances in computer technologies and the establishment of computer simulation as a robust tool for the study of container terminals, simulations have been extensively used for research purposes in container terminals (Sun et al., 2012). The following sections review papers that studied container terminals using simulation approaches. The papers that developed new simulation software/libraries throughout their research are reviewed first. The papers that used existing simulation software/libraries to develop their simulation models are reviewed afterwards. The papers are categorised based on their applications. Angeloudis and Bell (2011) provided a comprehensive survey on the simulation studies in container terminals. The authors reviewed different existing tools for the simulation of container terminals. This pa-

per also categorised and reviewed the simulation studies based on their case studies. Readers are referred to this survey paper for further details.

2.4.3 Simulation software/libraries

In the literature, many papers used existing simulation software/libraries to develop their simulation models to study container terminals. The following simulation products were mostly used in those papers: Arena, Plant, Automod, Witness, FlexSim, Simeview, Simprocess, Simfactory, Taylorand and MODSIM. These software products are generic simulation software with standard simulation objects that can be used for container terminals or any other applications such as simulating manufacturing shop floors, warehouses and health care systems. In addition to these software products, there are some simulation programming libraries that were used in the literature to develop simulation models such as TOMAS, DESMO-J, Must,Visual SLAM and AweSim. Developing simulation models with these libraries requires more efforts and coding compared with simulation software, since these libraries provide a minimal simulation framework and the remainder of the simulation model should be developed by users. Note that there are some commercial simulation software products that were developed specifically for container terminals such as SCUSY, TBA and PosPort CTS. These products, apart from SCUSY, were used mainly by the owner companies to offer consultancy services (Angeloudis and Bell, 2011). To the best of our knowledge, these software products were hardly used for simulation research in the literature.

In the literature there are some simulation packages that were developed specifically for container terminals. In Sun et al. (2012, 2013), a general simulation platform was proposed for the simulation of container terminals with a 3D visualisation feature. This platform was built upon MicroCity which is a spatial modelling framework for scientific analyses. This platform consists of three layers: function, application and

extensions layers. The function layer has a discrete-event scheduler to schedule events. The operations in response to events are performed by a multi-agent system in the application layer. In this layer, the interactions between equipment were considered to prevent collisions between equipment by taking into account the safety spacing between the objects. In the extensions layer, users can develop their desired optimisation algorithms and interact with the platform. A container terminal in Singapore was simulated using the proposed simulation platform and the gross crane rate was considered to be the measure to evaluate the performance of the case study. Bielli et al. (2006) developed a distributed discrete event simulator for container terminals using the Java programming language with a 2D visualisation feature. Relations between different entities of the simulator and the way events are managed were explained by providing unified modelling language diagrams. This simulator was calibrated using realistic data from the Casablanca container terminal in Morocco. Angeloudis and Bell (2010) developed the Limen terminal simulator. This simulator consists of two design layers: terminal emulation and terminal operation layers. The former layer deals with the physical behaviour of terminal equipment whereas the latter handles the operational decisions and controls. This software has the ability to be integrated with some optimisation software such as the Excel's solver and the IBM ILOG CPLEX optimisation software. It also benefits from 3D simulation visualisation tools.

Nevins et al. (1995); Nevins, Macal and Joines (1998) developed a port simulator, named PORTSIM, using the modular simulation language. This simulator was originally developed for the military mobility analysis to investigate the turnaround of military vessels to load and discharge military equipment in ports. This simulator supports different types of cargo such as breakbulk, container and roll-on/roll-off. This simulator was then improved in Nevins, Macal, Love and Bragen (1998) by adding the animation and 3D visualisation features to the simulator. The Java programming lan-

guage was used to develop animation and the authors used MicroStation and ModelGel to build the 3D representation.

2.4.4 Related work: Applications of simulation in container terminals

Comparing different transfer vehicles

One of the applications of simulations in container terminals is to compare the performance of container terminals using different types of transfer vehicles to transport containers inside terminals. Carlo et al. (2014*b*) classified the types of vehicles in container terminals into self-lifting and non-lifting vehicles. Self-lifting vehicles have the ability to pick up and drop off containers by themselves. IAVs, shuttle carriers, straddle carriers and ALVsare examples of self-lifting vehicles. Among these vehicles only straddle carriers can stack/unstack containers to/from more than one tier (usually up to four tiers). In contrast, non-lifting vehicles need external material handling equipment to load/unload containers to/from them. Examples of such vehicles are yard trucks and AGVs. With the lifting ability, the buffer of containers under cranes can be utilised to reduce the waiting time of cranes and vehicles. Among all the transfer vehicles, yard trucks are the most commonly used vehicles in container terminals, due to their low investment cost and ease of deployment. It should be noted that, the automatic vehicles (e.g. IAVs, ALVs and AGVs) need a central control system to manage the dispatching and movement of vehicles. In addition to this system, ALVs and AGVs also require special infrastructure under the ground to follow the paved track. This will increase the level of investment for the automatic vehicles significantly compared with manual vehicles. Thus, due to differences between vehicles' capability and required amount of investment, the performance and cost of container terminals

with each type of vehicles can be different and hence it needs careful investigation.

The following papers used simulation to compare the performance of container terminals with different types of transfer vehicles. Vis and Harika (2004) compared two different types of automatic vehicles, namely AGVs and ALVs by developing simulation models using the Arena simulation software. The authors identified the optimal number of AGVs and ALVs following a sensitivity analysis approach by considering the total discharging time of a vessel as a measure. In Duinkerken et al. (2006) and Ottjes et al. (1996), the Must simulation library was used to compare the performance of a container terminal using three different types of inter-terminal vehicles: Multi-Trailer-System (a man driver system with a chain of container-bearing trailers), AGVs and ALVs. Lee et al. (2007) compared the performance of a container terminal using prime movers (i.e. trucks) and shuttle carriers. The gross crane moves was set as the performance measure in this paper. In Liu et al. (2002), the authors compared four different types of equipment in an automatic container terminal. The four sets of equipment are AGVs, a linear motor conveyance system, an overhead grid rail system and a high-rise automated storage and retrieval structure. The authors developed a simulation model and used a cost model from the literature to compare the four types of equipment based on the average cost per container (calculated using the cost model). The results showed that with AGVs the automated container terminal can achieve the least average cost per container.

Evaluating different terminal layouts and vehicle travel routes

Transfer vehicles transport containers between quay and stack cranes. The travel routes of transfer vehicles between quay and stack cranes can greatly influence the travel time of vehicles inside the terminals (Carlo et al., 2014*a*). Each layout of the terminal may lead to different travel time of vehicles between two pickup and delivery

points. Thus, enhancing the layout of container terminals can effectively reduce the travel time of vehicles inside the terminals. The following studies used simulation to study the impact of different terminal layouts on the performance of container terminals. Liu et al. (2004) compared the performance of the Norfolk international terminal, USA, with two proposed layouts for AGVs using simulation models. The simulation was developed using the Matlab software. Control logic to prevent deadlock and collision was included in the simulation. A multi-attribute decision making method (MADM), the simple additive weighting method, was used to evaluate the performance of the container terminal given different measures such as average waiting rate of AGVs, average idle rate of AGVs, average stop rate of AGVs and total throughput of the container terminal. This MADM assigns a weight to each performance measure and by comparing the weighted value of each measure, the MADM identifies the optimal measure. The optimal measure was used to identify the best layout based on the results of the simulation.

In Kia et al. (2002), the current layout of an Australian port was compared with a newly proposed layout. In the proposed model a ship-to-rail direct loading approach was proposed to move containers directly from the berth to a distribution centre by trains. Results showed that using the proposed method the total occupancy of berth/yard was decreased compared with the current conventional method. In this paper, a collection of simulation tools such as Simeview, Simprocess, Simfactory, and Taylor was used.

In Lee et al. (2008), the authors provided a tool to generate layouts automatically for simulation models based on the given input parameters. To generate the simulation layout, it is required to set input first and the generator will then update the layout automatically based on the given input. The AutoMod simulation software was used to test the proposed layout generator.

Comparing different strategies/scenarios

The following papers used simulation to evaluate the impact of different possible scenarios/strategies on the performance of container terminals. Nam et al. (2002) developed a simulation model for the Gamman container terminal in Busan, South Korea. In this container terminal, four berths are available and each of the berths is dedicated to only one company for its import and export activities. The total throughput of each berth can be different from the other berths due to the differences in the schedule of ships. It can be a case where some of the berths are idle while ships queued to be served in other berths. In this paper, sharing berth scenarios between different terminal operators were proposed to increase the productivity of the container terminal. Using the developed simulation model, the authors investigated productivity of the terminal based on the proposed scenarios. Results of the simulation showed that using the berth-sharing scenario the productivity of the container terminal can be increased significantly.

Lee et al. (2003) modelled the Busan East container terminal as a supply chain network. The Arena simulation software was used to simulate the container terminal. Using this simulation model the authors investigated the impact of two different types of strategies on the performance of the container terminal. In the first strategy, the authors compared the impact of having enough resources such as quay cranes and vehicles with utilising speedy quay cranes. The results showed that speedy quay cranes provide shorter container handling time. In the second strategy, the impact of having more accurate vessels arrival time was investigated. The accurate data can be achieved by regular communication with the arrival vessels. The results showed that having accurate data can help the terminal to be better prepared in advance to be able to decrease the ship staying time.

In Soriguera et al. (2006), four types of transportation tasks were considered for

the straddle carriers: loading and unloading in the quay and gate sides. The way straddle carriers are assigned to each task type might have a significant impact on the jobs process time and on the empty travel time of straddle carriers. In addition, the allocation of containers to different slots in the yard can decrease or increase reshuffling of containers in the yard, and hence the productivity of unloading and loading activities might be changed. A simulation model was developed to analyse the aforementioned assignment of straddle carriers and allocation of containers strategies. A simplified model of the Barcelona container terminal was considered the case study of this simulation study. However, the simulation tool to develop the simulation model was not revealed. Moreover, the authors did not provide the results of the simulation.

A simulation model was developed in Hadjiconstantinou and Ma (2009) to evaluate the performance of straddle carriers in the Port Pireus container terminal. The simulation model was developed using the C# programming language. Three different policies for deployment of straddle carriers were considered. In the first policy, each straddle carrier was assigned to one stacking block. In the second policy, straddle carriers were shared between all the stacking blocks and in the last policy, each straddle carrier was assigned to two blocks. The results showed that the third policy provided better performance for the container terminal.

In Parola and Sciomachen (2005), the logistics chain connected to the Genoa and La Spezia ports in Italy was studied by developing a simulation model. The simulation model was developed using the Witness 2000 simulation software. In this model, not only the operations in ports, such as operations in the yard, quay side and transportation of containers inside the port were simulated, but also the roads and rails in the region connected to the ports were included in the simulation. Snapshots of the simulation model and some logical codes for the train transportation and ship berthing were provided. Due to the increase in the demand of the ports in future, the capacity

of inland transportation using road and rail will not be sufficient to handle the future demand of the ports effectively. As a result, three different scenarios were proposed to increase the capacity of the inland transportation i.e. rail and road that connects to these two ports. Among the studied scenarios, the one which gives a balanced transportation of containers between road and rails was considered the best favoured option.

In Briskorn et al. (2006), an inventory-based dispatching strategy was proposed to assign AGVs to containers. In this approach quay cranes were considered customers and AGVs were considered goods to satisfy demand of quay cranes i.e. to collect/drop off containers from/to quay cranes. The objective was to maintain the inventory of AGVs at a satisfactory level to minimise the waiting time of quay cranes. To evaluate the proposed approach a simulation model using the DESMO-J simulation framework was developed. The HHLA container terminal Altenwerder in Hamburg was considered the case study of this paper. However, due to the confidentiality of the data, the authors did not reveal the simulation input data and the distributions that were used in the simulation. Different variants of the dispatching strategies were tested to identify the best strategy.

Cortes et al. (2007) simulated the Seville inland port using the Arena simulation software. Spatial movements were not considered in this simulation. This port consists of three docks. Vessels can access the port through the Guadalquivir estuary and a lock by which the river is connected to a harbour. The port can deal with different types of cargo and each type of cargo is handled differently in a specific dock and berth. Different scenarios to handle cargo vessels were tested with the simulation. The simulation model was explained in detail by providing the detailed simulation modules such as vessel arrivals, dock assignment, vessel departure and lorry arrivals modules in addition to modules regarding the handling of each specific type of cargo

in the docks. Using the given performance measures such as containers per hour and tons per hour the traffic flow of the port was analysed.

Identifying the optimal settings and number of vehicles

This subsection presents the papers whose main focus is identifying the optimal settings and amount of equipment in container terminals. In Henesey, Aslam and Khurum (2006), a simulation model using the DESMO-J library was developed to evaluate an improved AGV system named IPSI AGV. The authors followed an agent based approach in which quay cranes, cassettes, containers and AGVs were considered to be the agents of this simulation model each with specific attributes and functions. A cassette is a table-like object which is proposed along with the IPSI AGV system, this is a similar concept to the cassette concept in the IAV system. The improved AGV combined with the cassette can pick up and drop off containers by itself. As a result, a buffer of containers under cranes can be utilised to reduce the waiting time of cranes and vehicles. Using realistic data sensitivity analysis was conducted to identify the optimal number of quay cranes, AGVs, and cassettes to minimise the container handling rate and maximise the terminal equipment utilisation rate.

The Plant simulation package was used in Ha et al. (2007) to simulate a container terminal. Two sets of objects were considered in this simulation: material flow objects and moving unit objects. The authors considered quay cranes, yard cranes transporters, external trucks, container vessel and containers to be moving unit objects. The material flow objects are the objects that generate, destroy and route the moving unit objects. No specific case study for this research was considered. However, the authors simulated an automatic container terminal that benefits from AGVs. AGVs motions were represented using virtual tracks in the simulation. The berth productivity was used as the performance measure to identify the optimal settings i.e. the

amount and specification of equipment following a case-sensitivity analysis. Lin et al. (2014) developed a simulation model using the Arena simulation software to identify the optimal settings to minimise the total cost to reach the required level of service. This paper applied a flexible berth allocation rule to make it possible for vessels to stay longer in the berth as long as there is enough space and resources. In this simulation bigger vessels were given a higher priority, given that their contributions to the port profit are higher. The case study in this paper was the Humen port in China.

Yun and Choi (1999) developed a simulation model using the SIMPLE++ object oriented simulation software developed to analyse the performance of Busan East Container Terminal. For the sake of simplicity, the authors considered a smaller case of the container terminal. The authors considered gate, container yard and berth in the simulation model. The objects in the simulation are divided into two groups: 1) moveable objects: they can change their positions and reside in other material flow elements; 2) stationary objects: objects with fixed positions. Detailed information regarding classes and methods in the simulation model was provided. The authors also provided details of the input data that were used in the experimental study. A number of relevant performance measures were provided to evaluate the performance of the container terminal. Result of the analysis was used to investigate whether this container terminal can handle a higher number of containers using existing equipment or it needs new equipment to be purchased, such as transfer vehicles and gantry cranes, to increase the capacity of the container terminal.

Various purposes

In the following papers the simulation approach was used for various purposes such as validating results of optimisation studies, estimating the total throughput of container terminals, evaluating the performance of container terminals using the given

performance measures etc. In Gambardella et al. (1998), a simulation model was developed to validate results of optimisation algorithms for allocation of resources in the La Spezia container terminal. The optimisation algorithm was developed using the LP_SOLVE library. The simulation model was validated using realistic data from the container terminal. The results of simulation showed that the results of the optimisation algorithms are valid and accurate, and hence can be used as a decision making tool in the port. Details of the yard and berth planners modules of the simulation tool were provided, however, the detailed technical information regarding the optimisation algorithms was not discussed. Gelareh et al. (2013) developed a Lagrangian relaxation-based decomposition algorithm to schedule IAVs in container terminals. In this algorithm, the pairing feature of IAVs was taken into account by which two 1-TEU (20-foot Equivalent Unit) IAVs can make a dynamic joint to be able to carry together a container with any size between 1-TEU and 1-FEU (40-foot Equivalent). The output of the algorithm was simulated using the FlexSim simulation software.

In Demirci (2003), a simulation model was developed using the AweSim computer simulation language to identify the bottlenecks in the Trabzon port in Turkey. The simulation model was developed based on realistic data which were provided in detail in the paper. To identify the bottlenecks, in addition to the existing state of the port, a situation where the full-capacity of the port is used was simulated. In the full-capacity situation, loading/unloading vehicles were considered to be the bottlenecks that impair the performance of the port. Due to the limitation in the port investment only a limited number of vehicles could be added to the fleet. As a result, performance of the port by considering the few added vehicles was investigated by the simulation model. The results showed that the performance of the port would be enhanced with the additional vehicles.

In Shabayek and Yeung (2002), the Kwai Chung container terminal in Hong Kong

was simulated using the Witness simulation software. This container terminal consists of four different operators to serve the incoming vessels. Each operator has a specific number of terminals. The purpose of this simulation is to estimate the vessel waiting time and degree of utilisation of each operator. In this simulation, the operations inside the terminals have not been simulated and the service time of vessels was estimated using the historical data. Henesey et al. (2002, 2003); Henesey, Davidsson and Persson (2006) developed agent-based simulation models to study the container terminals management system. Ottjes et al. (1994) developed a simulation model using the Must simulation library to reduce ships turnaround time and optimise the inter-terminal transportation of containers using a sailing container terminal in the port of Rotterdam. Ottjes et al. (2007) used the TOMAS toolkit to tackle the problem of transportation of containers between terminals of the port of Rotterdam. In Rebollo et al. (2000) a prototype of an agent-based system for simulation of container terminals was proposed. Kotachi et al. (2013) proposed a model for simulation of container terminals using the Arena simulation software. Hartmann (2004) proposed an approach to generate scenarios that can be used in simulation studies and optimisation problems in container terminals. In the scenarios deep sea vessels, feeder ships, trains and trucks arrivals, list of containers including the attributes of containers such as size, weight, empty, reefer, destination and dwell time are generated using the given parameters from users. The algorithm and input parameters to generate scenarios such as means of transportation (e.g. vessel, train and truck), arrival frequencies and container properties were explained in detail. The generated scenarios were validated with realistic data from the HHLA Container Terminal Burchardkai in Hamburg, Germany. The generated scenarios were used to study container stacking strategies in HHLA container terminal Altenwerderin Hamburg, Germany using a simulation model. The simulation model was developed using the emPlant software.

The equipment such as quay cranes, automated guided vehicles and stacking cranes were not modelled explicitly, but stacking strategies and stacks were the main focus of the simulation. The results of the simulation, however, due to confidentiality reasons, were not revealed. Klaws et al. (2011) developed a simulation model using FlexSim CT to investigate the possible improvement in the vessels turnaround time if a triple rail mounted gantry crane (TRMG) is used in a container terminal. A TRMG is a combination of three rail mounted gantry cranes to be assigned to one stacking block where these three cranes work in synchronisation with each other. The authors set out some rules for the TRMG to prioritise their work and also prevent any collision between the cranes.

Veenstra and Lang (2004) developed a simulation model to provide an economic analysis on a container terminal. A typical container terminal similar to the Delta container terminal in Rotterdam was modelled using the DSOL library in Java which consists of environments and transformation systems. The simulation model was combined with an economic appraisal model to analyse the performance of the container terminal with respect to some economic factors such as the investment policy, cost structure, income structure and net cash flow. The authors claimed that their economic approach is not limited only to container terminals and can be extended to any logistic systems. The economic appraisal model was a spreadsheet to calculate the financial figures using the results of the simulation for long term periods. Detailed simulation input data such as the number of automated stacking cranes, number of AGVs and other specifications of the container terminal were provided. However, the paper does not provide any detail of the economic appraisal model. The authors also had difficulties in the integration of operational and economic simulation models due to the differences in the ways the two models deal with time. The operational simulation model is event based while the economic simulation is time-step based. To overcome

this issue, the author proposed an approach to integrate the two simulation models by aggregation over objects and de-aggregation over time.

The major simulation papers that were reviewed in this section are summarised in Table 2.1.

2.4.5 Discussion on simulation studies in container terminals

The reviewed papers above reveal that there are few simulation tools available that were specifically developed for container terminals. Thus, many researchers in the community had to use generic simulation software packages to create container terminal simulation models. This adds much additional work to researchers adapting those tools for container terminals. This indicates a clear lack of enough simulation software/tools with built-in container terminal simulation objects. Thus, the thesis is trying to close this gap by proposing a new simulation framework for container terminals. This framework contains standard container terminal simulation objects that are needed to simulate any container terminal. This framework also gives the flexibility to users to develop their optimisation algorithms and incorporate them into their developed simulation models. This framework will be explained in Chapter 6.

Table 2.1: Simulation studies in container terminals

Reference	Purpose of simulation	Case study	Software /library	Buffers utilised?	Performance measures
(Ottjes et al., 1994)	Optimising inter-terminals transportation using a sailing container terminal	Port of Rotterdam, Netherlands	Must	No	Vessel turnaround time.
(Ottjes et al., 1996)	Comparing the impact of utilising a Multi-Trailer system, AGVs and ALVs on the performance of the case study	Port of Maasvlakte, Netherlands	Must	Yes	1) non performance: the number of cranes/vehicles divided by the terminal capacity; 2) the number of moves per hour for cranes; and 3) the service rate versus the system cost.
(Gambardella et al., 1998)	Validating the results of optimisation of resource allocation problems	La Spezia Container Terminal, Italy	Not specified	No	Vessel turnaround time.
(Yun and Choi, 1999)	Identifying the optimal number of vehicles/cranes that are needed to reach the satisfactory level of performance	Busan East Container Terminal, South Korea	SIMPLE++	No	1) the utilisation rate of equipment; 2) the container yard occupancy rate; and 3) average vessel waiting time for berth.

Reference	Purpose of simulation	Case study	Software /library	Buffers utilised?	Performance measures
(Liu et al., 2002)	Comparing the cost of utilising AGVs vs a linear motor conveyance system vs a high-rise automated storage and retrieval structure	Port of Rotterdam, Netherlands	Details of simulation were not provided.	Yes	1) moves per hour per quay crane; 2) throughput per acre; 3) vessel turnaround time; 4) the idle rate of equipment; 5) truck turnaround time; 6) the gate utilisation rate; 7) container dwell time; and 8) the authors developed a cost model to calculate the average cost of moving a container.
(Nam et al., 2002)	Investigating the sharing berth scenarios between different companies that operating the berths	Gamma Container Terminal in Busan, South Korea	AweSim	No	1) average port time; 2) average berth time; 3) the average berth occupancy ratio; 4) average waiting time; 5) the average number of cranes per vessel; and 6) the authors developed a cost model to calculate the total annual cost.
(Shabayek and Yeung, 2002)	Investigating the accuracy of simulation models to predict the actual operations in container terminals	Kwai Chung Container Terminal, China	Witness	No	1) vessel waiting time; and 2) the degree of utilisation of each operator to serve the incoming vessels.
(Kia et al., 2002)	Comparing the impact of two layouts	Not specified	Simeview, Simprocess, Simfactory and Taylor	No	1) occupancy of berth/yard; 2) containers dwell time; and 3) comparing the cost of the case study with the two layouts.

Reference	Purpose of simulation	Case study	Software /library	Buffers utilised?	Performance measures
(Lee et al., 2003)	Studying different strategies: 1) utilising speedy quay cranes vs using enough quay cranes; 2) impact of using more accurate vessel arrival times	Busan East Container Terminal, Korea	Arena	No	1) container handling time; 2) the number of vessels to be served; 3) service time per berth; and 4) the berth utilisation rate.
(Demirci, 2003)	Identifying bottlenecks that influence the performance of the case study	Trabzon Port, Turkey	AweSim	No	1) vessel service time; 2) vessel waiting time; and 3) vessel turnaround time
(Vis and Harika, 2004)	Identifying and comparing the optimal number of AGVs and ALVs	An automated container terminal	Arena	Yes	1) total discharging time of vessels; and 2) the purchasing cost of the optimal number of vehicles.
(Liu et al., 2004)	Comparing the impact of two layouts	Norfolk International Terminal, USA	Matlab	No	The authors used MADM to identify the most important performance measure among a list of performance measures: 1) the average stop rate of vehicles; 2) the average idle rate of equipment; 3) the average waiting rate of equipment; and 4) the number of loaded containers per hour per quay crane.

Reference	Purpose of simulation	Case study	Software /library	Buffers utilised?	Performance measures
(Hartmann, 2004)	Proposing an approach to generate scenarios for the simulation of container terminals	HHLA Container Terminal Al-tenwerder Hamburg, Germany	emPlant	No	The results of simulation were not revealed.
Veenstra and Lang (2004)	Developing a simulation model to provide an economic analysis for the case study	Delta Container Terminal, Netherlands	DSOL	No	The authors developed a cost model to estimate the cost of the case study in a 25-year period.
(Parola and Sciomachen, 2005)	The logistic chain of the northern Italian port system was analysed to evaluate the possible growth of the container flows in this region	Genoa and La Spezia ports, Italy	Witness	No	Capacity of inland transportation.
(Briskorn et al., 2006)	Identifying the best dispatching strategy of AGVs	HHLA Container Terminal Al-tenwerder, Germany	DESMO-J	No	1) quay crane productivity; 2) empty travel time of AGVs; 3) AGV waiting time; and 4) quay crane waiting time.
(Soriguera et al., 2006)	Impact of different dispatching strategies of straddle carriers	Barcelona Container Terminal, Spain	Not specified	Not specified	Not specified.

Reference	Purpose of simulation	Case study	Software /library	Buffers utilised?	Performance measures
(Duinkerken et al., 2006)	Comparing the impact of utilising a Multi-Trailer system, AGVs and ALVs on the performance and cost of the case study	Maasvlakte Container Terminal, Netherlands	Must	Yes	1) vessel turnaround time; and 2) the authors developed a cost model to estimate the cost of each vehicle system.
(Henesey, Aslam and Khurum, 2006)	Identifying the optimal number of a new type of AGVs that are combined with cassettes to pick up/drop off containers by themselves	Not specified	DESMO-J	Yes	1) the equipment utilisation rate; and 2) the container handling rate which was calculated as dividing the number of containers that a quay crane handles by the total number of containers.
(Ha et al., 2007)	Identifying the optimal number of cranes/vehicles	Not specified	Plant	No	Berth productivity rate.
(Lee et al., 2007)	Comparing the impact of utilising trucks vs shuttle carriers on the performance of the case study	A port in Europe (the name of the port was not specified)	AutoMod	Yes	1) gross crane moves; and 2) total working time (in hour).

Reference	Purpose of simulation	Case study	Software /library	Buffers utilised?	Performance measures
(Ottjes et al., 2007)	Estimating the requirements for designing a new terminal in Port of Rotterdam such as the quay length, stacking capacity, handling and transport equipment and the international traffic flows	Port of Rotterdam, Netherlands	TOMAS	No	1) the quay occupancy rate; and 2) traffic flow (the number of AGVs/hour).
(Cortes et al., 2007)	Evaluating different scenarios of handling cargo vessels to improve the traffic flow of vessels	Port of Seville, Spain	Arena	No	1) dock time for each type of cargo; 2) the vessels' queue size; 3) vessel turnaround time; and 4) the warehouse capacity level.
(Lee et al., 2008)	Generating layouts for container terminal simulation models based on the given input	A generic container terminal	AutoMod	No	This paper only proposed a layout generator and no experiment was conducted.
(Hadjiconstantinou and Ma, 2009)	Impact of different dispatching strategies of straddle carriers	Port Pireus Container Terminal, Greece	C# programming language	Not specified	1) the average stacking crane utilisation rate; 2) the average waiting time at gate; 3) average waiting time at yard; 4) average waiting time at quay; and 5) the total throughput of the case study.

Reference	Purpose of simulation	Case study	Software /library	Buffers utilised?	Performance measures
(Klaws et al., 2011)	Investigating the impact of using a triple rail mounted gantry crane system on the turnaround time of vessels	Not specified	FlexSim CT	No	Vessel turnaround time.
(Kotachi et al., 2013)	Proposing a model for the simulation of container terminals for the Arena simulation software	Not specified (the authors provided a generic simulation model)	Arena	No	Flow time of containers i.e. the time that containers exist in the system.
(Gelareh et al., 2013)	Evaluating the results of a scheduling algorithm developed in this paper	Dublin Ferry Terminal, Ireland	FlexSim	No	Service time of vessels.
(Lin et al., 2014)	Identifying the optimal settings to minimise the total cost to reach the required level of service	Humen port, China	Arena	No	1) turnaround time of vessels was used to calculate the cost; and 2) the utilisation rates of quay cranes and stacking cranes.

2.5 Conclusion

This chapter first reviewed research on the FSP in ESTTs. ESTTs are industrial settings with a number of PDPs where goods are transported repeatedly between these points by a fleet of vehicles. Each PDP is usually equipped with a machine to process goods. Next to each machine there can be a buffer for transition of goods between vehicles and machines to decrease their waiting time. Examples of ESTTs are manufacturing shop floors, warehouses and container terminals. The FSP in ESTTs is a very important tactical problem that needs to be addressed carefully. Having too few vehicles is not efficient and may impair the performance of the system whereas using too many vehicles is very expensive and can increase the possibility of deadlocks and collisions in the system. In the literature, this problem was addressed using various techniques: calculus-based approaches, queuing theory, simulation, exact optimisation and meta-heuristics. In ESTTs there exist some sources of uncertainties that might have a significant impact on the optimal fleet size such as changes in the travel time of vehicles due to any disruption such as breakdowns, deadlocks and collisions. The existing approaches, however, did not consider the uncertainty in the environment properly. This leaves an important gap in the current research. In addition, despite the supposed importance of buffers on the performance of ESTTs, the impact of buffers on the optimal fleet size was hardly investigated. This thesis will attempt to bridge this gap by developing an EA combined with the Monte Carlo simulation to identify the optimal number of vehicles that is robust to the changes due to uncertainties. These efforts are presented in Chapters 3 and 4.

In addition, the simulation research in container terminals was reviewed in this chapter (Section 2.4).This chapter first explained the existing simulation software and programming libraries used in the literature to develop simulation models. In addition,

it reviewed simulation software packages that were developed specifically for container terminals. It then reviewed different simulation studies in container terminals on various applications. The applications of simulation in container terminals are mainly to identify the optimal settings and number of cranes/vehicles in the terminals and also evaluating the performance of container terminals with different scenarios and strategies. A table summarising the major simulation research in container terminals was then provided. Despite the existing simulation tools that can be used for the simulation of container terminals, there is a clear lack of a flexible simulation tool specifically developed for container terminals. Chapter 6 will attempt to close this gap by developing a flexible simulation framework for simulation of container terminals based on the FlexSim CT software.

Chapter 3

Evolutionary Fleet Sizing in Static and Uncertain ESTTs

3.1 Introduction

This chapter proposes an EA to identify the optimal number of vehicles in ESTTs. The ESTTs are industrial settings where goods are transferred repeatedly between multiple PDPsby a fleet of vehicles. At each PDP there is a machine to process the goods. Once goods have been processed, they will be picked up by vehicles and be transferred to another machine for further processing. Next to each machine, there might be a buffer, which is a limited space designed to temporarily store goods in a queue. The purpose of the buffer is to reduce the waiting time. Vehicles can drop off goods in the buffer without having to wait for the machines to be available. Machines can also place the goods in the buffer for vehicles to collect later. ESTTs are very common in industrial applications. Typical examples are manufacturing factories, warehouses, container

terminals and distribution centres (Vis, 2006).

One very important problem in an ESTT is the FSP - identifying the optimal number of vehicles to transfer goods. Having too few vehicles may decrease performance while having too many vehicles is expensive and may introduce deadlocks[1]. This problem is not trivial. In real-world cases, it is highly complex and the optimal fleet size depends on many factors such as the uncertainty in travel time of vehicles; the dynamics of machines' process time; and the size of the buffer. These factors, however, have not been previously fully considered, leaving an important gap in the current research. This chapter attempts to close this gap by proposing an EA to solve the FSP by considering the above factors. The proposed algorithm will be tested on two case studies of container ports[2].

Specifically, the outcome of this chapter will help answering the following questions for the first time: 1) How to determine the optimal/robust number of vehicles in static/uncertain ESTTs, especially container terminals? 2) How to analyse the impact of uncertainties on the optimal number of vehicles? 3) What is the impact of the buffer size on the optimal/robust fleet size?

The novelty of this chapter can be summarised as follows: First, an EA is proposed to solve the FSP in this context, with better performance than existing state-of-the-art methods. Second, a new formulation for the FSP is developed so that EA components can be built upon. Third, for the proposed EA, the following elements are developed: a representation, a local search, two operators and an adaptive learning mechanism. Fourth, the uncertainties in the FSP in container terminals are taken into account and solved. Two high fidelity simulation models (with different scenarios) were also

[1] A deadlock is a situation in which one or more involved vehicles cannot move. There are different reasons for deadlocks, such as a lack of competent traffic control schemes or using too many vehicles etc.

[2] These two container ports have committed to consider the result of this research to improve their operations.

developed to serve as the benchmark for the EA in the uncertain cases. Finally, a set of test cases is developed using realistic data from real European container terminals to resolve the issue of lacking benchmarks in this problem.

The rest of this chapter is structured as follows. Section 3.2 describes the FSP in container ports. Section 3.4 describes the proposed EA for the static case and its different components are explained in detail. In Section 3.5, a combination of the proposed EA with Monte Carlo simulation to determine the robust number of vehicles under uncertainties is described. The general approach to generate the test cases is given in Section 3.6. The experimental results of the static case including comparison results with the CPLEX solver are presented in Section 3.7. Experiments to study the effectiveness of this robust optimisation approach are described in Section 3.8. Finally, the conclusion is provided in Section 4.4.

3.2 Terminologies and problem descriptions

A job is defined as the process of moving a good from one PDP to another by a vehicle. For each job a time window $[a_i, b_i]$ is associated where a_i is the release time of job i from a PDP and b_i is the latest time to start job i. The value of b_i is calculated based on the release time of successor jobs of job i and the size of buffer. For example with a buffer of size n (i.e. a buffer with the capacity of n goods) the due time of job i is calculated as: $b_i = a_{i+n}$. This means that job i should start before the release time of job $i + n$ to have at least one available slot for job $i + n$ in the buffer. To determine the exact pickup time of each job from the buffer, each time window needs to be discretized into multiple intervals, each with a duration of δ. The pickup time of a job is set at the beginning of one of the intervals.

3.2.1 Compatible jobs

Two jobs are compatible if they can be done consecutively by one vehicle. In other words, job i and job j are compatible if one vehicle can pick up job i from a pickup point P_i and deliver it to a delivery point D_i, and then can still travel to a pickup point P_j to pick up job j within the time window of job j. Mathematically, job i and job j are compatible if $s_i + t_{P_iD_i} + t_{D_iP_j} \epsilon [a_j,b_j]$, where s_i is the pickup time of job i, P_i is the pickup point of job i, D_i is the delivery point of job i, t_{PD} is the travel time from point P to point D and $[a_j, b_j]$ is the time window of job j.

3.2.2 Problem modelling

The FSP was modelled in Vis et al. (2005) as a graph where each node represents one of the possible pickup times for a job. This graph has a source node from which all other nodes originate and a sink to which all other nodes terminate. Nodes that are compatible, i.e. nodes whose jobs can be done by the same vehicle, are connected by arcs. A set of connected arcs going from the source to the sink is called a path. Each path represents the sequence of jobs to be done by one vehicle. The total number of paths represents the total number of vehicles. The objective is to find the minimum number of paths which start from the source node and end at the sink node subject to the following constraints: 1) Each job can start only once (it means that among all the possible pickup time nodes of a job, only one should be included in one of the paths); 2) Each job cannot be done by more than one vehicle (it means each node in the graph cannot be included in more than one path).

Figure 3.1 shows an example of using a graph to model one simple FSP with three jobs and two vehicles. Job 1 has two possible pickup times represented by nodes j_{11} and j_{12}. Job 2 has one pickup time node j_{21}. Job 3 has two possible pickup time nodes

j_{31} and j_{32}. Node s is the source node and node t is the sink node. The graph in Figure 3.1 represents a solution for the problem. This solution consists of two paths (i.e. two vehicles) in which the path of vehicle 1 passes through j_{11} and j_{32} and the path of vehicle 2 passes only through j_{21}. Using this graph model, Vis et al. (2005) formulated the FSP as an IP problem. Readers is referred to Vis et al. (2005) for details of this formulation. In the thesis, this IP formulation is reproduced and it is solved using the CPLEX solver. The results will be used as a benchmark to compare the proposed algorithm.

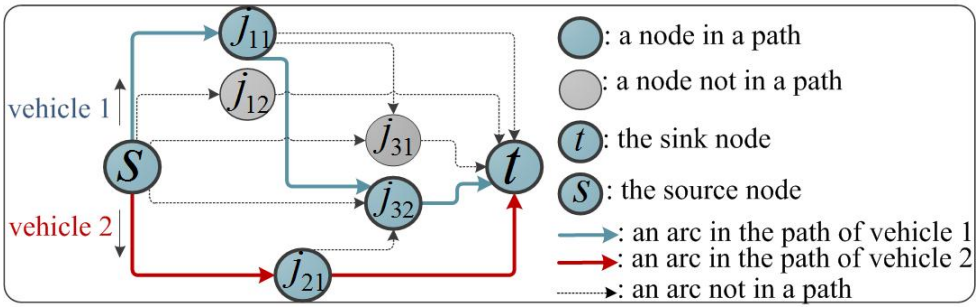

Figure 3.1: An example of graph representation of the FSP

3.3 The FSP in container terminals

In container terminals, a fleet of vehicles is responsible for transporting containers between two areas named quay and stack areas. The quay area is the place where vessels are berthed and there are QCs to discharge and load containers from/to vessels. The stack area consists of a number of stack blocks served by SCs. Each quay or stack area has a number of PDPs for vehicles to transfer containers. Each PDP is facilitated by one crane.

Once a vessel is berthed, the discharging and then loading operations are performed,

usually separately. In the discharging phase, QCs unload containers from vessels, pass them to transfer vehicles which then transport containers to the stack area. If vehicles can pick up containers by themselves, QCs place containers in the *buffer,* a place underneath the cranes for temporary storage of containers, for vehicles to collect. Otherwise, the QCs have to wait for vehicles to come and then place containers directly on the vehicles. The capacity of the buffer is limited and it is required that containers should be picked up from the buffer before it gets full i.e. there should always be at least one free slot available in the buffer for QCs to discharge containers. This is to minimise the expensive waiting times of QCs.

At the stack area, SCs collect containers from vehicles and place them in the stacks. After passing a container to SCs, transfer vehicles will come back to QCs to collect another container. The buffer for SCs is similar to the one for QCs: vehicles drop off containers in the buffer from which SCs pick up containers. In this chapter, it is assumed that there are always free slots available in the buffer of SCs for vehicles to drop off containers. After the discharging phase finishes, the loading phase will start to transport containers from the stack area to vessels. The loading tasks are similar to the discharging tasks but in an opposite direction. Due to such a similarity, in this chapter, the FSP for the discharging tasks is only considered.

3.4 An EA approach for static environments

As mentioned earlier, one of the main objectives of this research is to develop an EA to address the gaps in existing literature (Section 2.3), namely dealing with larger scale situations, handling uncertainties, and investigating the use of buffers in static and uncertain situations. The proposed EA will be named Fleet Sizing Evolutionary Algorithm (FSEA). In this section, a version of FSEA for static environments will be

explained. Another version of FSEA combined with a Monte Carlo approach to deal with uncertainties will be explained in Section 3.5.

3.4.1 EAs

EAs is a class of meta-heuristic population-based algorithms for solving optimisation problems. An EA normally consists of some general elements such as representations, selection methods, mutations, recombination operators and local searches. This section will propose new implementations for these elements as parts of FSEA.

3.4.2 Chromosome representation

This subsection defines chromosomes for the EA based on the graph representation in Subsection 3.2.2 (Figure 3.1). Given such a graph, each chromosome of FSEA is a solution comprising a number of paths. Each path represents a sequence of jobs to be done by one vehicle. Each chromosome is represented as a string of pairs $P_i =< x_i, y_i >$, $i = 1, ..., n$ where pair P_i corresponds to job i; x_i represents the pickup time for job i; y_i represents the chosen pickup time for the next job that will be done by the same vehicle as job i. Figure 3.2 shows a general representation and an example of a chromosome.

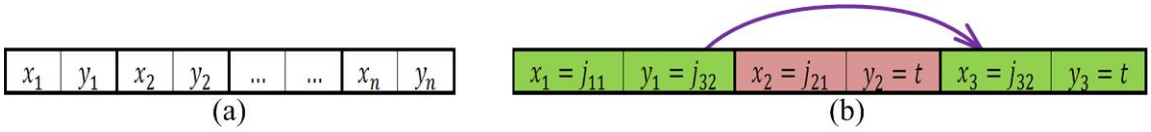

Figure 3.2: Chromosome representation. The left plot shows a general representation of a chromosome. The right plot shows an example of how such a chromosome can encode the paths of vehicles in the example in Figure 3.1. The two pairs in green represent the path for vehicle 1 and the pair in red represents the path for vehicle 2. Job 1 is linked with job 3 because both can be picked up by vehicle 1. As can be seen both y_1 and x_3 refer to the same value, j_{32}, which is the pickup time for job 3.

3.4.3 A new problem formulation for the FSP

Along with the representation, a new formulation is proposed for the objective function to make it solvable by FSEA. FSEA is developed based on this formulation. Note that this formulation is equivalent to the existing formulation in Vis et al. (2005). However, with different representation this formulation provides a better connection to the FSEA's components. This is because the decision variables in this formulation represent the pairs of jobs in the chromosome representation. In addition, in the case of producing infeasible solutions, thanks to this formulation, the causes of this violation in the chromosome can be easily identified and repaired accordingly. Thus, this formulation helps to understand FSEA components and their functionality better compared with the existing formulation in Vis et al. (2005) and hence it makes FSEA easier to comprehend.

The notation and formulation are as follows:

Let:

x_i: represents the pickup time for job i,

y_i: represents the chosen pickup time for the next job to be done by the same vehicle as job i,

n: number of jobs, $n \epsilon \mathbb{N}$

t: the sink node,

S_i: set of nodes that correspond to the different possible pickup times of job i,

C_{x_i}: set of nodes that are compatible with node x_i,

Define the $e(a,b)$ and $b(y_i, y_j)$ functions as follows:

$$e(a,b) = \begin{cases} 1 & if\, a = b \\ 0 & otherwise \end{cases} \qquad a, b \epsilon \mathbb{N},$$

$$b(y_i, y_j) = \begin{cases} 1 & if\ i \neq j\ and\ \exists job\ k\ so\ that:\ y_i \epsilon S_k\ and\ y_j \epsilon S_k \\ 0 & otherwise \end{cases}$$

Then, the objective function of this optimisation problem is as follows:

Find x_i and y_i to:

$$Min \sum e(y_i, t) \tag{3.1}$$

Subject to:

$$x_i \epsilon S_i,\ 1 \leq i \leq n \tag{3.2}$$

$$y_i \epsilon C_{x_i},\ 1 \leq i \leq n \tag{3.3}$$

$$\sum_{j=1}^{n} \sum_{i=1}^{n} b(y_j, y_i) = 0 \tag{3.4}$$

$$\sum_{j=1}^{n} \sum_{i=1}^{n} e(x_j, y_i) + \sum_{i=1}^{n} e(y_i, t) = n \tag{3.5}$$

Objective function (3.1) counts the number of times the y_i variables are set as t (the sink node). y_i being set as t means y_i is at the end of the path of one vehicle. Accordingly, the value of objective function (3.1) is the total number of paths, which is equivalent to the total number of vehicles. Constraints (3.2) and (3.3) define the domain range of the x_i and y_i variables, respectively. Constraint (3.4) ensures that a job must be in only one path (see Figure 3.3). If a job j is included in more than one path, j will be presented by more than one "y" variables in the chromosome. Constraint (3.4) prevents this from happening by ensuring that for each job that is not the first or the last in a path, there is one and only one "y" variable corresponding to the job. Figure 3.3 shows an example where Constraint (3.4) is violated. Constraint (3.5) ensures that in all paths there is no more than one node referring to one job (see Figure 3.4). In other words, each job must have only one node in only one path. The

following example clarifies the meaning of this constraint. Consider one path where job j is set to be transported after job i by the same vehicle. It means that, y_i refers to one of the possible pickup time nodes of job j. The path is only feasible if x_j is equal to y_i, i.e. both x_j and y_i refer to the same pickup time of job j if job i is not the last job in the path. It means e(x_j,y_i) = 1. Job i can only be picked up once, so $\sum_{i=1}^{n} e(x_j, y_i)$ must be equal to 1. Let $m < n$ be the number of jobs that are not last jobs in their paths, then $\sum_{j=1}^{n}\sum_{i=1}^{n} e(x_j, y_i)$ must be equal to m. Let $k = n - m$ be the number of jobs that are the last in their paths, then $\sum_{i=1}^{n} e(y_i, t)$ must be equal to k. Take the summation: $\sum_{j=1}^{n}\sum_{i=1}^{n} e(x_j, y_i) + \sum_{i=1}^{n} e(y_i, t)$, then $\sum_{j=1}^{n}\sum_{i=1}^{n} e(x_j, y_i) + \sum_{i=1}^{n} e(y_i, t) = m + (n - m) = n$. This is Constraint (3.5). Figure 3.4 shows an example of one solution that violates this constraint.

3.4.4 Initialisation and repair

To solve the FSP effectively, FSEA should start from a population of feasible solutions. The following approach is used to repair a random initial population into feasible individuals. At the start of the algorithm, each variable is initialised by a random value within its domain range. Most probably the produced solutions are not feasible. To repair such infeasible solutions, a repair operator is developed to make an infeasible individual feasible. Note that Constraints (3.2) and (3.3) are never violated, since individuals are initialised within the domain ranges given in these two constraints.

The repair operator repairs violations of Constraints (3.4) and (3.5) as follows. First, the repair operator checks the violation of Constraint (3.4). If it is violated, at least one job is placed in more than one path. Figure 3.3 shows an example of such

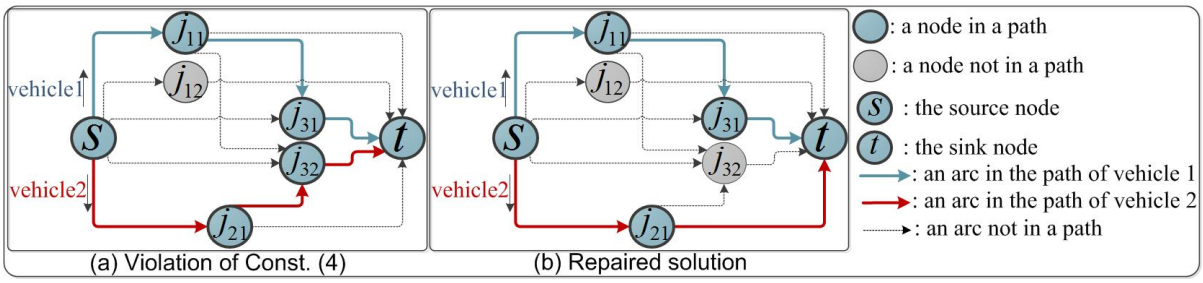

Figure 3.3: An example of violation of Constraint (3.4). The left plot shows an example of a solution violating Constraint (3.4) (chromosome: [<$x_1 = j_{11}$, $y_1 = j_{31}$>, <$x_2 = j_{21}$, $y_2 = j_{32}$>, <$x_3 = j_{31}$, $y_3 = t$>]) and the right plot shows the repaired solution for such violation (chromosome: [<$x_1 = j_{11}$, $y_1 = j_{31}$>, <$x_2 = j_{21}$, $y_2 = t$>, <$x_3 = j_{31}$, $y_3 = t$>]). In the left plot, job 3 (with two nodes j_{31} and j_{32}) is placed in two paths. To repair this violation (as in the right plot), job 3 must be kept in one of the paths e.g. the path of vehicle 1 and the path of vehicle 2 must be terminated by replacing j_{32} with the sink node (t).

violation.

In order to repair individuals regarding such violation, the duplicated jobs are removed and replaced by the sink node in all but one of the violated paths. In the example in Figure 3.3, job 3 must be removed from one of the paths, e.g. the path of vehicle 2 and be kept in the path of vehicle 1. Consequently, the violation regarding Constraint (3.4) is repaired.

The individuals must then be checked regarding the violation of Constraint (3.5). If this constraint is violated it means that a path has two different nodes corresponding to the same job. The process of repairing violations of this constraint is explained using the example in Figure 3.4. In this example, two different pickup time nodes of job 3 (j_{31} and j_{32}) are in the path of vehicle 1, hence a violation of Constraint (3.5). To repair the violation, the repair operator checks whether x_3 (j_{31}) in pair <x_3,y_3> of job 3 is compatible with x_1 (j_{11}) in pair <x_1,y_1>. If this is the case, the repair operator changes the value of y_1 to j_{31}. Otherwise, it updates the value of y_1 with the sink node (t) to terminate the path of vehicle 1 at job 1 and starts a new path from job 3. For this type of violation, the repair operator always changes the y variables

Algorithm 3.1 Initialisation

1: **for** i **from** 1 **to** *popSize* //*popSize* is the size of the population
2: **for** j **from** 1 **to** n //n is the number of jobs
3: Initialise x_j of individual $\vec{X}_i$ with a random integer within its domain range
4: Initialise y_j of individual $\vec{X}_i$ with a random integer within its domain range
5: **for** i **from** 1 **to** *popSize*
6: individual $\vec{X}_i$:= *Repair*(individual $\vec{X}_i$)

of the preceding job rather than the x variables of the violated job. This is because if the repair operator changes the x variable of the violated job, the consecutive jobs might not be compatible with this change. As a result, this might make the rest of the path invalid. Pseudo-codes of the initialisation and repair procedures are shown in Algorithms 3.1 and 3.2.

Algorithm 3.2 Repair

1: Identify $ViolConst_4$, the set of jobs in individual $\vec{X}$ that violate Constraint (3.4)
2: **for** all jobs $j \epsilon\ ViolConst_4$
3: Update all but one "y" variables corresponding to job j with the sink node //keep one of the "y" variables unchanged randomly
4: Identify $ViolConst_5$, the set of triples $< x_i, y_i, x_j >$ that violate Constraint (3.5)
5: **for** all triples $< x_i, y_i, x_j > \epsilon\ ViolConst_5$
6: **if** x_i is compatible with x_j
7: Update y_i with x_j
8: **else**
9: Update y_i with the sink node
where $ViolConst_5 = \{<x_i, y_i, x_j>, 1 \leq i, j \leq n$, and y_i refers to job j but to a different node than $x_j\}$

3.4.5 Reproduction

A reproduction method is proposed, which is a combination of a mutation operator and a local search, to produce new offspring. The mutation operator is used to create offspring and the local search is used to improve the fitness of each offspring[3]. The

[3]Subsection 3.7.6 provides justifications on why these reproduction operators are used for FSEA

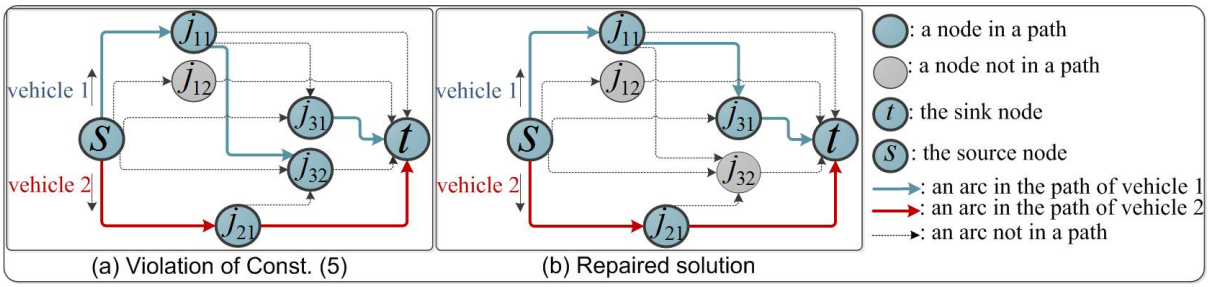

Figure 3.4: An example of violation of Constraint (3.5). The left plot shows a violation of constraint (3.5) where two different nodes of job 3 are placed in the path of vehicle 1 (chromosome: [$<x_1 = j_{11}, y_1 = j_{32}>$, $<x_2 = j_{21}, y_2 = t>$, $<x_3 = j_{31}, y_3 = t>$]). The right plot shows the repaired solution of this violation (chromosome: [$<x_1 = j_{11}, y_1 = j_{31}>$, $<x_2 = j_{21}, y_2 = t>$, $<x_3 = j_{31}, y_3 = t>$]).

mechanism for the local search and mutation operator are explained below.

Local search

Recall from Subsection 3.2.2 that (i) a solution is modelled as a graph containing multiple paths (job sequences) from the source to the sink and (ii) the total number of job sequences represents the total number of vehicles. Then, if the number of job sequences can be reduced in a solution, it would be possible to reduce the fleet size. To reduce the number of job sequences, all jobs in one randomly chosen sequence (let us call it *s_delete*) should be removed and, if possible, insert them to the other sequences. This is equivalent to asking other vehicles to take over all the jobs originally assigned to the s_delete vehicle. If this can be done, the s_delete job sequence will disappear and the fleet size will be reduced by one. This way, no constraints will be violated (Subsection 3.4.3) but the fitness of solutions can be improved.

Detailed implementation of this local search is provided in Algorithm 3.3.

Figure 3.5 shows an example of how the local search can be used to remove a job sequence in a solution.

Algorithm 3.3 ReduceJobSequences

1: **for** all jobs $j \in JobSeq_{s\_delete}$
2: **if** job j can be inserted to $JobSeq_l$ at position k ($1 \leq l \leq FS, l \neq s\_delete$ and $1 \leq k \leq length(J_l)$)
3: insert job j to J_l at position k
4: remove job j from $JobSeq_{s\_delete}$
5: **if** all jobs are removed from $JobSeq_{s\_delete}$ //vehicle s_delete was removed from the fleet size successfully
6: Randomly choose another vehicle to be s_delete.
7: **if** any job is removed from $JobSeq_{s\_delete}$
8: $\alpha := 0$
9: **else**
10: $\alpha := \alpha + 1$
11: *AdaptiveLearning*()

where $JobSeq_l$ is the array to show the sequence of jobs for vehicle l, α is the number of generations that $JobSeq_{s\_delete}$ has remained unchanged, and the *AdaptiveLearning()* is the proposed learning method to help FSEA to get out of local minima, this method is described in Subsection 3.4.6.

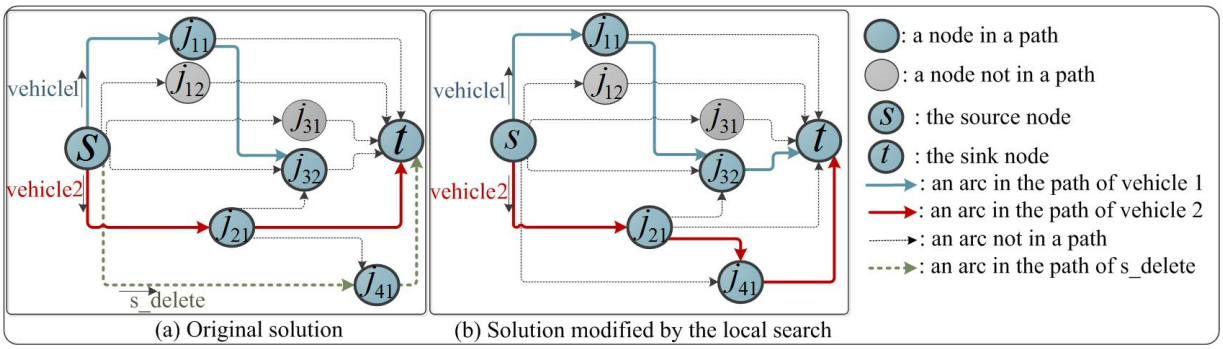

Figure 3.5: An example of the local search operator. This figure shows how the local search can help to improve fitness of individuals. In this example, the local search moves node j_{41} from vehicle s_delete into the sequence of jobs of vehicles 2. By this movement, the number of vehicles can be decreased from three to two.

Mutation

There might be a local optimum situation where the local search cannot remove the chosen s_delete sequence, because there is no available place in the other sequences to further insert the remaining jobs from s_delete. A mutation operator is proposed

to help the algorithm escape such a situation. Instead of moving jobs from s_delete to other sequences, jobs should be moved among the other sequences, in a hope that this movement would change the structure of these sequences, which in turn would open up space to which jobs from s_delete can be slotted in. A pseudo-code of the mutation operator is provided in Algorithm 3.4.

Algorithm 3.4 Mutate

1: Generate m, a random integer value //where $1 \leq m \leq FS$, $m \neq s\_delete$ and FS is the fleet size
2: **for** all jobs $j \in JobSeq_m$ //$JobSeq_m$ is the array of the sequences of jobs for vehicle m
3: **if** job j can be inserted to $JobSeq_l$ at position k ($1 \leq l \leq FS, l \neq s\_delete$ and $1 \leq k \leq length(J_l)$)
4: insert job j to $JobSeq_l$ at position k
5: remove job j from $JobSeq_m$

An example of the mutation operator is shown in Figure 3.6.

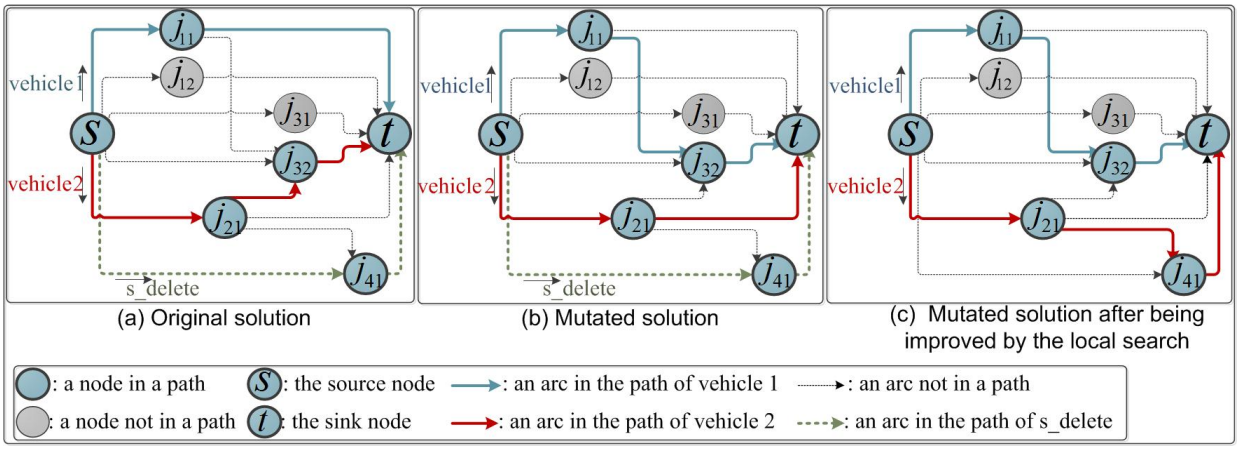

Figure 3.6: An example of the mutation operator. In this example, if job 3 in the original solution (left plot) is moved from vehicle 2 to vehicle 1 thanks to the mutation operator (middle plot), job 4 can be moved by the local search from vehicle s_delete to vehicle 2 (right plot). As a result, vehicle s_delete can be deleted and the fleet size is reduced by one.

3.4.6 Adaptive learning method

As recalled in Subsection 3.4.5, the algorithm might be trapped in a local optimum where it is not possible to move the remaining jobs from a randomly chosen s_delete

to any other sequences. The experiments showed that some jobs are significantly more difficult to be removed than others and they are the ones that normally remain in s_delete when the algorithm gets trapped. Worse, even if the algorithm is able to escape thanks to the mutation operator, later on it may still be likely to end up in a similar local optimum, where those most-difficult-to-remove jobs will again be the ones remaining in s_delete.

The best way to deal with this situation is to adaptively learn what are those most difficult jobs, remember them and then avoid removing any job sequence containing them when applying the local search *ReduceJobSequences()*. This process is similar to a Tabu search. This process is done in the proposed *AdaptiveLearning()* procedure (Algorithm 3.5). This procedure has two tasks: 1) identifying the most-difficult-to-remove jobs and 2) updating the index of s_delete to another job sequence (i.e. vehicle) if all the jobs of s_delete are the most-difficult-to-remove jobs.

To do the first task, this procedure keeps track of the changes in the jobs of s_delete to identify the most-difficult-to-remove jobs. Recall from Algorithm 3.3, this procedure counts the number of times where s_delete remained unchanged during consecutive generations using α as the counter (Algorithm 3.3, lines 7-10). Algorithm 3.5 checks the value of α and if $\alpha = \beta$ (β is the upper bound for α), Algorithm 3.5 then considers the jobs of s_delete the most-difficult-to-remove jobs. In such cases it adds the jobs of s_delete to the set of most-difficult-to-remove jobs (i.e. $DifficultJobs$). The algorithm then changes the index of s_delete to another job sequence (i.e. vehicle) in the solution, because all jobs of s_delete are difficult to be removed (Algorithm 3.5, lines 1-4). Algorithm 3.5 limits the size of $DifficultJobs$ to γ. When the size of this set reaches the limit, the algorithm randomly removes more jobs from this list until its size becomes r (a random integer value between 0 and γ) to make room for other potential jobs to become most-difficult-to-remove jobs (Algorithm 3.5, lines 8-10). To

do the second task, Algorithm 3.5 compares the jobs of s_delete with the elements of the $DifficultJobs$ set. If all the jobs of s_delete belong to this set, it is less likely that the jobs of s_delete can be removed, thus this algorithm updates the index of s_delete with another job sequence i.e. vehicle (Algorithm 3.5, lines 5-7).

An example of how the adaptive learning method helps in preventing the algorithm from getting trapped in local optima is shown in Table 3.1.

Algorithm 3.5 AdaptiveLearning

1: **if** $\beta = \alpha$
2: $DifficultJobs := DifficultJobs \cup JobSeq_{s\_delete}$
3: $\alpha := 0$
4: Update s_delete with a new random value between 1 and FS
5: **if** $JobSeq_{s\_delete} \subseteq DifficultJobs$
6: Update s_delete with a new random value between 1 and FS
7: $\alpha := 0$
8: **if** $length(DifficultJobs) \geq \gamma$
9: Generate r, a random integer value between 0 and γ
10: Remove elements of $DifficultJobs$ randomly until its size becomes r

where $DifficultJobs$ is the list of difficult jobs, $JobSeq_{s\_delete}$ is the array to show the sequence of jobs for s_delete, β is the maximum allowed number of generations that $Jobs_{s\_delete}$ can remain unchanged, α counts the number of generations that $Jobs_{s\_delete}$ has remained unchanged, γ is the maximum size of $DifficultJobs$ array.

3.4.7 The pseudo-code for FSEA

So far, all the components of FSEA have been explained. Now, it is the time to put together the components of FSEA and present the pseudo-code for the whole algorithm (Algorithm 3.6). FSEA starts by the initialisation of the population using Algorithm 3.1. It then evaluates the individuals based on the fleet size of each individual using Equation 3.1. FSEA selects individuals for the next generation using the rank selection. Algorithms 3.3 and 3.4 are then applied to individuals for possible improvement of their fitness. This loop will continue until the stopping criteria are met and the best found

Table 3.1: This table shows how the adaptive learning can help FSEA to get out of local minima. Without adaptive learning FSEA got trapped in local minima with job 83 in s_delete. With the adaptive learning (and remembering to avoid job 83), FSEA can get out of such local minima and hence reduce the fleet size by one in generation 100.

Without adaptive learning

	Generation 10	Generation 50	Generation 100
v1	24 0 26 27 30 55 56 59 36 86 87 42 43	24 0 74 78 33 58 83 17 21 44	24 0 26 28 27 55 56 59 36 86 87 42 43 97
v2	25 1 74 78 33 58 84 64 88 20 44	25 28 27 55 56 59 36 86 87 42 43 94 99	25 2 1 3 4 5 53 8 57 60 61 63 40 89 45 94 98
v3	48 3 75 31 80 12 15 17 41 22 94	26 52 6 79 84 38 88 20	48 52 7 79 34 35 33 84 38 39 88 20 96
v4	49 2 4 53 8 32 57 60 61 37 65 67 89 45 47	48 1 3 53 8 57 60 61 63 39 40 89 45 97	49 75 80 12 16 66 65 41 22
v5	50 52 7 79 82 38 40 21 46 98	49 2 75 31 80 12 35 16 66 65 41 22 96 98	50 29 77 82 85 19 69 91 70 71 95 47
v6	51 5 77 34 10 13 85 19 68 91 70 71 95 96 99	50 29 4 77 82 34 85 19 69 91 70 71 95 47	51 54 78 58 15 17 21 44 99
v7	54 83 (s_delete)	51 54 15 (s_delete)	72 76 10 11 13 62 64 67 90 68 23
v8	72 28 76 11 14 62 63 39 66 90 69 23 97	72 76 10 11 13 62 64 67 90 68 23	73 6 31 30 9 32 81 14 37 18 92 93 46
v9	73 29 6 9 81 35 16 18 92 93	73 5 7 30 9 32 81 14 37 18 92 93 46	74 83 (s_delete)

With adaptive learning

	Generation 10	Generation 50	Generation 100
v1	24 0 26 27 30 55 56 59 36 86 87 42 43	24 1 0 3 28 5 30 55 56 59 36 86 87 42 43 94	24 0 73 5 55 58 33 14 62 63 40 90 68 23
v2	25 1 74 78 33 58 84 64 88 20 44	25 27 52 6 9 81 84 18 92 93 44	25 27 75 8 32 81 15 37 18 92 93 46 99
v3	48 3 75 31 80 12 15 17 41 22 94	26 74 8 57 83 63 39 40 89 45 95 98	26 28 4 29 6 56 11 59 36 86 87 42 43 97
v4	49 2 4 53 8 32 57 60 61 37 65 67 89 45 47	48 2 53 54 33 32 10 35 61 37 64 88 20 46	48 2 1 3 74 78 12 84 64 88 20 47
v5	50 52 7 79 82 38 40 21 46 98	49 75 31 80 12 16 66 65 41 22	49 53 77 10 34 13 85 19 69 91 70 71 44
v6	51 5 77 34 10 13 85 19 68 91 70 71 95 96 99	50 29 77 82 34 85 19 69 91 70 71 47	50 52 7 79 82 38 39 66 65 21 94 95
v7	54 83 (s_delete)	51 7 78 58 15 17 21 97	51 54 30 31 80 83 17 41 22 96 98
v8	72 28 76 11 14 62 63 39 66 90 69 23 97	72 4 76 11 13 60 62 38 67 90 68 23 96 99	72 76 9 57 35 61 16 67 89 45 (s_delete)
v9	73 29 6 9 81 35 16 18 92 93	73 79 14 (s_delete)	

solution will be the optimal solution.

Algorithm 3.6 FSEA

1: **Initialise** population P_t by Initialisation() (Algorithm 3.1)
2: **Evaluate** population P_t
3: **while** stopping criteria not met
4: **Select** elements from P_t to copy into P_{t+1}
5: **Mutate** population P_{t+1} by Mutate() (Algorithm 3.4)
6: **Recombine** population P_{t+1} by ReduceJobSequences() (Algorithm 3.3)
7: **Evaluate** new population P_{t+1}
8: $P_t := P_{t+1}$
9: **return** the best individual

3.5 Extensions of FSEA for uncertain environments

Real-world problems usually have uncertainty (Jin and Branke, 2005). This is also the case with the FSP in ESTTs. However, none of the existing research in the FSP considers any uncertainty. This creates an important gap in this area of research. This section will try to bridge this gap by producing an extension of FSEA to deal with uncertain environments. In this section an extension of FSEA is developed to deal with uncertain environments.

3.5.1 Uncertainties in ESTTs

Uncertainties in the process time of machines and travel time of vehicles were identified as two important types of uncertainties that have significant impacts on the optimal fleet size. The first type of uncertainties can be caused by variations in the quality of machine parts, skills of operators and so on. The second type of uncertainties can be caused by weather conditions, vehicle breakdowns, or congestion. Both types may render a static optimal fleet size ineffective (as will be shown in the experiments later). Due to that, it is needed to take these uncertainties into account by identifying a fleet

size that is robust against uncertainties, so that if there is any change, the fleet size is still effective. A Monte Carlo approach is proposed to simulate and guide FSEA towards the most robust solutions. The proposed method will measure the quality robustness by which it can be ensured the optimal fleet size is capable of performing all the transportation tasks without imposing any delay to machines.

3.5.2 Monte Carlo simulation for the uncertainties in vehicle travel time

As explained in Section 3.4, the solutions (chromosomes) produced by FSEA contain not only the total number of vehicles but also the schedules (sequences) of jobs for each vehicle. It is possible that two individuals have the same number of vehicles but different schedules i.e. sequences of jobs can be different for the same number of vehicles. In the static case FSEA evaluates fitness of individuals based only on the number of vehicles, regardless of the produced schedules. However, in the uncertain case such an evaluation may not be totally realistic. Different schedules for the same number of vehicles may not behave similarly under uncertainties. Some schedules may show more robustness against changes than others.

To evaluate the robustness of a schedule of vehicles, a Monte Carlo (MC) approach is proposed. The proposed MC evaluates the schedules of vehicles to answer the following questions: 1) How robust is the schedule of vehicles under uncertainty in travel time? 2) Is it needed to add more vehicles to the system to reduce the possible waiting time of machines caused by this type of uncertainty? If those additional vehicles are needed, how many more vehicles should be added?

Before explaining the proposed MC, some concepts will be defined: the frequency of disruptions of vehicles and the time to resolve the disruptions. Different types of

vehicle disruptions can happen in ESTTs such as breakdowns, collisions and deadlocks. The disruption rate ($\lambda(t)$) of a vehicle is defined as the frequency of disruptions until the time t. Without any loss of generality, it is assumed that the disruption rate is a constant value in the period of evaluation i.e. $\lambda(t) = \lambda_0$. It is commonly assumed that the disruption of vehicles follows the exponential distribution with the parameter λ_0. Once disruptions happen to the vehicles, they will not be available until they get repaired. Mean time to repair (MTTR) is a parameter that shows the average of unavailable time. The MTTR is used in the MC to estimate the duration in which vehicles become unavailable.

The proposed MC evaluates the robustness of schedules as follows. First, for each vehicle the MC estimates the first moment and the duration of disruptions using a random exponential value with the parameter λ (the given disruption rate) and a MTTR value, respectively. Let us assume that the first disruption is at time t_1. This means that the vehicle can work from time $t = 0$ until $t = t_1$. Then, the vehicle will not be available for a period equal to $MTTR$ until the time t_2, $t_2 = t_1 + MTTR$. This process is repeated to reach the makespan - the time by which the last job has finished. Then this process of simulating disruptions is repeated for all the vehicles until they all reach the makespan. In order to prevent any delay in the system, the jobs that are uncovered due to vehicles being unavailable must be assigned to another available vehicle. The MC searches through all the available vehicles in the solution to find a suitable substitution to do the uncovered jobs. If such substitution vehicles are found then MC assigns the uncovered jobs to those vehicles. Otherwise, new vehicles must be added to the system to carry out the uncovered jobs. Those vehicles are called *additional vehicles*. In order to produce a robust number of vehicles, the additional number of vehicles must be estimated and added to the fleet size.

In order to have an accurate estimation of the additional vehicles, the algorithm

applies the above MC approach to the schedule of each individual m_1 times and stores the fleet size of all these m_1 replications (the higher m_1, the more accurate estimation). At the end of m_1 replications, the MC produces an average number of additional vehicles. This average number is considered a measure to evaluate the robustness of schedules. This MC approach is combined with FSEA to determine the robust number of vehicles. This combined algorithm is MC1-FSEA. In MC1-FSEA, the fitness of an individual is calculated as the number of vehicles decoded from the individual's chromosome plus the additional vehicles needed to deal with the uncertainty, as estimated by the MC. The objective function for MC1-FSEA is defined as: $MC1 = f + \frac{\sum_{i=1}^{m_1} av_i}{m_1}$, where f is calculated using Equation (3.1); av_i is the number of required additional vehicles to cover the uncovered jobs during disruptions in the ith MC simulation; and m_1 is the number of replications. The pseudo-code of MC1-FSEA is exactly the same as FSEA (Algorithm 3.6) except that the objective function step 7 (Evaluation) is replaced by $MC1 = f + \frac{\sum_{i=1}^{m_1} av_i}{m_1}$.

3.5.3 Uncertainty in machine process time

The process time of machines can be changed due to many reasons, but it usually follows a probabilistic distribution (Celen et al., 1997). The distribution is assumed to have already captured various uncertainty factors that have an impact on the machine process time such as mechanical faults, operator's skills etc. To capture this uncertainty, another variant of FSEA is developed which is called MC2-FSEA. MC2-FSEA uses m_2 different sets of generated process times of machines that are estimated by the above distribution. It then applies FSEA to each set of the process time separately to determine the optimal number of vehicles. Results of runs over the m_2 sets show the impacts of the uncertainty in machine processing time on the optimal fleet size. The average of results of m_2 runs is considered the robust fleet size in regard to this type of

uncertainty. Algorithm 3.7 shows a pseudo-code for MC2-FSEA. The robust fleet size for MC2-FSEA is defined as: $MC2 = \frac{\sum_{i=1}^{m_2} FS_i}{m_2}$, where FS_i is the static optimal fleet size identified by FSEA for run i and m_2 is the number of runs. Note that, in line 4 of Algorithm 3.7 if instead of FSEA MC1-FSEA is used then the robust number of vehicles under both types of uncertainties can be produced. This version of FSEA is called MC12-FSEA. For MC12-FSEA, the robust fleet size is defined as: $MC12 = \frac{\sum_{i=1}^{m_2} MC1_i}{m_2}$, where $MC1_i$ is the robust fleet size against the uncertainty in vehicles travel time for run i and m_2 is the number of runs.

Algorithm 3.7 MC2-FSEA

1: $sum := 0$
2: **for** i **from** 1 **to** m_2 //m_2 is the number of replications for MC2-FSEA
3: Estimate the process time of machines for all the jobs using the given distribution
4: Determine the optimal fleet size FS_i by FSEA using the estimated set of process time
5: $sum := sum + FS_i$
6: **return** $MC2 := sum \ / \ m_2$

3.6 Case studies

As mentioned earlier, container ports were considered to be the case study for this research. The FSP in ports is a new problem. There has been no published test case for this problem in ports in the literature. As a result, all the test cases were created for this research from scratch using the real-world data from two of the European port partners.

3.6.1 Real-world test cases

The test cases are generated using some real-world data such as the number of QCs, the number of containers, the size of buffers, the distances between QCs and SCs (Table 3.2), and so on. In the two European ports considered in this case study (let us call them port A and port B[4]), the maximum numbers of QCs that can work on one vessel are three and two QCs, respectively. The numbers of containers to be discharged are considered 100, 200, and 300 to be in line with the actual transactions in the two ports. Currently port A has 6 blocks and port B has 2 container stack blocks and it is assumed containers are distributed evenly between the blocks. Each stack block is facilitated with one SC. Given the actual space under or next to the cranes in the two ports, the sizes of buffers are varied from 0 to 10.

Table 3.2: This table shows the distances (in meters) between QCs and SCs for ports A and B.

		QC1	QC2	QC3
Port A	SC1	220	250	340
	SC2	257	287	377
	SC3	298	328	418
	SC4	467	497	587
	SC5	441	471	561
	SC6	428	458	648

		QC1	QC2
Port B	SC1	784	682
	SC2	795	693

The only assumptions that have been made (in agreement with the port partners) in generating the test cases are as follows: 1) the speeds of vehicles are constant (Subsection 3.6.3); 2) the process time of quay cranes follows the given distributions (Subsection 3.6.2) and 3) no waiting time of vehicles for stack cranes has been considered, similar to Vis et al. (2005).

[4]Due to confidentiality agreements, their actual identities cannot be revealed.

3.6.2 Time windows of jobs

As mentioned earlier in Section 3.2, a container has to be picked up within a certain time window, which is defined as the duration between the release time and due time of this container. The release time of a container depends on crane cycle time - the time it takes for a crane to complete moving a container from the ship to the quay side. To make the test cases as realistic as possible, the crane cycle distribution time table (Table 3.3) from a real-world scenario (Celen et al., 1997) is used to generate container release time. For example, there is a 5% chance that the QC would take 30-40 seconds to process a container. Given this distribution, container release time can be generated as in Algorithm 3.8. Given the release time of a specific container, its due time can then be generated following the calculation outlined in Subsection 3.2.

Algorithm 3.8 GenerateContainerReleaseTime

```
1: releaseTime[0] := 0
2: for i from 1 to n
3:     cycle_time := generateCycleUsingDistribution()
4:     releaseTime[i] := releaseTime[i − 1] + cycle_time //release time of each container
```

Table 3.3: Distribution of QCs cycle time (Celen et al., 1997)

Percentage	Cycle time (sec)
5%	30-40
15%	40-50
20%	50-60
19%	60-70
19%	70-80
10%	70-90
8%	90-120
3%	120-150
1%	150-180

3.6.3 Graph model of the FSP

The approach in Section 3.2 is followed to create a graph model for the FSP. To do so, first all the possible pickup times for each container are calculated by dividing the time window by a discretization unit δ. To create the test cases δ is considered to be 60 seconds. Then, among all nodes those that are compatible with each other are identified. To do so, the travel time of vehicles between PDPs (QCs and SCs) is calculated based on the distances between PDPs and the speeds of vehicles, which are set to be 4m/s and 2m/s for the empty and loaded vehicles, based on common industrial specifications. Given the pickup and travel times, now all the compatible nodes of a given node can be calculated using the procedure in Subsection 3.2.1. By connecting all the compatible nodes a graph model of the FSP can be created.

3.7 Experimental results in static environments

3.7.1 IBM ILOG CPLEX optimiser as a benchmark

To the best of our knowledge there has been no existing EA research for the FSP in ports. As a result, the IP model in Vis et al. (2005), which was solved using the commercial solver CPLEX from IBM, was considered to be the benchmark of this problem. CPLEX is the commercial, state-of-the-art solver and mathematically it is proved that the IP model in CPLEX can reach the global optima given unlimited time and resources. For each test case, the CPLEX source code was developed for the IP model in Vis et al. (2005) and then the model was solved by CPLEX to find global optima. FSEA is then applied to all the test cases to find the optimal solutions by EA approaches. The results of FSEA are then compared with CPLEX to investigate the strengths and weaknesses of each solver/method, as well as to find out which method

is more suitable for the real-world scenarios.

3.7.2 Parameter settings of FSEA and CPLEX

A series of experimental studies were conducted to determine the optimal values for the parameters of FSEA which are 15, 20, and 5 for the $popSize$, γ, and β respectively, where $popSize$ is the size of the population, γ and β are the parameters of the adaptive learning method that is explained in Subsection 3.4.6. FSEA uses the rank selection to select individuals for the next generation. The local search and mutation operators are applied to all individuals in each generation. FSEA stops when one of the following criteria is met first: FSEA reaches the global optima found by CPLEX (if such global optima are available) or FSEA reaches its 2000th generation. In the second criterion, the first time that FSEA finds the best solution will be reported as the process time of FSEA.

For CPLEX, the best estimate search for the node selection and the strong branching for the variable selection were considered. The relative gap tolerance was set to 0.01% of the optimal value. To help address the CPLEX's out-of-memory issue due to the gap tolerance, for the cases where CPLEX runs out of memory the gap tolerance will be increased to 0.1%.

3.7.3 Performance measures

To evaluate the performance of FSEA, the results of FSEA were compared with the results of CPLEX. FSEA was ran 30 times and CPLEX once on each test case (because CPLEX uses an exact technique). There are four possible comparison outcomes. First, FSEA can find the same optimal fleet size as CPLEX in all the 30 runs. Second, FSEA can only find the optimal fleet size as CPLEX in less than 30 runs. Third, CPLEX runs

out of memory and cannot solve test cases but it provides an integer lower bound with no proof of optimality. Fourth, CPLEX runs out of memory and cannot find any lower bounds. In the first case, the time to reach the global optimum is used to compare the two algorithms, using the Mann-Whitney statistical test with a significance level 95%. In the second case, since FSEA cannot find the global optima in all runs, obviously CPLEX outperforms FSEA. In the third case, the algorithm with the lower objective value is considered significantly better. In the case of equal objective value, the Mann-Whitney statistical test identifies the superior algorithm based on the process time. In the fourth case, since CPLEX cannot solve the algorithm, obviously FSEA outperforms CPLEX.

3.7.4 Experimental results

FSEA was coded in C++. All the experiments were conducted on, a Core 2 Duo CPU 2.98 GHz with 3 GB RAM. 165 test cases were created using the settings given in Section 3.6. Detailed results of experiments (i.e. the optimal fleet size and the process time of algorithms) are shown in Table 3.4. In this table, the test cases are indicated by the number of containers and size of buffer, for example c100-b0 is a test case with 100 containers and the size of buffer equals 0. For each test case, the results of the outperforming algorithm are shown by the bold-face font.

For the sake of readability, experimental results in Table 3.4 are summarised into Tables 3.5 and 3.6. Table 3.5 shows the overall comparison results of the two algorithms. This table shows that FSEA significantly outperforms CPLEX in 128 out of 165 cases. CPLEX outperforms FSEA in 32 out of 165 cases. In 5 cases no algorithm is statistically better. CPLEX can only solve 56 cases (the smaller-scale ones) and it cannot solve 109 larger cases due to out-of-memory issues. Out of the 56 cases where CPLEX finds the global optimum, FSEA is able to find the same global optimum in

Table 3.4: This table shows the detailed results of FSEA and CPLEX for ports A and B. For FSEA, the average and standard deviation of the fleet sizes out of 30 runs including the average process times are reported. For CPLEX, where applicable, the optimal fleet size and the process time are reported and the cases where CPLEX aborted due to the lack of memory are shown by MemLim. The cases where an algorithm (i.e. FSEA or CPLEX) outperforms the other one are shown by the bold-face font. Note that the process times of algorithms are in seconds.

Test case	Port A						Port B			
	1 quay crane		2 quay cranes		3 quay cranes		1 quay crane		2 quay cranes	
	FSEA (FS ± SD, time)	CPLEX (FS, time)	FSEA (FS ± SD, time)	CPLEX (FS, time)	FSEA (FS ± SD, time)	CPLEX (FS, time)	FSEA (FS ± SD, time)	CPLEX (FS, time)	FSEA (FS ± SD, time)	CPLEX (FS, time)
c100-b0	(7.00 ± 0.00, 2.55)	(7, 2.10)	(13.00 ± 0.00, 1.18)	(13, 1.86)	(20.00 ± 0.00, 38.41)	**(20, 1.66)**	(11.00 ± 0.00, 27.61)	**(11, 1.88)**	(20.00 ± 0.00, 4.24)	**(20, 1.59)**
c100-b1	**(7.00 ± 0.00, 1.36)**	(7, 2.17)	(12.00 ± 0.00, 21.16)	**(12, 1.92)**	(20.00 ± 0.00, 25.58)	**(20, 1.69)**	(11.00 ± 0.00, 22.37)	**(11, 2.12)**	(20.00 ± 0.00, 4.33)	**(20, 1.68)**
c100-b2	**(7.00 ± 0.00, 1.36)**	(7, 9.41)	(12.00 ± 0.00, 6.81)	(12, 6.35)	(19.00 ± 0.00, 12.27)	**(19, 8.25)**	(11.00 ± 0.00, 12.75)	**(11, 7.63)**	(19.03 ± 0.18, 11.66)	**(19, 5.40)**
c100-b3	**(6.00 ± 0.00, 7.58)**	(6, 35.59)	(10.07 ± 0.25, 389.63)	**(10, 25.79)**	(17.00 ± 0.00, 30.89)	(17, 22.76)	(10.00 ± 0.00, 91.35)	**(10, 28.08)**	(18.00 ± 0.00, 17.36)	**(17, 18.75)**
c100-b4	**(6.00 ± 0.00, 1.33)**	(6, 83.50)	**(10.00 ± 0.00, 14.65)**	(10, 56.84)	(15.17 ± 0.38, 156.58)	**(15, 53.72)**	**(10.00 ± 0.00, 25.32)**	(10, 66.03)	(17.00 ± 0.00, 62.25)	**(17, 42.21)**
c100-b5	**(5.00 ± 0.00, 21.01)**	MemLim	(9.00 ± 0.00, 162.36)	(9, 110.38)	(14.00 ± 0.00, 1341.00)	**(13, 90.44)**	(9.97 ± 0.18, 620.59)	**(9, 125.79)**	(17.00 ± 0.00, 13.38)	**(16, 77.25)**
c100-b6	**(5.00 ± 0.00, 4.00)**	MemLim	**(9.00 ± 0.00, 7.44)**	MemLim	**(13.00 ± 0.00, 50.40)**	(13, 145.06)	**(9.20 ± 0.41, 651.47)**	MemLim	(16.03 ± 0.18, 123.35)	**(16, 135.17)**
c100-b7	**(5.00 ± 0.00, 1.01)**	MemLim	**(8.00 ± 0.00, 261.54)**	MemLim	**(12.10 ± 0.31, 542.34)**	MemLim	**(9.00 ± 0.00, 191.68)**	MemLim	**(16.00 ± 0.00, 9.28)**	MemLim
c100-b8	**(5.00 ± 0.00, 0.92)**	MemLim	**(8.00 ± 0.00, 15.54)**	MemLim	**(12.00 ± 0.00, 4.84)**	MemLim	**(9.00 ± 0.00, 67.03)**	MemLim	**(15.00 ± 0.00, 143.90)**	MemLim
c100-b9	**(5.00 ± 0.00, 1.03)**	MemLim	**(8.00 ± 0.00, 2.87)**	MemLim	**(11.00 ± 0.00, 13.08)**	MemLim	**(9.00 ± 0.00, 19.01)**	MemLim	**(15.00 ± 0.00, 15.73)**	MemLim
c100-b10	**(5.00 ± 0.00, 1.12)**	MemLim	**(7.00 ± 0.00, 159.19)**	MemLim	**(10.00 ± 0.00, 489.74)**	MemLim	**(9.00 ± 0.00, 9.67)**	MemLim	**(14.13 ± 0.35, 255.59)**	MemLim
c200-b0	**(7.00 ± 0.00, 10.68)**	(7, 15.18)	**(13.00 ± 0.00, 6.54)**	(13, 14.59)	(19.00 ± 0.00, 268.91)	**(19, 13.68)**	(11.00 ± 0.00, 34.87)	**(11, 14.42)**	(19.00 ± 0.00, 69.93)	**(19, 13.52)**
c200-b1	**(7.00 ± 0.00, 8.04)**	(7, 16.41)	**(13.00 ± 0.00, 5.39)**	(13, 15.19)	(19.00 ± 0.00, 140.97)	**(19, 13.64)**	(11.00 ± 0.00, 32.80)	**(11, 15.73)**	(19.00 ± 0.00, 76.97)	**(19, 14.18)**
c200-b2	**(7.00 ± 0.00, 6.12)**	(7, 65.81)	**(12.00 ± 0.00, 26.52)**	(12, 63.96)	(18.00 ± 0.00, 314.57)	**(18, 53.55)**	**(11.00 ± 0.00, 21.14)**	(11, 69.60)	(18.20 ± 0.41, 354.03)	**(18, 60.18)**
c200-b3	**(6.00 ± 0.00, 133.00)**	MemLim	**(11.00 ± 0.00, 52.90)**	MemLim	**(17.00 ± 0.00, 325.45)**	MemLim	**(10.00 ± 0.00, 177.71)**	MemLim	**(18.00 ± 0.00, 109.28)**	MemLim
c200-b4	**(6.00 ± 0.00, 23.06)**	MemLim	**(10.70 ± 0.47, 1107.30)**	MemLim	**(16.03 ± 0.18, 513.96)**	MemLim	**(10.00 ± 0.00, 66.64)**	MemLim	**(17.30 ± 0.47, 752.28)**	MemLim
c200-b5	**(6.00 ± 0.00, 6.32)**	MemLim	**(10.00 ± 0.00, 292.26)**	MemLim	**(16.00 ± 0.00, 37.05)**	MemLim	**(10.00 ± 0.00, 31.69)**	MemLim	**(17.00 ± 0.00, 294.07)**	MemLim
c200-b6	**(6.00 ± 0.00, 1.74)**	MemLim	**(10.00 ± 0.00, 27.11)**	MemLim	**(15.00 ± 0.00, 127.96)**	MemLim	**(9.77 ± 0.43, 1984.78)**	MemLim	**(17.00 ± 0.00, 108.79)**	MemLim
c200-b7	**(6.00 ± 0.00, 1.90)**	MemLim	**(10.00 ± 0.00, 4.13)**	MemLim	**(14.47 ± 0.51, 1692.71)**	MemLim	**(9.17 ± 0.38, 1456.62)**	MemLim	**(16.93 ± 0.25, 305.05)**	MemLim
c200-b8	**(6.00 ± 0.00, 2.09)**	MemLim	**(9.33 ± 0.48, 1034.42)**	MemLim	**(14.00 ± 0.00, 74.07)**	MemLim	**(9.00 ± 0.00, 856.76)**	MemLim	**(16.20 ± 0.41, 1601.10)**	MemLim
c200-b9	**(5.70 ± 0.47, 1687.44)**	MemLim	**(9.00 ± 0.00, 209.31)**	MemLim	**(13.20 ± 0.41, 1427.55)**	MemLim	**(9.00 ± 0.00, 465.28)**	MemLim	**(16.00 ± 0.00, 448.66)**	MemLim
c200-b10	**(5.30 ± 0.47, 1225.72)**	MemLim	**(9.00 ± 0.00, 35.95)**	MemLim	**(13.00 ± 0.00, 70.74)**	MemLim	**(9.00 ± 0.00, 157.96)**	MemLim	**(16.00 ± 0.00, 124.39)**	MemLim
c300-b0	**(7.00 ± 0.00, 1.94)**	(7, 50.16)	(13.00 ± 0.00, 222.76)	**(13, 49.74)**	(21.00 ± 0.00, 50.11)	(21, 47.35)	**(12.00 ± 0.00, 33.31)**	(12, 49.01)	(19.00 ± 0.00, 158.23)	**(19, 51.81)**
c300-b1	**(7.00 ± 0.00, 1.70)**	(7, 57.51)	(13.00 ± 0.00, 256.52)	**(13, 55.16)**	**(21.00 ± 0.00, 32.55)**	(21, 50.91)	**(12.00 ± 0.00, 29.42)**	(12, 53.44)	(19.00 ± 0.00, 160.71)	**(19, 50.34)**
c300-b2	**(7.00 ± 0.00, 1.60)**	MemLim	**(13.00 ± 0.00, 117.85)**	MemLim	**(19.77 ± 0.43, 1187.86)**	MemLim	**(12.00 ± 0.00, 25.76)**	MemLim	**(18.17 ± 0.38, 693.46)**	MemLim
c300-b3	**(6.00 ± 0.00, 8.17)**	MemLim	**(12.03 ± 0.18, 581.30)**	MemLim	**(18.27 ± 0.45, 1492.78)**	MemLim	**(11.00 ± 0.00, 200.31)**	MemLim	**(18.00 ± 0.00, 271.42)**	MemLim
c300-b4	**(5.50 ± 0.51, 2091.20)**	MemLim	**(12.00 ± 0.00, 113.38)**	MemLim	**(18.00 ± 0.00, 129.58)**	MemLim	**(11.00 ± 0.00, 110.73)**	MemLim	**(17.57 ± 0.50, 2129.63)**	MemLim
c300-b5	**(5.03 ± 0.18, 878.96)**	MemLim	**(11.90 ± 0.31, 2974.58)**	MemLim	**(17.03 ± 0.18, 1044.85)**	MemLim	**(10.63 ± 0.49, 2200.29)**	MemLim	**(17.00 ± 0.00, 1275.46)**	MemLim
c300-b6	**(5.00 ± 0.00, 231.81)**	MemLim	**(11.00 ± 0.00, 1053.25)**	MemLim	**(17.00 ± 0.00, 148.80)**	MemLim	**(10.00 ± 0.00, 1578.42)**	MemLim	**(17.00 ± 0.00, 547.73)**	MemLim
c300-b7	**(5.00 ± 0.00, 93.97)**	MemLim	**(11.00 ± 0.00, 218.70)**	MemLim	**(16.00 ± 0.00, 1686.53)**	MemLim	**(10.00 ± 0.00, 671.38)**	MemLim	**(17.00 ± 0.00, 158.72)**	MemLim
c300-b8	**(5.00 ± 0.00, 42.56)**	MemLim	**(11.00 ± 0.00, 72.49)**	MemLim	**(16.00 ± 0.00, 206.25)**	MemLim	**(10.00 ± 0.00, 411.31)**	MemLim	**(16.97 ± 0.18, 7348.12)**	MemLim
c300-b9	**(5.00 ± 0.00, 23.21)**	MemLim	**(11.00 ± 0.00, 25.71)**	MemLim	**(16.00 ± 0.00, 113.56)**	MemLim	**(10.00 ± 0.00, 257.02)**	MemLim	**(16.87 ± 0.35, 6027.53)**	MemLim
c300-b10	**(5.00 ± 0.00, 8.53)**	MemLim	**(11.00 ± 0.00, 18.24)**	MemLim	**(16.00 ± 0.00, 98.32)**	MemLim	**(10.00 ± 0.00, 97.45)**	MemLim	**(16.13 ± 0.35, 3760.33)**	MemLim

50 cases. In all the cases where FSEA cannot find the global optima, the differences between the optima found by FSEA and that of CPLEX is only one vehicle.

Table 3.6 shows the performance of the two algorithms with respect to the size of problems. It shows 1) the percentages of times that CPLEX runs out of memory; 2) the average of process time (in seconds) that FSEA and CPLEX need. Generally, it can be seen that CPLEX cannot be used when the sizes of the problems increase. For example, in the cases with 100 containers and the size of buffer is equal to 6 or more CPLEX runs out of memory. Worse, when the number of containers increases to 200 or 300, CPLEX can only solve the problems if the size of buffer is very small, i.e. less than or equal to three or two, respectively. In contrast, FSEA can solve all the larger scale problems in a reasonable time.

Based on the results provided by Tables 3.5 and 3.6, it can be seen that FSEA not only has a reliable performance regarding finding global optima, but it also is able to solve the larger scale problems where state-of-the-art CPLEX fails[5]. To investigate whether FSEA can deal with larger-scale problems, FSEA was also tested on the cases with the number of containers more than 300, namely from 400 - 3000. The results show that FSEA manage to find optimal solutions in reasonable time. Those results, however, are not reported in this chapter because they are unrealistic given the capacity of the studied ports[6].

3.7.5 Optimal fleet sizes and advantages of using buffers

This experiment studies what is the optimal fleet size in different scenarios when the buffer size changes for the studied ports. The results are shown in Figure 3.7. The

[5]Readers are referred to Subsection 3.7.6 for detailed analysis on the FSEA's operators and the impact of the buffer on the optimal fleet size.

[6]The results of FSEA on the larger scale problems are accessible through the following link: http://www.staff.ljmu.ac.uk/enrtngu1/Papers/CIM_large_scales.pdf

Table 3.5: This table shows a summary of the comparison results between FSEA and CPLEX.

Test cases	Number of QCs	Number of test cases	Cases where FSEA is better / total cases	Cases where CPLEX is better / total cases	Cases where no algorithm is statistically better	Solvable cases / total cases		Cases where global optima were found / total cases with known global optima	
						CPLEX	FSEA	CPLEX	FSEA
Port A	1	33	32/33	0/33	1/33	10/33	33/33	10/10	10/10
	2	33	26/33	4/33	3/33	11/33	33/33	11/11	10/11
	3	33	23/33	8/33	2/33	12/33	33/33	12/12	11/12
Port B	1	33	26/33	7/33	0/33	11/33	33/33	11/11	10/11
	2	33	21/33	12/33	0/33	12/33	33/33	12/12	9/12
Total comparison		165	128/165	31/165	6/165	56/165	165/165	56/56	50/56

Table 3.6: This table shows the following: 1) percentages of times that CPLEX ran out of memory, 2) average time for CPLEX and FSEA to solve the test cases. Note that in the cases where CPLEX runs out of memory, CPLEX time is shown by "N/A".

Buffer size	Number of containers								
	100			200			300		
	CPLEX out of memory (%)	FSEA time (sec)	CPLEX time (sec)	CPLEX out of memory (%)	FSEA time (sec)	CPLEX time (sec)	CPLEX out of memory (%)	FSEA time (sec)	CPLEX time (sec)
0	0	14.80	1.82	0	80.02	14.27	0	93.27	49.61
1	0	14.96	1.913	0	52.83	15.02	0	96.18	53.46
2	0	8.97	7.40	0	144.48	62.62	100	405.31	N/A
3	0	107.36	26.19	100	159.67	N/A	100	506.91	N/A
4	0	52.12	60.46	100	492.65	N/A	100	914.90	N/A
5	20	431.67	N/A	100	132.28	N/A	100	1674.83	N/A
6	60	167.33	N/A	100	450.08	N/A	100	712.00	N/A
7	100	201.17	N/A	100	692.08	N/A	100	565.86	N/A
8	100	46.45	N/A	100	713.69	N/A	100	1616.15	N/A
9	100	10.34	N/A	100	847.65	N/A	100	1289.40	N/A
10	100	183.06	N/A	100	322.95	N/A	100	769.58	N/A

results show that when the buffer size increases, the optimal fleet size decreases significantly. For instance, in port A with 100 containers, the optimal fleet size decreases significantly from 20 to 10 when the buffer size increases from 0 to 10. Similar behaviours can also be seen when numbers of containers are 200 and 300. A similar trend is also observed in the case of port B. One reason for this behaviour is that by increasing the size of buffer more space is available in the buffer, so it takes more time for the buffer to become full. Therefore, the length of time window for each job can be increased. Consequently, more jobs can become compatible and the number of jobs assigned to each vehicle can be increased.

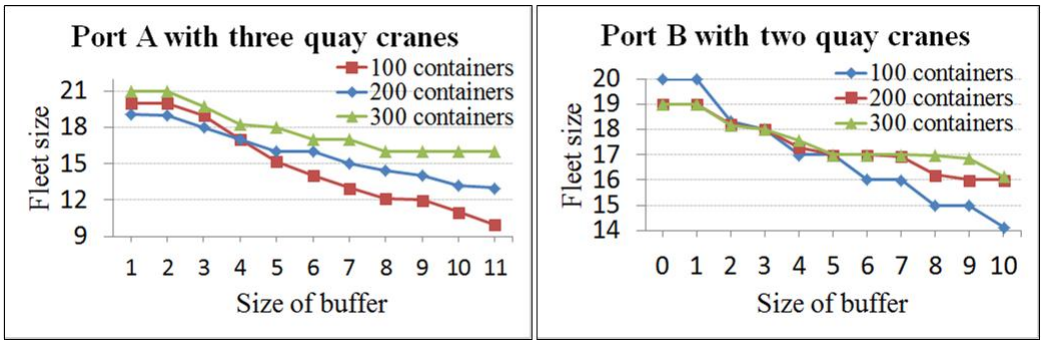

Figure 3.7: Impact of buffers on the optimal fleet size.

3.7.6 Why FSEA works? Analysing the contribution of different algorithmic components

Analysing the impact of the adaptive local search

Two special versions of FSEA were developed where the adaptive local search was replaced by a crossover and a standard mutation operator. By comparing the original FSEA with these two new versions, it can be seen that how the adaptive local search approach can improve the performance over the proposed crossover and standard mutation operators. The standard mutation operator that was used here works by changing values of the variables. For each individual, the mutation randomly selects a number of variables to mutate. It then assigns the selected variables with some random uniform values. Finally, the algorithm applies the repair operator to the mutated solution to repair any possible infeasibility caused by the mutation.

There are some standard crossover operators in the literature such as the one-point and two-point crossover operators. Such crossover operators, however, do not make sense for the FSP. As a result, a new crossover operator is defined that is relevant to the FSP. The proposed crossover works as follows. It selects randomly two individuals and extracts the sequences of jobs of vehicles from the individuals. From each individual,

it selects one vehicle randomly and swaps the jobs of the two vehicles. By such job swapping, the crossover produces two new individuals. Finally, the repair operator will be applied to the new individuals to repair any possible infeasibility caused by the crossover operator.

The impact of the adaptive learning approach

The proposed FSEA also benefits from the adaptive learning approach to get out of local minima. To show how significantly this approach can help FSEA, in the experiments an FSEA was compared with adaptive learning with another FSEA without adaptive learning, i.e. with the local search and mutation operators only.

Experimental results

Because the behaviour of the various FSEA versions (original, +standard mutation, +crossover, +local search and mutation only) are similar in all test cases, to save space here only two representative cases for the two ports will be presented. The average of results over 30 runs over each test case are summarized in Figure 3.8. Note that each bar in the figure has two values inside, the lower value is the average and the upper value is the standard deviation of the optimal solutions over the 30 runs. Figure 3.8 shows that FSEA with the standard mutation or crossover operators cannot find high quality solutions (its best solution has six or seven more vehicles than that of the original FSEA). This shows the significant improvement brought by the local search and mutation operator. Figure 3.8 shows that the adaptive learning can improve the fleet size further by at least one vehicle.

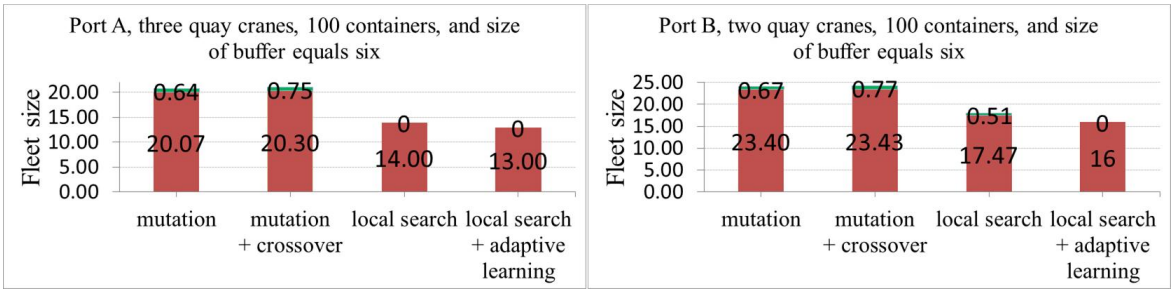

Figure 3.8: Analysis on different components of the EA. This figure analyses the adaptive local search approach by comparing it with the standard mutation and the proposed crossover operators. Each bar in the figure has two values inside, the lower value is the average and the upper value is the standard deviation of found solutions over 30 runs.

3.8 Experimental results in uncertain environments

3.8.1 Case studies

The same test cases used in Section 3.7 are used in this subsection, but with the addition of uncertainties simulated by MC simulation as described in Section 3.5. However, to save space, this section only reports the results of two representative sets of test cases, one from port A and one from port B. It should be noted that although for the static case in Section 3.7 an IP model (Vis et al., 2005) was considered as the benchmark in the uncertain case it is impossible to keep using this IP model as the benchmark. The reason is that there has been no existing research on how to extend the IP model for the FSP to deal with uncertainties[7]. Thus, due to a lack of available benchmarks, the results of MC1-FSEA and MC2-FSEA are compared with the results of FSEA to show the impact of uncertainties on the optimal number of vehicles. High fidelity simulation models are then developed to simulate the real operations in the ports to analyse what would be a robust fleet size in the real ports. This result from the simulations is then compared with the robust fleet size suggested by MC12-FSEA.

[7] One possible way to extend the IP in Vis et al. (2005) for the uncertain case is using the stochastic programming approach. However, it has not been done before and is out of the scope of this thesis.

The purpose is to see if the EA approach can provide an accurate robust fleet size, in comparison to the high-fidelity simulations.

3.8.2 Disruption rates and MTTRs

As recalled in Subsection 3.5.2, the robustness of an FSP solution depends on the disruption rates of vehicles and the MTTRs. The disruption rate and the MTTR of AGVs are chosen from Hoshino and Ota (2007) and Farling et al. (2001)[8]. These two references provide two representative examples of AGVs under low (Hoshino and Ota, 2007) and high (Farling et al., 2001) disruption (called "failure rate" in these papers) rates. In Hoshino and Ota (2007) the disruption rate and the MTTR are 5.0×10^{-6} (disruptions/sec) and 1620 (sec), respectively. In Farling et al. (2001) the disruption rate and the MTTR are 1.0×10^{-3} (disruptions/sec) and 500 (sec), respectively.

3.8.3 Calculating the number of samples for MC simulation in MC1-FSEA

It is important to identify the appropriate number of samples (replications) for the MC simulation in MC1-FSEA. Using too many replications will make the algorithm inefficient, while using too few will make the simulation inaccurate. In this thesis, the *length of confidence interval* [Rubinstein and Kroese, 2011, Section 4] is used to determine the best number of replications in MC1-FSEA.

A series of pilot experiments were conducted to measure the lengths of confidence intervals for a different number of replications. Figure 3.9 shows changes in the lengths of confidence intervals when varying the number of replications from 10 to 100 with

[8]Given that IAVs have not manufactured commercially yet, their actual disruption rate and MTTR are not available yet. Thus, the available disruption rate and MTTR of AGVs were chosen for the experimental analysis in the thesis.

95-99% levels of confidence (port A, 100 containers, buffer size of 6). Figure 3.9 shows that at 100 replications, MC1-FSEA achieves a length of confidence interval of less than 0.3, which is satisfactory. Thus, in this research 100 replications were considered for MC1-FSEA.

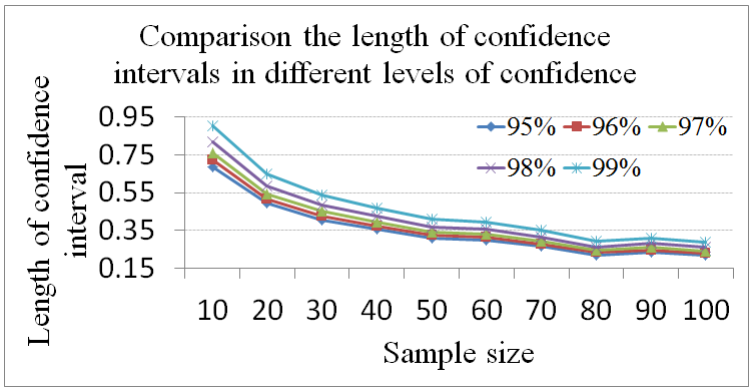

Figure 3.9: Length of confidence intervals in different levels of confidence 95-99% for port A with 100 containers

3.8.4 Comparison results between static and uncertain cases

Results of the proposed algorithms for the static case, the uncertainty in travel time case, the uncertainty in machine process time case, and the uncertainty in both travel and process time case are shown in Table 3.7. All the algorithm settings are the same as in Subsection 3.7.2. The stopping criterion for the algorithms in the experiments is considered 100 generations. Regarding the case of the uncertainty in process time, the sample size for MC simulation is set to be 30, i.e. algorithms like MC2-FSEA and MC12-FSEA will run 30 times with 30 different process times. The fleet size achieved after these 30 runs will then be averaged to calculate the final robust fleet size. The values for the uncertainty in machine process time are the average of 30 runs including the standard deviations. The values after the sign ± are the standard deviation.

It can be seen in Table 3.7 that if the disruption rate is low, uncertainties in travel

Table 3.7: A comparison of the optimal fleet size in different static and uncertain scenarios. The values for MC2-FSEA and MC12-FSEA are averaged fleet sizes ± standard deviations. The process times of the algorithms are reported as minutes.

Port	No. of conts.	Static (FSEA)		Uncertainty in process time (MC2-FSEA)		Uncertainty in travel time (MC1-FSEA)				Uncertainty in process and travel time (MC12-FSEA)			
						Low disruption rate		High disruption rate		Low disruption rate		High disruption rate	
		F.S.	Time	F.S.*	Time	F.S.	Time	F.S.	Time	F.S.	Time	F.S.	Time
Port A	100	10	0.94	10.53±0.49	31.03	10.20	1.93	12.90	29.25	10.46±0.49	58.03	12.86±0.53	975.64
	200	14	2.71	15.10±0.53	77.57	14.30	6.60	18.77	155.49	15.00±0.67	188.35	18.61±0.58	4850.37
	300	16	12.61	16.30±0.64	381.58	16.20	13.77	20.05	313.72	16.20±0.47	404.10	20.10±0.59	9693.48
Port B	100	15	0.92	15.83±0.45	28.34	15.07	5.1	20.22	117.02	15.59±0.60	105.29	20.87±0.75	4031.73
	200	17	2.87	17.13±0.42	83.21	17.27	10.65	23.57	429.69	17.27±0.65	243.41	23.46±0.64	11867.79
	300	17	5.40	17.70±0.52	167.92	17.24	20.05	24.84	775.36	17.88±0.55	473.82	24.44±0.47	23138.02

*F.S. stands for fleet size.

time will not have a significant impact on the optimal number of vehicles. One reason for such behaviour is that since the disruption rate is low, at any time there is likely to be no disruption so there is no need for additional vehicles. Even if there is a rare disruption, the disruption is likely to be isolated so there might be the chance that the current fleet will be able to deal with it without the need for additional vehicles. On the contrary, when the disruption rate is high, disruptions are likely to occur more often and in a more global scale. In such case, the current fleet might not have enough available vehicles to cover all the possible failed vehicles and hence additional vehicles are likely to be required. Table 3.7 also shows that the optimal fleet size under process-time uncertainties is very similar to the optimal fleet size in the static case. This indicates that uncertainties in machine process time seem not to have a significant impact on the fleet size. This result suggests that, at least for the two studied ports, perhaps the uncertainty of process time can be ignored to be able to provide an optimal fleet size solution faster.

3.8.5 Simulation with high fidelity to validate the robust fleet sizes

In this subsection, the effectiveness of MC12-FSEA is evaluated using a simulation approach. The reason to consider only MC12-FSEA is that this algorithm considers both types of uncertainties, and hence is the most general form of the three proposed robust EAs.

To evaluate the effectiveness of MC12-FSEA, ports A and B were simulated with high fidelity to simulate exactly their real operations under uncertainties when using different fleet sizes. To do so, a simulation framework was developed for container terminals to simulate the virtual environments of the case studies. Details of this

framework will be explained in Chapter 6. Figure 3.10 shows the snapshot of the developed simulation model for port A.

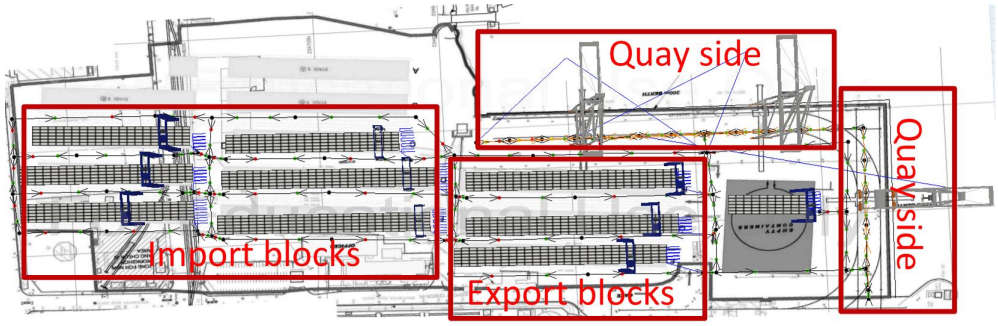

Figure 3.10: A snapshot of the simulation model of the case study container terminal with high fidelity

In the developed simulation models, it is attempted to reproduce the real-world operations, which are the same as the 12 test cases in Table 3.7 with the same properties (e.g. distances between QCs and SCs, speeds of vehicles, number of containers etc). In addition, the simulations incorporate the same uncertainties (travel time and machine time) as considered by MC12-FSEA.

The simulation model was run under different fleet sizes (from 8 - 30 vehicles) to identify the total discharging time of vessels for each value of fleet size. These simulation results helped showing the impact of changing the fleet size on the port performance. More importantly, by running the simulations with uncertainties under different fleet sizes, it is possible to identify an optimal fleet size that is robust against changes. Because the high-fidelity simulation models have been validated by the data from the port partners, it is expected that the robust fleet size observed by the simulation is the most accurate that can be achieved. Hence, comparing the robust fleet sizes found by simulation with the robust fleet sizes by MC12-FSEA can validate the effectiveness of MC12-FSEA. To improve the accuracy of the simulation results, each

simulation was replicated 10 times and the average results are reported.

Note that although a high fidelity simulation can provide the most accurate robust fleet size, it is port-specific. It means that for each different port a new simulation needs to be developed. This is a very time consuming process. In addition, a simulation model generally takes significantly more time to run than an optimisation algorithm like MC12-FSEA. These two disadvantages makes an EA like MC12-FSEA (if it can be proved to be effective) a much better alternative than a simulation model, because it is only needed to develop the EA once to apply it to all different ports.

Figure 3.11 shows the results of the simulation experiments. As can be seen in this figure, in all of the cases for a certain range of fleet sizes the discharging time is almost unchanged. The smallest fleet size in this range can be considered the optimal robust fleet size, because (1) if the fleet size is decreased further, the discharging time will increase, and (2) if the fleet size is increased, any further improvement will not be achieved. In Figure 3.11, the optimal robust fleet sizes found by simulations are squared in red and the optimal robust solutions found by MC12-FSEA are circled in black. Note that the fleet sizes found by MC12-FSEA are converted to integers by adding up the average fleet sizes and standard deviations, then rounding them to the closest integer.

As can be seen in Figure 3.11, in a majority of the cases the robust fleet sizes found by MC12-FSEA are very close to the optimal robust fleet sizes found by the simulation. In 8/12 scenarios the robust solutions found by the two approaches are identical. In the other four scenarios (three from port A under high disruptions), the EA underestimates the fleet size, but the differences are not significant: between one and four vehicles and the difference in discharging time is less than 20 minutes. The possible reason for the EA to underestimate in the scenarios of port A under high disruptions is that these are the most complex scenarios. There might be some other

factors in these scenarios that the EA model has not considered. The results prove that MC12-FSEA is able to accurately estimate the optimal robust fleet size in a majority of the cases.

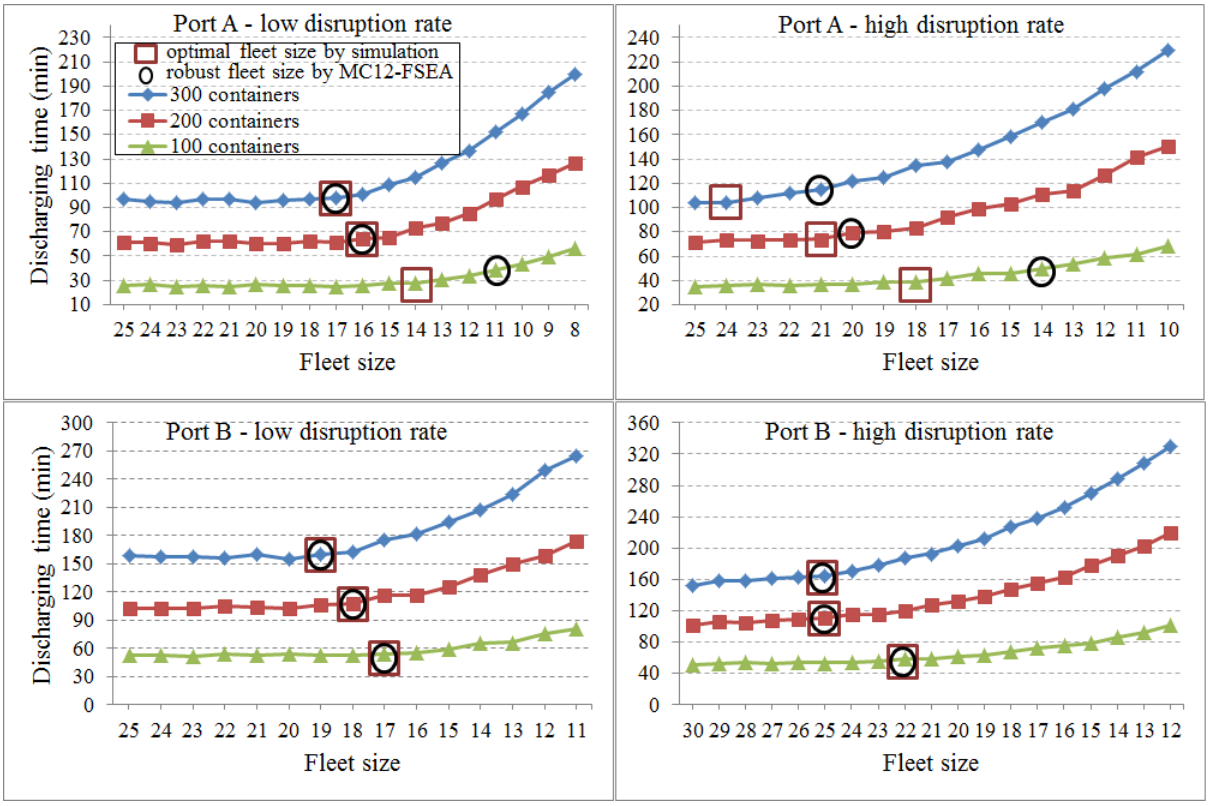

Figure 3.11: Validation of the robust solutions achieved by the EA for uncertain environments using high fidelity simulation. This figure shows the average discharging time of vessels in simulation for the instances in Table 3.7. The robust fleet sizes found by MC12-FSEA for each instance are shown as black circles and the optimal fleet sizes found by simulation are shown as red squares.

3.9 Conclusion

The contributions of this chapter can be summarised as follows:

1. EAs were developed to identify the suitable number of vehicles in ESTTs, in

both static and uncertain situations. The proposed EAs are capable of solving large-scale test cases where the existing approach in the literature (Vis et al., 2005) is limited to only small-scale problems.

2. Two European container terminals were considered as the case studies of this research. Due to the lack of available test cases in the literature, 165 test cases based on the specifications of these two container terminals were generated. This can partially bridge the gap of the lack of available test cases for the FSP in ESTTs.

3. In the static case, the results of FSEA were compared with those of the commercial state-of-the-art CPLEX solver. The results showed that:

   (a) FSEA was significantly better than CPLEX in a majority of cases (i.e. in 128 cases out of 165 cases).

   (b) In 50 cases out of 56 cases FSEA found the global optima found by CPLEX. This shows the effectiveness FSEA to solve the FSP.

   (c) In 109 cases CPLEX failed to solve the test cases due to the memory limitations. In contrast, FSEA was able to solve all the 165 instances in reasonable times.

4. Different components of FSEA were analysed and compared with the standard mutation and crossover operators to show the importance of the developed components.

5. The travel time of vehicles and the process time of machines were considered the main sources of uncertainties. To take into account the uncertainties in the proposed EA, Monte Carlo simulation was combined with the EA to evaluate the fitness of solutions under uncertainties.

6. In the tested uncertain cases, results showed that the effect of uncertainties on the optimal number of vehicles is significant when the disruption rate is high.

7. A simulation with high fidelity was developed to validate the results of the EA for uncertain situations. In this simulation, the same test cases that generated in Section 3.6 were reproduced and uncertainties in the travel time of vehicles and process time of quay cranes were introduced to the simulation. This is to mimic the dynamic of container terminals in the simulation as accurately as possible.

8. Results of high fidelity simulation showed that in a majority of cases the EA found the correct robust solutions and for a few cases the EA underestimated the robust solutions.

3.9.1 Advantages of the proposed methods

The first advantage of the proposed algorithm for the static case (i.e. FSEA) is its efficiency. This algorithm can solve large-scale problems within the reasonable algorithm process time, whereas the existing IP approach for the static FSP (Vis et al., 2005) is unable to tackle large-scale problems due to memory limitations (Tables 3.5 and 3.6). This advantage makes it possible to apply the proposed algorithm to the real-world problems which are usually large-scale problems.

The second advantage of the proposed algorithms (i.e. FSEA-MC1 and FSEA-MC12) is their generality. These algorithms use Monte Carlo simulation to evaluate the robustness of solutions in the population. This Monte Carlo simulation is independent from the EA and it can be generalised to encompass other possible uncertainties that might exist in other case studies.

The third advantage of the proposed algorithm (i.e. FSEA-MC12) is its effectiveness to identify the robust solutions. The results of validation of the robust solutions

achieved by FSEA-MC12 using the high fidelity simulation showed that in most of the cases the optimal solutions achieved by the proposed algorithm are robust against the existing uncertainties in the environment.

3.9.2 Shortcoming of the proposed methods

The biggest disadvantage of the proposed methods for uncertain environments is that they are computationally expensive which is due to the evaluation of solutions using the Monte Carlo simulation at each generation. This, however, is the disadvantage of robust optimisation algorithms that are based on Monte Carlo simulation. Therefore, when the size of problems increases the proposed algorithms become very time consuming to achieve robust solutions with high accuracy. Thus, to solve the problems in reasonable times the accuracy of the algorithms should be reduced by using a lower number of samples. This is because, the higher the number of samples, the more accurate the robust solutions. However, to address this shortcoming, an improved dynamic sampling technique will be developed in Chapter 4 to improve theperformance of the algorithms. This improvement will help the algorithms to achieve high quality solutions using a lower number of samples.

Chapter 4

The Improved EA for the FSP

4.1 Introduction

This chapter is based on the algorithms proposed in Chapter 3. In Chapter 3, uncertainties in the travel time of vehicles were considered as a major source of uncertainties that has a significant impact on the optimal number of vehicles. Such uncertainties may arise from any breakdowns, collisions, or deadlocks. In Chapter 3, the static FSEA was extended for uncertain environments [1] by combining the EA with the MC simulation, to identify the optimal number of vehicles that is robust to the changes in travel time of vehicles. Each solution of FSEA represents a particular number of vehicles and the sequence of jobs (with expected duration) that these vehicles need to carry out. To encapsulate uncertainties, whenever FSEA evaluates a particular solution, it uses a MC simulation to generate n replications of this solution, of which in each replication some possible uncertainties (e.g. vehicle disruptions) may occur. Results of n replications are then combined using an aggregation function to produce fitness of individuals.

[1] This refers to MC1-FSEA and MC12-FSEA, the two variants of FSEA for uncertain environments. In this chapter, for the sake of simplicity these two variants are denoted by FSEA.

Chapter 3 adopted the aggregation function commonly used in robust optimisation: averaging the fitness values over all replications. However, just taking the average might not produce the most appropriate robust solution for some specific scenarios. For instance, if the worst case scenario is desired, the worst fitness value received out of n replications should be considered the fitness of individuals. Now, the challenge is how different robust solutions can be produced and then be compared to identify the most appropriate one. More importantly, the process of Monte Carlo sampling is very time consuming. As a result, when being combined with an EA, an MC simulation will significantly decrease the performance of the EA in terms of computational time. This behaviour in FSEA is also observed. Therefore, the second challenge in this research is how to improve the performance of an EA when being combined with an MC simulation.

This chapter contributes to answering the above questions by proposing some extensions on FSEA. Firstly, to improve performance of combining MC simulation with EAs, the number of samples are reduced on poor solutions and more samples on high quality solutions are used. This can help to reduce the number of samples and improve performance of FSEA significantly. Secondly, different aggregation functions in the MC simulation are incorporated to produce different robust solutions. The robust solutions are then statistically compared to identify the most appropriate robust solutions for port operators.

4.2 Extensions on FSEA

This section first explains a new dynamic sampling strategy to improve the performance of FSEA. It then discusses the proposed approach to aggregate results of the samples in MC simulation to produce different robust solutions.

4.2.1 A new dynamic sampling strategy

As mentioned in Section 3.5, in FSEA the robustness of individuals is evaluated using MC simulation. In FSEA, the same number of samples to all individuals are applied, regardless of whether the quality of individuals is good or poor. The higher the number of samples, the more accurate the robustness of individuals. Evaluating the robustness of poor individuals as accurately as good individuals may not be totally efficient, because those poor individuals would be eliminated in the process of evolution. Therefore, it is a waste of resources. If those poor solutions can be identified and the algorithm spends less time on them, the performance of the algorithm can be improved significantly. It is obvious that at the earlier generations the quality of solutions is poor and in the later generations, the quality of solutions is increased. So, a dynamic strategy can be used to adjust the number of samples along with the increase in the generations. In this chapter, this dynamic strategy will be integrated in a new algorithm named improved FSEA (iFSEA).

iFSEA considers a lower number of samples at the earlier generations and it increases the number of samples step by step during the evolution. The pseudo-code for this is shown in Algorithm 4.1. In this algorithm, an initial number of replications is set as n_0. After g generations it increases the number of replications by s. The number of replications will be increased until it reaches the maximum number of samples, n. From that point to the end of the evolution, the number of replications will be kept as n.

4.2.2 Extension on MC simulation (eMCS)

As recalled in Section 3.5, individuals are evaluated using MC simulation. MC simulation evaluates the robustness of individuals by estimating the possible disruptions of

Algorithm 4.1 NumberOfReplication

```
1: if genCounter % g == 0
2:     if repNo + s > n
3:         return n
4:     repNo := repNo + s
5:     return repNo
6: else
7:     return repNo
```

where *genCounter* is the current generation of iFSEA, g is the generations interval to increase the number of replications, s is the step to increase the number of replications, n is the maximum number of replications and *repNo* is the number of replications.

vehicles. MC simulation in each replication estimates the number of vehicles including the additional vehicles needed to cover disruptions in each individual. An average of n replications in MC simulation is considered the fitness of an individual.

In this research, MC simulation, named eMCS, is extended by considering robustness measures not only an average but also the maximum, minimum, and the most frequently occurred (mode) values of the fleet size as observed out of n replications. Each of those robustness measures can drive the EA to find a different robust solution and hence may be applicable to different scenarios. By using the maximum function, the problem is turned into a minimax problem which looks for the best solution in the worst case scenarios. Specifically, using the Maximum function, iFSEA will try to minimise the largest fleet size that can be observed when applying eMCS with uncertainties to each individual. For a formal description of minimax problems and their applications in robust optimisation, readers are referred to Beyer and Sendhoff (2007).

Similarly, by using the mode and minimum functions, iFSEA will try to minimise the most frequently occurring fleet size and the smallest fleet size, respectively, that eMCS observes for each individual under uncertainty. The pseudo-codes for eMCS and iFSEA are in Algorithms 4.2 and 4.4 respectively.

Algorithm 4.2 eMCS

```
1: Identify f, the fleet size in the given individual
2: D := φ
3: FSL := φ /*FSL is the list of fleet size for the samples*/
4: repNo := NumberOfReplication()
5: for j from 1 to repNo
6:     UJ := φ /*UJ is the list of uncovered jobs*/
7:     a := 0 /*a is the number of additional vehicles*/
8:     for i from 1 to f
9:         D := EstimateDisruptions()
10:        Identify uncovered jobs of vehicles i based on D and add them to UJ
11:    for i from 1 to length(UJ)
12:        if job UJ[i] can be covered by an available vehicle k
13:            Assign job UJ[i] to vehicle k
14:        else
15:            a := a + 1
16:            Assign job UJ[i] to the newly added vehicle
17:    FSL:=FSL ∪ {f + a}
18: switch (aggregationType)
19:    case Max:
20:        return the maximum of FSL
21:    case Min:
22:        return the minimum of FSL
23:    case Avg:
24:        return the average of FSL
25:    case Mode:
26:        return the most frequent element of FSL
```

where $D = \{<t_f, t_r>, t_f$ is the time of disruption and t_r is the time of repair$\}$ and $repNo$ is the number of replications.

4.3 Experimental results

This section first compares the performance of FSEA and iFSEA. It then statistically compares the robust solutions of iFSEA to identify the most appropriate robust solution for port operators.

4.3.1 Test cases and parameter settings

Port A (see Section 3.6) was selected as the case study for the experimental study in this chapter. All settings are from real-world data of this terminal. To generate the

Algorithm 4.3 EstimateDisruptions()

1: $D := \phi$ //D is the set of durations of disruptions for one vehicle
2: $t := 0$ //t is the simulation time
3: **while** $t < makespan$ //$makespan$ is the time that the last job is done
4: Generate t_e, a random exponential value using the parameter λ //λ is the disruption rate of vehicles
5: $t_f := t_e + t$
6: $t_r := t_f + MTTR$
7: $D := D \cup \{< t_f, t_r >\}$
8: $t := t_r$
9: **return** D

Algorithm 4.4 iFSEA

1: **Initialise** population P_t
2: **Evaluate** population P_t by $eMCS()$
3: **for** *genCounter* **from** 1 **to** m /*m is the maximum generations*/
4: **Select** elements from P_t to copy into P_{t+1}
5: **Mutate** population P_{t+1} by Mutate() (Algorithm 3.4)
6: **Recombine** population P_{t+1} by ReduceJobSequences() (Algorithm 3.3)
7: **Evaluate** new population P_{t+1} by $eMCS()$
8: $P_t := P_{t+1}$
9: **return** the best individual

test cases similar to Section 3.6, the number of QCs and size of buffer are varied. The number of QCs is varied from 1 to 3 because in this container terminal at most three QCs can work on one vessel simultaneously. The size of buffer (number of cassettes) under the cranes is varied from 0 to 10. The number of containers to be discharged is 100. In this container terminal, 6 SCs are available and it is assumed that containers are divided evenly between those SCs. Similar to Section 3.6, IAVs are considered transfer vehicles in this port. The speed of vehicles is considered 4m/s for empty IAVs and 2m/s for loaded IAVs. As explained in Chapter 3, the actual disruption rate and MTTR for IAVs are not available yet. Thus, the same disruption rate and MTTR as in Farling et al. (2001) are used. In these experiments, the case with the high disruption rate is considered, given that the experiments results in Section 3.8 showed that the impact of the low disruption rate on the optimal fleet size is minor. The parameter

settings for FSEA and iFSEA are shown in Table 4.1.

Table 4.1: Parameter settings for FSEA and iFSEA

	Parameter	Value	Descriptions
iFSEA	n_0	60	Initial number of replications
	s	20	Step to increase the number of replications
	g	10	Generations interval to increase the number of replications
	n	100	Maximum number of replications
iFSEA & FSEA	$popSize$	15	Size of population
	λ(disruption/sec)	1.0×10^{-3}	Disruption rate of IAVs
	MTTR(s)	500	Mean time to repair
	other	As in Section 3.8	

4.3.2 Performance of iFSEA compared with FSEA

One of the purposes of proposing iFSEA is to improve the computational time of FSEA without decreasing the quality of solutions. To compare the quality of solutions of the two algorithms, FSEA and iFSEA were applied to the same test cases. The Mann-Whitney statistical test is then used to see whether the results of the two algorithms are significantly different. The significance level is 95%.

The results showed that there is no significant difference between the solutions of iFSEA and FSEA. The p-values of the statistical analysis are 0.24, 0.48, 0.68 for the test cases of 1, 2, and 3 QCs, respectively. The results show considerably high p-values and it confirms that the quality of solutions has not deteriorated in iFSEA.

Figure 4.1 shows differences between the process time of FSEA and iFSEA. It shows that iFSEA in all the cases could solve the problem considerably faster than FSEA.

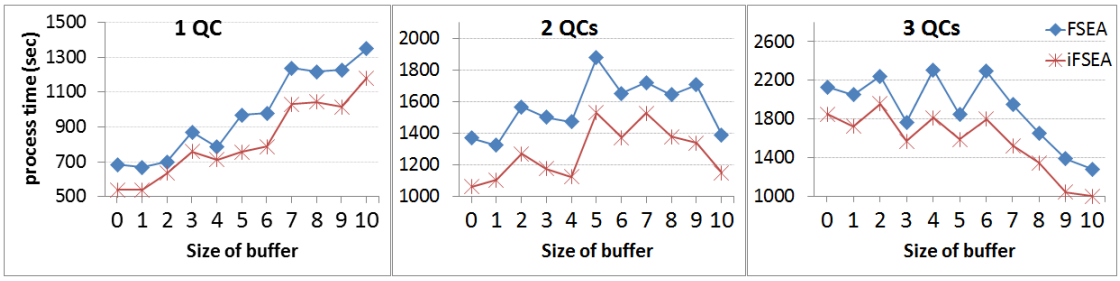

Figure 4.1: Performance comparison: FSEA vs iFSEA

4.3.3 Comparison between robust solutions

Results of applying iFSEA to the test cases using different aggregation approaches (e.g. Min, Max, Avg, and Mode) are shown in Table 4.2. Note that a higher number of IAVs may increase productivity under uncertainties, however, it can be expensive to deploy too many IAVs. Therefore, the port operators need an accurate comparison between the robust approaches, and based on that they can carefully select the most appropriate robust solution. As a result, the robust approaches are compared using the Mann-Whitney test to provide a tool that can help port operators to identify the most appropriate robust solution.

Table 4.2: Different robust numbers of IAVs using different aggregation functions

Buffer size	1 QC				2 QCs				3 QCs			
	Min	Max	Mode	Avg	Min	Max	Mode	Avg	Min	Max	Mode	Avg
0	7	10	9	8.96	13	17	15	15.55	19	23	21	21.81
1	6	9	8	8.27	12	17	14	14.64	17	22	19	20.25
2	6	9	8	7.51	11	15	13	13.64	15	20	18	18.16
3	6	9	8	7.87	10	14	12	12.32	13	17	15	15.02
4	5	8	7	6.00	9	12	10	10.29	12	17	15	15.49
5	5	8	7	6.76	9	12	11	11.00	12	15	13	13.51
6	5	8	7	6.75	9	11	11	10.85	12	15	14	16.69
7	5	7	7	6.87	9	11	11	10.79	11	14	13	12.53
8	5	7	7	6.67	9	11	10	10.00	10	13	12	11.76
9	5	7	6	6.55	8	9	9	9.44	9	11	10	10.30
10	5	7	5	6.00	8	9	9	9.18	9	11	10	9.00

Table 4.3 shows results of which aggregation functions are significantly different

from those of other aggregation functions. For instance, with 1 QC, results of Min are significantly different from the results of Max, Avg, and Mode with the p-values equal 0.0017, 0.0011, and 0.0470, respectively. In contrast, in this case, results of Avg and Max are not significantly different from the p-value equals 0.0409.

The port operators can look at the results of Tables 4.2 and 4.3 to select the most appropriate number of IAVs. For example, in the case of 1 QCs, Min is not a reasonable choice, because Min is significantly different from Mode and Avg. This means that in a majority of cases the number of IAVs achieved by Min is not enough. In contrast, if the port operators want to be on the safe side, Max is a good option for them. This is because, Max is not significantly different from Mode, meaning that the worst case is also likely the most frequently occurred case. In addition, even though results of Max and Avg in this case are significantly different, the p-value for this case is not considerably high. As a result, for the case with 1 QCs, Max is a reasonable choice.

Table 4.3: Mann-Whitney comparisons of iFSEA using different aggregation functions. The sign "+" or "-" means there is or there is no significant difference, respectively.

Aggregation function	1QC		2 QCs		3 QCs	
	significantly different	p-value	significantly different	p-value	significantly different	p-value
Min vs Max	+	0.0017	+	0.0354	-	0.0538
Min vs Avg	+	0.0011	+	0.0098	-	0.0790
Min vs Mod	+	0.0470	-	0.1252	-	0.1705
Max vs Min	+	0.0017	+	0.0354	-	0.0538
Max vs Avg	+	0.0409	-	0.2452	-	0.3347
Max vs Mod	-	0.3589	-	0.3347	-	0.2883
Avg vs Min	+	0.0011	+	0.0098	-	0.0790
Avg vs Max	+	0.0409	-	0.2452	-	0.3347
Avg vs Mod	-	0.6410	-	0.5130	-	0.3589
Mod vs Min	+	0.0470	-	0.1252	-	0.1705
Mod vs Max	-	0.3589	-	0.3347	-	0.2883
Mod vs Avg	-	0.6410	-	0.5130	-	0.3589

4.4 Conclusion

This chapter enhanced the EAs proposed in Chapter 3 in terms of performance and identifying robust solutions based on various robustness measures. This chapter has the following contributions:

1. It improves the performance of the algorithm in Chapter 3 by proposing a new dynamic sampling strategy. The new sampling strategy evaluates the fitness of poor solutions at the earlier generations with low accuracy i.e. it uses a lower number of samples for poor solutions. By increasing the generation number, the quality of solutions increases and hence the EA evaluates the fitness of high quality solutions with the better accuracy i.e. the algorithm uses a higher number of samples in MC simulation for high quality solutions.

2. Different robustness measures were proposed to produce different robust solutions based on different preferences of port operators.

3. The robust solutions were statistically compared to identify the robust solutions that are significantly different from each other. The statistical comparison gives better insight to port operators to identify the most relevant robust solutions based on their requirements.

Chapter 5

A simulation model to compare the performance and cost of the case study with IAVs and trucks

5.1 Introduction

Container terminals play a vital role in international supply chains, since container terminals are major interfaces to transfer/distribute containers (carrying 90% of non-bulk world trade goods as of 2009 (Ebeling, 2009)). How container terminals handle goods greatly influences emissions and final cost, because up to 50% of cost could be due to handling and logistics (Rodrigue et al., 2013, Chapter 5). Thus, improving container terminals' efficiency is an important/practical issue (Ha et al., 2007). The growth in the global container market has made container terminals key hubs of global supply chain networks. Therefore, if a container terminal wants to be successful in this market, it should improve its performance and also be able to keep its operational costs at the lowest level (Soriguera et al., 2006). Moreover, with the growth of

containerisation, container terminals have to face with the problems of limited space (Gambardella et al., 1998). Some container terminals, especially European ports, have difficulties in coping with congestion caused by the increase in equipment and activities in ports. Due to the limited available land it is not possible to increase the area of container terminals despite the need for increasing capacity (Henesey, Aslam and Khurum, 2006). Thus, the capability of equipment to perform in confined spaces has become a competitive advantage.

Due to the aforementioned issues, container terminals have been looking for new technologies to improve their performance. The first step is to identify the most suitable sets of equipment. However, since the introduction of containers in 1960, identifying the optimal amount of equipment and capacity of container terminals has always been a challenging task due to the complex nature of the problem. One possible way to solve this challenging task is to use simulation. Simulation is a scientific approach not only to study a system without actually disturbing it (Demirci, 2003), but also to evaluate concepts that have not been used in the real world (Henesey, Aslam and Khurum, 2006; Yun and Choi, 1999). Therefore, for a container terminal, a simulation study can be carried out to predict the effect of applying different types of equipment, as well as the ideal amount of equipment to meet the performance target (Ha et al., 2007; Yun and Choi, 1999; Parola and Sciomachen, 2005; Bielli et al., 2006). This is the focus of this chapter.

In this chapter, a simulation model is developed to identify the optimal fleet size in terms of cost and performance to assist investment decisions for the case study container terminal. The impact of using IAVs in comparison with trucks on the performance and cost in this terminal is investigated.

This chapter is structured as follows. Section 5.2 describes the developed simulation model to investigate how IAVs can be accommodated in container terminals and

whether they can contribute to the improvement of performance of container terminals. All the specifications and settings of the model are explained in this section. Section 5.3 discusses the results of the simulation study. It first explains the chosen performance measures to evaluate the results of using trucks and IAVs to identify the optimal fleet size of IAVs and trucks. It then provides the results of the cost model based on the given optimal fleet size. Finally, Section 5.4 concludes this chapter.

5.2 The simulation model

This section first provides a brief introduction to the FlexSim CT simulation software. It then explains the specifications of the developed simulation model to study the productivity of trucks and IAVs, as well as their optimal fleet sizes, in the case study port.

5.2.1 FlexSim simulation software

FlexSim CT is a purpose-built container terminal simulation tool to develop simulation models. FlexSim CT is an extension of the FlexSim software (Nordgren, 2003) where it offers specific features for simulating container terminals such as the berth planner, quay cranes, stacking blocks and stacking cranes. The benefit of FlexSim and FlexSim CT is that, in addition to the standard discrete-event simulation features, they support good 3D visualisations, as well as the ability to rewrite some part of the source code (written in C).

5.2.2 The case study and its layout

In this chapter, similar to Chapters 3 and 4, port A is considered the case study [1]. Figure 5.1 shows a snapshot of the simulation model with the map of port A as the background. It can be seen that port A has three quay cranes at berth, six blocks to stack importing containers and three blocks to stack export containers. Each of the stacking blocks is equipped with one rubber-tyred gantry crane to stack/unstack containers. The positions of quay-side and stack-side areas are shown in Figure 5.1.

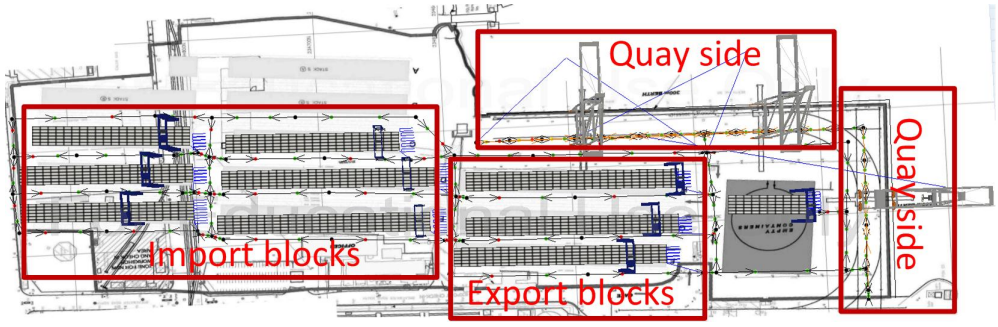

Figure 5.1: The layout of port A. This figure shows the position of import/export blocks in the stack-side area and also the berths at the quay-side area.

In port A, trucks are currently being used to transport containers between the quay-side and stack-side areas. Trucks follow a loop-shaped layout between the quay-side and stack-side areas. It means that once a truck drops off/collects containers to/from a block, it will have to travel all the way to the end of the block, then takes a long circle round the port to go back to the quay-side area (Figure 5.2). This is because trucks cannot turn in the narrow space inside the stack-side areas.

Different from trucks, IAVs are better at manoeuvring in confined places thanks to their novel 180-degree-rotation wheels. The wheels allow them to move in any direction, including moving forward, backward and sideways without having to turn.

[1] This container terminal has committed to considering the results of this chapter to enhance their operations. Due the confidential agreements with this container terminal its identity cannot be revealed.

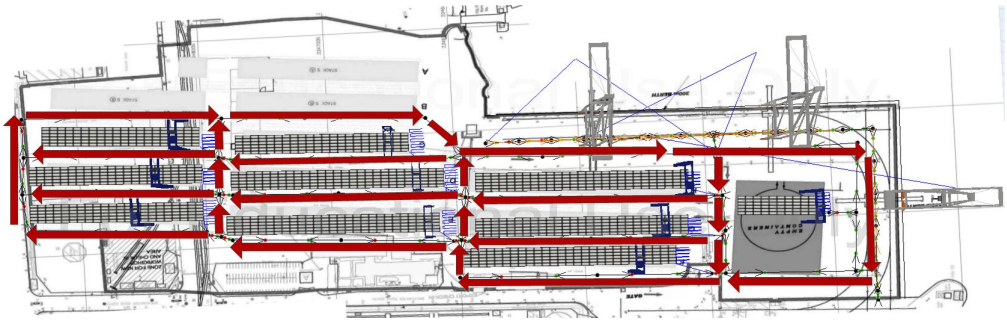

Figure 5.2: The travel routes of trucks in port A.

It means that, in port A, once an IAV has picked up/dropped off a container along a block, it can reverse back to the quay-side using the same route without having to take a long circle like the trucks, or it can change direction at any point (Figure 5.3). Thus, to do the same job in the same container terminal, an IAV has a significantly shorter travel distance than a truck. This potentially leads to time and money savings.

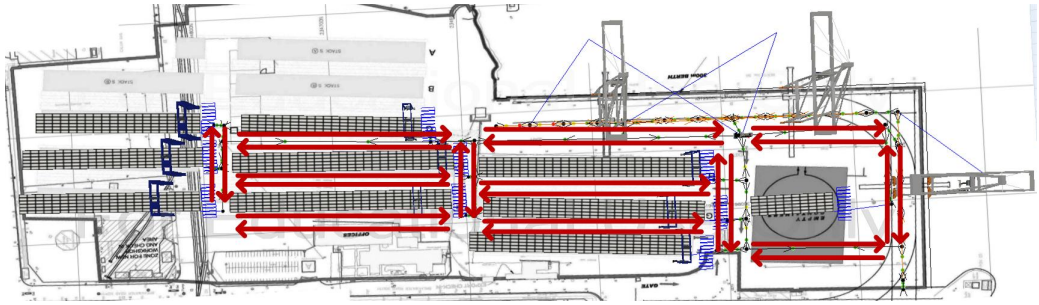

Figure 5.3: The proposed travel routes of IAVs. In these routes, IAVs do not need to go to the end of the roads to turn around or follow a loop like trucks. Instead, they can move forward, backward, or sideways using the shortest available path.

5.2.3 Berth configuration

The berths' layouts is simulated using real-world data as in Figure 5.3. Following real-data, weekly transactions of port A were simulated, of which the busiest transactions has about 300 containers to be discharged from the vessels and 300 containers to be loaded to the vessels. Containers were assumed to be distributed evenly between the

quay cranes and import/export blocks.

5.2.4 Quay and stacking cranes

Based on real data from this container terminal, the cycle time of each quay crane was considered two minutes, i.e., it takes on average two minutes for a quay crane to locate a container, pick it up and then place it on top of a vehicle, an empty cassette or a vessel. Based on the real data, the cycle time for a stack crane to stack/unstack a container was considered to be on average 3.5 minutes. The container placement strategy used in the simulation is keeping container stacks at the lowest height possible i.e. the number of containers that are stacked on top of each other should always be minimal. This is the strategy that is currently being used in port A.

5.2.5 Vehicles

To investigate the difference in port productivity using IAVs against trucks, two simulation models were developed: one for trucks and one for IAVs. There are two main differences between the simulation models. The first difference is the travel routes. As mentioned in Subsection 5.2.2, IAVs' better manoeuvrability help them to travel shorter distances (compared with trucks) to carry out the same task (Figure 5.3), so the two simulations have two different travel routes. The second difference is the (in)ability of vehicles to pick up/drop off containers. IAVs can pick up/drop off containers by themselves when being combined with the cassettes while trucks cannot do so.

The current version of FlexSim CT supports only two types of transfer vehicles: truck and straddle carrier (a vehicle able to top-lift containers and stack them to a container block without the need of a stacking crane). FlexSim CT does not support

IAVs or any similar type of vehicle. It means that it is needed to create a new vehicle object for the IAV in FlexSim CT.

To address this limitation, the straddle carrier object is modified. A straddle carrier is a vehicle that is somewhat similar to an IAV in the sense that it can also pick up and drop off containers. However, there are some significant issues, resulting from the differences between a FlexSim CT's straddle carrier (CTSC) and an IAV: 1) the appearances of two vehicles are very different; 2) CTSC can only pick up/drop off containers from/to the ground in the quay side while IAVs need to pick up/drop off containers in both quay side and stack side; 3) CTSC does not work with stack cranes - they can stack containers to the storage blocks by themselves. On the contrary, IAVs need to work with stack cranes - they can only deliver containers to the ground next to a container block, and then the crane in that block will do the stacking.

To overcome the first issue, difference in appearances, in the simulation the 3D image of a straddle carrier is just simply replaced with that of an IAV. To overcome the second and third issues, the straddle carrier object is combined with another FlexSim CT object - the transfer area. In FlexSim CT, a transfer area is a waiting area dedicated to truck-like vehicles to wait before being served by stacking cranes (Figure 5.4).

For the purpose of overcoming the two aforementioned issues, transfer areas are used for a different purpose: to connect CTSCs with stack cranes. To do so, a transfer area is placed next to each container block, which in turn is served by one stack crane. As mentioned previously, CTSCs do not work with stack cranes because CTSCs can stack containers in blocks directly without the cranes. However, CTSCs do work with transfer areas because transfer areas can be considered special blocks of containers. So, CTSCs and stack cranes can work together by asking CTSC to bring containers to transfer areas, then asking stack cranes to pick up those containers from the transfer

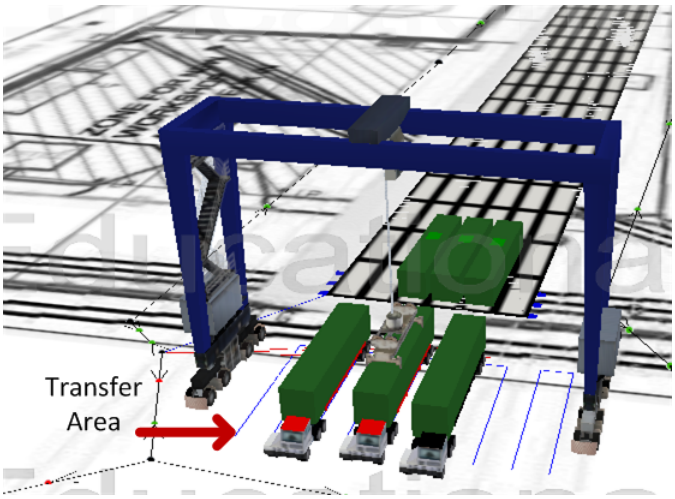

Figure 5.4: The transfer area object for the implementation of buffers at the stack-side area

areas and stack them to the blocks (Figure 5.5). This way, issue 3 is resolved.

Because transfer areas are placed in the stack areas, this makes it possible for CTSCs to pick up/drop off containers in the stack side. This resolves issue 2.

By modifying the existing straddle carrier object and adding the transfer object, all the three issues are resolved and hence a CTSC works exactly like an IAV, i.e. to pick up containers from a buffer in the quay side, then bring them to another buffer in the stack side and vice versa. It means a CTSC can be used to represent an IAV. Similarly, a transfer area can be used to represent a buffer for cassettes in the stack side. Note that in the quay side it is not needed to use the transfer area to represent a buffer because CTSC does support pick up/drop off container from/to the ground on the quay side by default, i.e. CTSC have their own buffer on quay side by default.

Figures 5.5 and 5.6 show how all the modified objects work together to simulate the behaviour of IAVs and cassettes in the port.

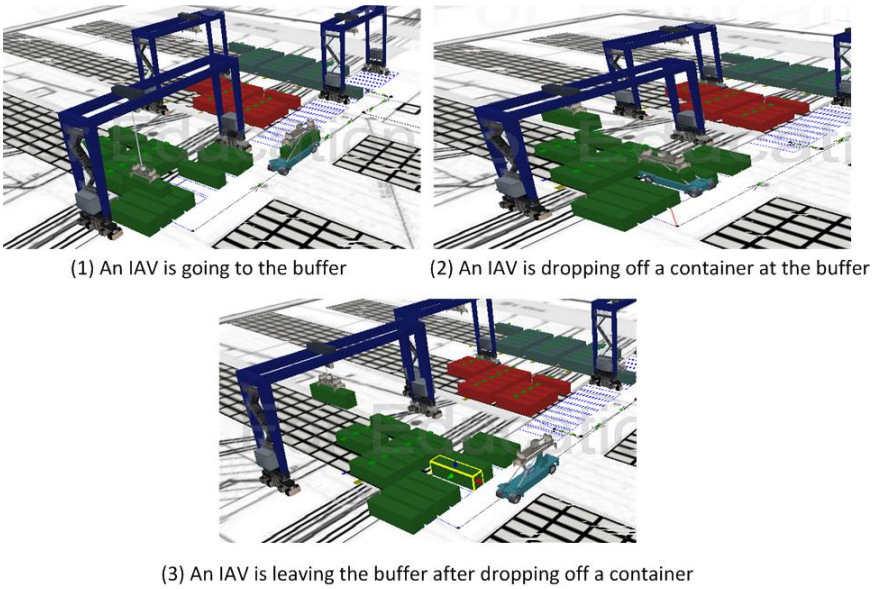

Figure 5.5: Buffers of containers at the stack-side area

5.2.6 Vehicles speed and dispatching strategy

In this chapter, for both IAVs and trucks the same realistic dispatching strategy is considered. In this strategy, for each container, the closest available vehicle to that particular container would be dispatched to handle the container. Table 1 shows the speeds of vehicles when being loaded and empty. It can be seen that the speeds of trucks (from real-world data in the port) are significantly higher than the speed of IAVs (hypothetical, worst-case scenario value). Note that IAVs actually can move much faster than the values used in this thesis. However, since IAVs have not been implemented commercially yet the worst-case scenario with the lower bounds for the IAV speeds is considered.

Table 5.1: Vehicle speeds in the simulation models.

Speed (m/s)	Truck	IAV
empty speed	13.41	4
loaded speed	11.18	2

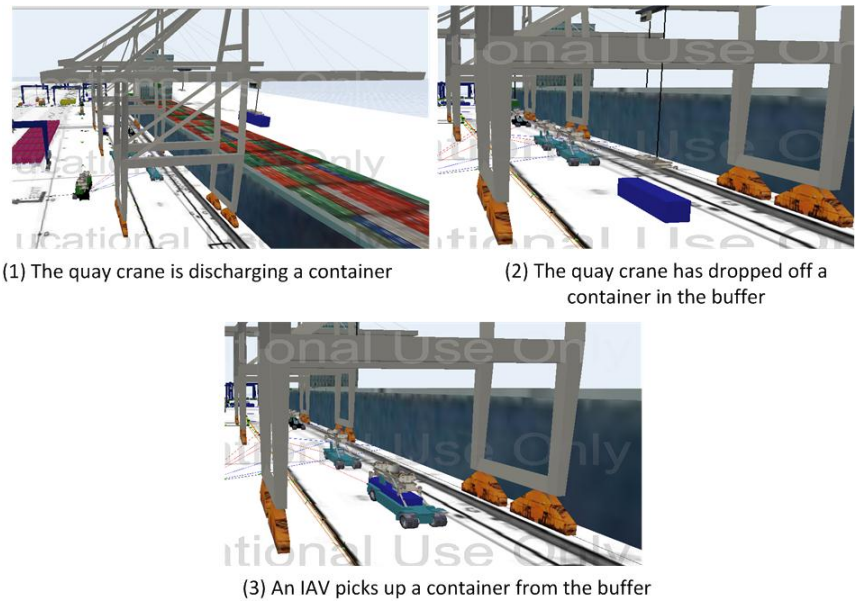

Figure 5.6: Buffers of containers at the quay-side area

5.3 Experimental studies

This section first compares results of the simulation models of the terminal in two cases: using trucks and using IAVs without cassettes (i.e. IAVs do not pick up/drop off containers by themselves). To do so, a sensitivity analysis approach is followed by varying the number of vehicles from 3 to 25 to investigate the performance of port A using these different numbers of IAVs and trucks. The impact of using cassettes on the port performance is studied by varying the size of the buffers (number of cassettes) from 1 to 10 and also varying the number of vehicles from 3 to 25. Finally, the results of the experiments are used to identify the optimal type and number of vehicles and also the size of the buffers for port A. To have a better understanding of the performance of port A, the results of discharging and loading are reported separately. This is because the optimum number of vehicles for discharging and loading can be significantly different, given the differences between the number of import and export blocks and also the geographical positions of import and export blocks in regard to the quay-side area

(Figure 5.1).

5.3.1 Performance measures

In container terminals, it is very important to minimise the total discharging/loading time, because vessels at ports are much more expensive than they are on the sea (Steenken et al., 2004). The total discharging/loading time at ports is highly dependent on the total loading/discharging time when containers are loaded/discharged to/from the vessel. The shorter the loading/discharging process time is, the shorter time the vessel has to stay. The total loading/discharging time, in turn, is dependent on the quay crane net moves per hour. This is because containers are discharged/loaded using quay cranes from/to vessels and hence the higher the quay crane moves per hour, the shorter total loading/discharging time. Therefore, the quay crane net moves per hour was chosen as the performance measure for the simulated port. The total discharging/loading time at berth is then calculated given the quay crane net moves per hour. Using the total discharging/loading time at berth the optimal number of vehicles is identified.

5.3.2 Simulation validation

Before the simulation model can be used to study the impact of trucks and IAVs, it is needed to validate the simulation model against historical data from the real environment (port A). In the validation phase, the simulation was ran using exactly the same settings as recorded in the port's historical data to see if it is possible to simulate the same average productivity (average number of moves per hour) as recorded in historical data. The same fleet size of 10 as currently used in the port is used. In these experiments, the number of containers was varied from 100-300 and the number quay

cranes was varied from 1-3 per transaction to cover almost all the possible realistic scenarios. The simulation produced an average quay crane net moves per hour of 24.1, which is close to the real-world value of 25 moves per hour as recorded by the port. This validation shows that the simulation is valid and accurate. It hence can be used to analyse the difference between trucks and IAVs, as will be shown in the next sections.

5.3.3 Experiment settings

Two simulation models were created for the experimental study, one for trucks and one for IAVs. Each model was run 30 times and the average results of the 30 runs were reported. All the experiments were conducted on a 32-bit Intel(R) Core(TM)2 Duo 2.93 GHz with 3 GB RAM.

5.3.4 Trucks versus IAVs - without cassettes

This subsection compares trucks and IAVs where no buffers for IAVs are considered. In other words, in this comparison the ability of IAVs to pick up and drop off containers by themselves is not considered. Therefore, the main differences between IAVs and trucks in this comparison are: the different travel routes for IAVs and trucks (Section 5.2) and different speeds of vehicles (Table 5.1).

Figure 5.7 shows the comparison results based on crane net moves per hour. It can be seen that the performances of the two vehicles are quite similar. Obviously the IAVs will give a much better performance if they are allowed to move faster. This suggests that the ability of IAVs to manoeuvre better can have a positive impact on the port performance. Figure 5.7 also shows that without the cassettes, both trucks and IAVs cannot increase the quay crane net moves per hour to more than 28 (i.e. no waiting time of quay cranes for vehicles) even when the fleet size is 25.

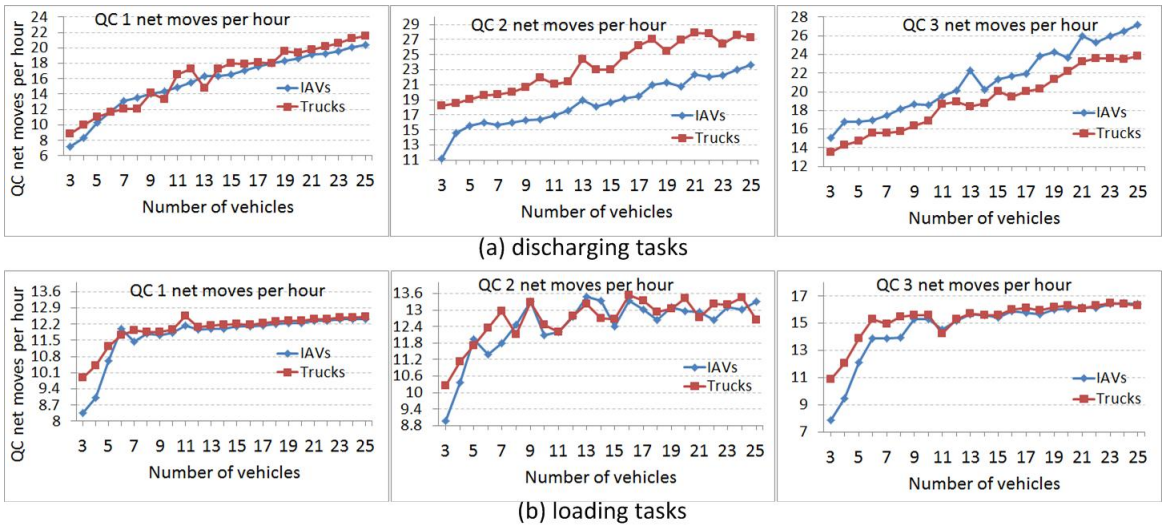

Figure 5.7: Quay crane net moves per hour comparison: IAVs without cassettes vs trucks. Plot (a) shows the quay crane net moves per hour for the discharging tasks. Plot (b) shows the quay crane net moves per hour for the loading tasks.

The results of Figure 5.7 are used to calculate the total discharging/loading time at berth. This measure is very important, since it shows how long vessels have to stay at berth. To do so, the net moves per hour of the slowest quay crane is used and by considering the number of containers that are moved by that particular quay crane, the total discharging/loading time at berth can be calculated. Note that in the experiments, all quay cranes have to move roughly the same number of containers due to the way containers are distributed to cranes i.e. the total throughput of the quay cranes are equal (Subsection 5.2.3). The calculation for the total discharging/loading time at berth is as follows:

Let:

q: number of quay cranes

m_i: net moves per hour for quay crane i, $1 \leq i \leq q$

n_l: total number of containers to be loaded

n_d: total number of containers to be discharged

s_l: average vessel loading time

s_d: average vessel discharging time

$$s_l = \frac{n_l/q}{min(m_1, ..., m_q)} \tag{5.1}$$

$$s_d = \frac{n_d/q}{min(m_1, ..., m_q)} \tag{5.2}$$

Using Equations 5.1 and 5.2, the total discharging/loading time is calculated (Figure 5.8). It can be seen that by using IAVs without the cassettes, vessels can be served in almost the same amount of time as by trucks in most of the cases. Note that in this experiment, the ability of IAVs to utilise buffers has not been considered. Given that IAVs are at a significant disadvantage due to their speed being severely restricted to be much lower than that of trucks, the fact that they still are able to get the same total discharging/loading time highlights the advantages of IAVs in being able to move in more flexible routes (Figure 5.3). Figure 5.8 also shows that there is a significant increase in the total discharging/loading time for IAVs against trucks where the number of vehicles is less than 5. This is because when the number of vehicles is very small, the higher speed of trucks can compensate for the shorter travel routes of IAVs. The other interesting finding about these results is: for the loading case (Figure 5.8-b) the total discharging/loading time at berth is not significantly affected by the number of vehicles. For example, the difference between the total discharging/loading time at berth for 25 vehicles and 6 vehicles is only 0.4 hours. This is because for the loading scenario the stack cranes are the bottlenecks due to two facts: (a) the vehicle travel time is short (due to the short distance between the quay side and the stack area used for loading a.k.a an export area), and (b) stack cranes are much slower than quay cranes. The combination of (a) and (b) means that once a vehicle has delivered a container to the quay crane and come back to the stack crane to get another one,

it will have to wait because the stack crane has likely not finished picking up its next container yet. Because vehicles will likely have to wait for stack cranes regardless of how many vehicles are there, the fleet size does not play a major role in reducing loading time. To reduce loading time, the port operator would have to add more stack cranes, or use a more effective type of stack crane.

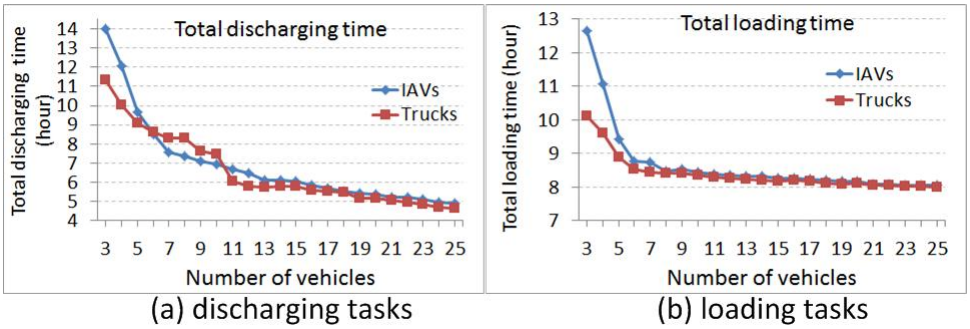

Figure 5.8: Total loading and discharging times: IAVs without cassettes vs trucks. This figure compares the total discharging/loading time at berth using IAVs (without cassettes) and trucks. As can be seen the total discharging/loading time at berth for IAVs and trucks are similar specially for the number of vehicles greater than five.

5.3.5 Trucks versus IAVs - with cassettes

Recall from Subsection 5.3.4, the performance of IAVs without cassettes (i.e. no buffers) is quite similar to that of trucks and in some cases IAVs are even better than trucks (Figure 5.8) thanks to IAVs' manoeuvrability. In this subsection, the impact of utilising the buffers of containers with IAVs on the performance of port A is investigated. Note that in this section, to save space, the results of discharging tasks are only reported. This is because as explained in Subsection 5.3.4, the impact of the optimal number of vehicles on the quay crane net moves per hour for the loading tasks is not significant.

To investigate the impact of utilising buffers in port A, a sensitivity analysis approach is followed by varying the size of buffers (i.e. the number of available places

for cassettes next to a crane) from 1 to 10 and the number of IAVs from 3 to 25. Note that because IAVs need some additional time to pick up/drop off cassettes, this has to be taken into account in the simulation with cassettes. It is estimated that the IAVs will need an average 48 seconds to either pick up or drop off containers. Results of the simulation are presented in Tables 5.2 and 5.3.

Results in this experiment clearly show the advantages of using buffers. As mentioned earlier, Figure 5.7 shows that without cassettes it is not possible to increase net moves per hour to around 30 (no waiting time of quay cranes). Table 5.2 shows that, however, with the use of cassettes a zero crane waiting time can be achieved with a much smaller fleet size (11 vehicles) if 9 cassettes or more are used. There is also a wide range of combinations of different fleet sizes and buffer sizes to achieve no waiting time for quay cranes as shown by the blue cells in Table 5.2. The use of cassettes also allows achieving a reasonably high crane net moves per hour (just over 25 moves) with just 9 or 10 vehicles.

As can be seen in Table 5.2 the impact of buffers on the productivity of quay cranes is significant. To investigate how much the total discharging time can be reduced by the utilisation of the buffers, this measure is calculated using Equations 5.1 and 5.2. The results are reported in Table 5.3. It can be seen that with 11 IAVs and the size of buffer equals 10 the total discharging time is 3.37 hours, 2.68 hours shorter than the discharging time achieved by the same number of trucks (6.05 hours). In addition, if trucks are used it will not be possible to achieve the small total discharging time achieved by IAVs (3.37 hours). Even if the number of trucks are increased to a large number of 25, the discharging time is still 4.64 hours, significantly larger than the value achieved by IAVs (Figure 5.8).

Table 5.2: This table shows the quay crane net moves per hour by varying the number of vehicles from 3 to 25 and size of the buffer from 1 to 10 (for discharging tasks).

Size of buffer	Number of IAVs																					
	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	Quay crane 1 net moves per hour																						
10	9.7	12.2	14.5	16.8	19.1	22.4	25.7	29.1	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0
9	9.6	12.1	14.2	16.1	18.7	21.1	24.1	26.5	29.9	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0
8	9.5	12.0	13.8	15.7	17.8	20.2	22.4	24.6	27.5	29.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0
7	9.3	11.8	13.3	15.0	16.9	18.8	20.8	22.4	25.2	27.0	28.5	29.8	29.9	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0
6	9.2	11.7	13.0	14.5	15.9	18.2	19.7	21.0	22.8	24.5	25.9	27.5	28.5	29.3	29.6	29.8	30.0	30.0	30.0	30.0	30.0	30.0	30.0
5	9.0	11.3	12.6	13.6	14.8	17.2	18.5	19.8	20.8	22.3	23.2	24.3	25.4	26.4	27.3	28.1	28.6	29.2	29.5	29.9	30.0	30.0	30.0
4	8.9	10.7	11.9	13.2	14.4	15.8	17.5	18.9	19.1	19.8	20.9	22.4	23.2	23.6	24.1	24.8	25.4	26.3	27.0	27.9	28.4	28.8	29.4
3	8.7	10.8	11.6	12.6	13.6	14.5	16.2	17.4	18.2	18.8	19.6	20.1	20.5	21.1	22.4	22.2	23.0	23.5	23.8	24.3	24.8	25.6	26.3
2	8.5	10.2	11.2	11.8	12.9	14.0	14.8	16.4	16.1	17.6	18.6	18.3	19.3	19.6	20.3	20.4	20.8	21.0	21.4	22.6	23.0	23.2	23.9
1	8.3	9.7	10.8	11.6	13.2	13.8	14.2	15.1	15.7	16.1	16.6	17.6	18.2	18.7	18.9	19.2	19.5	20.1	20.5	20.9	21.5	22.0	22.4
	Quay crane 2 net moves per hour																						
10	12.7	17.2	21.0	24.5	27.3	28.7	29.4	29.8	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9
9	13.4	16.8	20.7	24.3	27.0	28.4	28.8	29.4	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9
8	12.1	16.6	20.2	23.5	26.6	28.0	27.9	28.7	29.7	29.7	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9
7	12.6	16.5	20.0	23.1	25.7	27.3	27.3	27.3	28.8	29.5	29.8	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9
6	12.0	16.5	19.8	22.6	24.6	26.2	26.7	26.5	27.4	28.3	29.0	29.5	29.7	29.8	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9	29.9
5	12.8	16.0	19.4	22.2	23.8	25.0	25.5	26.0	26.0	26.6	27.3	28.3	28.7	29.1	29.3	29.6	29.7	29.8	29.9	29.9	29.9	29.9	29.9
4	11.4	16.2	19.0	21.5	22.6	23.8	24.6	25.1	25.9	26.0	26.2	26.7	27.2	27.6	28.0	28.3	28.7	28.9	29.3	29.5	29.7	29.7	29.9
3	12.0	16.1	18.3	20.6	21.5	22.6	23.3	23.8	24.5	25.1	25.2	25.5	26.2	26.5	26.3	27.0	27.3	27.6	28.0	28.5	28.6	28.8	29.2
2	11.5	15.9	17.9	19.5	20.4	21.3	21.9	22.1	23.5	23.7	23.6	25.1	24.7	24.8	25.0	25.7	26.1	26.6	27.0	26.9	27.0	27.5	27.8
1	11.6	15.9	17.4	18.6	19.2	19.9	20.7	21.1	21.8	22.7	23.1	23.3	23.4	23.9	24.6	25.2	25.6	25.8	26.7	26.5	26.7	26.6	26.9
	Quay crane 3 net moves per hour																						
10	10.2	14.8	18.3	21.0	23.8	26.1	27.9	29.4	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7
9	10.1	14.5	17.7	20.2	23.3	25.6	26.8	28.2	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7
8	10.0	14.6	17.2	19.8	22.5	24.8	25.5	27.1	28.6	29.3	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7
7	9.8	14.5	17.1	19.3	21.8	23.5	24.2	25.3	27.0	28.4	29.5	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7
6	9.8	14.6	16.6	18.9	20.5	22.6	23.7	24.1	25.3	26.4	27.5	28.7	29.3	29.6	29.6	29.7	29.7	29.7	29.7	29.7	29.7	29.7	29.7
5	9.5	13.8	16.2	17.6	19.5	21.7	22.4	23.0	23.6	24.6	25.4	26.4	27.2	27.7	28.6	28.9	29.1	29.4	29.6	29.7	29.7	29.7	29.7
4	9.4	13.5	15.3	17.1	18.6	20.1	21.1	21.7	22.8	23.3	23.6	24.6	25.3	25.7	26.0	26.8	27.2	27.9	28.4	28.7	29.0	29.3	29.5
3	9.1	14.0	15.0	16.2	17.5	18.8	19.2	20.1	21.3	22.3	22.5	23.2	23.5	23.8	24.4	24.6	25.0	25.4	26.1	26.3	26.9	27.6	27.9
2	9.0	13.2	14.6	15.3	16.4	17.4	18.2	18.7	19.2	20.2	20.7	21.6	21.8	22.1	22.6	23.0	23.1	23.5	24.3	24.9	25.2	25.6	26.1
1	8.8	12.8	14.2	15.1	16.1	16.3	16.9	17.8	18.9	18.9	19.2	20.2	20.8	20.9	21.4	21.9	22.2	22.7	23.5	23.8	24.1	24.8	24.9

5.3.6 IAVs versus trucks: A total cost comparison

This section compares the values of IAVs and trucks based on the total capital and operational cost in a 15-year period.

Identifying the optimal number of vehicles

To compare the total cost of the two types of vehicles, it is first needed to identify the smallest number of vehicles (e.g. IAVs and trucks) that can meet the target set

Table 5.3: This table shows the total discharging time by utilising the buffers of containers under cranes.

Size of buffer	Number of IAVs																						
	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	Total discharging time (hour)																						
10	10.30	8.17	6.88	5.94	5.22	4.47	3.89	3.44	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37
9	10.43	8.26	7.07	6.20	5.36	4.73	4.15	3.78	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37
8	10.55	8.34	7.25	6.37	5.60	4.96	4.47	4.07	3.63	3.45	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37
7	10.72	8.50	7.52	6.67	5.93	5.33	4.80	4.46	3.98	3.70	3.50	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37
6	10.93	8.52	7.70	6.88	6.28	5.50	5.08	4.76	4.38	4.09	3.86	3.64	3.51	3.41	3.38	3.37	3.37	3.37	3.37	3.37	3.37	3.37	3.37
5	11.13	8.89	7.94	7.36	6.77	5.81	5.40	5.06	4.81	4.48	4.32	4.11	3.93	3.78	3.67	3.56	3.49	3.42	3.39	3.37	3.37	3.37	3.37
4	11.28	9.35	8.43	7.60	6.93	6.32	5.71	5.30	5.24	5.04	4.77	4.46	4.30	4.24	4.15	4.03	3.93	3.80	3.70	3.58	3.53	3.47	3.41
3	11.44	9.24	8.59	7.94	7.35	6.89	6.16	5.75	5.50	5.32	5.10	4.97	4.87	4.75	4.46	4.51	4.34	4.26	4.19	4.12	4.03	3.91	3.80
2	11.76	9.82	8.95	8.46	7.77	7.13	6.74	6.08	6.20	5.67	5.37	5.48	5.19	5.11	4.93	4.90	4.81	4.76	4.68	4.42	4.35	4.30	4.18
1	12.01	10.34	9.23	8.60	7.58	7.23	7.03	6.62	6.35	6.22	6.03	5.69	5.49	5.34	5.29	5.21	5.13	4.97	4.88	4.78	4.64	4.54	4.46

out by the port. Current port A suggests a target of 25 moves per hour if using two quay cranes, which is equivalent to 17 moves per hour if using three quay cranes. To identify such an optimal fleet size, simulation is used to identify the minimum number of vehicles that can meet the required target moves per hour for the largest transaction available in the port, in which 300 containers are discharged. The reason to only consider the largest transaction is that for smaller transactions naturally fewer vehicles are required to meet the target. By comparing Figure 5.8 and Table 5.3, it can be seen that with 6 IAVs and a buffer size of 5 or with 10 trucks this target of 17 moves per hour for 3 quay cranes can be achieved. Therefore, 6 IAVs (with a buffer size of 5) and 10 trucks are considered to be the optimal numbers of vehicles. Note that to identify the optimal number of vehicles the loading cases are not considered, given that in this container terminal the fleet size needed for loading is always less than the fleet size for discharging (as explained in the last paragraph of Subsection 5.3.4 and also shown in Figure 5.8).

Cost model of port A

Identifying only the minimum fleet size for trucks and IAVs, however, does not answer the question of which type of vehicles is economically better and what would be the

total cost of those vehicles. To answer this question, in this subsection, a cost model (see details in the technical report in McGinley and Murray (2013)) is enhanced to compare the total cost that port A needs to spend for its vehicles in 15 years when being used with the optimal fleet sizes of 6 IAVs against 10 trucks.

The cost model calculates the total cost that port A has to spend on each type of vehicles, taking into account the vehicles' capital and operational cost for a 15-year period. The purpose of this cost model is to estimate the total present values of each system (e.g. IAVs and trucks). The present value is a metric to show the total cash flows of an investment over a given period, discounted to today's cash value (Bazargan et al., 2013). For this calculation a discount rate of 5% and a 15-year period are considered. Ten years is considered to be the lifetime of trucks and IAVs. The factors that were considered in this cost model are explained in this section.

The first factor in the cost model is the vehicles' capital. The IAVs' and trucks' capital can have a significant impact on the total cost of port A. Note that by the time of submission, IAVs have not been manufactured commercially, therefore, the final price of IAVs has not been determined. However, the price of an IAV is estimated to be €500,000 plus €8,000 for a cassette and €2,000 for charger installation cost. The truck's capital was considered €113,000 including €90,000 for a shunter and €23,000 for a trailer. It can be seen that an IAV is almost 5 time more expensive than a truck.

Trucks consume diesel and IAVs use electricity, therefore the price of energy for the two types of vehicles can be different. To calculate the energy cost per year, the vehicles working hours per year is needed. The same working hours for IAVs and trucks are considered, given that the two types of vehicles are supposed to provide the same performance in port A. The total fuel costs of IAVs and trucks for one year are calculated as below:

Let:

h: total working hours per vehicle per year

d: diesel litre consumed per truck per hour

p_d: price per diesel litre

p_c: price per charge per IAV

w: IAV working hours per charge

E_{IAV}: total energy cost per IAV per year

E_{truck}: total energy cost per truck per year

$$E_{IAV} = p_c * (h/w) \tag{5.3}$$

$$E_{truck} = p_d * h * d \tag{5.4}$$

The next cost that is explained is the cost of periodic services. To calculate the service cost n_s services per year for IAVs and trucks are considered. The cost per service is shown by s_{IAV} for IAVs and for trucks by s_{truck}. Note that by the time of submission the exact maintenance and repair costs of IAVs were not available. Existing literature indicates that electric vehicles (like AGVs, IAVs etc) usually cost less to maintain and repair than diesel vehicles (like trucks) (Funk and Rabl, 1999; Nam and Ha, 2001; Lin et al., 2013). Despite that, in this thesis the worst-case scenario is considered where the service cost of IAVs is the same as that of trucks. Using this information the total service cost of one year for an IAV, S_{IAV} and for a truck, S_{trucks} can be calculated as below:

$$S_{IAV} = s_{IAV} * n_s \tag{5.5}$$

$$S_{truck} = s_{truck} * n_s \tag{5.6}$$

Six IAVs need two operators and ten trucks need ten drivers. The cost of wages, insurance and annual leave of an operator for IAVs and a driver for trucks were calculated based on the following parameters:

Let:

h: total working hours per year per vehicle

w_{IAV}: wage cost per hour per IAV operator

w_{truck}: wage cost per hour per truck driver

v_{IAV}: provision for holiday pay per year per IAV operator

v_{truck}: provision for holiday pay per year per truck driver

i_{IAV}: employers insurance per year per IAV operator

i_{truck}: employers insurance per year per truck driver

a_{IAV}: annual leave hours per year per IAV operator

a_{truck}: annual leave hours per year per truck driver

W_{IAV}: total wage cost per year per IAV operator

W_{truck}: total wage cost per year per truck driver

$$W_{IAV} = (w_{IAV} * h) + v_{IAV} + i_{IAV} + (a_{IAV} * w_{IAV}) \tag{5.7}$$

$$W_{truck} = (w_{truck} * h) + v_{truck} + i_{truck} + (a_{truck} * w_{truck}) \tag{5.8}$$

By calculating the above intermediate parameters (E, S and W), the cash flows for the operational costs of IAVs and trucks can be calculated. Equations 5.9 and 5.10 show how the cash flows for operational costs in year 0 (O_0) can be calculated.

Let:

d_{truck}: number of drivers for trucks

d_{IAV}: number of operators for IAVs

n_{truck}: optimal number of trucks

n_{IAV}: optimal number of IAVs

$$O_0^{truck} = (E_{truck} + S_{truck}) * n_{truck} + (W_{truck}) * d_{truck} \tag{5.9}$$

$$O_0^{IAV} = (E_{IAV} + S_{IAV}) * n_{IAV} + (W_{IAV}) * d_{IAV} \tag{5.10}$$

The cash flows for operational cost of the next 15 years are calculated using the cash flow for year 0 and the inflation rate i. This is shown by Equation 5.11.

$$O_t = O_0 * (i+1)^t,\ 1 \leq t \leq 15 \tag{5.11}$$

Equation 5.12 estimates the vehicle capital for the next 15 years in a similar way to that of the operational cost. Note that since the lifetime of the vehicles was considered 10 years, the capital costs were taken into account only in year 0 and 10 (Table 5.6).

$$C_t = \begin{cases} C_0 * (i+1)^t,\ if\ t = 10 \\ 0,\ otherwise \end{cases} \tag{5.12}$$

Equation 5.13 calculates R_t, the total cash flow of year t. To do so, it takes the summation of the operational cash flow (Q_t) and vehicle capital cost (C_t).

$$R_t = O_t + C_t,\ 0 \leq t \leq 15 \tag{5.13}$$

By calculation of R_t, $0 \leq t \leq 15$, the present value of the cash flow of each year can be calculated using Equation 5.14 where r is the risk adjusted discount rate.

$$P_t = R_t/(1+r)^t \tag{5.14}$$

Finally, Equation 5.15 calculates the total present value of the vehicle (TPV) by taking the summation of the present values of the cash flow of each year.

$$TPV = \sum_{t=0}^{15} P_t \tag{5.15}$$

Tables 5.4 and 5.5 show the values of the initial and intermediate parameters used in the cost model. The intermediate parameters were calculated by Equations 5.3-5.8. Table 5.6 shows the cash flows for the 15-year period that were calculated using Equations 5.9-5.13.

Table 5.4: This table shows the parameters that were used in the cost model and their values, as provided by the port.

Parameter description	Symbol	Unit	Value
Total working hours per year	h	h/year	3,000
Diesel litre consumed per hour per truck	d	l/h	8
Price per diesel litre	p_d	€/l	0.9
Price per charge per IAV	p_c	€/c	3.89
IAV working hours per charge	w	h/c	4
Wage cost per hour per IAV operator	w_{IAV}	€/h	19
Wage cost per hour per truck driver	w_{truck}	€/h	19
Provision for holiday pay per year per IAV operator	v_{IAV}	€/year	6,080
Provision for holiday pay per year per truck driver	v_{truck}	€/year	6,080
Employers insurance per year per IAV operator	i_{IAV}	€/year	6,779
Employers insurance per year per truck driver	i_{truck}	€/year	6,779
Annual leave hours per year per IAV operator	a_{IAV}	h/year	320
Annual leave hours per year per truck driver	a_{truck}	h/year	320
Number of services per year per vehicle	n_s	1/year	10
Cost of a service per IAV	s_{IAV}	€	800
Cost of a service per truck	s_{truck}	€	800
Number of operators for the IAV system	d_{IAV}	person	2
Number of drivers for the truck system	d_{truck}	person	10
Optimal number of IAVs	n_{IAV}	vehicle	6
Optimal number of trucks	n_{truck}	vehicle	10
Risk adjusted discount rate	r	-	0.05
Inflation rate	i	-	0.02
IAV capital (IAV + cassette + charger)	C_0^{IAV}	€	510,000
Truck capital (shunter + trailer)	C_0^{truck}	€	113,000

Table 5.5: This table shows the intermediate parameters that were calculated using the parameters in Table 5.4 and Equations 5.9-5.8 for year 0.

Parameter description (for year 0)	Symbol	Unit	Value
Total energy cost per IAV	E_{IAV}	€	2,916
Total energy cost per truck	E_{truck}	€	21,600
Total wage cost per IAV operator	W_{IAV}	€	75,939
Total wage cost per truck driver	W_{truck}	€	75,939
Total service cost per IAV	S_{IAV}	€	8,000
Total service cost per truck	S_{truck}	€	8,000

Table 5.6: This table shows the cash flows for IAVs and trucks for the 15-year period. The unit for O_t, C_t and R_t is Euro (€). These cash flows were calculated using Equations 5.9-5.13. Note that since the lifetime of trucks and IAVs is 10 years, at year 0 and year 10 a new fleet should be purchased and thus C_t in all years apart from years 0 and 10 have the value of 0.

Year	Trucks			IAVs		
	Q_t*	C_t	R_t	Q_t	C_t	R_t
0	2,072,390	113,000	2,185,390	2,767,374	510,000	3,277,374
1	1,076,498	0	1,076,498	221,721	0	221,721
2	1,098,028	0	1,098,028	226,156	0	226,156
3	1,119,988	0	1,119,988	230,679	0	230,679
4	1,142,388	0	1,142,388	235,293	0	235,293
5	1,165,236	0	1,165,236	239,998	0	239,998
6	1,188,541	0	1,188,541	244,798	0	244,798
7	1,212,311	0	1,212,311	249,694	0	249,694
8	1,236,558	0	1,236,558	254,688	0	254,688
9	1,261,289	0	1,261,289	259,782	0	259,782
10	2,526,232	137,746	2,663,978	3,373,413	621,687	3,995,101
11	1,312,245	0	1,312,245	270,277	0	270,277
12	1,338,490	0	1,338,490	275,683	0	275,683
13	1,365,260	0	1,365,260	281,196	0	281,196
14	1,392,565	0	1,392,565	286,820	0	286,820
15	1,420,416	0	1,420,416	292,557	0	292,557

*Q_t: operational cost at year t
C_t: vehicles capital at year t
R_t: total cash flow (i.e. $Q_t + C_t$) at year t

Figure 5.9-a compares the present value of the cash flow in each year for IAVs and trucks. At year 0 the present value of IAVs is €3,277,374 and that of trucks is

€2,185,390. The present value in year 0 is the present value of cash flow, which is the summation of operational cost and vehicles capital, because in year 0 new fleet should be purchased. In year 1, the present cash flow value for IAVs is €211,161 which is significantly lower than €1,025,236 of cash flow for trucks. In the next following years apart from year 10, similar trend as for year 1 can be observed. This shows that the operational cost of IAVs is much lower than trucks. This is mainly because of the higher price of energy for trucks compared with that of IAVs (Table 5.5) and also the optimal number of trucks is higher than IAVs (Table 5.4). The reason to have a significant increase in the present cash flow values of trucks and IAVs in year 10 is that new vehicles should be replaced with the current fleet (the lifetime of vehicles was considered 10 years). Next, the total present values for IAVs are compared with those for trucks. As in Figure 5.9-b, the total present cash flow values for IAVs is €8,032,693 and for trucks is €15,000,740. As one can see, the total present value for the IAV system is significantly lower than that for trucks despite the fact that IAVs are much more expensive than trucks. Thanks to the IAV's unique feature of utilising the buffers of containers, fewer IAVs are needed compared with trucks. Being electric, IAVs also lead to lower energy costs than trucks.

5.4 Conclusion

This chapter has made the following contributions.

1. A simulation model was developed to evaluate the performance and cost of the case study with IAVs compared with the existing vehicles system (i.e. trucks). The FlexSim CT simulation software was used to develop the simulation model. This software, however, does not have the IAV object in its simulation library. Thus, to simulate IAVs and cassettes the "straddle carrier gang" object was com-

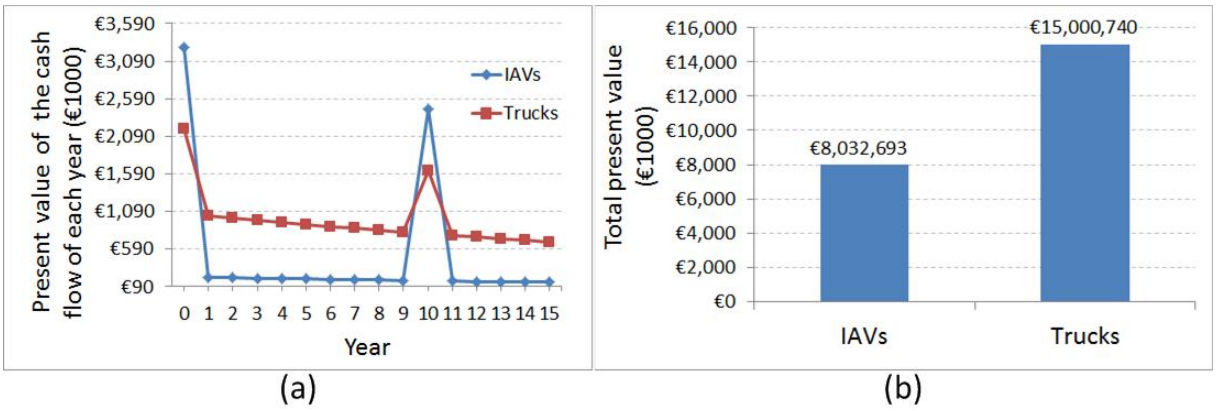

Figure 5.9: The comparison between present values: IAVs vs trucks. Plot (a) compares the present cash flow values of cost of trucks and IAVs in each year. Plot (b) compares the total present value of trucks against that of IAVs over 15 years. As can be seen the total present value for IAVs is much lower than that of trucks.

bined with "transfer area" object to be able to simulate IAVs features in the simulation

2. The simulation results reveal two findings:

   (a) When not using the cassettes, IAVs are still shown to have the same efficacy as regular trucks if five or more vehicles are used, even though the IAVs were chosen to operate at a much slower speed than the trucks. Due to their ability to move in all directions without having to turn, IAVs can save on travel time compared with trucks, leading to better efficiency. Of course, the efficacy could be improved considerably if IAVs are allowed to travel at a higher speed.

   (b) Combining IAVs with cassettes significantly improves port performance in terms of the number of crane moves per hour and total loading/discharging time.

3. A cost model was enhanced to estimate the cost of the case study with IAVs and trucks. By comparing the total present values of the two vehicle systems, it

can be concluded that the total present value of IAVs is much lower than that of trucks even though the IAVs' capital is much higher than trucks' capital.

4. This is the first research that uses simulation to study the impact of using IAVs in container terminals. With the potential improvements shown to be significant, this study is expected to have practical impacts and the research results are being considered by the studied port.

Chapter 6

A Discrete-event Simulation Framework for Container Terminals

6.1 Introduction

Container terminals play a vital role in global trade (carrying 90% of non-bulk world trade goods as of 2009 (Ebeling, 2009)). In the last decade, there has been a significant increase in world container trades (with a growth rate of 201% during the 2003-2012 period (UNCTAD, 2013)). To deal with this growth of containerisation, container terminals have been seeking to adopt new technologies and equipment to enhance their productivity (Henesey, Aslam and Khurum, 2006). Given the size of investments involved, it is essential for container terminals to identify the most suitable technologies/equipment that provide the best performance. A container terminal, however, is a very complex system and developing an analytical model to evaluate the performance of container terminals has always been a very challenging task (Demirci, 2003). This is because, a container terminal encompasses different pieces of equipment that interact with and depend on each other. For example, the performance of a quay

crane depends not only on its designed capacity, but also on the skills of drivers, the size and dimension of vessels, the weather conditions and the availability and performance of transfer vehicles. In addition, the operations in container terminals have a stochastic nature and hence the performance of each piece of equipment is subject to different uncertainties. For instance, the travel time of vehicles between quay-side and stack-side areas can be changed due to possible deadlocks, breakdowns or collisions. So far, such complexity can only be modelled in computer simulations (Angeloudis and Bell, 2011). The simulation approach for container terminals is the focus of this chapter.

This chapter proposes a new simulation framework using the FlexSim CT software for the simulation of container terminals. FlexSim CT is discrete-event simulation software specialised in simulating container terminals, specifically the yard and the quay sides of ports. It is based on FlexSim general (Nordgren, 2003), a more generic simulation platform. FlexSim CT has specific container terminal simulation objects such as berth planner, yard planner, stack blocks and cranes which make the development of simulation models very straightforward. In addition, FlexSim CT benefits from a very good 3D simulation feature and also the access to the FlexSim simulation library. Despite these advantages, FlexSim CT has some limitations that might prevent users from fully simulating certain realistic scenarios: while its simulation for the yard side is fairly flexible, FlexSim CT's simulation for the quay side is limited in that it offers almost no access to the source code to change/configure the quay side's behaviours. For example, modifying the predefined berth, quay crane assignment or the task sequence of quay cranes is not possible in FlexSim CT. Furthermore, it is not possible to define any routing and/or scheduling algorithm to control the behaviour of equipment such as transfer vehicles and quay cranes. It is also not feasible to fully define and control any new type of equipment that does not exist in the FlexSim CT

library, for example automated guided vehicles, shuttle carriers or freight rail. To address these limitations, based on FlexSim CT, a new simulation framework is developed. This new framework was created by developing an entire new quay side area and then linking it with FlexSim CT's yard simulation, skipping FlexSim CT's own simulation of the quay. Some features on FlexSim CT's yard simulation are improved, making it more flexible and supporting user-defined algorithms. By doing so, users will have full control of all simulation details. This helps users to create customised equipment/vehicles/scenarios in container terminals that are not currently supported by FlexSim CT. It also allows users to develop a wide range of optimisation algorithms and what-if analyses for their container terminal.

The remainder of this chapter is structured as follows. Section 6.2 explains the operations in container terminals that are included in the framework. Section 6.3 explains the developed simulation framework. Results of the experimental study are provided in Section 6.4. Finally, Section 6.5 concludes this chapter.

6.2 Operations in container terminals

This section explains the processes of discharging and loading of ships in container terminals to provide a better understanding of operations in container terminals. These operations can be simulated using the proposed simulation framework. The relations between the processes are explained using an activity diagram (Figure 6.1). In this diagram, ships, quay cranes, vehicles and stack cranes are considered the main objects that are involved in the discharging and loading processes. Each of these objects is represented by a swimlane to differentiate the processes in which each object is involved.

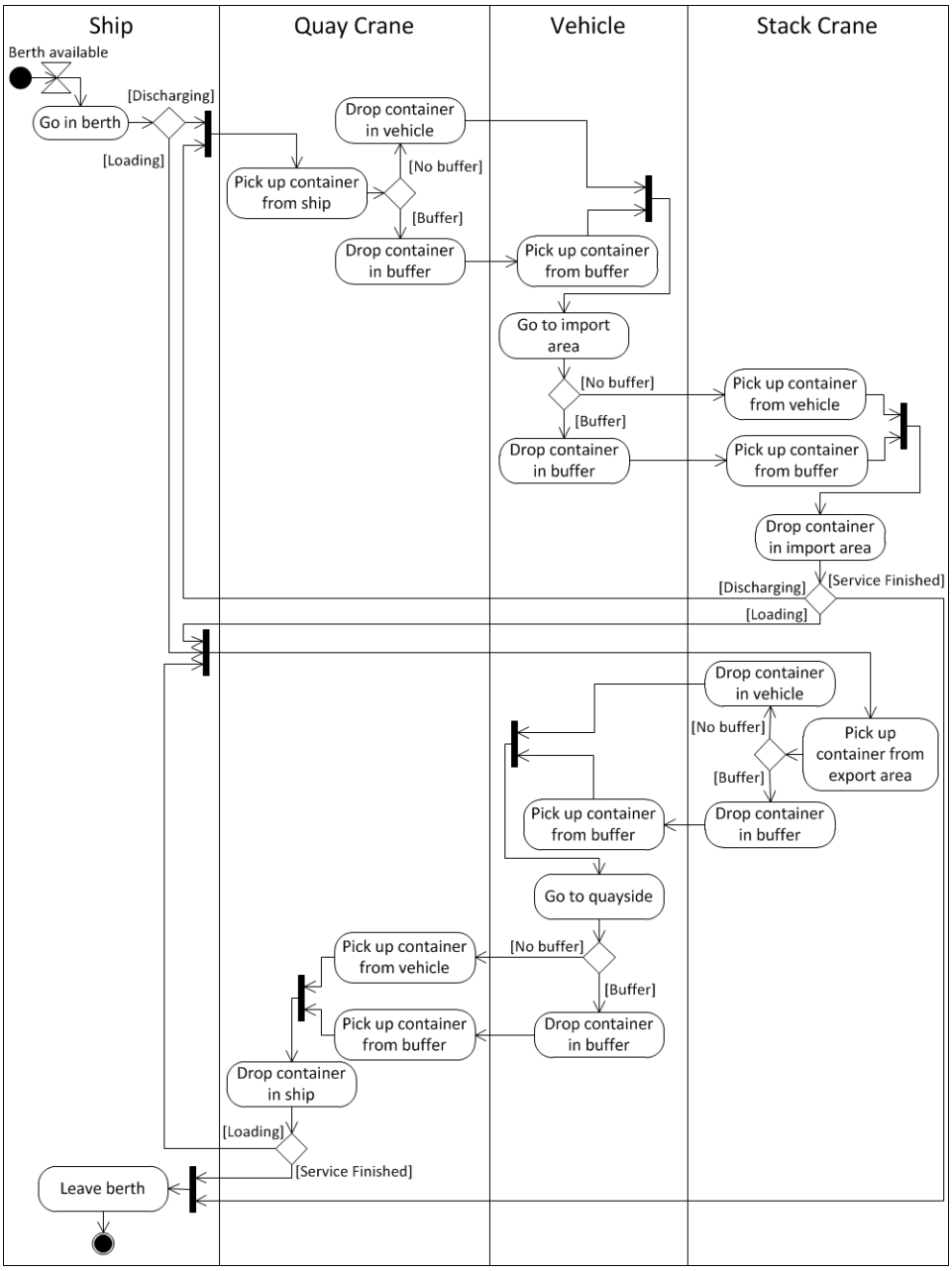

Figure 6.1: The activity diagram to show the operations in container terminals. This figure shows the relations between the processes in container terminals using an activity diagram. Ships, quay cranes, transfer vehicles and stack cranes are the main objects that are involved in container terminal operations.

Container terminals generally consist of quay-side and stack-side areas. The quay-side area is a place where ships are berthed. In this area, quay cranes are used to

discharge/load containers from/into ships. The stack-side area consists of a number of stack blocks to stack import and export containers using stack cranes.

This diagram starts by the arrival of a ship at a container terminal and ends when the ship leaves the container terminal. Once a ship arrives at a container terminal, a berth should be assigned to it. If no berth is available, the ship should wait until a berth becomes available. A berth is a place where ships are moored. Each container terminal has a number of berths with a specific length to allocate to ships. Each berth is equipped with a number of quay cranes which can be shared between the adjacent berths. When a quay crane is assigned to a berth it moves alongside of the quay to reach the designated berth. To allocate a berth to a ship, the length of the berth should be taken into account to ensure that the ship can fit in the berth. By allocating a berth to a ship, a number of quay cranes need to be assigned to the berth. Each ship upon arrival may request a specific number of quay cranes to discharge and load containers. As a result, there may be a number of combinations of berths and quay cranes that can be assigned to a ship. Each may lead to different waiting times of ships, because the available times of berths and quay cranes can be different. Thus, the berth and quay cranes assignment are two important decision making problems that should be addressed by the time ships arrive at container terminals. In the literature, there is a large number of algorithms to address these two decision problems (Steenken et al., 2004; Stahlbock and Voss, 2008).

As mentioned earlier, each ship has a number of containers to be discharged and loaded. For each ship, usually the discharging process is performed first and the loading process is performed afterwards. However, in busy cases the two processes may be performed in parallel.

Quay cranes start the discharging tasks by removing containers from a ship. Once a quay crane picks up a container from a ship, the container should be collected by a

vehicle (based on the vehicles schedule) and transported to the stack-side area (based on the containers placement strategy). Collection of containers from quay cranes depends on whether vehicles can pick up and drop off containers by themselves. If this is the case, quay cranes do not need to wait for vehicles, they can place containers in buffer, a temporary storage space under the crane, for vehicles to collect later. Utilising the buffers has the great advantage of decreasing the waiting time of both quay cranes and vehicles. In contrast, if vehicles cannot pick up/drop off containers by themselves, quay cranes have to wait for vehicles to come and then they have to place containers directly on vehicles.

After a vehicle collects a container from either a buffer or directly from a quay crane, it transports the container to the stack-side area to deliver it to the designated import block. The stack crane assigned to that particular stack block then picks the container up from the vehicle and stacks it in the import block. The vehicle then transports to the quay-side area for the next transportation tasks. Note that similar to the quay-side area, buffers can also be used in the stack-side area under stack cranes. This way, vehicles can drop off containers in the buffers for stack cranes to collect later, and hence minimise the waiting time. The discharging process will be continued until all importing containers are removed from the ship.

After finishing the discharging process, the loading process starts. In the loading process, containers are moved from exporting blocks into ships. The loading process is similar to the discharging process but in the opposite direction. Stack cranes pick up containers from the export blocks and place them in the buffers for vehicles or directly on vehicles depends on whether vehicles can utilise the buffers. The utilising of buffers for loading tasks is similar to the one for discharging process. The loaded vehicles then transport to the quay-side area and drop the containers in the buffers for quay cranes to collect later or wait for quay cranes to pick up containers directly from them (in the

case without buffers). Quay cranes pick up containers from the buffers/vehicles and place them in the ship based on the ship stowage plan. Once all the loading containers are placed in the ship, the ship can leave the berth and container terminal.

6.3 Simulation framework

This section explains the developed simulation framework for container terminals, named FlexFrame. It first explains the FlexSim CT simulation software upon which FlexFrame was built. It then explains the components of FlexFrame and how FlexFrame can be used for developing simulation models of container terminals.

6.3.1 FlexSim basics

FlexSim (Nordgren, 2003) is general purpose discrete-event simulation software which has been widely used for manufacturing and warehousing simulations. The benefits of FlexSim are that in addition to the standard discrete-event simulation features, it supports good 3D visualisations, as well as the flexibility to rewrite some parts of the source code. FlexSim has an extension for simulation of container terminals, named FlexSim CT[1]. This software has special container terminal simulation objects for creating container terminal simulation models. In addition to those objects, FlexSim CT has access to FlexSim library objects. This opens the possibility of integrating a container terminal simulation model with a general simulation model. Despite these remarkable features, FlexSim CT has some limitations that prevent users from simulating certain realistic scenarios. As mentioned previously, between the two main simulation components in FlexSim CT: yard and quay simulations, these limitations mainly arise from the quay simulation. Due to the complexity in the development of

[1]Recently Moffatt & Nichol has acquired FlexSim CT.

the berth planner and quay side operations, FlexSim CT hides most of its source code for the quay simulation and does not allow users to rewrite or change some of the default settings. For example, modifying/changing berth and quay crane assignment and also more importantly any change in quay cranes task sequences are not possible.

To address the above-mentioned limitations, FlexFrame was developed with the help of some of the simulation objects in FlexSim CT. Notably the stack blocks, stack cranes, transfer vehicles and network nodes objects of FlexSim CT were used to develop FlexFrame. For this framework, a user library was developed for users to create simulation models using the drag-and-drop feature. This library gives users the flexibility to incorporate their optimisation algorithms (e.g. berth assignment, quay crane assignment and vehicles schedules etc) to simulation models. This flexibility, however, is not possible in FlexSim CT. In addition, advance users, if necessary, can rewrite the source code of FlexFrame.

FlexSim provides two types of functions for the development purposes, namely *user commands* and *user events*, which were used to develop FlexFrame. A user command is similar to an ordinary function or method in programming languages (e.g. C++, Java and Python). A user event is also similar to a function but with the following additional features: 1) it is triggered automatically at a specific t_0 time; 2) it cannot be called by any user command and user event; and 3) after the first time of being triggered it will be triggered repeatedly at each Δt time unit.

User events are the basis of FlexFrame to perform simulation. FlexFrame uses user events to: 1) keep track of events and changes in the simulation, and 2) control objects' behaviours according to the new events. Figure 6.2 depicts Ship Management, one of the FlexFrame user events as an example to show how a user event works. The figure depicts that at each Δt time unit the user event is triggered. In this case, the user event tests the availability of a berth for a ship. Following this test, the ship either

waits or goes in the berth. If the berth is available and the ship goes in the berth, a sub-process for the berth and quay crane assignment is called.

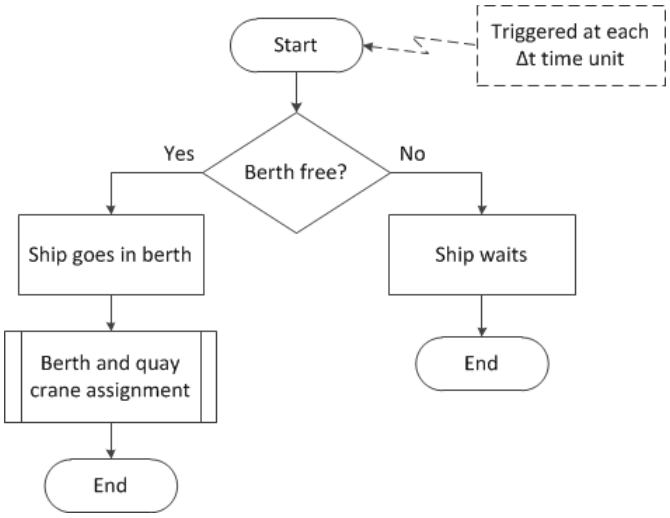

Figure 6.2: An example of a user event in FlexSim. As can be seen, a user event is similar to an ordinary function in programming languages but it can be triggered repeatedly at each Δt time unit.

6.3.2 FlexFrame structure

In this subsection, the structure of FlexFrame from the level of complexity and accessibility is explained. Recall from the previous subsection, FlexFrame was developed on top of FlexSim CT. Some of the objects in the FlexSim CT simulation library, specifically stack blocks, stack cranes, transfer vehicles and network nodes, are used to develop the new library with better capabilities. This library consists of a number of user events, user commands and 3D simulation objects.

FlexFrame basically has four levels in terms of complexity and accessibility. These levels are shown in Figure 6.3. In FlexFrame, the top level is Interface where users interact with FlexFrame. This level encompasses top level library objects for setting and configuring the input parameters and viewing the output results (i.e. statistics). Examples of inputs are setting schedules and properties of ships, configuring the po-

sition of port equipment and setting their properties as well as rewriting optimisation algorithms such as quay crane assignment, berth assignment and vehicles schedulers. The outputs consist of a real-time 3D simulation and a collection of the simulation statistics (e.g. cranes net/gross moves, ship staying time and berth utilisation rate etc). This level is explained in Subsection 6.3.4.

The next level is FlexFrame Components which includes the developed user events and user commands (functions). To develop simulation models users do not need to rewrite/modify the source code of this level, but they have access to it for any desired change. This level is explained in more detail in Subsection 6.3.3. The next level is FlexSim CT. The FlexFrame Components uses basic FlexSim CT objects and functions to perform the simulation. Note that this does not impose any limitation on the flexibility of FlexFrame, given that the objects and functions of FlexSim CT that users have full control on were only chosen. The lowest level is FlexSim which FlexSim CT was built upon. Thus, users in addition to FlexFrame objects have access to all simulation objects in FlexSim and can use them in their models.

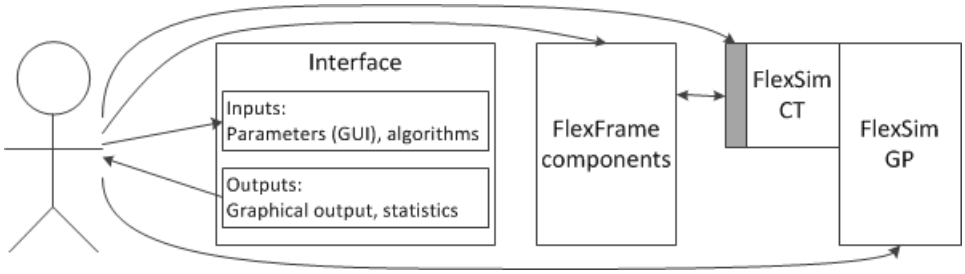

Figure 6.3: Relations between different components of FlexFrame and accessibility of users to these components. This figure depicts the different levels of FlexFrame. The top level that users have access is Interface for setting the simulation input parameters and viewing the simulation output. The middle level is FlexFrame Components which is the developed library of FlexFrame. The lowest levels are FlexSim CT and FlexSim which FlexFrame was built upon.

6.3.3 FlexFrame simulation flow

Recall from Subsection 6.3.1, FlexFrame is mainly controlled and operated by user events. This subsection explains the FlexFrame user events and the relations between them. FlexFrame has five main user events: Ship Management, Discharging Process, Loading Process, Queue Handling and Vehicles Dispatching (Figure 6.4).

In FlexFrame, the first user event that is triggered is Ship Management. This user event is responsible for the handling of arrival ships. It allocates berths to the arrival ships and depending on the availability of berths it either sends a ship to the designated berth or puts the ship in a queue until the berth becomes available. Once a ship is sent to the berth it assigns a number of quay cranes to the ship to perform discharging and loading tasks. At the end when a ship is served, this user event requests the ship to leave the berth.

After a ship gets berthed the discharging and loading processes should start. The Discharging Process user event handles discharging of containers from ships and the Loading Process user event handles loading of containers to ships. Discharging Process sends request to quay cranes to pick up containers from ships and synchronises vehicles with quay cranes to collect the containers from quay cranes. Once containers are discharged from ships, they will be moved by vehicles to stack blocks to be stacked. This step is carried out by Queue Handling and Vehicles Dispatching.

Vehicles Dispatching sends a vehicle to pick up an available container (either from a buffer or directly from a quay crane) and to transport and deliver it to its destination (either to a buffer or directly to a stack crane). The Queue Handling user event is responsible for managing the queue of vehicles. Depending on whether buffers are used this responsibility is different. In the case with buffers, when a vehicle wants to drop off a container in a buffer, it checks whether any slot is available in the buffer. If there is no available buffer (i.e. all the slots are full) it keeps vehicles in a queue until a

slot in the buffer becomes available. It then lets a vehicle based on the queue strategy (e.g. First-in First-out; Last-in First-out; or any other queuing strategy) drop off the container in the buffer. In the case without buffers, each time a vehicle arrives at a crane, it stops the vehicle at the queue until the crane becomes available. Otherwise, if the crane is available, the vehicle goes to the crane to collect or drop off containers.

The functionality of Loading Process is similar to that of Discharging Process, but the containers are moved from the stack-side area to the quay-side area to load the ships (Figure 6.1). Based on the loading plan, containers should be moved from the stack blocks to ships. The Loading Process user event chooses containers and sends request to stack cranes to unstack the designated containers. The Queue Handling and Vehicles Dispatching in collaboration with Loading Process moves containers to the quay cranes that are assigned to the ships. The quay cranes then load containers onto the ships. Ships stay in the berth as long as there are containers to be loaded to them. Once a ship service is completed, Ship Management requests the ship to leave the berth and make the berth available.

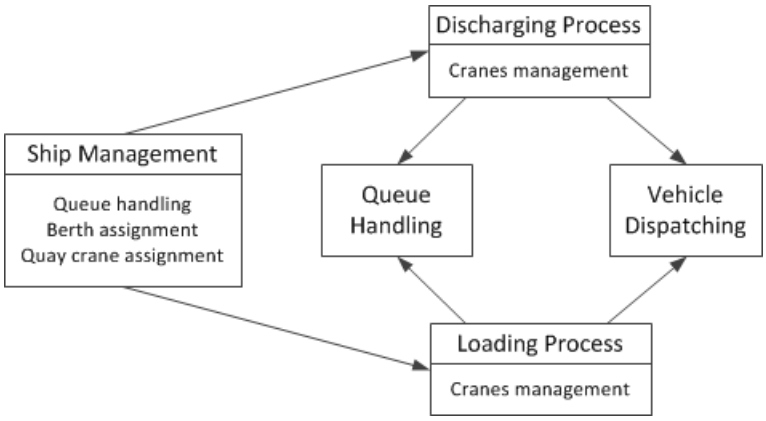

Figure 6.4: The interaction between different user events of FlexFrame.

6.3.4 Applying FlexFrame

This section explains how users can use FlexFrame to develop simulation models. The FlexFrame library includes the newly developed 3D objects to create simulation models, the developed user events and user commands to perform the simulation. Users can use the 3D objects in the library to create the 3D models. The objects that are needed to develop simulation models are the top three objects in Figure 6.5 (inside the red oval). Those objects are quay cranes, import and export blocks. For each import/export block one stack crane is associated as the default crane. However, the assignment of stack cranes to the stack blocks can be modified by users.

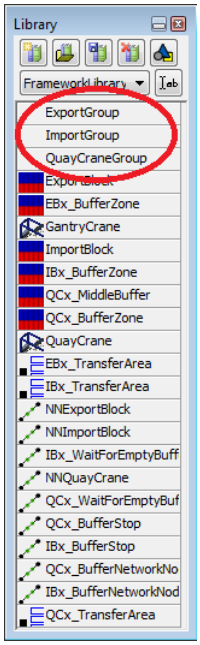

Figure 6.5: The developed user library of FlexFrame

Creating the network (travel routes) of vehicles is the next step to do. The network is the path that vehicles can follow to access simulation objects (e.g. quay cranes, stack cranes and stack blocks) and transport containers. The simulation objects need to be connected to the network to be accessible by vehicles. Once the layout of the 3D model is set, users should configure the simulation parameters. The parameters are stored in

a database in FlexFrame. If users wish to apply their required optimisation algorithms to the simulation, they can develop their algorithms in the library. The functions that users can modify/overwrite in FlexFrame are shown using sequence diagrams of the unified modelling language (Figures 6.6, 6.7 and 6.8).

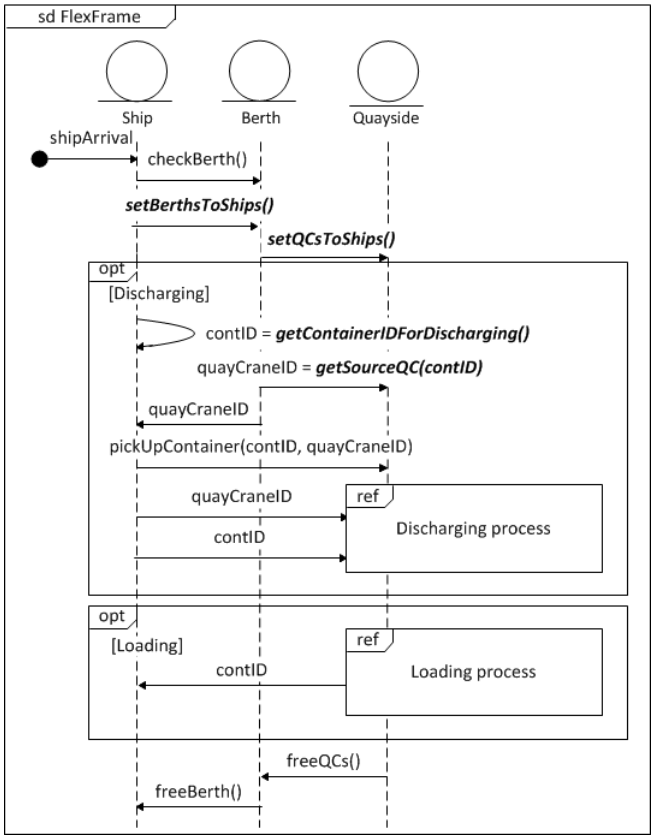

Figure 6.6: The sequence diagram to show main functionality of FlexFrame. Ship, Berth and Quayside were considered the main entities of FlexFrame. The functions that are shown by the boldface font can be overwritten by users to incorporate their optimisation algorithms to their simulation models.

These diagrams show the relations between the entities of FlexFrame and also specify the functions that users can rewrite to incorporate their optimisation algorithms to the framework. These functions are shown by the boldface font in these diagrams. Figure 6.6 shows the main sequence diagram that represents FlexFrame. In this diagram, the relations between three main entities (e.g. Ship, Berth and Quayside) are shown.

Figures 6.7 and 6.8 show discharging and loading processes as two sub-diagrams of the main diagram in Figure 6.6.

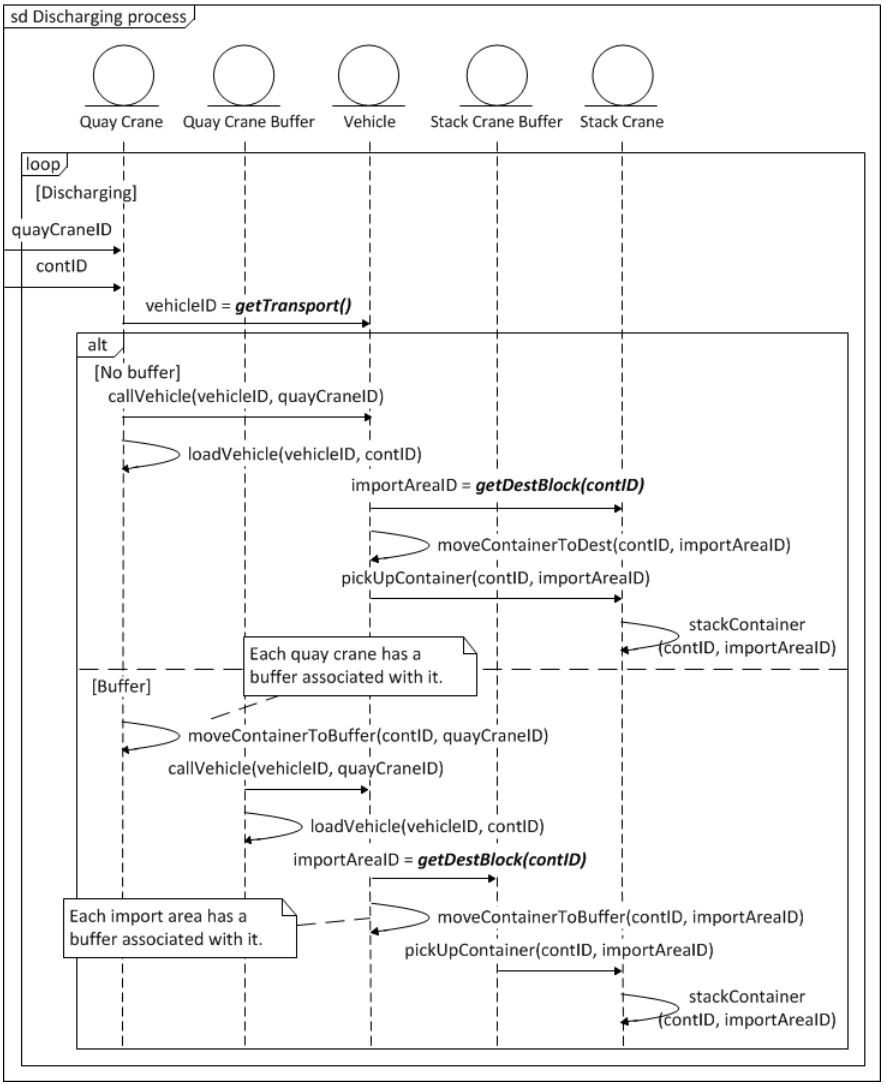

Figure 6.7: The sequence diagram to show the discharging process in FlexFrame. For the discharging process, Quay Crane, Quay Crane Buffer, Vehicle, Stack Crane Buffer and Stack Crane were considered to be the main entities. This diagram elaborates the sequence of calls of functions of each entity by other entities in the discharging process. The functions that are shown by the boldface font can be overwritten by users to incorporate their optimisation algorithms to their simulation models.

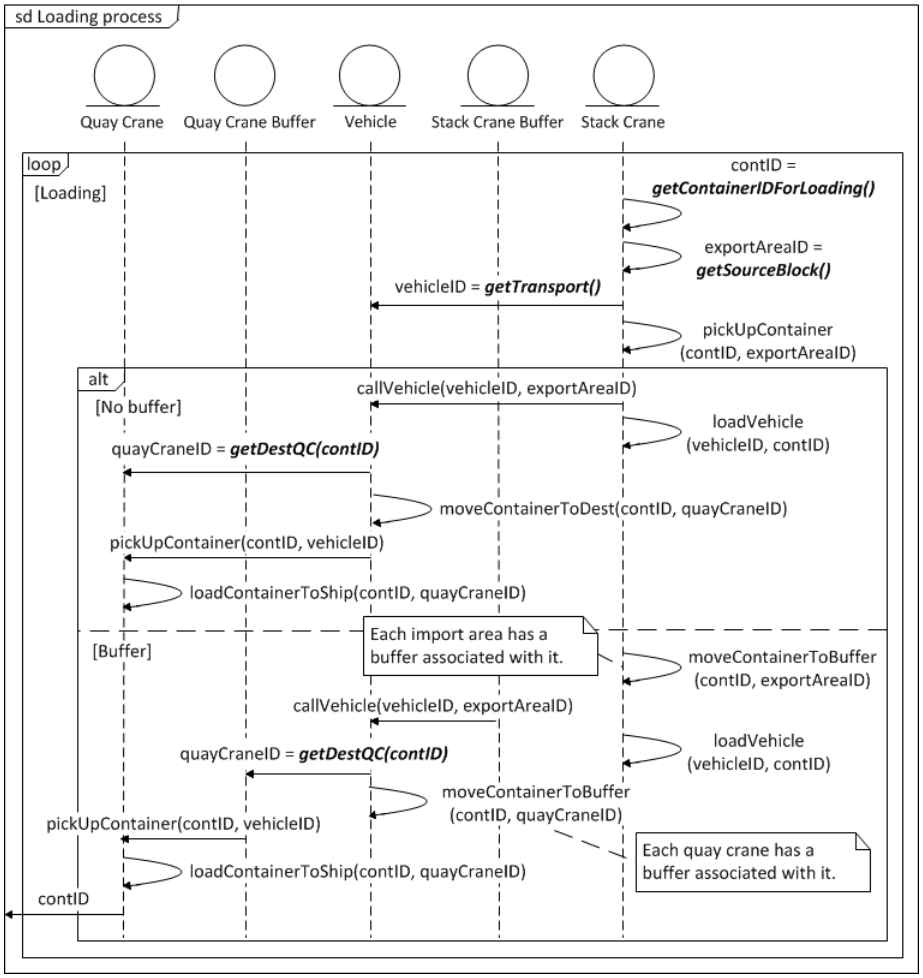

Figure 6.8: The sequence diagram to show the loading process in FlexFrame. For the loading process, Quay Crane, Quay Crane Buffer, Vehicle, Stack Crane Buffer and Stack Crane were considered to be the main entities. This diagram elaborates the sequence of calls of functions of each entity by other entities in the loading process. The functions that are shown by the boldface font can be overwritten by users to incorporate their optimisation algorithms to their simulation models.

After these configurations, users can run the simulation. The simulation runs using FlexFrame functions and it displays the 3D model at a chosen speed. While the simulation is running, FlexFrame and FlexSim collect data for the statistics, therefore users can observe them online while the simulation is running. There is a range of different statistics that users can get from FlexFrame such as crane net/gross moves per hour, ship turnaround time, ship waiting time, berth occupancy rate and vehicle

waiting time. Figure 6.9 shows the sequence of steps to develop simulation models using FlexFrame.

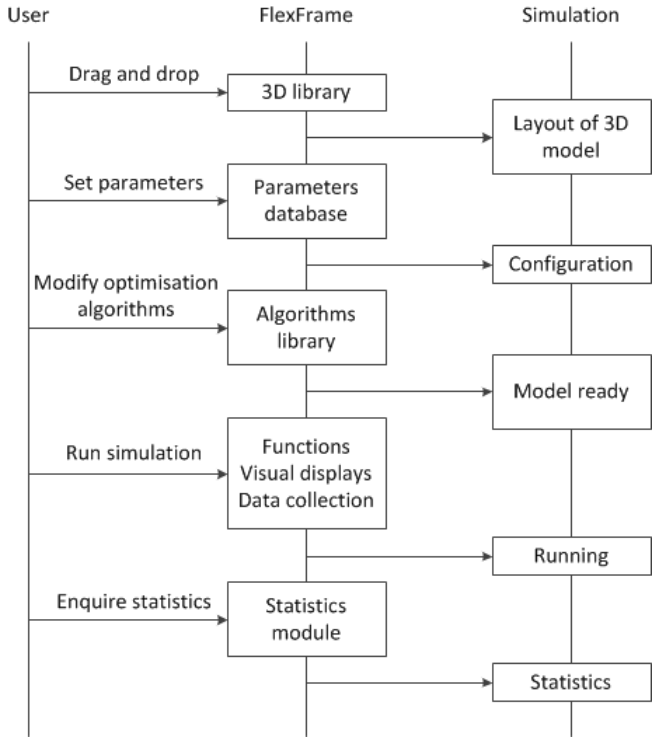

Figure 6.9: How users can create simulation models and get statistics from it.

6.4 Experimental results

This section provides the results of the simulation experiments. In the experiments, FlexFrame is used to develop a simulation model to compare the performance of a European container terminal with IAVs against trucks.

6.4.1 Case study layout

For the experimental study, similar to Chapter 5, port A was considered the case study. Figure 6.10 shows the layout of this container terminal. As can be seen, this container

terminal has three stack blocks to stack export containers and six stack blocks to stack import containers. This container terminal has three quay cranes at the quay side to discharge/load ships. As shown in this figure, the case study has two berths. Berth 1 can have at most three quay cranes whereas berth 2 has only one quay crane. In the cases where ships in berth 1 need three quay cranes, the quay crane in berth 2 should travel alongside the quay to berth 1 and joins the two quay cranes in berth 1.

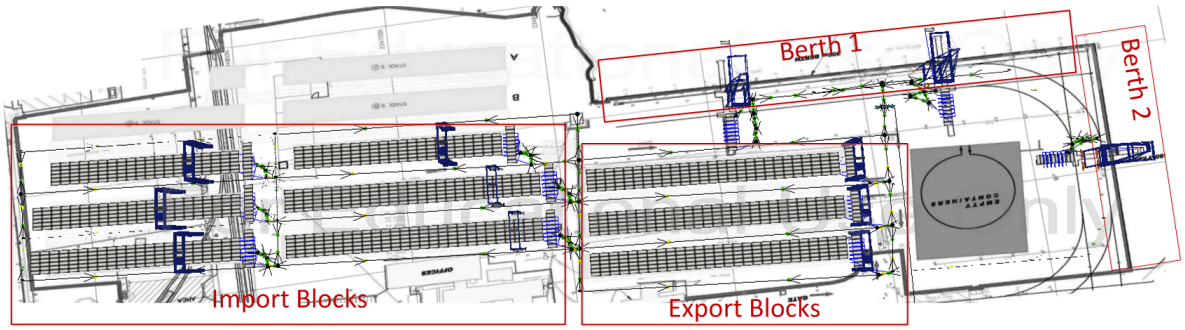

Figure 6.10: The layout of port A. This container terminal has two berths. The quay crane in berth 2 is shared between berths 1 and 2 and hence in the cases where ships in berth 1 need three quay cranes, this quay crane moves to berth 1.

6.4.2 Vehicles

IAVs and trucks were considered to be the transfer vehicles in this container terminal to transport containers. Figure 6.11 shows the travel routes for IAVs and trucks. As in this figure, the travel routes of IAVs are much shorter than those of trucks. Given that IAVs can move in any direction without turning, when an IAV delivers a container to a stack crane, it does not need to travel to the end of that stack block to turn, unlike trucks, it can reverse from that point and travel back to the quay-side area. However, the speeds of IAVs are much slower than those of trucks (Table 6.1) and hence it is not clear that the travel time of IAVs can be shorter than that of trucks, given their shorter travel routes.

Table 6.1: This table shows the speeds of vehicles in the simulation experiments. As can be seen, the speeds of IAVs are much slower than those of trucks.

Speed (m/s)	Truck	IAV
empty speed	13.41	4
loaded speed	11.18	2

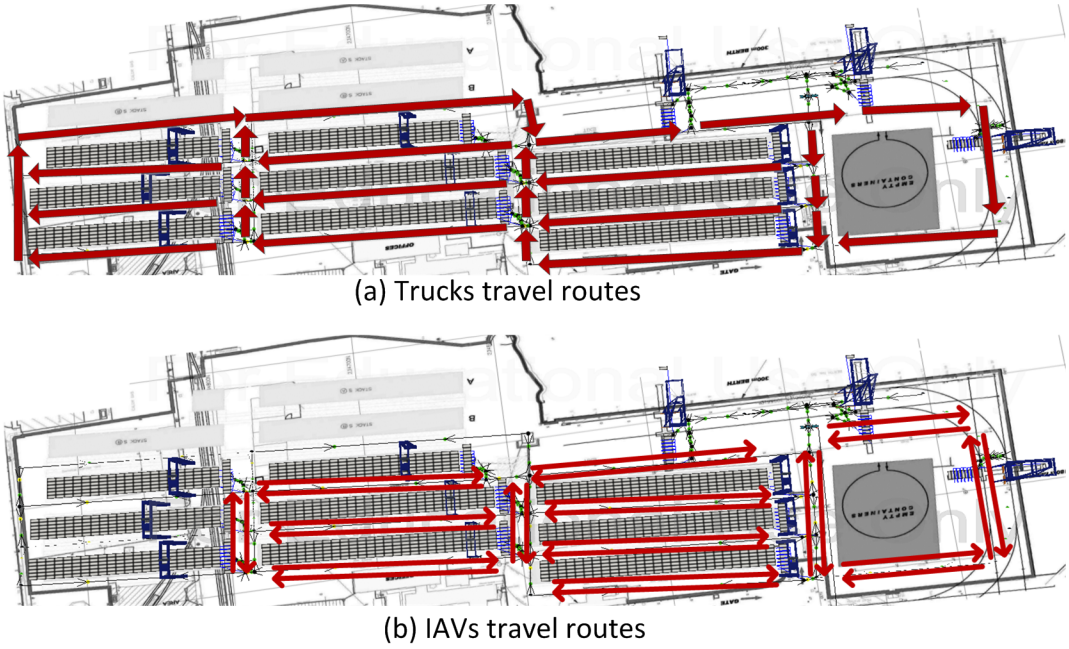

Figure 6.11: Travel routes of IAVs and trucks in port A. Plot a shows the travel routes for trucks and plot b depicts those of IAVs. Thanks to IAVs manoeuvrability, they do not need to go to the end of blocks to turn. Instead, from the point that they deliver containers to the stack cranes, they can move backward to travel to the quay side. This leads to shorter travel routes for IAVs.

6.4.3 Simulation results

This subsection provides the results of the developed simulation. In this experiment, quay crane net moves per hour and berth occupancy rate were considered to be the performance measures. For the experimental study, weekly schedules of this container terminal were simulated. The schedule of one week of this port is shown in Table 6.2. Note that simulating such a schedule with a combination of one, two and three quay

cranes by FlexSim CT for "straddle carrier"[2] objects is not possible, due to run time errors that are prompted during simulation. This is another limitation of FlexSim CT that highlights the importance of developing FlexFrame.

Table 6.2: This table shows the schedule of one week of the case study.

Day	Service	Berth	Containers		Arrival time
			Discharge	Load	
Monday	1	1	301	29	07:00
	2	2	157	52	22:15
	3	1	99	0	23:30
Tuesday	4	1	0	74	07:00
	5	2	5	193	16:00
	6	1	0	63	16:00
Thursday	7	2	187	0	00:00
	8	1	78	0	00:45
	9	1	149	118	14:45
Friday	10	2	3	282	09:30
	11	1	149	237	12:30
Saturday	12	1	83	98	07:00
Sunday	13	2	177	0	20:00
	14	1	233	0	23:00

This section follows a sensitivity analysis approach to report the results of the simulation. It first compares the performance of the case study with trucks against IAVs without cassettes (i.e. without considering the ability of IAVs to pick up and drop off containers by themselves - no buffers). Thus, in this case, the main differences between IAVs and trucks are the travel routes (Figure 6.11) and their speeds (Table 6.1). Figure 6.12 depicts the results of such comparison by varying the number of vehicles from 3 to 25 and considering the berth occupancy rate to be the performance measure. As can be seen, the performance of IAVs and trucks on berth 1 for the number of vehicles greater than 10 is quite similar. However, there are sharp differences between the berth occupancy rate with the lower number of vehicles. This means that

[2]As recalled from Subsection 5.2.5, the straddle carriers and transfer area objects were used to simulate IAVs in FlexSim CT.

by decreasing the number of vehicles the impact of the slower speeds of IAVs becomes more severe. The performance of trucks on the berth 2 for all the cases is better than that of IAVs. This is because berth 2 is farther than berth 1 and due to the fact that IAVs are much slower than trucks (Table 6.1), IAVs' better manoeuvrability cannot compensate for their slower speeds.

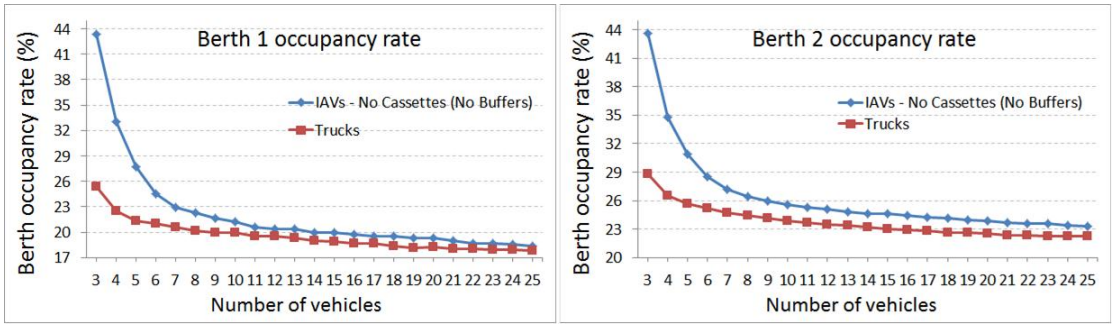

Figure 6.12: The comparison between the berth occupancy rate: IAVs without cassettes vs trucks

Figure 6.13 compares the berth occupancy rate for the case study with trucks versus IAVs with cassettes (i.e. with buffers). In this figure, the number of vehicles vary from 3 to 15 and the size of buffers from 1 to 10. For berth 1, in all the cases with more than 4 vehicles, IAVs perform much better than trucks. Moreover, it can be seen that by increasing the number of slots in the buffers (i.e. the size of buffers), the berth occupancy rate decreases significantly. For berth 2 the similar behaviour as for berth 1 can be observed, but in this case the performance of trucks compared with IAVs with one slot in buffers is much better. Similar to the case of IAVs without buffers, since berth 2 is farther than berth 1, the buffer size of one for IAVs is still not enough to provide better performance of IAVs against trucks.

Figure 6.14 depicts the average quay crane net moves per hour by varying the size of buffers from 1-10 and number of IAVs from 3-15. It shows that the impact of the size of buffers on the quay crane net moves per hour is significant. It also can be observed that there is a sharp increase in the quay crane net moves per hour when the

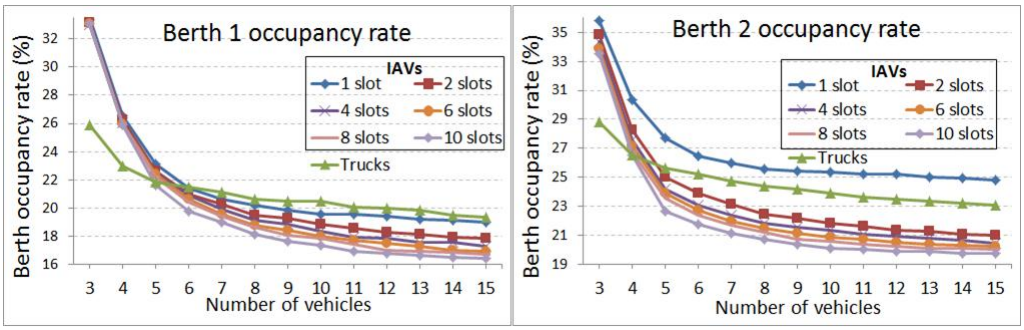

Figure 6.13: The comparison between the berth occupancy rate: IAVs with cassettes vs trucks. In this figure, the number of vehicles was varied from 3-15 and the size of buffers (i.e. number of slots in the buffers) from 1-10.

buffer size equals one compared with when the buffer size equals two.

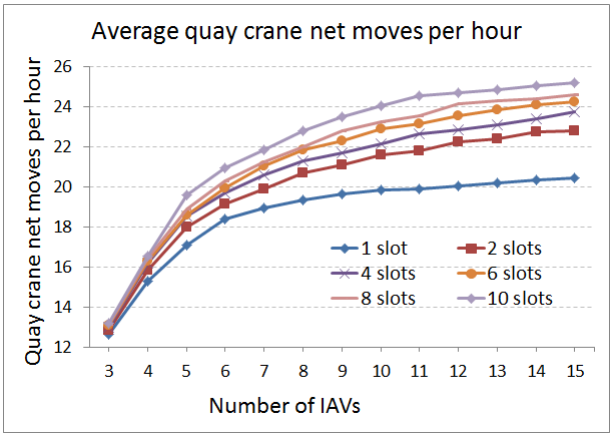

Figure 6.14: The comparison between the average quay crane net moves per hour: IAVs with cassettes vs trucks.

6.5 Conclusion

The contribution of this chapter is to develop a new simulation framework for the simulation of container terminals. This framework was developed upon the FlexSim CT simulation software to address its limitations. FlexSim CT is discrete-event simulation software for the simulation of container terminals. Despite its advantages such as having specific container terminal simulation objects and good 3D simulation fea-

tures, it has some limitations that hinder users to simulate realistic scenarios. The limitations arise from the quay-side operations. Due to some technical difficulties in the quay side simulation, FlexSim CT does not allow users to make any change in quay crane operations or quay crane and berth assignment. To address these limitations, a container terminal simulation framework with new object libraries was developed. This framework offers better flexibility to users and lets them develop their optimisation algorithms and use them in their simulation models. The framework was applied to simulate a European container terminal. In the experiments, the berth occupancy rate and quay crane net moves per hour were the performance measures and IAVs and trucks were considered to be the transfer vehicles. Results showed that the performance of the case study can be increased significantly if IAVs with buffers are used compared with trucks.

Chapter 7

Conclusions and Future Work

The thesis has attempted to answer the research questions raised in Section 1.3 about the important decision making problems related to the exploitation of IAVs in container terminals: what is the optimal number of IAVs in container terminals; how can IAVs be accommodated in container terminals; and more importantly how to estimate the performance and cost of container terminals with IAVs. These are crucial questions that any container terminal needs to deal with if they want to consider using the IAVs. To answer these questions, this thesis used advanced operational research techniques such as EAs and simulation. A European container terminal was considered the case study of this research to evaluate the applicability and generalisation of the proposed approaches in real environments.

7.1 Summary of major contributions

At the end of each chapter the main contributions have been provided. Here the main contributions of the thesis are summarised as follows.

Following an extensive literature review on the FSP in ESTTs, an EA is developed

to identify the optimal fleet size of vehicles in ESTTs. To evaluate the proposed EA, a series of test cases are generated based on the case study container terminals. The state-of-the-art CPLEX solver is considered to be the benchmark to evaluate the performance of the EA on the generated test cases. The comparison results show that in the majority of the cases (i.e. 128 out of 165 cases) the developed EA is significantly better than CPLEX. The EA is then combined with Monte Carlo simulation to take into account the uncertainty in the environment such as the travel time of vehicles and machine process times. This is to make the results of the EA close to the real practice in container terminals. This is the first research that considers these types of uncertainty for the FSP in ESTTs. The combined algorithm identifies the optimal number of vehicles that is robust to the aforementioned uncertainties in the environment. Comparing the results of the robust EA with a high-fidelity simulation shows that in the majority of the cases the robust solutions identified by the EA can be used effectively in real environments.

The developed EA is then enhanced further to be able to tackle large-scale problems in a reasonable time. To do so, a new dynamic sampling strategy is proposed and incorporated to the EA. Using this dynamic strategy, the samples are used more effectively as before. Thus, using fewer samples more accurate results can be achieved compared with the case without the dynamic sampling strategy. The experiment results on the generated test cases show a significant improvement of the EA in terms of performance. Furthermore, new robustness measures are proposed to achieve various robust solution based on the requirement of the user. A statistical test is then used to compare the robust solutions to provide better insights on these various robust solutions. The results of this statistical test can help port operators to identify the best robust solution based on their requirements.

A simulation model is developed to compare the performance of the case study

container terminal with IAVs against the existing vehicle system (i.e. trucks). The developed simulation model then identifies the optimal number of vehicles that can reach the provided targeted port performance. This shows the number of IAVs that are required to provide the same performance that are currently being delivered by trucks. To evaluate the cost of the case study container terminal with IAVs a cost mode is developed. In this cost model a 15-year period for analysis is considered. The cost of the case study with IAVs and trucks is estimated within this 15-year period. The results of cost model reveal that the cost of the case study container terminal can be decreased significantly if IAVs are being used.

Due to the lack of a proper tool for simulation of container terminals, a new simulation framework is developed. This simulation framework is developed upon the FlexSim CT simulation software. Though the FlexSim CT simulation software is a good tool for simulation of container terminals, this software has some limitations that prevent users to simulate realistic models. The developed simulation framework in the thesis addresses the existing limitations in Flexsim CT. This framework offers flexible simulation software that makes it possible for users to simulate various realistic scenarios in container terminals.

7.2 Future work

Container terminals are very complex systems and there are many decision making problems in container terminals that are related to IAVs. This thesis has mainly focused on the very important FSP from two different perspectives: optimisation and simulation approaches. This research is the first step to identify the robust number of IAVs in container terminals and also to study the performance of container terminals with IAVs. Thus, there are a lot of future works that need to be addressed. Some

possible directions for future study on this topic are discussed below:

1. *New robustness measures for robust fleet sizing:* In this thesis (Chapters 3 and 4), EAs were developed to identify the robust fleet size. In each generation, the robustness of solutions against the existing uncertainties was evaluated using a Monte Carlo simulation. However, there might be some alternative approaches for the evaluation of the robustness of solutions. In the context of robust scheduling and planning (Liu et al., 2007; Al-Fawzan and Haouari, 2005; Jorge et al., 1994), some surrogate robustness measures were defined based on different factors such as the workload of machines, total free slacks etc. These measures can be extended for the context of robust fleet sizing and used as alternative approaches to the Monte Carlo simulation used in this thesis. Using these surrogate measures might help to improve the performance of the proposed EA for uncertain environments. This is because, despite the fact that Monte Carlo simulation can produce accurate data and it is also easy to implement, one of the downsides of using Monte Carlo simulation is that it is computationally expensive. Thus it is of interest to investigate whether using surrogate robust measures can produce high quality solutions within better algorithm process time.

2. *Stochastic programming for the FSP:* Developing a stochastic programming model for the FSP under uncertainty can be another alternative approach for the uncertain FSP. In the literature (Vis et al., 2005), there is an IP model for the FSP in container terminals. This IP was used in this thesis as a benchmark for the proposed EA in static environments. However, for the uncertain case it was not possible to use the existing IP for the comparison. This IP needs to be extended to encompass uncertain elements in the formulation using stochastic programming approaches. The results of the extended IP for the uncertain case can then

be used as a benchmark for the developed EA in uncertain environments.

3. *Test cases for the FSP:* Due to the lack of available test cases in the literature, in Section 3.6, a set of test cases to evaluate the performance of the proposed EAs were generated. These test cases were generated based on the two case studies of the thesis. Although these test cases can partially fill the gap of the lack of available test cases for the FSP in container terminals, they are limited to the specifications of the two case studies. Thus, generating more test cases containing various characteristics of different container terminals is essential for further study on the FSP.

4. *Developing new meta-heuristic algorithms for the FSP in ESTTs:* In the thesis, an EA was developed to tackle the FSP in ESTTs. However, in the literature, there are many other meta-heuristic algorithms that could be used to tackle the FSP in ESTTs. Examples of such meta-heuristics are ant colony optimisation, particle swarm optimisation, tabu search, guided local search and simulated annealing. Thus, new algorithms based these meta-heuristics can be developed and compared with the results of the proposed EA in the thesis. Results of such a comparison can lead to identifying the best algorithm for the FSP in ESTTs.

5. *Combined simulation-optimisation framework for container terminals:* Various simulation studies in container terminals were reviewed in Chapter 2. This literature review identified a clear lack of available flexible simulation tools developed specifically for container terminals. The thesis has attempted to partially bridge this gap by developing a new flexible simulation framework capable of: 1) simulating realistic scenarios; and 2) being integrated with various optimisation algorithms. However, in container terminals there are many optimisation problems (e.g. berth and quay crane allocation, vehicles scheduling and routing etc)

that need to be addressed within the simulation model. As mentioned above, the developed simulation framework in Chapter 6 is capable of being integrated with any optimisation algorithms. However, in order to have a powerful simulation tool it is essential that the simulation tool not only is flexible to incorporate different optimisation algorithms but also has various embedded optimisation algorithms to address different decision making problems in container terminals. Developing such an integrated simulation-optimisation tool can effectively fill the current gap of simulation tools for container terminals.

References

Abbass, H. A., Alam, S. and Bender, A. (2009), 'Application notes: Mebra: multiobjective evolutionary-based risk assessment', *IEEE Computational Intelligence Magazine* **4**(3), 29–36.

Abbass, H. A., Bender, A., Dam, H. H., Baker, S., Whitacre, J. and Sarker, R. (2008), Computational scenario-based capability planning, *in* 'Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation', ACM, pp. 1437–1444.

Abbass, H. A., Bender, A., Gaidow, S. and Whitbread, P. (2011), 'Computational red teaming: Past, present and future', *Computational Intelligence Magazine, IEEE* **6**(1), 30–42.

Al-Fawzan, M. A. and Haouari, M. (2005), 'A bi-objective model for robust resource-constrained project scheduling', *International Journal of Poduction Economics* **96**(2), 175–187.

Al-Hinai, N. and ElMekkawy, T. (2011), 'Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm', *International Journal of Production Economics* **132**(2), 279–291.

Alvarez, F. J., Tsilingiris, P., Engebrethsen, E. S. and Kakalis, N. M. (2011), 'Ro-

bust fleet sizing and deployment for industrial and independent bulk ocean shipping companies', *INFOR: Information Systems and Operational Research* **49**(2), 93–107.

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

Angeloudis, P. and Bell, M. G. (2011), 'A review of container terminal simulation models', *Maritime Policy & Management* **38**(5), 523–540.

Arifin, R. and Egbelu, P. J. (2000), 'Determination of vehicle requirements in automated guided vehicle systems: A statistical approach', *Production Planning & Control* **11**(3), 258–270.

Asef-Vaziri, A., Laporte, G. and Ortiz, R. (2007), 'Exact and heuristic procedures for the material handling circular flow path design problem', *European Journal of Operational Research* **176**(2), 707–726.

Bartholdi, J. J. and Platzman, L. K. (1989), 'Decentralized control of automated guided vehicles on a simple loop', *IIE Transactions* **21**(1), 76–81.

Bazaraa, M. S., Jarvis, J. J. and Sherali, H. D. (2011), *Linear programming and network flows*, John Wiley & Sons.

Bazargan, M., Lange, D., Tran, L. and Zhou, Z. (2013), 'A simulation approach to airline cost benefit analysis', *Journal of Management* **14**(2), 55.

Ben-Tal, A. and Nemirovski, A. (2002), 'Robust optimization–methodology and applications', *Mathematical Programming* **92**(3), 453–480.

Beyer, H. G. and Sendhoff, B. (2007), 'Robust optimization–a comprehensive survey', *Computer Methods in Applied Mechanics and Engineering* **196**(33), 3190–3218.

Bianchi, L., Dorigo, M., Gambardella, L. M. and Gutjahr, W. J. (2009), 'A survey on metaheuristics for stochastic combinatorial optimization', *Natural Computing: An International Journal* **8**(2), 239–287.

Bielli, M., Boulmakoul, A. and Rida, M. (2006), 'Object oriented model for container terminal distributed simulation', *European Journal of Operational Research* **175**(3), 1731–1751.

Borshchev, A. and Filippov, A. (2004), From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools, *in* 'Proceedings of the 22nd International Conference of the System Dynamics Society'.

Boussaïd, I., Lepagnot, J. and Siarry, P. (2013), 'A survey on optimization metaheuristics', *Information Sciences* **237**, 82–117.

Briskorn, D., Drexl, A. and Hartmann, S. (2006), 'Inventory-based dispatching of automated guided vehicles on container terminals', *OR Spectrum* **28**(4), 611–630.

Bui, L. T., Abbass, H. A., Barlow, M. and Bender, A. (2012), 'Robustness against the decision-maker's attitude to risk in problems with conflicting objectives', *Evolutionary Computation, IEEE Transactions on* **16**(1), 1–19.

Cantoni, M., Marseguerra, M. and Zio, E. (2000), 'Genetic algorithms and Monte Carlo simulation for optimal plant design', *Reliability Engineering & System Safety* **68**(1), 29–38.

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

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

Celen, H., Slegtenhorst, R., van der Ham, R. T., Nagel, A., de Vos Burchart, R., Berg, J., van den Evers, J., Lindeijer, D., Dekker, R., Meersmans, P. et al. (1997), 'Famas newcon', *Concept TRAIL report. TU of Delft and E.U. Rotterdam* .

Chen, W., Allen, J. K., Tsui, K.-L. and Mistree, F. (1996), 'A procedure for robust design: minimizing variations caused by noise factors and control factors', *Journal of Mechanical Design* **118**(4), 478–485.

Choobineh, F. F., Asef-Vaziri, A. and Huang, X. (2012), 'Fleet sizing of automated guided vehicles: A linear programming approach based on closed queuing networks', *International Journal of Production Research* **50**(12), 3222–3235.

Chtourou, H. and Haouari, M. (2008), 'A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling', *Computers & Industrial Engineering* **55**(1), 183–194.

Cortes, P., Munuzuri, J., Nicolas Ibanez, J. and Guadix, J. (2007), 'Simulation of freight traffic in the Seville inland port', *Simulation Modelling Practice and Theory* **15**(3), 256–271.

Cruz, C., Gonzalez, J. R. and Pelta, D. A. (2011), 'Optimization in dynamic environments: A survey on problems, methods and measures', *Soft Computing* **15**(7), 1427–1448.

Deb, K. and Gupta, H. (2006), 'Introducing robustness in multi-objective optimization', *Evolutionary Computation* **14**(4), 463–494.

Dehghan, S., Kazemi, A. and Amjady, N. (2014), 'Multi-objective robust transmission expansion planning using information-gap decision theory and augmented ϵ-constraint method', *IET Generation, Transmission & Distribution* **8**(5), 828–840.

Delage, E. and Ye, Y. (2010), 'Distributionally robust optimization under moment uncertainty with application to data-driven problems', *Operations Research* **58**(3), 595–612.

Demirci, E. (2003), 'Simulation modelling and analysis of a port investment', *Simulation* **79**(2), 94–105.

Duinkerken, M. B., Dekker, R., Kurstjens, S. T., Ottjes, J. A. and Dellaert, N. P. (2006), 'Comparing transportation systems for inter-terminal transport at the Maasvlakte container terminals.', *OR Spectrum* **28**(4), 469 – 493.

Ebeling, C. (2009), 'Evolution of a box', *Invention and Technology* **23**(4), 8–9.

Egbelu, P. J. (1987), 'The use of non-simulation approaches in estimating vehicle requirements in an automated guided vehicle based transport system', *Material Flow* **4**(1), 17–32.

Farling, B., Mosier, C. and Mahmoodi, F. (2001), 'Analysis of automated guided vehicle configurations in flexible manufacturing systems', *International Journal of Production Research* **39**(18), 4239–4260.

Fitzgerald, K. R. (1985), 'How to estimate the number of AGVs you need', *Modern Materials Handling* **10**, 79.

Funk, K. and Rabl, A. (1999), 'Electric versus conventional vehicles: social costs and benefits in France', *Transportation Research Part D: Transport and Environment* **4**(6), 397–411.

Gabrel, V., Murat, C. and Thiele, A. (2014), 'Recent advances in robust optimization: An overview', *European Journal of Operational Research* **235**(3), 471–483.

Gambardella, L. M., Rizzoli, A. E. and Zaffalon, M. (1998), 'Simulation and planning of an intermodal container terminal', *Simulation* **71**(2), 107–116.

Gaspar-Cunha, A. and Covas, J. A. (2008), 'Robustness in multi-objective optimization using evolutionary algorithms', *Computational Optimization and Applications* **39**(1), 75–96.

Gelareh, S., Merzouki, R., McGinley, K. and Murray, R. (2013), 'Scheduling of intelligent and autonomous vehicles under pairing/unpairing collaboration strategy in container terminals', *Transportation Research Part C: Emerging Technologies* **33**, 1–21.

Gharehgozli, A. H., Roy, D. and De Koster, M. (2014), 'Sea container terminals: New technologies, or models, and emerging research areas', *ERIM Report Series Reference No. ERS-2014-009-LIS* .

Gu, J., Gu, M., Cao, C. and Gu, X. (2010), 'A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem', *Computers & Operations Research* **37**(5), 927–937.

Ha, B. H., Park, E. J. and Lee, C. H. (2007), A simulation model with a low level of detail for container terminals and its applications, *in* 'Proceedings of the 39th Conference on Winter Simulation: 40 Years! The Best is Yet to Come', IEEE Press, pp. 2003–2011.

Hadjiconstantinou, E. and Ma, N. L. (2009), 'Evaluating straddle carrier deployment policies: a simulation study for the Piraeus container terminal', *Maritime Policy & Management* **36**(4), 353–366.

Hartmann, S. (2004), 'Generating scenarios for simulation and optimization of container terminal logistics', *OR Spectrum* **26**(2), 171–192.

Helton, J. C. and Davis, F. J. (2003), 'Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems', *Reliability Engineering & System Safety* **81**(1), 23–69.

Henesey, L., Aslam, K. and Khurum, M. (2006), Task coordination of automated guided vehicles in a container terminal, *in* '5th International Conference on Computer Applications and Information Technology in the Maritime Industries', Leiden, Netherlands.

Henesey, L., Davidsson, P. and Persson, J. A. (2006), Agent based simulation architecture for evaluating operational policies in transshipping containers, *in* 'Multiagent System Technologies', Springer, pp. 73–85.

Henesey, L., Notteboom, T. and Davidsson, P. (2003), 'Agent-based simulation of stakeholders rela-tions: An approach to sustainable port and terminal management', *Multi-Agent Systems for Container Terminal Management* p. 99.

Henesey, L., Wernstedt, F. and Davidsson, P. (2002), A market based approach to container port terminal management, *in* 'Proceedings of the 15th European Conference on Artificial In-Multi-Agent Systems for Container Terminal Management', Citeseer.

Hillier, F. S. and Lieberman, G. J. (2010), *Intorduction to Operations Research*, McGraw Hill.

Hoshino, S. and Ota, J. (2007), Design of an automated transportation system in a seaport container terminal for the reliability of operating robot, *in* 'Intelligent

Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on', IEEE, pp. 4259–4264.

Ilić, O. R. (1994), 'Analysis of the number of automated guided vehicles required in flexible manufacturing systems', *The International Journal of Advanced Manufacturing Technology* **9**(6), 382–389.

Ji, M. and Xia, J. (2010), 'Analysis of vehicle requirements in a general automated guided vehicle system based transportation system', *Computers & Industrial Engineering* **59**(4), 544 – 551.

Jin, Y. and Branke, J. (2005), 'Evolutionary optimization in uncertain environments-a survey', *Evolutionary Computation, IEEE Transactions on* **9**(3), 303–317.

Jin, Y. and Sendhoff, B. (2003), Trade-off between performance and robustness: an evolutionary multiobjective approach, *in* 'Evolutionary Multi-Criterion Optimization', Springer, pp. 237–251.

Johnson, M. E. (2001), 'Modelling empty vehicle traffic in AGVS design', *International Journal of Production Research* **39**(12), 2615–2633.

Jorge, L., David Wu, S. and Storer, R. H. (1994), 'Robustness measures and robust scheduling for job shops', *IIE Transactions* **26**(5), 32–43.

Kasilingam, R. G. (1991), 'Mathematical modeling of the AGVS capacity requirements planning problem', *Engineering Costs and Production Economics* **21**(2), 171 – 175.

Kia, M., Shayan, E. and Ghotb, F. (2002), 'Investigation of port capacity under a new approach by computer simulation', *Computers & Industrial Engineering* **42**(2), 533–540.

Kılıç, M., Ulusoy, G. and Şerifoğlu, F. S. (2008), 'A bi-objective genetic algorithm approach to risk mitigation in project scheduling', *International Journal of Production Economics* **112**(1), 202–216.

Klaws, J., Stahlbock, R. and Voß, S. (2011), Container terminal yard operations–simulation of a side-loaded container block served by triple rail mounted gantry cranes, *in* 'Computational Logistics', Springer, pp. 243–255.

Kobylański, P. and Kuchta, D. (2007), 'A note on the paper by MA Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling', *International Journal of Production Economics* **107**(2), 496–501.

Koo, P. H., Lee, W. S. and W, J. D. (2004), 'Fleet sizing and vehicle routing for container transportation in a static environment.', *OR Spectrum* **26**(2), 193 – 209.

Kotachi, M., Rabadi, G. and Obeid, M. F. (2013), 'Simulation modeling and analysis of complex port operations with multimodal transportation', *Procedia Computer Science* **20**, 229–234.

Kuhn, A. (1983), Efficient planning of AGVS by analytical methods, *in* 'Proceedings of the 2nd International Conference on Automated Guided Vehicle Systems and 16th IPA Conference'.

Le-Anh, T. and De Koster, R. (2006), 'A review of design and control of automated guided vehicle systems', *European Journal of Operational Research* **171**(1), 1 – 23.

Lee, D., Gonzalez, L. F., Periaux, J. and Srinivas, K. (2011), 'Efficient hybrid-game strategies coupled to evolutionary algorithms for robust multidisciplinary design optimization in aerospace engineering', *Evolutionary Computation, IEEE Transactions on* **15**(2), 133–150.

Lee, L. H., Chew, E. P., Tan, K. C., Huang, H. C., Lin, W., Han, Y. and Chan, T. H. (2007), A simulation study on the uses of shuttle carriers in the container yard, *in* 'Simulation Conference, 2007 Winter', IEEE, pp. 1994–2002.

Lee, L. H., Chew, P. E., Cheng, H. X. and Han, Y. B. (2008), A study on port design automation concept, *in* 'Proceedings of the 40th Conference on Winter Simulation', Winter Simulation Conference, pp. 2726–2731.

Lee, T.-W., Park, N.-K. and Lee, D.-W. (2003), 'A simulation study for the logistics planning of a container terminal in view of SCM', *Maritime Policy & Management* **30**(3), 243–254.

Lin, C., Wu, T., Ou, X., Zhang, Q., Zhang, X. and Zhang, X. (2013), 'Life-cycle private costs of hybrid electric vehicles in the current chinese market', *Energy Policy* **55**, 501–510.

Lin, J., Gao, B. and Zhang, C. (2014), 'Simulation-based investment planning for Humen port', *Simulation Modelling Practice and Theory* **40**, 161–175.

Liu, C.-I., Jula, H. and Ioannou, P. A. (2002), 'Design, simulation, and evaluation of automated container terminals', *Intelligent Transportation Systems, IEEE Transactions on* **3**(1), 12–26.

Liu, C.-I., Jula, H., Vukadinovic, K. and Ioannou, P. (2004), 'Automated guided vehicle system for two container yard layouts', *Transportation Research Part C: Emerging Technologies* **12**(5), 349–368.

Liu, L., Gu, H.-y. and Xi, Y.-g. (2007), 'Robust and stable scheduling of a single machine with random machine breakdowns', *The International Journal of Advanced Manufacturing Technology* **31**(7-8), 645–654.

Mahadevan, B. and Narendran, T. (1990), 'Design of an automated guided vehicle-based material handling system for a flexible manufacturing system', *The International Journal of Production Research* **28**(9), 1611–1622.

Mahadevan, B. and Narendran, T. (1993), 'Estimation of number of AGVs for an FMS: An analytical model', *International Journal of Production Research* **31**(7), 1655–1670.

Mahdavi, I., Shirazi, B. and Solimanpur, M. (2010), 'Development of a simulation-based decision support system for controlling stochastic flexible job shop manufacturing systems', *Simulation Modelling Practice and Theory* **18**(6), 768–786.

Malmborg, C. J. (1990), 'A model for the design of zone control automated guided vehicle systems', *International Journal of Production Research* **28**(10), 1741–1758.

Mantel, R. J. and Landeweerd, H. R. (1995), 'Design and operational control of an AGV system', *International Journal of Production Economics* **41**(1), 257 – 266.

Marseguerra, M. and Zio, E. (2000), 'Optimising maintenance and repair policies via a combination of genetic algorithms and Monte Carlo simulation', *Reliability Engineering & System Safety* **68**(1), 69–83.

Marseguerra, M., Zio, E. and Podofillini, L. (2002), 'Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation', *Reliability Engineering & System Safety* **77**(2), 151–165.

Marseguerra, M., Zio, E. and Podofillini, L. (2007), Genetic algorithms and Monte Carlo simulation for the optimization of system design and operation, *in* 'Computational Intelligence in Reliability Engineering', Springer, pp. 101–150.

McGinley, K. and Murray, R. (2013), Qualitative study on the implementation of ITS in port environments: Part 1: Economic suitability of viability of the IAV, Technical report, Dublin Institute of Technology.

Meng, Q. and Wang, T. (2010), 'A chance constrained programming model for short-term liner ship fleet planning problems', *Maritime Policy & Management* **37**(4), 329–346.

Meng, Q., Wang, T. and Wang, S. (2012), 'Short-term liner ship fleet planning with container trans-shipment and uncertain container shipment demand', *European Journal of Operational Research* **223**(1), 96–105.

Milenković, M. and Bojović, N. (2013), 'A fuzzy random model for rail freight car fleet sizing problem', *Transportation Research Part C: Emerging Technologies* **33**, 107–133.

Moradi, E., Fatemi Ghomi, M. T. and Zandieh, M. (2011), 'Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem', *Expert Systems with Applications* **38**(6), 7169–7178.

Muller, T. (1983), *Automated guided vehicles*, IFS Ltd.

Mulvey, J. M., Vanderbei, R. J. and Zenios, S. A. (1995), 'Robust optimization of large-scale systems', *Operations Researchesearch* **43**(2), 264–281.

Murty, K. G. (1994), *Operations Research: Deterministic Optimization Models*, Prentice Hall.

Nam, K.-C. and Ha, W.-I. (2001), 'Evaluation of handling systems for container terminals', *Journal of Waterway, Port, Coastal, and Ocean Engineering* **127**(3), 171–175.

Nam, K.-C., Kwak, K.-S. and Yu, M.-S. (2002), 'Simulation study of container terminal performance', *Journal of Waterway, Port, Coastal, and Ocean Engineering* **128**(3), 126–132.

Nevins, M. R., Macal, C. M. and Joines, J. C. (1995), PORTSIM: An object-oriented port simulation, Technical report, Argonne National Lab., IL (United States).

Nevins, M. R., Macal, C. M. and Joines, J. C. (1998), 'A discrete-event simulation model for seaport operations', *Simulation* **70**(4), 213–223.

Nevins, M. R., Macal, C. M., Love, R. J. and Bragen, M. J. (1998), 'Simulation, animation and visualization of seaport operations', *Simulation* **71**(2), 96–106.

Nguyen, S., Zhang, M., Johnston, M. and Tan, K. C. (2014), 'Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming', *Evolutionary Computation, IEEE Transactions on* **18**(2), 193–208.

Nguyen, T. T. (January 2011), Continuous Dynamic Optimisation Using Evolutionary Algorithms, PhD thesis, School of Computer Science, University of Birmingham, http://etheses.bham.ac.uk/1296.

Nguyen, T. T., Yang, S. and Branke, J. (2012), 'Evolutionary dynamic optimization: A survey of the state of the art', *Swarm and Evolutionary Computation* **6**, 1–24.

Nordgren, W. B. (2003), Flexsim simulation environment, *in* 'Simulation Conference, 2003. Proceedings of the 2003 Winter', Vol. 1, IEEE, pp. 197–200.

Ottjes, J. A., Duinkerken, M. B., Evers, J. J. and Dekker, R. (1996), Robotised inter terminal transport of containers, *in* 'Proceeding 8th European Simulation Symposium', Citeseer, pp. 621–625.

Ottjes, J. A., Hengst, S. and Tutuarima, W. (1994), A simulation model of a sailing container terminal service in the port of Rotterdam, *in* 'Proceedings of the 1994 Conference on Modelling and Simulation', Citeseer, pp. 876–880.

Ottjes, J. A., Veeke, H. P., Duinkerken, M. B., Rijsenbrij, J. C. and Lodewijks, G. (2007), Simulation of a multiterminal system for container handling, *in* 'Container Terminals and Cargo Systems', Springer, pp. 15–36.

Pantuso, G., Fagerholt, K. and Hvattum, L. M. (2014), 'A survey on maritime fleet size and mix problems', *European Journal of Operational Research* **235**(2), 341–349.

Parola, F. and Sciomachen, A. (2005), 'Intermodal container flows in a port system network: Analysis of possible growths via simulation models', *International Journal of Production Economics* **97**(1), 75–88.

Rajotia, S., Shanker, K. and Batra, J. (1998), 'Determination of optimal AGV fleet size for an FMS', *International Journal of Production Research* **36**(5), 1177–1198.

Raman, D., Nagalingam, S. V., Gurd, B. W. and Lin, G. C. (2009), 'Quantity of material handling equipment: A queuing theory based approach', *Robotics and Computer-Integrated Manufacturing* **25**(2), 348 – 357.

Rashidi, H. and Tsang, E. P. (2013), 'Novel constraints satisfaction models for optimization problems in container terminals', *Applied Mathematical Modelling* **37**(6), 3601–3634.

Ray, T. (2002), Constrained robust optimal design using a multiobjective evolutionary algorithm, *in* 'Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on', Vol. 1, IEEE, pp. 419–424.

Rebollo, M., Julian, V., Carrascosa, C. and Botti, V. (2000), A multi-agent system for the automation of a port container terminal, *in* 'Proceedings of Autonomous Agents 2000 Workshop on Agents in Industry'.

Rodrigue, J.-P., Comtois, C. and Slack, B. (2013), *The Geography of Transport Systems*, Routledge.

Rubinstein, R. Y. and Kroese, D. P. (2011), *Simulation and the Monte Carlo method*, Vol. 707, Wiley.

Saanen, Y., van Meel, J. and Verbraeck, A. (2003), The design and assessment of next generation automated container terminals, *in* 'Proceedings 15th European Simulation Symposium (ESS 2003)', pp. 577–584.

Sayarshad, H. R. (2010), 'Using bees algorithm for material handling equipment planning in manufacturing systems', *The International Journal of Advanced Manufacturing Technology* **48**, 1009–1018.

Sayarshad, H. R. and Tavakkoli-Moghaddam, R. (2010), 'Solving a multi periodic stochastic model of the rail–car fleet sizing by two-stage optimization formulation', *Applied Mathematical Modelling* **34**(5), 1164–1174.

Sevaux, M. and Sörensen, K. (2004), 'A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates', *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* **2**(2), 129–147.

Shabayek, A. and Yeung, W. (2002), 'A simulation model for the Kwai Chung container terminals in Hong Kong', *European Journal of Operational Research* **140**(1), 1–11.

Shadrokh, S. and Kianfar, F. (2007), 'A genetic algorithm for resource investment

project scheduling problem, tardiness permitted with penalty', *European Journal of Operational Research* **181**(1), 86–101.

Sinriech, D. and Tanchoco, J. (1992), 'An economic model for determining AGV fleet size', *International Journal of Production Research* **30**(6), 1255–1268.

Solberg, J. J. (1977), A mathematical model of computerized manufacturing systems, *in* 'Proceedings of the 4th International Conference on Production Research', pp. 1265–1275.

Soriguera, F., Espinet, D. and Robuste, F. (2006), 'A simulation model for straddle carrier operational assessment in a marine container terminal', *Journal of Mritime Research* **3**(2), 19–34.

Stahlbock, R. and Voss, S. (2008), 'Operations research at container terminals: a literature update.', *OR Spectrum* **30**(1), 1 – 52.

Steenken, D., Voss, S. and Stahlbock, R. (2004), 'Container terminal operation and operations research - a classification and literature review.', *OR Spectrum* **26**(1), 3 – 49.

Sun, Z., Lee, L. H., Chew, E. P. and Tan, K. C. (2012), 'Microport: A general simulation platform for seaport container terminals', *Advanced Engineering Informatics* **26**(1), 80–89.

Sun, Z., Tan, K. C., Lee, L. H. and Chew, E. P. (2013), 'Design and evaluation of mega container terminal configurations: An integrated simulation framework', *Simulation* pp. 684–692.

Taguchi, G. (1986), *Introduction to Quality Engineering: Designing Quality into Products and Processes*, Tokyo : Asian Productivity Organization.

Taguchi, G. (1989), *Introduction to quality engineering*, American Supplier Institute.

Tanchoco, J., Egbelu, P. J. and Taghaboni, F. (1987), 'Determination of the total number of vehicles in an AGV-based material transport system', *Material Flow* **4**(1-2), 33–51.

Turnquist, M. A. and Jordan, W. C. (1986), 'Fleet sizing under production cycles and uncertain travel times', *Transportation Science* **20**(4), 227–236.

UNCTAD (2013), *(United Nations Conference on Trade and Development) Secretariat. Review of Maritime Transpor*, United Nations Publication.

Veenstra, A. W. and Lang, N. (2004), 'Economic analysis of a container terminal simulation', *International Journal of Logistics Research and Applications* **7**(3), 263–279.

Vidal, T., Crainic, T. G., Gendreau, M. and Prins, C. (2013), 'Heuristics for multi-attribute vehicle routing problems: a survey and synthesis', *European Journal of Operational Research* .

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

Vis, I. F., De Koster, R., Roodbergen, K. J. and Peeters, L. W. P. (2001), 'Determination of the number of automated guided vehicles required at a semi-automated container terminal', *European Journal of Operational Research* **52**(4), pp. 409–417.

Vis, I. F., De Koster, R. and Savelsbergh, M. W. (2005), 'Minimum vehicle fleet size under time-window constraints at a container terminal.', *Transportation Science* **39**(2), 249 – 260.

Vis, I. F. and Harika, I. (2004), 'Comparison of vehicle types at an automated container terminal', *OR Spectrum* **26**(1), 117 – 143.

Wang, S. and Yu, J. (2010), 'An effective heuristic for flexible job-shop scheduling problem with maintenance activities', *Computers & Industrial Engineering* **59**(3), 436–447.

Wiesmann, D., Hammel, U. and Back, T. (1998), 'Robust design of multilayer optical coatings by means of evolutionary algorithms', *Evolutionary Computation, IEEE Transactions on* **2**(4), 162–167.

Wojtaszek, D. and Wesolkowski, S. (2012), 'Military fleet mix computation and analysis [application notes]', *Computational Intelligence Magazine, IEEE* **7**(3), 53–61.

Wysk, R., Egbelu, P. J., Zhou, C. and Ghosh, B. (1987), 'Use of spread sheet analysis for evaluating agv systems', *Material Flow* **4**(1-2), 53–64.

Xiong, J., Liu, J., Chen, Y. and Abbass, H. A. (2013), 'A knowledge-based evolutionary multi-objective approach for stochastic extended resource investment project scheduling problems', *Evolutionary Computation, IEEE Transactions on* .

Xiong, J., Yang, K.-w., Liu, J., Zhao, Q.-s. and Chen, Y.-w. (2012), 'A two-stage preference-based evolutionary multi-objective approach for capability planning problems', *Knowledge-Based Systems* **31**, 128–139.

Yao, D. D. and Buzacott, J. (1985), 'Queuing models for a flexible machining station part I: The diffusion approximation', *European Journal of Operational Research* **19**(2), 233–240.

Yim, D.-S. and Linnt, R. (1993), 'Push and pull rules for dispatching automated

guided vehicles in a flexible manufacturing system', *The International Journal of Production Research* **31**(1), 43–57.

Yun, W. Y. and Choi, Y. S. (1999), 'A simulation model for container-terminal operation analysis using an object-oriented approach', *International Journal of Production Economics* **59**(1), 221–230.

Zuo, X., Mo, H. and Wu, J. (2009), 'A robust scheduling method based on a multi-objective immune algorithm', *Information Sciences* **179**(19), 3359–3369.