Control of Free-ranging Automated Guided Vehicles in Container Terminals

Jung-Yong Seo

A thesis submitted for the degree of Doctor of Philosophy and Diploma of Membership of Imperial College London

Port Operations Research & Technology Centre

Centre for Transport Studies

Department of Civil & Environmental Engineering

Imperial College London

December 2017

This work is dedicated to my family.

Acknowledgements

This study would not have been completed without certain people and organisations to whom I would like to express my acknowledgements:

- To Dr Panagiotis Angeloudis and Prof Washington Yotto Ochieng who have supervised me during my PhD. Without their guidance and support, this work would have not been completed.
- To all the members in the Centre for Transport Studies (CTS) who have always been with me.
- To my friends at Imperial College London and in my country: ICKS 에 소속된 모든 박사과정 학생 및 포닥 형 누나들 너무 고맙습니다. 항상 힘내라고 응원 해줘서 너무 고마워요. 그리고 한국에 있는 내 친구들, 너무 고맙다. 너희들이 정말 큰 힘이 되었다.
- 장윤석, 채준재 교수님: 이 자리에 오기까지 학문적, 정신적으로 도움을 주신 한국항공대학교 장윤석, 채준재 교수님 너무 감사합니다. 교수님들의 지도가 없었다면, 지금과는 다른 길을 가고 있었을 것입니다.
- 나의 가족들에게: 가장 사랑하는 사람들, 늘 뒤에서 묵묵히 응원해준 우리 가족 모두 너무 고맙습니다. 또한, 건강하게 있어줘서 너무 고마워요. 앞으로도 아프지 말고, 모두 건강하게 잘 지냈으면 좋겠습니다.

Declaration of Originality

I hereby declare that the work described in this thesis has been performed on my own.

_____

Jung-Yong Seo

December 2017

Copyright Declaration

Abstract

Container terminal automation has come to the fore during the last 20 years to improve their efficiency. Whereas a high level of automation has already been achieved in vertical handling operations (stacking cranes), horizontal container transport still has disincentives to the adoption of automated guided vehicles (AGVs) due to a high degree of operational complexity of vehicles. This feature has led to the employment of simple AGV control techniques while hindering the vehicles to utilise their maximum operational capability. In AGV dispatching, vehicles cannot amend ongoing delivery assignments although they have yet to receive the corresponding containers. Therefore, better AGV allocation plans would be discarded that can only be achieved by task reassignment. Also, because of the adoption of predetermined guide paths, AGVs are forced to deploy a highly limited range of their movement abilities while increasing required travel distances for handling container delivery jobs. To handle the two main issues, an AGV dispatching model and a fleet trajectory planning algorithm are proposed. The dispatcher achieves job assignment flexibility by allowing AGVs towards to container origins to abandon their current duty and receive new tasks. The trajectory planner advances Dubins curves to suggest diverse optional paths per origin-destination pair. It also amends vehicular acceleration rates for resolving conflicts between AGVs. In both of the models, the framework of simulated annealing was applied to resolve inherent time complexity. To test and evaluate the sophisticated AGV control models for vehicle dispatching and fleet trajectory planning, a bespoke simulation model is also proposed. A series of simulation tests were performed based on a real container terminal with several performance indicators, and it is identified that the presented dispatcher outperforms conventional vehicle dispatching heuristics in AGV arrival delay time and setup travel time, and the fleet trajectory planner can suggest shorter paths than the corresponding Manhattan distances, especially with fewer AGVs.

Contents

List of figures

List of tables

Acronyms and abbreviations

A	Anticlockwise steering
AABB	Axis-aligned Bounding Boxes
ACO	Ant Colony Optimisation
ACT	Automated Container Terminal
ACV	Automated Container Vehicle
AGV	Automated Guided Vehicle
ALV	Automated Lifting Vehicle
AYC	Automated Yard Crane
C	Clockwise steering
DTSP	Dubins Travelling Salesperson Problem
ECT	Europe Container Terminals
EDD	Earliest Due Date
EPETWD	Earliest Possible Event Time without Delay
ETSP	Euclidean Travelling Salesperson Problem
FEFS	First-encounter-first-served
FTPP	Fleet Trajectory Planning Problem
GA	Genetic Algorithm
GNRON	Goals Non-reachable with Obstacles Nearby
GV	Ground Vehicle
HHLA CTA	Hamburger Hafen und Logistik AG Container Terminal Altenwerder
HPC	High-performance Computing
ID	Identification
LP	Linear Programming
m	Metre(s)
MDE	Map Definition Error
MILP	Mixed Integer Linear Programming

MIP	Mixed Integer Programming
mTSP	Multiple Travelling Salesperson Problem
N	Non-additional steering
NSE	Navigation System Error
NV	Nearest Vehicle
OBB	Oriented Bounding Boxes
OHT	Overhead Hoist Transport
P&D	Pickup and Delivery
PSO	Particle Swarm Optimisation
RIC	Region of Inevitable Collision
RRT	Rapidly-exploring Random Tree
S	Straight
SA	Simulated Annealing
sec	Second(s)
STT/D	Shortest Travel Time/Distance
TEU	Twenty-foot Equivalent Unit
TS	Tabu Search
TSE	Total System Error
TSP	Travelling Salesperson Problem
TTI	Total Terminal International
UAV	Unmanned Aerial Vehicle
UNCTAD	United Nations Conference on Trade and Development
VCE	Vehicle Control Error

Chapter 1. Introduction

Sea container terminals are becoming increasingly busy and have been struggling with high labour costs. Global trade with containers doubled from 2002 to 2016 and is likely to rise by 5% in the next five years. This resulted in a rise in the number of containers handled in container terminals of 1.9% in the last year and projected to increase by 2.3% worldwide by 2018, requiring additional labour as well as equipment (UNCTAD, 2017). The financial burden of labour constitutes a significant proportion of the total operating costs; for example, over 50% for the ports of Sydney and Melbourne (Castalia Strategic Advisors, 2012).

In order to reduce the operating costs, terminal operators have been trying to replace human labour with automated equipment. These types of the equipment are mainly container vehicles that convey containers between the quay and yard, and cranes that move containers from container stacks onto the vehicles or vice versa.

However, a high level of automation is still to be achieved in container deliveries by container vehicles compared to container handling by cranes. Unlike cranes, the vehicles have a high degree of freedom in terms of movement and operate as a large fleet. This makes it challenging to manage container vehicle operations automatically, and therefore they are constrained regarding delivery assignments and movement in practice. This, in turn, causes low quay crane productivity which is a key performance measure in container terminals while requiring more equipment and longer vehicular journeys. There has also been little research on the aspect of terminal simulation models for unconstrained automated vehicle systems, design, development, implementation and operational impact.

Hence, this thesis focuses on improving the operation of automated container vehicles (ACVs) for use in sea container terminals. The remainder of the chapter includes a discussion on current issues concerning the horizontal container transport operation and relevant simulation models. Motivations, aims, and objectives are also provided alongside an outline of this thesis.

1.1. Background

First introduced in 1956, containers are standardised metal boxes used in international trade (Levinson, 2016). Several types of shipping containers exist in circulation. The basic container is 20 feet long, 8 feet wide and 8 feet 6 inches high, referred to as the Twenty-Foot Equivalent unit - TEU (Meisel, 2009). Along with TEU containers, 40-foot long intermodal containers are also popular in the market.

Container vessels, also called containerships, vary in size. They refer to cargo ships that move intermodal containers between maritime container terminals. Small container vessels carry up to 1,000 TEUs; however, large ones can hold over 10,000 TEUs. Panamax containerships are well-known, able to hole between 3,000 and 4,400 TEUs (Gilman, 1999).

Maritime container terminals act as interfaces between sea and land transport modes. They handle containers for export, import and transhipment by terminal operation that consists of several sub-operation units. Cranes (i.e. vertical container handling equipment) in the quay and the yard move containers from container stacks onto container vehicles (i.e. horizontal container handling equipment) or vice versa. Container vehicles transport containers between the quay and the yard. Gate and rail operations connect the terminals to inland areas by exchanging containers.

Terminal automation has been being employed around the world. Automated container terminals (ACTs) are container terminals where automated equipment, such as automated cranes and vehicles, operates for container transfers. Although human labour is still relied on in most terminal environments, several terminals have been automated to replace workers (Günther and Kim, 2006; Yang and Shen, 2013).

Since 1993, terminal operators have been utilising automated yard cranes (AYCs) and automated guided vehicles (AGVs) for container handling – these vehicles are the most popular ACV type. The first ACT is the Europe Container Terminals (ECT) Delta Terminal in the Netherlands. Subsequently several terminals, including Thamesport in the UK and the

HHLA CTA Terminal in Hamburg, started deploying automation. Outside of Europe, the Fisherman Island Terminal in Brisbane has been using automated container handling equipment since 2006. The Busan New Port in South Korea and the Port of Kaohsiung in Taiwan followed suit in 2009 and 2011 respectively (Yang and Shen, 2013). Several container terminals, such as those in the Busan New Port, have yet to employ ACVs.

1.2. Current Issues and Research Motivations

This subsection introduces and explains limitations in relevant previous research and current horizontal container transport systems in sea container terminals. The limitations fall into three areas this study seeks to address.

Inflexible Vehicle Dispatching

Vehicle dispatching is a decision-making process that aims to achieve goals, such as the minimisation of job delays, for vehicles and other resources undertaking cargo delivery jobs. It is required in various industrial environments, such as manufacturing systems and warehouses. In ACTs, it refers to a process of assigning AGVs to container delivery jobs while minimising job delays or vehicular travel.

Three approaches to dispatching exist regarding task reassignment. The first approach does not amend assignment links once these have been planned. The second approach makes vehicles maintain their current job but can change subsequent tasks. The third can even re-allocate such ongoing tasks when their designated vehicle has yet to take the corresponding container; therefore, it has the highest dispatching flexibility among the three approaches.

A remarkable issue is that most previous research on dispatching have rarely pursued the highest dispatching flexibility. Because the third approach provides the largest possible assignment set among the three approaches, maintaining planned assignments means a

potential loss of better vehicle dispatch plans that would be achieved by task reassignment. Although several dispatching models and algorithms have achieved complete re-assignability, they do not include task-sequencing. Vehicles move to task origins without cargos, and such journeys are affected by the destinations of their previous delivery task. Therefore, vehicular travel time tends to increase without considering task-sequencing impacts.

Inefficient AGV Trajectories

AGV systems are mainly based on magnetic flow-paths or transponders installed on the floor. AGVs perform their duties by moving along such preinstalled lines or positioning themselves through triangulation. The latter configuration provides more flexible vehicular movements since transponders do not constrain trajectories. In other words, AGVs can freely move under their kinematic limitations without any predetermined networks (i.e. the free-ranging vehicle setting). Thus, more sophisticated trajectory planners are required. To detect radio signals transmitted from magnetic flow-paths or transponders, AGVs are equipped with radio sensors. The vehicles also have laser sensors that identify the distances from other vehicles or obstacles by measuring the time taken by light to reach a target object and return.

Existing ACTs have adopted magnetic flow-path systems which lead to two remarkable drawbacks. Firstly, these systems are subject to the Manhattan distance, which is the sum of the absolute differences of all the Cartesian coordinate pairs of two points. Therefore, resulting trajectories cannot be shorter than those subject to the Euclidean distance, which is pursued by the free-ranging vehicle setting (*Figure 1-1*). Secondly, magnetic flow-path systems constrain trajectories in shape and therefore provide less options in avoiding vehicular collisions and deadlocks (i.e. system stalling). Especially, AGVs should take long detours when vehicular conflicts occur on the vehicles' planned routes. On the other hand, the free-ranging vehicle setting allows AGVs to escape such adverse situations by changing their heading or path without a substantial path length increase.

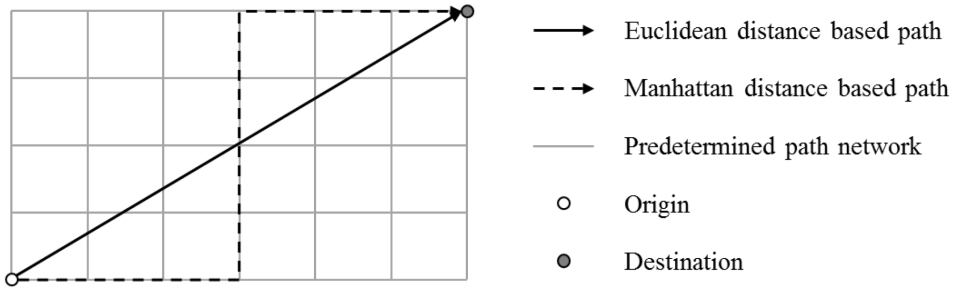


Figure 1-1. Comparison between the Euclidean and the Manhattan distances.

A considerable amount of research has been performed on path and trajectory planning for use in various application settings, such as manufacturing systems, warehouses and ACTs. However, no category can be appropriate for potential ACTs where a large AGV fleet operates under the free-ranging vehicle setting while requiring detailed trajectories and travel safety. Firstly, network-based trajectory planning is not applicable in the potential ACTs because it computes trajectories based on a finite set of predetermined path segments (i.e. edges) the free-ranging vehicle setting cannot use. Therefore, network-based models and algorithms cannot be used in the target ACTs.

Secondly, the fact that AGVs can move faster in straight lines than in curves has been ignored in the literature. In practice, it requires AGVs to decelerate before they move on curves when travelling at over their maximum turning speed. Thus, resulting trajectories by previous trajectory planning models would make AGVs travel on curves at over their maximum curve speed, which is inaccurate and unsafe.

Thirdly, a large vehicle fleet has led to simple trajectories unable to be adopted in the target ACTs. Research concerning a single vehicle provides sophisticated, short, smooth and detailed paths while considering various vehicular kinematic characteristics. However, it probably results in significant computation and vehicular conflicts in fleet trajectory planning since the optimal path of a vehicle would cause a collision with another. On the other hand, fleet trajectory planning has tended to ignore vehicular constraints or limit vehicular manoeuvres. For example, paths by Qu, Xing and Alexander (2013) contain zig-zags and Pallottino, Feron and Bicchi (2002) allow a fleet of aircraft to alter their heading only once.

Absence of Simulation Models in Integrated Operation Planning for Free-ranging AGVs

As mentioned by Le-Anh and De Koster (2006), problems in the interaction between AGV dispatching and trajectory planning demonstrate a clear need for the integration of the two processes: The former determines the destinations of AGVs that influence their trajectories, and the latter the locations of vehicles to calculate expected travel time to their goal locations, which is used in the process of dispatching.

Therefore, there is a need for simulation models that provide a function of testing and evaluating integrated operation planning for free-ranging AGVs in potential ACTs. However, previous research has even ignored the integration of the two decision-making problems. Although Corman *et al.* (2016) introduce the free-ranging vehicle setting to integrated AGV control, they do not consider task reassignment and several critical elements in trajectory planning, such as the maximum speed rate difference for turning and moving straight, the maximum vehicular steering angle and required postures at destinations.

Commercial simulation packages, such as Arena, have been used in various port environments, such as oil terminals. However, they are not appropriate for testing sophisticated vehicle dispatching and trajectory planning algorithms since they are more suitable for generating process diagrams and flowcharts based on predetermined functional blocks that can simply be dragged-and-dropped by users. Without a significant amount of programming work, they cannot be utilised to realise detailed vehicle operations.

1.3. Aim and Objectives

This study seeks to develop an integrated transport operation solution for AGVs in potential free-ranging ACTs to improve container handling in the area of the horizontal container carriage by addressing the issues identified in the previous section. The study also provides a simulation model to test and evaluate the solution in a port-like domain. The following objectives are part of this research:

- Identify the limitations of existing vehicle dispatching methods, path and trajectory planning approaches and simulation models for use in ACTs (*Chapter 2*)
- Construct the methodology of the research (Chapter 3)
- Define a flexible AGV dispatching model and provide a solution for real-time applications (*Chapter 4*)
- Develop a fleet trajectory planning algorithm for AGVs without utilising pre-fixed flow-paths (*Chapter 5*)
- Construct a simulation model for testing and evaluating a vehicle operation model that integrates the AGV dispatching algorithm and the fleet trajectory planning model (*Chapter 6*)
- Evaluate the AGV operation models in a simulated container terminal with various scenarios and settings (*Chapter 7*)

1.4. Thesis Structure

The thesis consists of eight chapters, and each falls into several subsections.

Chapter 1 firslty introduces the background of the research and identifies several issues on vehicle dispatching and fleet trajectory planning in container terminals in consideration of relevant simulation models. It also outlines the aim and the objectives of the study.

Chapter 2 starts with an explanation on terminal automation and ACTs, which is followed by a discussion on previous research on vehicle dispatching, path planning and trajectory planning in various application settings, such as aviation and container terminals. It also includes a review of previous studies on deadlock handling for AGV operation environments. It then classifies and reviews simulation models for vehicle operation analysis. The second chapter concludes with a critical evaluation of the literature.

Chapter 3 introduces the methodology of the research. It describes research design and process used in the doctoral degree while recapping the research aim and objectives. It also explains

ideas, techniques and methods that were adopted in or are closely-related to vehicle operation models suggested by this research.

Chapter 4 begins with an explanation of AGV tasks in container terminals and introduces a new AGV dispatch model that allows AGVs to change their ongoing job while considering task-sequencing. A metaheuristic framework for the dispatching model is introduced to handle the model's time complexity for real-time applications. A series of experimental results with randomly-generated problem scenarios are also provided.

Chapter 5 describes a new path generation algorithm with an explanation of its sub-processes. Additionally, it shows how to generate acceleration rates on generated paths and provides a new fleet trajectory planning algorithm while considering trajectory safety in the environment where multiple AGVs operate. Like *Chapter 4*, it presents simulation results to evaluate the fleet trajectory planning model.

Chapter 6 provides a simulation model specially designed for free-ranging AGVs that operate in potential ACTs. The bespoke model mainly focuses on vehicle operation analysis with a function of providing results subject to customised performance indicators.

Chapter 7 provides a simulation domain, settings and scenarios used for the research and identifies performance indicators to evaluate the developed dispatching and trajectory planning models. It also discusses the results of performed simulation experiments.

Chapter 8 summarises the contributions and the limitations of the research. The thesis concludes by suggesting further work to improve the study.

Chapter 2. Literature Review

This chapter describes terminal automation and outlines previous research on vehicle operations for cargo deliveries in generic environments as well as container terminals. The literature review covers vehicle dispatching, trajectory planning and the integration of the two areas. It also contains studies on deadlock handling potentially required in vehicle fleet control.

2.1. Terminal Automation

Terminal automation can be represented by ACTs where various automated equipment units operate in container handling with the help of sophisticated resource control systems. Throughout the thesis, resource control (or control of resources) refers to a type of planning and managing the operations of resources. Although ACTs bring about a high initial investment cost, they have three key benefits compared to traditional container terminals.

The first one is the decrease in human error. Unmanned handling equipment can perform simple, repetitive work more precisely. For resources with a high degree of operational freedom and potential interactions between them, such as container vehicles, sophisticated control methods are essential. Secondly, automation contributes to reducing operating costs in regions with high staffing costs (Stahlbock and Voß, 2008). For instance, replacing straddle carriers with automated counterparts can reduce operating costs by over 70% (Saanen, 2016). Automation can also yield performance increases, such as in quay crane productivity (Liu, Jula and Ioannou, 2002). Higher quay crane productivity reduces ship turnaround time, thus increasing the competitiveness of container terminals.

2.1.1. Vertical Container Handling Automation

Automation of vertical container handling means the replacement of yard and quay crane operators with their corresponding automated equipment. Terminal operators have automated

yard cranes with a high level of success, given the limited degree of movement freedom, repetitive movements and semi-isolated operational nature. The cranes transfer containers between vehicles and yard blocks. A yard block refers to an un-shareable area where containers are stacked up and is essentially assigned to each yard crane. There exist two types of yard block layouts as described in *Figure 2-1*. Unlike the vertical layout, the horizontal layout makes container vehicles share the workplace with outbound trucks that move containers between terminals and inland areas.

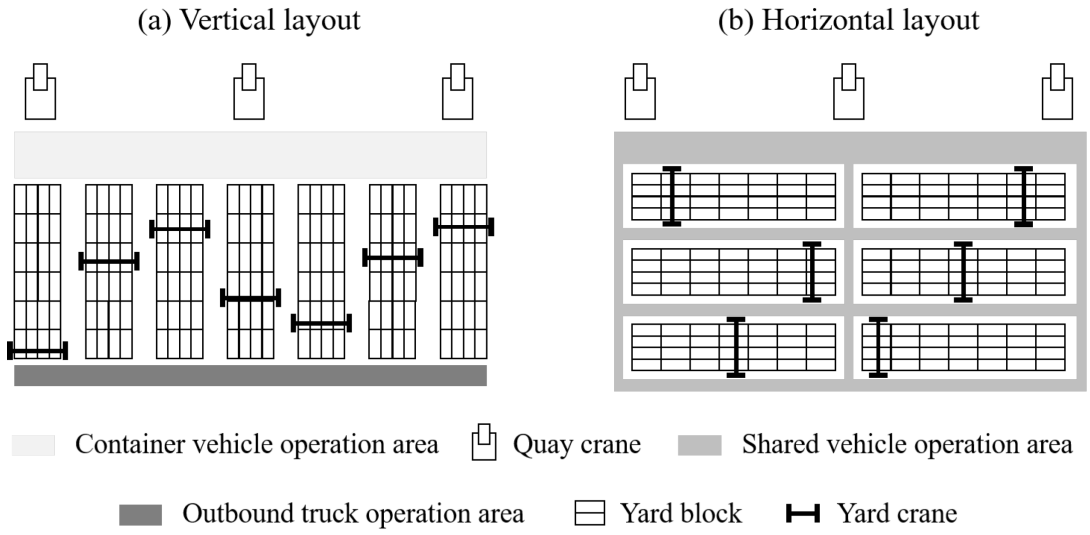


Figure 2-1. Yard block layouts.

As an advanced version, two AYCs can operate in the same yard block (Gharehgozli *et al.*, 2015). The technology allows one AYC to serve container vehicles while the other provides services for outbound trucks or rearranges containers in the stack. The two cranes form a twin- or a dual-configuration (Carlo and Martínez-Acevedo, 2015). In the first configuration shown in *Figure 2-2* (a), two cranes share a set of rails on which the cranes move; therefore, careful control is essential to avoid collisions. In the second configuration, the smaller crane moves underneath the larger one without sharing their metal guidelines; therefore, no collision threat exists (*Figure 2-2* (b)).

On the other hand, it is not easy for quay crane operations to be automated. Unlike yard cranes, quay cranes work with various types of container vessels floating on water. Since their

spreader is supported by several wires sensitive to winds, it is difficult to place the spreaders precisely onto container slots on floating ships. Such work is hard to be programmed due to a high degree of uncertainty; therefore, terminal operators still require experienced quay crane operators.

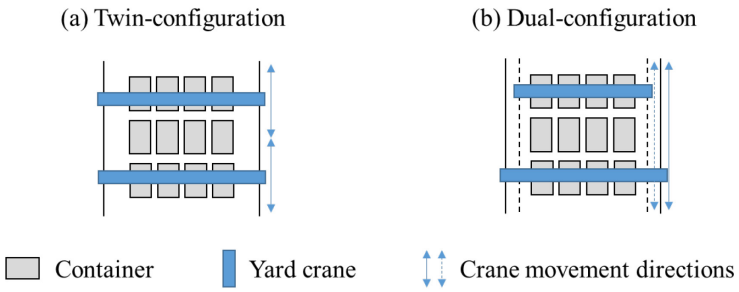


Figure 2-2. Two AYC types.

Researchers and practitioners achieved partial automation to address the adverse working conditions to which quay crane operators are exposed. Dual-trolley, semi-automated quay cranes were consequently introduced, dividing vertical container handling into two distinct activities (Günther and Kim, 2006). The vessel-side remains manually controlled and focuses on container transfers between the vessel and an intermediate platform that acts as a temporary storage point. The second trolley operates between this platform and the crane buffer area, which can be easily automated since it does not interact directly with the ship (*Figure 2-3*).

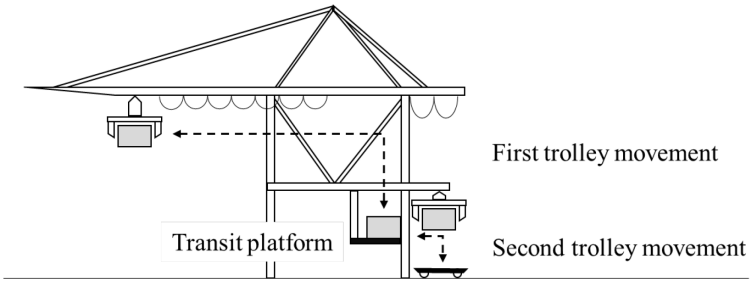


Figure 2-3. Dual-trolley quay crane.

2.1.2. Horizontal Container Transport Automation

Automation of horizontal container transport refers to the system of equipment and enabling infrastructure that is responsible for the movement of containers between the yard and the quay. Attempts to automate horizontal container handling equipment have been less successful

compared to vertical container handling. Whereas quay and yard cranes mainly operate in an un-shareable workspace, container vehicles have a high degree of movement freedom and operate in a shared area. This feature makes it difficult for ACV systems to be programmed since vehicular journeys should carefully be planned to avoid potential threats of collisions between the vehicles.

Traffic Flow Design of ACVs

So far, all ACT operators have utilised pre-defined flow-path systems to achieve easy traffic control. The systems include magnetic guide-wires on the ground to restrict the movement of ACVs. These vehicles can only move along the inductive lines even though they have a free-ranging (or free-range) capability implying that they can operate with transponder navigation, not using the constrained paths.

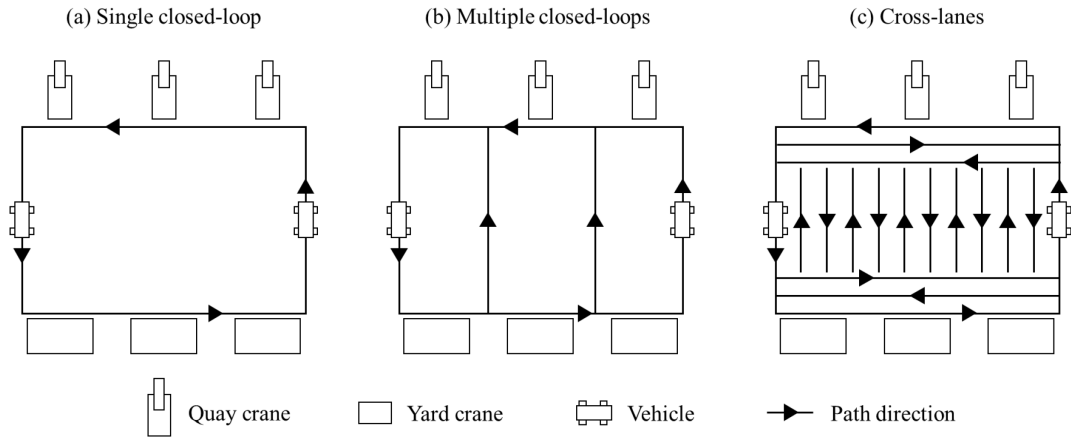


Figure 2-4. Closed-loop and cross-lane guide path designs.

There exist two types of predetermined path systems (*Figure 2-4*). The first type (i.e. closed-loop system) guarantees simple vehicle control but requires long vehicular journeys. *Figure 2-4* (a) shows the application of a single closed-loop whereas *Figure 2-4* (b) depicts a multiple-loop system, which was adopted in ECT Rotterdam. *Figure 2-4* (c) describes the second type, named cross-lane system. This is the most recent of the three path installation approaches while being used in ECT's Euromax. The path system type includes horizontal cross-lanes and vertical corridors that suggest multiple path selections per origin-destination pair.

However, it requires advanced vehicular journey planning since it has many intersections where potential vehicular conflicts can occur.

The free-ranging setting for ACVs can be regarded as an alternative vehicle flow system. It does not adopt predetermined guide-paths but employs transponder navigation. Therefore, under the free-ranging setting vehicles can fully utilise their kinematic limitations while being only subject to other vehicles' movements to avoid conflicts.

The free-ranging vehicle setting can be a favourable option in the transport area between the quay and yard whereas it is less appropriate in the yard area. Since the yard area includes explicit corridors between yard blocks, it can be represented by a road network for ACVs; therefore, network-based path planning is more appropriate in the yard than free-ranging vehicle operations. Container terminals with vertically-arranged yard blocks would be managed by the free-ranging vehicle setting, but those with horizontally-arranged yard blocks additionally require network-based path planners.

Three Types of ACVs

ACVs mainly fall into three groups: AGVs, lift-AGVs, and automated lifting vehicles (ALVs) with the first group being the most popular option in current ACTs (*Figure 2-5*). Although the term 'AGVs' can also mean automated vehicles employed in other application settings, throughout this thesis it refers to the vehicles used in maritime container terminals (*Figure 2-5* (a)). Since all the vehicle types are car-like, they are designed to be able to move faster in linear sections than in curves.

AGVs are usually tasked with moving ISO containers between crane buffers and as they are not equipped with lifting equipment, they depend upon the cranes to position containers safely and securely; therefore, they should cooperate with cranes. Most commercial AGVs are capable of either carrying a single 40ft container or up to two 20ft containers at any time (i.e. the dual-load capability of AGVs).

(a) AGV (b) Lift-AGV (c) ALV

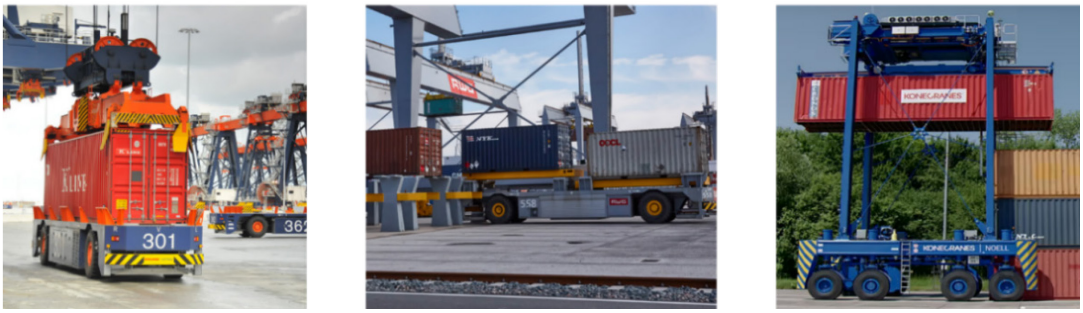


Figure 2-5. Types of automated vehicles for use in ACTs. Source: Konecranes[1]

A fleet of this vehicle type has two main drawbacks. Firstly, the inability to independently load or unload containers necessitates the presence of a larger fleet to achieve desired service levels in selected performance areas, such as quay crane productivity. Although this vehicle type is the cheapest among the three vehicle types regarding unit vehicle prices, it potentially causes a more expensive vehicle system in total. Secondly, AGV systems need sophisticated trajectory planning models to manage a relatively large fleet.

More recently Lift-AGVs, a new class of ACVs, were developed to address the main weaknesses of AGVs (*Figure 2-5* (b)). They are capable of independent container handling through the help of lift platforms. In the case where a Lift-AGV needs to take a container on such a cargo handover platform, the vehicle moves into the platform, raises its plate to decouple the container from the lift platform, moves out from the area and lowers the plate. The platform allows crane and container vehicle operations to decouple. Therefore, the waiting time of vehicles at crane buffers decreases while resulting in a smaller fleet size compared to AGV systems.

ALVs have almost the same functionality with straddle carriers and are capable of lifting and placing containers on the ground and container stacks without handover platforms (*Figure 2-5* (c)). Since they can also replace yard cranes as well as container vehicles, their use can simplify terminal resource management. However, ALVs have their own set of drawbacks.

[1] Available at: http://www.konecranes.com/equipment [Accessed 13 July. 2017]

Their unit cost is the highest of the three automated vehicle types, being around 60% more expensive than AGVs (Saanen, 2016). Furthermore, they tend to be relatively unstable as a result of their increased height and high level of gravity – in a well-known incident that occurred in 2009, a straddle carrier toppled in the Port of Auckland (*Figure 2-6*).

Figure 2-6. Toppled straddle carrier. Source: Cargolaw[2]

Table 2-1 shows performance and cost comparisons between AGV systems, Lift-AGV systems and ALV systems, included in the study by Saanen (2016). Regarding vehicle net productivity (containers/hour), ALVs have almost two times better performance than AGVs. The result means that a small ALV fleet can show equivalent performance to an AGV fleet that is double in size, which would however incur only a quarter of the operating costs. Lift-AGVs show similar performance to ALVs with a low operating cost of €14 per hour.

Table 2-1. Comparisons between the three automated vehicle types

Performance indicator	ALV	Lift-AGV	AGV
Required vehicles (vehicles/QC)	3.5	4.0	6.5
Net productivity (containers/hour)	12.0	10.5	6.5
Price (euro/vehicle)	850k	680k	540k
Operating cost (euro/hour)	37	14	10

* QC: quay crane

[2] Available at: http://www.cargolaw.com/2009nightmare_auk_straddle.html [Accessed 13 July. 2017]

Control Strategies for ACVs

Such ACVs are controlled by the following two strategies: the centralised vehicle control strategy and the decentralised vehicle control strategy. Under the first strategy, there exists a central vehicle controller that observes the states of all ACVs and manages their operations in a cooperative manner. It is also possible to consider interactions between ACVs and cranes in the process of container transfers. Therefore, this strategy is adopted when (1) assigning container delivery jobs to vehicles (i.e. dispatching) and (2) planning vehicular journeys to crane buffers in consideration of deadlocks, explained in *Section* 2.5 in detail.

On the other hand, under the decentralised vehicle control strategy ACVs individually control themselves with the help of onboard sensors and computers. In the case where two vehicles are involved in a collision, for example, each keeps detecting the other vehicle's movement and adjusts its trajectory without changing the other's trajectory. Since ACVs cannot always be moved as planned by a central controller, the vehicles should also be controlled under the decentralised control strategy to follow planned trajectories while avoiding expected collisions.

Unlike ACV systems, autonomous vehicle (AV) systems are generally subject to the decentralised strategy only. Whereas ACV systems are subject to known domains and capable of foreseeing systemic changes, AVs operate in unknown environments with unknown, uncontrollable objects, such as pedestrians. Therefore, the control of AVs rather focuses on reactions against sudden environmental changes near the vehicles, as well as considering normal situations and controllable factors.

2.2. Vehicle Dispatching

A common terminology of vehicle dispatching does not exist in the current literature. Instead, it is referred to as dispatching (Lim *et al.*, 2003; Bish *et al.*, 2005; Briskorn, Drexl and Hartmann, 2006) or scheduling (Le, Yassine and Moussi, 2012; Skinner *et al.*, 2013). Throughout this thesis, the term vehicle dispatching will be used.

As discussed by Dantzig and Ramser (1959), the dispatching problem can be regarded as a generalised version of the travelling salesperson problem (TSP). The TSP is a decision-making problem in which a seller determines their visiting order of n cities to minimise the total travelled distance (Flood, 1956). They should visit every city once. By increasing the number of salespeople from 1 to m, the problem changes into the multiple TSP (mTSP), which is a form of the vehicle dispatching problem.

From a different point of view, the dispatching problem represents the resource allocation problem of m resources to n transport jobs (Grunow, Günther and Lehmann, 2006). Given a vehicle dispatching plan request, each idle resource (i.e. vehicle) receives one or more missions; or pending tasks are assigned to resources.

When each task refers to a cargo delivery from a place to another, task sequencing affects vehicular journeys for receiving cargos. Given a vehicle with two delivery tasks, the vehicle finishes its first task at the destination and moves to the origin of the second task. Such journeys become short where the destination of the former task is close to the origin of the following task. Thus, the capability of assigning a list of delivery tasks to each vehicle helps to improve dispatching performance in vehicular travel distance and time.

The following reviewed studies in this section show three tendencies. Firstly, they are based on predetermined flow-path systems, not on the free-ranging vehicle setting. Secondly, most of them provide single tasks or large task lists for cargo transporters at a time, except for the research by Grunow, Günther and Lehmann (2006) and Angeloudis and Bell (2010). Thirdly, they ignore task reassignment able to be considered when designated transporters have yet to take designated cargos.

Egbelu and Tanchoco (1984) identify various vehicle dispatching rules categorised into two groups, *vehicle-initiated* dispatching and *station-initiated* dispatching. Any rules in the first group, represented by the shortest travel time/distance (STT/D) rule, assign a newly available vehicle to a cargo delivery mission. In the second group, a work centre that has recently

released a task initiates a vehicle-dispatching process. The nearest vehicle (NV) rule is one of the most popular methods.

Bartholdi and Platzman (1989) develop a multi-load AGV dispatching model based on the first-encounter-first-served (FEFS) rule for use in single closed-loop configurations. In such a configuration, there exists only one closed loop on which AGVs travel. When a station needs to send a cargo to another, it assigns the delivery request to the vehicle that visits the station first with an empty slot to accommodate the cargo. Loaded AGVs release their cargos when they visit the goal locations. Their simulation results show that the method outperforms the first-in-first-out rule in waiting time on delivery tasks.

Kim, Tanchoco and Koo (1999) consider workload balance in dispatching. They provide a priority index for each transport task. The index consists of two sub-indices: (1) the workload difference between the origin and destination (i.e. balancing index) and (2) the corresponding task urgency index. The first index rises when there exist many jobs at the source location compared to the goal. The second index is utilised to break the tie of balancing indices.

Kim and Bae (1999) focus on the fact that the sequences of quay crane tasks are determined before container handling operations, and they are not likely to change. In their scenario, quay cranes alternatively perform loading and discharging operations (i.e. dual-cycle operations) to reduce the empty traversal distance of the cranes' spreaders. The suggested model can assign many tasks to each AGV and is designed to minimise the completion time of quay cranes and reduce vehicular travel time.

Lim *et al.* (2003) regard AGV dispatching in manufacturing systems as the *m: n* allocation problem. They utilise the concept of auction markets where competition exists. Cargo transport missions are likely to choose vehicles suggesting lower costs; however, vehicles are likely to select delivery missions with higher margins. Except for the ongoing missions of vehicles, the model can amend vehicular assignments to reduce the total cost of the vehicle system.

Kim and Bae (2004) develop a new dispatching method that mainly tries to minimise the delay of quay cranes while reducing vehicular travel time for container pickup. Their simulation results show that the presented model outperforms several alternative dispatching rules, such as the STT/D rule and the earliest due date (EDD) rule, regarding the two objectives. Bish *et al.* (2005) utilise the feature of predetermined quay crane schedules to overcome the myopic nature of greedy algorithms. They provide each delivery task a weight value that means the minimum time required to finish all the following jobs processed by the same quay crane.

Briskorn, Drexl and Hartmann (2006) adopt a basic idea of inventory management for AGV dispatching in container terminals. The idea is that the level of inventory should not be zero and too high. Each quay crane's inventory level is defined as the number of AGVs in its buffer and on the way to the crane. Therefore, the model can prevent quay cranes from being crowded by vehicles.

Grunow, Günther and Lehmann (2006) consider the dual-load capability of container AGVs. Since long-term planning increases the complexity of dispatching, their model assigns two tasks per vehicle. An event, such as the delay of a delivery job, initiates a dispatching process. ongoing vehicular jobs cannot be cancelled, and a newly allocated job to an AGV should be performed after the vehicle finishes its current task. Such assignment rigidity may decrease service qualities in several performance areas, such as the empty travel time. Their simulation results show that the single-load setting is almost equivalent to the dual-load setting in performance.

Kim *et al.* (2007) suggest an event-based vehicle dispatching rule for use in the overhead hoist transport (OHT) system in which vehicles are connected to ceiling-mount rails and transport loads along the rails. When a vehicle finishes its delivery task, the proposed rule identifies (1) idle stations and (2) the process stations in which their load has yet to be taken by vehicles and to which these vehicles are farther compared to the newly-available vehicle. The rule then assigns the vehicle to the best-fitted station by utilising a distance-based fitness function. It

also considers line-blocking issues in which vehicles' journeys are blocked by a vehicle that is on the same path and in the process of load transfer.

Kim *et al.* (2009) perform m to n reassignment by adopting the Hungarian algorithm, where m refers to the number of vehicles and n refers to the number of jobs. The proposed approach is different from the rule by Kim *et al.* (2007) in that the earlier rule limits the number of re-assignable vehicle-job links to one, although both studies are subject to predetermined path systems. Their simulation results show that the Hungarian algorithm-based approach can outperform the previous rule in terms of lead time.

Angeloudis and Bell (2010) consider uncertainty in estimated travel time for AGV dispatching. Their simulator keeps gathering travel time data to calculate the various levels of travel time uncertainties in origin-destination pairs. Where a journey link has been unstable in travel time, a high cost is assigned to the link; therefore, this link becomes unattractive to AGVs. Similar to the research by Grunow, Günther and Lehmann (2006), their model allocates two transport tasks to each vehicle at most. It leads to rapid computation time that allows the model to be valid in real-time applications. Task reassignment is unavailable when designated vehicles are heading to their pickup station.

Rashidi and Tsang (2011) formulate an AGV dispatching problem in container terminals as the minimum cost flow problem for finding the cheapest route in a given network to convey a certain quantity of cargo. They utilise two solvers: a simplex algorithm and a greedy vehicle search method. The first solver is more appropriate when the time complexity of a give problem instance is not significant. It can provide the optimal solution but requires long computation time. The greedy solver is more attractive when the magnitude of a problem is too large to be solved by the simplex method within a limited period of time.

Some researchers have paid attention to the different types of ACVs. Nguyen and Kim (2009) perform research on ALV dispatching that is a different version of the AGV dispatching

research by Kim and Bae (2004). Both studies use predetermined quay crane schedules and perform long-term planning.

Le, Yassine and Moussi (2012) devise a dispatching model for lifting vehicles with the objective of minimising the total lateness of quay cranes and the total travel time of the vehicles. Similar to the research by Kim and Bae (2004), the higher weight is assigned to the first objective term, and each vehicle can take many container delivery jobs.

Skinner *et al.* (2013) develop a dispatching algorithm for straddle carriers. Unlike the studies by Kim and Bae (2004) and Le, Yassine and Moussi (2012), they add a term for reducing the total waiting time of vehicles on their objective function. Additionally, urgent tasks have higher priority scores in order to be assigned as early as possible. The model allows each vehicle to hold multiple container transport missions (i.e. long-term vehicle dispatching).

2.3. Path and Trajectory Planning

Path planning refers to a process of generating a geometric line or curve that connects a source to its goal location, and trajectory planning determines locational points or postures to be observed at specific time points (Gasparetto *et al.*, 2015). Both problems are subject to the kinematic limitations of target objects and environmental features, such as the locations of obstacles or the size of a given domain. In the literature, the two terms (i.e. path planning and trajectory planning) are likely to be used interchangeably. In such cases, path planning models mostly apply a constant speed rate to their moving objects.

There exist two categories for path and trajectory planning: one is based on predetermined path networks, each of which consists of multiple nodes and edges (*Section* 2.3.1). The other is subject to the free-ranging vehicle setting (*Section* 2.3.2 to 2.3.6). It is sub-divided into five groups by the following approaches: conventional methods (*Section* 2.3.2), curves (*Section* 2.3.3), mathematical formulation (*Section* 2.3.4), vehicle dynamics (*Section* 2.3.5) and metaheuristics (*Section* 2.3.6).

2.3.1. Guide Path-based Approaches

Many studies on path and trajectory planning have been performed subject to one of the flow-path systems explained in *Section* 2.1.2. A limited number of studies are based on the single closed-loop setting. They guide vehicles to their destinations by controlling vehicular speed rates only while considering a safety distance between the vehicles; therefore, it is not regarded as a complex decision-making problem. Instead, Tanchoco and Sinriech (1992) focused on the design of single-loop configurations.

In multiple-loop topology, a group of cargo handling stations are located on each loop that is served by a single vehicle while allowing adjacent loops to exchange loads (Lin and Dgen, 1994). So, if the origin and the destination of a task are not on the same loop, multiple AGVs perform the task. Lin, Chang and Liu (1994) devise a two-phase fleet trajectory planning model. The model suggests candidate trajectory sets based on steady states and then evaluates the candidates by several performance measures, such as the queue lengths of transit stations or the utilisation of AGVs. Lin and Dgen (1994) apply a task-list time-window concept to their algorithm to find a path set that minimises the travel time of vehicles.

During the past two decades, many researchers have presented fleet trajectory planning models for use in mesh-like path design (Zeng and Hsu, 2008; Chiew and Qin, 2009; Ghasemzadeh, Behrangi and Azgomi, 2009; Jeon, Kim and Kopfer, 2011; Chiew, 2012).

Chiew and Qin (2009) utilise a sorting technique. Their assumed environment consists of multiple rectangular working areas, and each area has four P&D stations (one at each corner). Also, two unidirectional edges connect two stations in different intersections. The model labels each station by using its row, column and junction indices. In the initial stage, AGVs each stay in an un-shareable station. The model then labels the AGVs with their destination identification (i.e. ID). It divides the given ID set of the stations into multiple groups of rows and sorts the station IDs (i.e. vehicle destinations) vertically and horizontally in ascending (or

descending) order for odd- (or even-) numbered groups. Although it does not guide the AGVs onto their theoretical shortest paths, it guarantees conflict-free travels.

Jeon, Kim and Kopfer (2011) propose a learning-based fleet trajectory planning model. Their learning engine ultimately creates an optimal-next-node table. Each tabular element (i, j) refers to the best next node for a journey from a current node i to a destination node j, determined by expected travel time. The model continues to update the matrix by trial and error and finds a steady state of the table elements in the end.

Similar to Chiew and Qin (2009), Chiew (2012) develops an algorithm to achieve conflict-free AGV fleet journeys by using a coordinate-ordering concept. The proposed algorithm labels AGVs with their destination coordinates. Then, it vertically sorts the vehicles based on their column index to make all elements in each row have different destination column indices. It implements similar processes horizontally and vertically to match the vehicles' destination row and column numbers to the corresponding stations' IDs.

Vehicle fleets as well as single vehicles have been of interest to researchers in the field of path and trajectory planning subject to predetermined path systems. There has been a significant amount of effort to solve vehicular conflicts rather than only reduce travel time.

2.3.2. Conventional Approaches

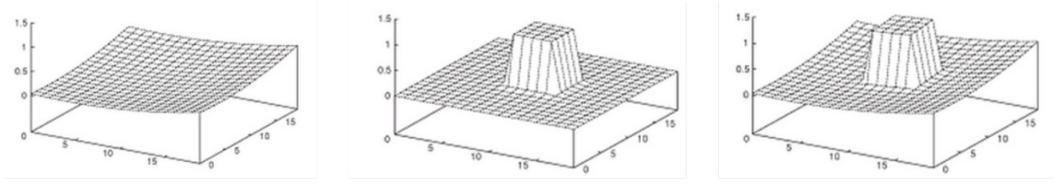
There are three conventional approaches to path planning: the cell decomposition approach, the potential field approach and the roadmap approach (Latombe, 1991). The first path planning approach is a process of (1) dividing a given domain into multiple areas (i.e. cells), (2) generating a network by connecting the cells and (3) finding the shortest path in the network.

Seneviratne, Ko and Earles (1997) perform single agent path planning by adopting a triangulation-based cell decomposition technique. Their model converts a given environment with polygonal static obstacles into a simple polygon by adding virtual edges between the

boundaries of the obstacles and the environment. Based on the simplified polygon, it creates non-overlapping triangles by adding non-intersecting diagonals and implements path planning with the midpoints of the triangles' edges.

Williams and Jones (2001) utilise the cell decomposition approach to three-dimensional unmanned aerial vehicle (UAV) path planning. Ghita and Kloetzer (2012) devise a trajectory planning model for a mobile robot considering its kinematic characteristics and size. Dugarjav *et al.* (2013) and Paull *et al.* (2013) perform coverage path planning. This is a type of decision-making problem to find a collision-free path covering a set of must-visit points (Galceran and Carreras, 2013).

The second path planning approach utilises potential fields. Given a path planning problem instance with a single controllable agent, this approach generates attractive potential and repulsive potential at the destination and all obstacles respectively. The attractive potential pulls the agent whereas the repulsive potential pushes it away. So, the goal point and the obstacles each have a potential field in a three-dimensional free space (*Figure 2-7* (a) and *Figure 2-7* (b)), and all the fields are summed up to form a total potential field domain as shown in *Figure 2-7* (c). If the destination is influenced by adjacent obstacles and therefore is not the global minimum in the summed field, the agent would not reach the destination, which is called "Goals Non-reachable with Obstacles Nearby" (GNRON).



(a) Goal field (b) Obstacle field (c) Summed field

Figure 2-7. Potential fields. Source: Dudek and Jenkin (2010)

Ge and Cui (2000) suggest a new potential calculation function in order to deal with the GNRON problem. Given a single agent and an unreachable destination, this function makes the destination's potential zero (i.e. the most attractive point) by considering the distance from

the agent to the destination. Park and Lee (2003) prevent agents from being trapped at local optimal points by introducing virtual obstacles near the local optima.

Similar to Ge and Cui (2000), Zhang, Lü and Song (2004) propose a new repulsive potential function to handle the GNRON problem. They consider the distance between the destination of a target vehicle and the vehicle's current location, in order to prevent the total potential at the goal from not being the global minimum. Kim and Shin (2006) suggest analytical design guidelines on potential functions to escape from local minima. Ge and Cui (2002) develop a potential field based trajectory planning method for use in dynamic environments. Cetin and Zagli (2012) and Cetin, Zagli and Yilmaz (2013) focus on UAV journeys with potential fields.

Roadmap-based path planning (i.e. the third approach) starts with generating a road network or a directed tree behind a given domain and finds the shortest path in the network. Lam and Srikanthan (2001) present a rapid map configuration method for dynamic path planning. Given a domain, they extract an area that can contain the shortest path to decrease search space size. Kim, Kim and Kim (2011) suggest a method that converts each arbitrarily-shaped object into a polygon.

Rapidly-exploring random trees (RRTs) are one of the most popular methods in the roadmap approach. An RRT is an algorithm in which a tree iteratively grows from a location origin until the tree reaches a destination (Lavalle and Kuffner, 2001). It ends up producing a directed tree that consists of a root (i.e. the origin) and numerous branches. Such trees are utilised as predetermined path systems.

In each iteration, the algorithm randomly generates a candidate node x and identifies the closest node of the current tree to the node x, denoted by y. If the distance between the nodes is over a distance of d, the algorithm newly defines a node z away by d from y towards x (i.e. linear extrapolation). If x or z does not cause a collision, the algorithm connects y to x or z and then updates the tree; otherwise, it tries the process again.

Since RRTs include many non-optimal connections (i.e. longer edges), they mainly focus on finding feasible paths rather than optimal ones. To handle this issue, Karaman and Frazzoli (2011) provide RRT* that ignores redundant feasible connections during the iterative process of RRTs. Their simulation results prove that RRT* outperforms RRT in path length with the Euclidean distance cost function.

Otani *et al.* (2009) adopt the framework of RRTs for cooperative path planning in a maze-like environment. In their scenario, two circular robots transport a box alternately from its origin to the destination. Each robot has a series of sub-journeys for passing the box to the other robot. Therefore, unlike general path planning scenarios the travel path of the box is not equal to any vehicular paths. Kim *et al.* (2015) develop an RRT-based path planning method for single articulated robots that have a high degree of movement freedom. For rapid computation, they focus on decreasing the size of solution search space by selecting key parts and joints of a robot.

Cui, Li and Yan (2016) suggest a path planning algorithm based on the idea of RRT* for generating a scalar field of an interesting underwater area. In such a scalar field generation problem, each vehicle collects local data by using its onboard sensor and shares the data with its neighbour vehicles. The observation vehicle fleet then estimates a scalar field of the target area. During the process, the algorithm tries to find optimal vehicular paths subject to estimation uncertainty.

Pharpatara, Herisse and Bestaoui (2017) develop a hybrid, decentralised trajectory planning algorithm for nonholonomic aerial vehicles. A nonholonomic vehicle is a type of vehicle with one or more nonholonomic constraints, all of which are based on the derivatives of the vehicular postures (i.e. x- and y-coordinates with directions) and cannot form an integrated constraint of the vehicle configuration variables (Corke, 2011). Their algorithm combines potential fields with the framework of RRT* to increase the convergence speed of finding the first solutions as well as the optimal solutions of trajectory planning problems.

In the literature, the three conventional approaches to path and trajectory planning tend to be applied to the environment where a single agent and static obstacles exist. Although there also exist previous studies able to implement fleet trajectory planning, they are mostly subject to decentralised planning, instead of centralised planning employed for container AGVs.

2.3.3. Curve-based Approaches

Two types of curves are mainly utilised in path and trajectory planning: Bézier curves and Dubins curves. A Bézier curve is a parametric curve without the inclusion of a straight section and defined by ordered reference points (Marsh, 2005). A Dubins curve (or path) is the shortest path drawn with the minimum steering radius of a target vehicle and can contain a straight section (Tsourdos, White and Shanmugavel, 2011).

In the following literature survey, two tendencies are observed: Firstly, both curve types have mainly been utilised in single agent path and trajectory planning. Secondly, Bézier curves have been used for smoothing or avoiding swift turns whereas Dubins curves have worked for drawing the shortest paths without amending the basic structure of the curves.

Jolly, Sreerama Kumar and Vijayakumar (2009) provide a trajectory planning model for a single robot that operates in a multi-agent football system. There exists only one controllable robot acting like a football player that cooperates with its teammates to defeat its opponents. So, the controllable robot is exposed to potential collisions against the opponent robots as well as its teammates. When the target robot detects collision events on its planned path, it decelerates and regenerates a path subject to a Bézier curve to avoid the expected threats. The model evaluates the quality of a path regarding its expected travel time.

Škrjanc and Klančar (2010) suggest a fleet trajectory planning model based on Bézier curves for nonholonomic robots. They focus on the fact that some collisions between robots cannot be avoided by changing a single trajectory. Therefore, their model updates multiple trajectories while minimising the total path length and considering a predefined safety distance.

Sahingoz (2014) proposes a fleet trajectory planning model for UAVs. In their scenario, UAVs share a central depot where they should return after their operation. They travel at a constant speed rate, and each needs to pass over near its designated operation areas. The model firstly creates a path per vehicle by connecting its assigned mission points with straight lines. The algorithm then converts sudden turns near the connected mission points into smooth curves by using Bézier curves.

Savla, Frazzoli and Bullo (2008) employ Dubins curves in the TSP (i.e. DTSP), not using the Euclidean distance (i.e. ETSP) only. Given a problem instance that includes a set of must-visit points, their algorithm solves the corresponding ETSP as the first step in order to approximate the optimal solution to the corresponding DTSP. Then, it adjusts the waypoint areas of the ETSP-based route to draw followable curves subject to Dubins curves.

Ding, Rahmani and Egerstedt (2010) perform fleet trajectory planning for UAVs that are employed to escort a group of ground vehicles (GV). In their model, GVs can stay or move along a straight path, and each should be in sight of at least one UAV. UAVs are equipped with onboard sensors that have a limited detection range. They cooperate to maximise their monitoring period to the GV fleet under some critical constraints, such as a fixed altitude for the UAVs' flights and a constant speed rate for the aerial vehicles.

Shanmugavel *et al.* (2010) develop a fleet trajectory planning model for UAVs that need to arrive at their destination at the same time. Their approach consists of three phases. The first phase generates followable paths for UAVs based on Dubins curves and parametric arcs. In the second phase, the model achieves collision-free journeys by using a minimum separation distance and additional waypoints. Since the vehicles travel at a constant speed rate, the model does not allow the planned journeys to intersect at their equal lengths. Therefore, the third phase is to adjust the generated path lengths to be equal to that of the longest route.

De Filippis, Guglieri and Quagliotti (2011) propose a trajectory planning model for use in the environment where ground obstacles can affect the journeys of a UAV. Their model has three

steps. Firstly, it generates a risk map by using available orographic data and the relative altitudes of a given UAV from ground obstacles. Secondly, it transforms the map into a graph with nodes and edges and utilises a graph-based shortest path finding algorithm while minimising the degree of collision threats. Thirdly, the proposed model smooths each turning area by adopting Dubins curves.

Yang, Li and Sun (2013) consider a specific type of domain where there exists only one circular obstacle. They also utilise Dubins curves for path planning and prove that the shortest path consists of no more than five segments, each of which can be a curve or a straight line. Askari *et al.* (2016) use Bézier curves and Dubins curves together. Their approach finds the shortest path based on Dubins curves and then generates a Bézier curve based on the generated path. Wehbe, Bazzi and Shammas (2017) implement three-dimensional trajectory planning by separately generating horizontal Dubins curves and vertical ones.

2.3.4. Mixed Integer Linear Programming Based Models

Linear programming (LP) is a process of modelling a mathematical model expressed by linear functions and finding its optimal objective value, and mixed integer linear programming (MILP) is a branch of LP which contains integer decision variables. An MILP problem instance consists of an objective function and constraints formulated by decision variables and parameters. It is designed to find the optimal values of the decision variables that minimise (or maximise) the objective function value while satisfying the constraints (Di Natale *et al.*, 2010).

MILP-based models for path and trajectory planning have been developed for various vehicle types, such as aerial vehicles (Pallottino, Feron and Bicchi, 2002; Richards *et al.*, 2002; Cetin, Bikdash and Hadaegh, 2007; Chen, Han and Zhao, 2012; Habib, Jamal and Khan, 2013; Wang and Ge, 2014), underwater vehicles (Yilmaz *et al.*, 2008) and rescue vehicles (Berger *et al.*, 2012).

In the literature, MILP-based path and trajectory planning shows two remarkable issues. Firstly, they usually handle a few vehicles (i.e. a small vehicle fleet) due to inherent time complexity in MILP. Secondly, they tend to utilise basic constraints, such as the maximum acceleration and speed rates of vehicles, without considering more sophisticated features including the difference between the maximum linear and turning speed rates.

Pallottino, Feron and Bicchi (2002) propose two manoeuvre methods in multi-aircraft trajectory planning. The first method changes the velocity rates of aerial vehicles, and the second allows them to alter their heading angle once. The first cannot provide safe trajectories in head-to-head journey cases; therefore, the use of the method is limited. They do not suggest a combined model of the methods due to following non-linearity.

Richards *et al.* (2002) consider the thruster plumes of spacecraft which consist of gas particles with high energy. Since a spacecraft's plume negatively influences other spacecraft flying in the emission region, all spacecraft need to be separated between them. Earl and D'Andrea (2005) handle high time complexity in MILP-based trajectory planning models that include many controllable vehicles. They apply iterative schemes to generate collision-free trajectories.

Cetin, Bikdash and Hadaegh (2007) suggest a trajectory planning algorithm for a spacecraft fleet to minimise their fuel consumption increased by vehicular acceleration and deceleration changes. The model represents each spacecraft as a mass point and bounds each with a cube-shaped safety region that is constant in size.

Berger *et al.* (2012) introduce a threat-level idea to MILP-based path planning for a rescue vehicle that operates in unknown areas. They define threat zones according to threat levels and provide opportunities for a vehicle to pass through the adversarial zones while considering a vehicular survivability threshold. In addition to environmental threat exposure to the vehicle, its travel distance is considered as the objective.

Chen, Han and Zhao (2012) develop a trajectory planning algorithm for use in uncertain environments. Like the research by Richards *et al.* (2002) and Cetin, Bikdash and Hadaegh (2007), they represent an aerial vehicle as a particle. The vehicle is capable of detecting static and moving obstacles within its detection range to avoid collisions against them.

Habib, Jamal and Khan (2013) perform fleet path planning in the open multiple depot vehicle routing problem in which UAVs start their journey at their location origin, visit pick-up points, and arrive at their delivery location. They consider the situation where some UAVs in a fleet are damaged during the operations. When any vehicles in the fleet detect the damaged members, they share the information with the other undamaged members. Their model then re-generates paths that cover the unvisited areas of the damaged UAVs while implementing time separation for collision avoidance.

Wang and Ge (2014) utilise general flight rules in the UK, the US and China for planning two UAVs' trajectories. The rules specify the right-of-way of two aerial vehicles for collision avoidance. In their scenario, the vehicle with the higher right-of-way keeps its trajectory whereas the other needs to increase its altitude or turn to the right. Such trajectory adjustments are intended to cause the minimisation of the travelled distance and the fuel consumption. Both vehicles are represented as mass points and separated by a predetermined safety distance.

2.3.5. Dynamics-based Approaches

Although vehicular posture lists are enough to express graphical paths and trajectories, vehicle dynamics can be considered as constraints to suggest more followable paths and trajectories. Many researchers and practitioners have taken angular velocities into account; as a result, their approaches and models include control of the yaw rate, the roll rate or the lateral rate. However, such angular velocity control has been considered more in path following and trajectory tracking than in path and trajectory planning. This is because path following and trajectory

tracking cover relatively short vehicle-operation periods and therefore are capable of considering more control parameters.

Lauffenburger *et al.* (2003) focus on the lateral control of road vehicles in the framework of a driver-assistance system. Such systems are designed to minimise the interactions between drivers and vehicles to achieve a high level of driving safety and comfort. Their controller is capable of detecting and warning dangerous situations in vehicular speed and trajectories. Also, it adjusts planned paths when vehicles' predicted trajectories significantly deviate from their reference path.

Song, Cao and Huang (2013) consider the yaw rate and the lateral rate in decentralised trajectory planning for road vehicles. Their approach generates artificial potentials in the front and rear sides of obstacle vehicles, named guide potentials. It uses such forces and road borders to create a risk map for a host vehicle. The approach then utilises the risk map to determine the host vehicle's trajectories in various situations, such as lane-changes and turnings.

Li *et al.* (2017) integrate trajectory planning and tracking for automated ground vehicles. Their integrated controller calculates vehicular trajectories to be used for short periods of time based on a state-sampling method. Given a reference path for a target ground vehicle, the controller examines potential terminal states of the vehicle subject to a predefined lateral offset. The controller then connects the initial vehicular state to all terminal states to generate candidate paths. Among these paths, it selects a safe path by using a defined cost function. In the process of velocity planning and trajectory tracking, it considers the maximum lateral acceleration to achieve yaw stability and trajectory safety.

2.3.6. Metaheuristic Based Algorithms

Metaheuristics are experienced-based iterative procedures designed to find satisfactory or near-optimal solutions under limited time and computation capacity conditions. Although they

do not guarantee solution optimality, these approaches are attractive when there is a possibility of non-metaheuristic techniques failing to discover solutions in a limited period of time.

They are designed (1) to find a better solution than the previous best solution in each iteration, (2) to allow them to search solution space more although they have failed to find better solutions and (3) to terminate if they find an acceptable solution or have experienced sufficient iterations. Over the past few decades, many types of metaheuristics have been suggested, such as genetic algorithms (GAs), particle swarm optimisation (PSO), ant colony optimisation (ACO), tabu search (TS) and simulated annealing (SA).

Various metaheuristics have been applied to path and trajectory planning, for example, PSO (Deepak and Parhi, 2012; Zhang, Gong and Zhang, 2013; Deepak, Parhi and Raju, 2014), ACO (Gigras and Gupta, 2012; Chen *et al.*, 2013), TA (Khaksar *et al.*, 2012) and SA (Miao and Tian, 2013; Wei, 2013). However, a significant number of studies have adopted GAs for the decision making problem. Some studies are subject to application settings; however, most do not.

The basic framework of GAs was proposed by John Holland in 1975, and a GA is an approach based on the idea of *Survival of the fittest* (Sivanandam and Deepa, 2008). The algorithm imitates the evolutionary procedure in which individuals in a population evolve in the next generation, and the process continues. It employs a fitness function to evaluate each individual that refers to a chromosome in nature that consists of multiple genes.

The algorithm improves individuals by crossover and mutation. Crossover is a process of producing offspring by combining two parental chromosomes whereas mutation creates a new chromosome by changing genes in its original chromosome. More specifically, the latter helps to maintain the diversity of populations and to prevent premature convergence in which individuals become similar in the early stage of the evolutionary procedure. New chromosomes generated by crossover or mutation can replace their parents to form a new

population to be used in the next generation. The algorithm terminates if it has discovered satisfactory chromosomes (i.e. solutions) or has experienced enough iterations.

Previous studies show three trends in metaheuristic-based path and trajectory planning. Firstly, GAs are dominant compared to different metaheuristics, such as SA, TA and PSO. Secondly, a small-sized vehicle fleet is usually handled, especially from one to three vehicles. Thirdly, basic vehicular constraints are usually taken into account to simplify models and algorithms.

Several previous studies handle a single agent that operates in a domain with randomly-scattered obstacles. Nearchou (1998) converts a given environment with P&D stations into a visibility graph and utilises a GA framework for path planning. Al-Taharwa, Sheta and Al-Weshah (2008) implement path planning for a single robot that is capable of operating in various environments with static obstacles. They convert a domain into a grid system, and therefore a robot's moving directions are only right, left, up and down. Regardless of the complexity of the obstacle layout of a domain, their suggested algorithm can find a collision-free path while trying to minimise the length of the path.

Chiu (2010) attempts to reveal the validity of GA-based path planning in multi-layer barrier environments. Liu, Liu and Yang (2011) suggest an adaptive GA-based path planning model for use in static environments. They utilise reactive crossover and mutation operators according to maximum and average fitness values to prevent premature convergence. Zhou and Jiang (2012) perform knowledge-based path planning in the environment where obstacles emerge and disappear.

Tuncer and Yildirim (2012) devise a new mutation operator to avoid premature convergence. The method memorises all the possible neighbour points of a selected mutation position and calculates the fitness values of the neighbour nodes to choose the best mutation point. Their simulation tests showed the superiority of the proposed method regarding the frequency of finding optimal paths, compared to random mutation and other mutation methods suggested by Li *et al.* (2006).

Shiltagh and Jalal (2013) employ a different population size according to environmental characteristics, such as the number of obstacles, in single agent path planning. They also suggest an engine to identify infeasible paths that cause collisions against static obstacles. The engine also deletes such infeasible solutions from populations in order to avoid unnecessary computation work. Zhu *et al.* (2015) consider path smoothness as an objective in path planning. They define the smoothness of a path as the less sum of the involved turning angles.

Research has also been performed on path and trajectory planning for a vehicle fleet while considering vehicular conflicts. Cai and Peng (2002) number the vertices of polygonal pickup stations. Each vertex has goods to be delivered, and there exists only one destination. Cargo transporters should visit as many intermediate vertices as possible while minimise the total travel distance.

Conde *et al.* (2012) suggest a fleet trajectory planning model for cooperative UAVs. They assume aerial vehicles to move along their planned route while travelling at a constant speed rate. When they detect possible collisions on their ongoing trajectory, the model tries to improve the trajectory set based on path length and collision penalty.

Qu, Xing and Alexander (2013) present a centralised AGV fleet trajectory planning model based on coevolutionary genetic algorithms. Their model consists of two parts. The lower part improves each AGV's candidate paths without considering the other AGVs and selects a few elite paths from each. The upper part identifies vehicular conflicts in possible trajectory sets generated by combining elite paths They evaluate the quality of paths by path length and smoothness.

2.4. Integration of Vehicle Dispatching and Fleet Trajectory Planning

Although significant research has been performed on AGV dispatching and fleet trajectory planning separately, the integration of the two research areas has relatively been neglected so

far. Also, studies involving both are mostly based on predetermined path systems instead of the free-ranging vehicle setting concerned with this thesis.

Huang and Chung (2005) attempt to integrate the two vehicle operation problems for use in pipe-less batch plants. Their research a couple of different system layouts, each of which can be expressed as a graph by connecting workstations. Corréa, Langevin and Rousseau (2007) focus on the integrated vehicle operation setting for use in flexible manufacturing systems with bidirectional flow paths. When providing a dispatching plan, their model employs the shortest path rule to minimise the total delay of delivery tasks. As the generated plan does not consider vehicular conflicts, the model calculates one collision-free travel plan without any specific objectives, such as the minimisation of the total travel time of AGVs.

Ghasemzadeh, Behrangi and Azgomi (2009) devise an integrated vehicle operation model for use in mesh-like path systems. They transform mesh-like systems into lattice-shaped maps by representing each intersection as a vertex. For simplicity, they decompose the integrated problem and apply a constant speed rate to vehicles.

Corman *et al.* (2016) propose a hybrid vehicle operation model for use in potential ACTs where free-ranging AGVs operate (i.e. the free-ranging vehicle setting). They decompose the integrated vehicle operation problem into a discrete problem (i.e. AGV dispatching) and a constant decision-making problem (i.e. AGV fleet trajectory planning). Their dispatching model mainly focuses on determining the delivery sequences of containers while ignoring possible task reassignment. They represent fleet trajectory planning as the determination of the acceleration rates of AGVs with the consideration of the maximum speed rate and collision avoidance.

2.5. Deadlocks in AGV Operations

A deadlock refers to a situation in which a system stalls because multiple processes are waiting for resources other processes hold. Deadlocks arise if the following four conditions coincide:

- Mutual exclusion condition: Each resource cannot be shared by multiple processes.
- Wait for condition: A process holding resources is waiting for other resources being possessed by different processes.
- No preemption condition: Each resource held by a process is released after the process terminates.
- Circular wait condition: Where there exist N processes, $P = \{P_1, P_2, P_3 \dots P_N\}$, P_1 is requesting a resource possessed by P_2, and P_2 is requesting for a resource in P_3 and so on until P_N is requesting for a resource possesses by P_1 (Coffman, Elphick and Shoshani, 1971).

There exist four approaches to handling deadlocks: (1) ignorance, (2) prevention, (3) avoidance and (4) detection and resolution (Chung, Yoon and Maeng, 1994). The first approach ignores negligible deadlocks that rarely affect system performance. The second approach prevents the occurrence of one of the four deadlock conditions. On the other hand, the third approach utilises resource and process information to avoid unsafe states that can cause deadlocks. The fourth allows control software to get into deadlock states. Once diagnosed, the controller attempts to resolve the detected deadlocks through a range of methods.

Considerable research has been performed on deadlock handling for AGV travels in the last decade. Wu and Zeng (2002) focus on preventing circular waits in automated manufacturing systems with unidirectional flow paths. Their model examines system states and identifies assignable empty zones to AGVs for deadlock-free vehicle operations. Moorthy *et al.* (2003) present a deadlock detection and resolution algorithm applicable in ACTs where dozens of AGVs operate. Before the algorithm allocates AGVs their next zone, it examines a couple of the following zone steps per vehicle in order to predict upcoming cyclic deadlock situations. Under a deadlock situation, the algorithm sends an AGV trapped in the deadlock to a zone connected to a common node that does not cause another deadlock state.

Yoo *et al.* (2005) devise a simple deadlock avoidance algorithm based on a graph-theoretic deadlock detection method. Their model can be used in real-time control environments, such as flexible manufacturing systems, because of the simplicity of the model. Kim, Jeon and Ryu (2006) propose a method to detect and prevent deadlock situations for AGVs in ACTs. In their model, AGVs possess their reserved zones in advance to prevent deadlocks.

Park, Kim and Lee (2009) suggest a deadlock prevention method focusing on the circular wait condition. They define special interest areas where circular waits are likely to occur. The model allocates each interest area a control node that possesses a limited number of tokens. When a vehicle enters a special area, it should receive a token that is returned to the connected control node when moving out.

AGV dispatching is also exposed to operational deadlocks (Lehmann, Grunow and Günther, 2006). Such deadlocks result from inappropriate interactions between AGVs and other container handling resources, such as quay cranes. For instance, a deadlock happens when a loaded AGV is waiting to be served by a quay crane attempting to place a container on an empty AGV.

Lehmann, Grunow and Günther (2006) focus on dealing with deadlocks between AGVs and cranes. Firstly, they suggest two deadlock detection methods based on matrices and graphs respectively. Since the former method requires long computation time, the latter is more attractive in the environment where many container handling resources operate. Secondly, they present a deadlock resolution algorithm that adjusts stacking cranes' schedules or vehicle dispatching plans according to the state of the system.

Singh, Sarngadharan and Pal (2011) suggest a deadlock handling model for AGV dispatching in the machine shop environment where multiple AGVs work with a single loading station and multiple delivery points on predetermined guide paths. The model avoids expected deadlock states by preventing AGVs from travelling on shared paths without permission. Gudelj, Kezić and Vidačić (2012) propose an AGV dispatching algorithm in consideration of

deadlocks. The algorithm is based on the structural analysis of the state of a given system. It tries to minimise the makespan of AGV schedules by decreasing the unloading and travelling time of containers.

Zheng *et al.* (2013) develop a regional control model to resolve possible deadlocks in AGV dispatching. They assign a local controller to each manufacturing sector. When a deadlock is detected, the relevant local controllers cooperate in finding the most undisturbed path to be allocated to the nearest AGV from the path.

Previous research on deadlock handling in fleet trajectory planning tends to be independently performed, not being a sub-part of trajectory planning. The reason is that deadlocks in fleet trajectory planning are complicated to solve, especially when many objects are involved. The previous work is subject to predetermined path networks, and the fleet trajectory planning models introduced in *Section* 2.3 do not deal with deadlock situations.

On the other hand, some vehicle dispatching models include deadlock handling functions since such system stalling situations in vehicle dispatching are relatively easily solved than those in fleet trajectory planning.

2.6. Simulation Packages for Container Terminal Operation Analysis

Container terminals are complex and include various types of container handling equipment. This feature renders mathematical analysis unadoptable to researchers in the field of terminal automation. Instead, many terminal operators have been utilising simulation to observe potential performance changes by different terminal layouts or new technologies. Henesey (2004) shows a list of used simulation tools (or packages) by port operators (*Table 2-2*).

Several operators utilised commercial packages, such as Arena and Automod. These tools are user-friendly because they provide predefined functional components. So, users can visually design new models and approaches without advanced programming knowledge. Such components are also generic, therefore being able to cover a wide analysis scope and various

application settings, such as manufacturing systems and warehouses. In other words, commercial simulation models are not specific to container terminals. In the literature, Merkuryev, Kamperman and Visipkov (2000) and Cortés *et al.* (2007) employ Arena, and Lee and Cho (2007) use Automod.

Table 2-2. Used tools by container terminal operators. Source: Henesey (2004)

Port	Terminal type	Used tool
La Spezia	Containers	Modsim III
Riga	Containers	Arena
Genoa	Containers	C/C++
Genoa	Ferry	Arena
Durban	Containers	ITE
Zeebrugge	Bulk	Automod
Savannah	Complete	Modsim/Java
Multedo	Oil	Java
Voltri	Containers	Arena & C
Torres	Bulk	C/C++
Savona	Containers	C/C++

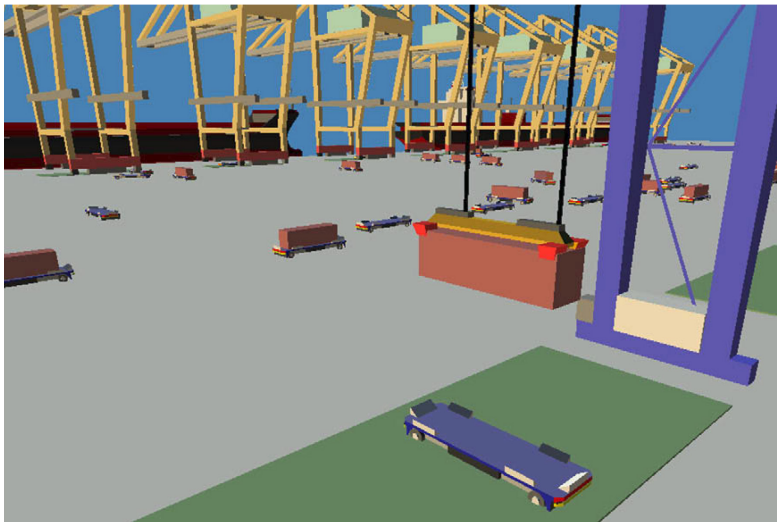


Figure 2-8. Screenshot on Limen. Source: Angeloudis and Bell (2010)

On the other hand, bespoke tools have also been utilised. Such in-house simulation models are developed in conventional programming languages, such as C/C++, C# and Java. Although developing simulation models is challenging and time-consuming, user can suggest detailed modelling requirements during the process of development. It also specifies target analysis

environments and extracts experimental results subject to user-defined performance indicators. *Figure 2-8* shows a screenshot of the model developed by Angeloudis and Bell (2010) coded in C#; it is named Limen.

2.7. Knowledge Gap

Free-ranging AGVs have the following operational features and requirements that should be considered in potential ACTs:

- By changing the container handling sequences of AGVs, the vehicles could achieve less travel for container pickup. While considering task sequencing impacts, such changes need to include ongoing tasks when designated AGVs have yet to receive the containers of the tasks.
- The free-ranging vehicle setting is free of predetermined flow-paths. Thus, line-blocking issues in dispatching do not need to be considered directly although congested environments would have similar issues. Vehicular trajectories need to be designed subject to the Euclidean distance, which is the minimum length between two points, instead of the Manhattan distance.
- Unlike AVs, AGVs' journeys are centrally planned in compact, known domains (i.e. AGV transport areas in container terminals). They also tend to operate as a large fleet in a cooperative manner. So, deadlocks would happen, and central vehicle operation planners need to have deadlock handling engines.
- The maximum linear speed of AGVs is higher than the maximum turning speed. Therefore, planned trajectories need to include as less curves as possible.
- AGV dispatching and fleet trajectory planning closely interact. So, integrated operation planners for the two decision making problems would be required.

So far, a considerable amount of research has been performed in two vehicle operational problems, dispatching and trajectory planning. However, considering all the features and

requirements together has been ignored. Especially, only few previous studies allow delivery vehicles to cancel their ongoing missions; however, they are subject to predetermined path systems and assign a single task per vehicle (i.e. a limited degree of reassignment).

From the viewpoint of fleet trajectory planning, there is a trade-off relationship between the level of trajectory detail and the size of a target vehicle fleet. The trajectories of a vehicle fleet also tend to be undetailed and simple especially when they are centrally planned. Although there exist many trajectory planners for detailed trajectories, they are subject to onboard planners for use in unknown environments. The difference between the maximum linear and turning speed of vehicles has also been ignored, and deadlock handling has separately been considered.

The knowledge gap initiated the doctoral research on implementing AGV dispatching and fleet trajectory planning while handling the operational requirements summarised above. Due to the complexity of resulting AGV control, commercial simulation packages are less appropriate than customised tools. Thus, the thesis also involves developing a simulation tool to analyse the operations of free-ranging AGVs in potential ACTs.

Chapter 3. Research Methodology

This chapter describes research design by showing a process diagram that defines the flow of this study and identifies the relationships between performed research activities. It also explains several ideas and existing techniques considered in the process of model development during the study.

3.1. Research Design

In order to achieve the aim of this research explained in *Chapter 1*, research objectives were carefully designed and performed. To begin with, this research includes a review of container terminal automation to identify the current state of container handling automation in the quay, the yard side and the transport area. Also, the review covers vehicle dispatching and trajectory planning in a range of application areas in consideration of simulation packages and models for vehicle operation analysis. Consequently, *Chapter 2* identifies key limitations of previous work and current ACTs and research methodology was carefully constructed to handle the identified limitations (*Chapter 3*).

The methodology includes three development activities for the following models and algorithms: (1) a flexible AGV dispatcher, (2) a free-ranging AGV fleet trajectory planner and (3) a simulation model for integrated AGV operation planning (i.e. vehicle dispatching and fleet trajectory planning). Firstly, vehicle dispatching with task reassignment and task-sequencing is mathematically formulated in consideration of key issues in vehicle dispatching (*Section* 3.4). This results in mixed integer programming (MIP) in which some decision variables are subject to integers. Since branch-and-cut, explained in *Section* 3.2, is a common approach to solving MIP, it is applied to the dispatching model first. However, due to the time complexity of the model with the approach, this study adopts the framework of SA and suggests a vehicle dispatching algorithm as a result (*Chapter 4*).

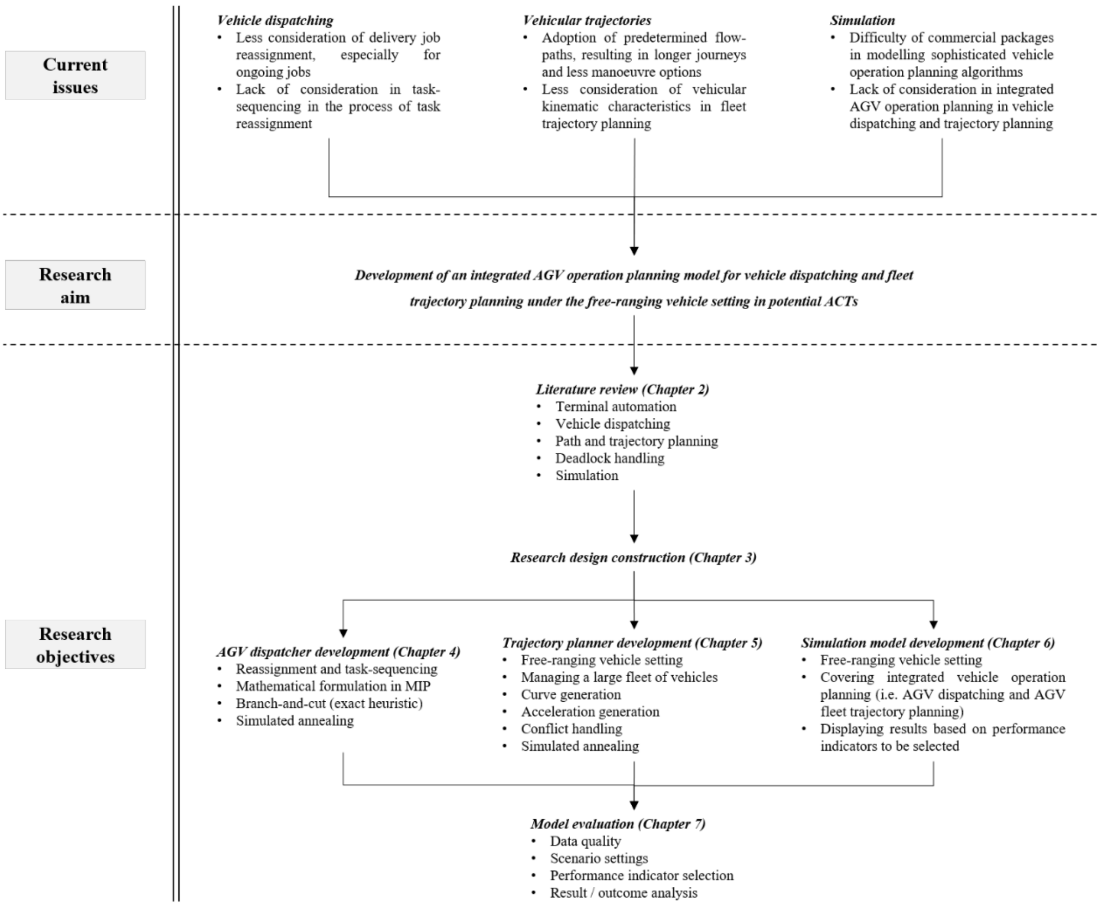


Figure 3-1. Research flow diagram.

The second development activity was designed to provide a fleet trajectory planning method for use in the free-ranging vehicle setting. Similar to the first development activity, mathematical formulation would have been performed; however, the suggested method is based on the core idea of Dubins curves without requiring mathematical curve functions. In other words, it does not work based on vehicles' reachable areas (*Section* 3.5). Therefore, the development activity separates curve generation with acceleration generation without complicated mathematical expressions.

Since there exist potential collision and deadlock threats in vehicle fleet operation, handling such adverse situations is considered in the development process (*Chapter 5*). In addition, this research employs SA in the trajectory generation method to cope with the time complexity of fleet trajectory planning. The framework of GAs is also applied and tested to compare the two metaheuristics for AGV fleet trajectory planning in calculation time and vehicular travel.

The development of a simulation model comes as the third development activity. Since mathematical analysis and direct applications to real terminals are not appropriate to evaluate sophisticated vehicle operation algorithms, including the suggested ones, this research adopts simulation (*Section* 3.7). The simulation model provides experimental environments, mainly focusing on the transport area in ACTs. It also covers container handling operations in the quay and the yard to simulate more realistic AGV operations (*Chapter 6*).

This research includes a series of simulation experiments to evaluate resulting AGV operations by the developed models (*Chapter 7*). Since real terminal data still have been difficult to be directly obtained because of safety and security issues, previous research is utilised to obtain or calculate scenarios and parameters. A selected simulation domain represents ACTs while sharing the core layout of terminal resources (i.e. cranes and container stacks). Based on key performance indicators and various scenarios, the suggested AGV operation planning model is rigorously evaluated in comparison to conventional dispatching methods and path design subject to the Manhattan distance.

3.2. Branch-and-cut

This approach guarantees solution optimality by computing solutions as follows: Given an MIP minimisation problem, it firstly solves the problem without being constrained to the integer condition. Then, it adopts additional constraints (i.e. cutting planes) that make all feasible integer sets valid while being violated by the found partial solution. The process stops when it cannot discover more cutting planes or has found a valid integer solution. In the former case, one of the partial solutions, such as the one with the lowest objective value, can be used as the lower bound of the problem and each of the integer solutions as an upper bound. A node (i.e. branch start point) is defined as a partially determined decision variable set, and it is pruned when the problem's current upper bound is lower than the node's lower bound. When the algorithm prunes all branches, the current upper bound becomes the optimal solution.

3.3. Metaheuristics

Metaheuristics are a customised heuristic framework to solve a specific type of problem (Hillier and Lieberman, 2005). Especially, they are widely used to solve combinatorial optimisation problems with numerous decision variables and possible combinations. They suggest good enough solutions that are not necessarily optimal. As explained in *Section* 2.3.6, they have an iteration engine with the ability to escape from local optimal solutions.

Metaheuristics fall into two categories: trajectory-based and population-based approaches (Hussein *et al.*, 2012). Trajectory-based approaches, represented by TS and SA, try to improve an objective function value by searching for a neighbour solution iteratively from a randomly generated initial solution. On the other hand, population-based approaches iteratively enhance a solution by using a set of candidate solutions (i.e. population); GAs represent the second category.

TS is a solution search procedure that uses a tabu list to generate neighbour solutions; the list refers to a set of partial solutions that rarely construct optimal or near-optimal solutions causing significantly degraded solution quality. The metaheuristic avoids utilising such inferior sub-solutions to efficiently find competitive neighbour solutions.

SA is one of the most common metaheuristics which imitates an annealing process in metallurgy to increase the stability of materials by a cooling schedule (Kirkpatrick, Gelatt and Vecchi, 1983). Similarly, the metaheuristic advances solutions while lowering a temperature. To be more specific, in each iteration, it creates several neighbour solutions of its current reference solution; the neighbours share most parts of the solution structures with the reference. Where the best neighbour is superior to the reference solution, the algorithm adopts it as the reference in the next iteration. Otherwise, it utilises a probability function to determine whether it updates its reference solution. When the temperature is relatively high, it is less repulsive to worse solutions. Since the temperature gradually decreases, the probability of accepting a worse solution as a reference also gradually decreases (i.e. stabilisation).

A GA is an experience-based heuristic approach that imitates the evolutionary procedure in nature. Each generation has a population which consists of individuals (i.e. solutions), and each individual refers to a chromosome composed of multiple genes. The individuals experience structural changes and improvements by crossover or mutation through the evolutionary procedure. The crossover is a technique to produce offspring by using parent individuals whereas the mutation provides a new individual by changing one or multiple genes in a chromosome. Such new chromosomes form a new population. When the algorithm has found a satisfactory chromosome or reached a predefined maximum generation number, it terminates.

3.4. Key Issues in AGV Dispatching

AGV dispatching models and algorithms have various aspects. These aspects can also be considered in the process of developing AGV dispatching ideas and strategies. This section explains the following six aspects in detail: planning length, operational environments, workload balancing, objectives, task re-assignability and load types.

3.4.1. Planning Length

Three approaches to AGV dispatching exist, categorised by planning length: short-, mid-, and long-term planning (or dispatching) approaches. Short-term dispatching models assign a single delivery job to each vehicle. Some give multiple vehicles jobs at a time, and the others provide a job for one vehicle per decision-making step. One of the most popular methods is the nearest work centre dispatching rule, which sends a vehicle that has finished its mission to the closest work centre having a pending task. Long-term dispatching allocates several or many delivery tasks to each vehicle. On the other hand, mid-term dispatching assigns only a few tasks to each vehicle. For instance, Angeloudis and Bell (2010) let each AGV receive two container transport tasks at most.

Each approach has its own strengths and weaknesses. Long-term dispatching is theoratically capable of offering better solutions in vehicluar travel distance and time, compared to short-term planning. This is because it includes task-seqeuncing and therefore can easily minimise the setup travel of vehicles. However, vehicle dispatch plans are likely not to be performed as scheduled due to environmental and operational uncertainty. Thus, such assignments need to be amended to improve vehicle work orders. In addition, long-term dispatching potentially requires long computation time due to long planning horizons, whereas short-term planning is able to frequently and quickly provides solutions.

The mid-term approach has the merits of both to a certain extent. It can consider the influence of task orders on AGVs and calculate a new dispatch plan relatively quickly to react environmental changes. As an advanced technique, Angeloudis and Bell (2010) combine their mid-term dispatching model with rolling horizon decision-making to cope with uncertainty in container terminals. Such integrated methods assign a few cargo transport tasks (i.e. two missions) to each AGV and periodically generate new plans that are positioned alongside ongoing dispatch schedules.

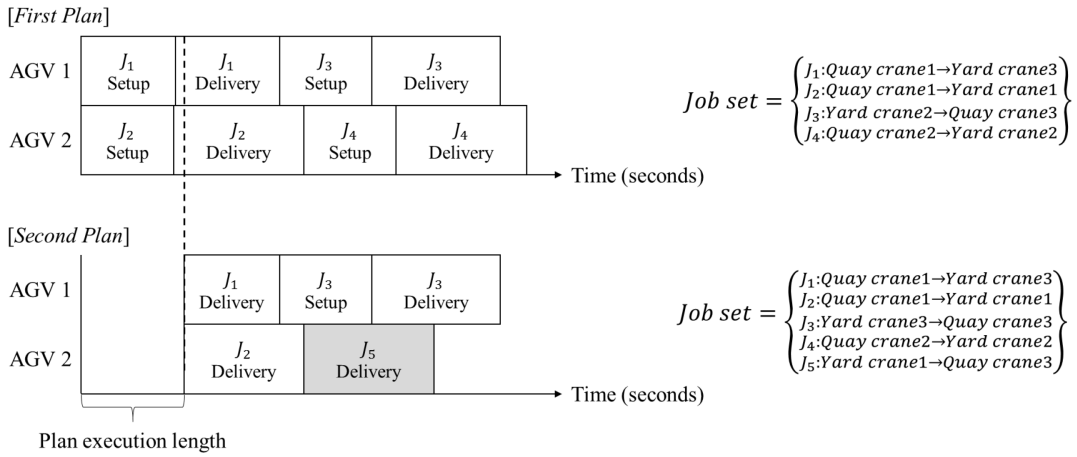


Figure 3-2. Mid-term dispatching with a rolling horizon.

Figure 3-2 illustrates a series of AGV dispatch plans based on mid-term dispatching and rolling horizon decision-making. In the first plan, there were four tasks in the job set. AGV 1 was assigned to tasks $J1$ and $J3$, while AGV 2 was assigned with $J2$ and $J4$. After a planned

execution time conditions in the terminal change, and a new job (i.e. $J5$) is introduced into the job list. Since its origin is the destination of $J2$, no setup travel occurs in the order $J2$ -$J5$. In the second plan, the AGV dispatcher has cancelled the assignment from AGV 2 to $J4$, and assigned the new mission to the vehicle.

As seen in the example (*Figure 3-2*), the integrated approach can consider the impact of such sequences on delivery vehicles. Also, although there is a degree of uncertainty in an operation environment, it can adjust an AGV dispatch schedule periodically to improve vehicle operations.

3.4.2. Operational Environment

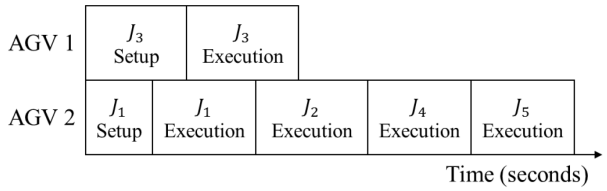
A range of AGV dispatching models have been developed, and some have been designed for specific AGV configurations. For instance, the FEFS rule was initially deployed for AGV systems with a single unidirectional loop. On such configurations, AGVs travel on a circular path on which cargo handling stations are located. Where a station requires a vehicle to send a cargo unit to another station, the first vehicle that is expected to encounter the origin station takes the delivery mission. This scheme cannot be used in any other environment, such as domains with cross-lanes.

However, there also exists environment-independent dispatching, such as the STT/D rule. When an AGV finishes its current task and becomes idle, the rule dispatches the vehicle to the closest work centre requiring a cargo delivery vehicle. This rule can work with all types of guide path-based systems, even the free-ranging vehicle setting.

In addition, vehicular journeys are closely related to operational environments, especially flow-path settings. These settings determine the shapes of paths and trajectoeis and resulting (or expected) travel time. The latter is key information in vehicle dispatching in order to predict vehicle arival and job finish time.

3.4.3. Workload Balancing

Workload balancing seeks to evenly assign tasks to resources while preventing uneven allocations (i.e. workload unbalance). It reduces the makespan of workloads and increases resource utilisation. Workload unbalance is likely to happen when long-term dispatching only attempts to minimise the total setup travel distance or time of delivery vehicles. To be more specific, if an AGV has a transfer job whose destination is the origin of another mission, the vehicle will take this as the second job for a setup-free operation. *Figure 3-3* illustrates an example of unbalanced workloads that possibly happen in the described setting.



$$Job\ set = \begin{Bmatrix} J_1{:}Quay\ crane1 \rightarrow Yard\ crane2 \\ J_2{:}Yard\ crane2 \rightarrow Quay\ crane2 \\ J_3{:}Yard\ crane3 \rightarrow Quay\ crane1 \\ J_4{:}Quay\ crane2 \rightarrow Yard\ crane3 \\ J_5{:}Yard\ crane3 \rightarrow Quay\ crane2 \end{Bmatrix}$$

Figure 3-3. Workload unbalance.

On the other hand, in other dispatching schemes, workload unbalancing rarely occurs. Short-term planning inherently prevents such adverse situations since each vehicle receives a new delivery task as soon as it finishes its current mission. In mid-term dispatching, these situations can be prevented by limiting the number of tasks assignable to each AGV.

3.4.4. Objective

An AGV dispatching model tries to optimise its objective function value while being constrained by conditions. The users of the model can define the function based on their requirements. For example, they can set the minimisation of the total vehicular late arrival time as their goal.

From the viewpoint of vehicle operation optimisation, reducing the total setup travel distance or time of AGVs can be the aim of vehicle dispatching. In this case, as mentioned in *Section* 3.4.3, there is a possibility of a few AGVs taking most given tasks in their operating system.

It is likely to lead to unbalanced resource utilisation and decreases in the throughput of cargo handling resources, such as quay cranes in container terminals.

In fact, the aim of an AGV dispatching model does not need to originate in vehicle systems. In other words, it can also come from different systems that interact with AGVs. In a container terminal, the AGV system is part of the whole terminal system and helps to optimise the flow of containers between the quay and the yard. Therefore, an available objective is to reduce the late arrival time of AGVs at quay and yard cranes. An objective function can also be designed in a way that incorporates the external and internal objectives while applying different weights to the selected objectives.

3.4.5. Task Re-assignability

The state of a transport task assignment can be either breakable or unbreakable depending on the designated vehicle's travel state. Where an AGV is travelling to the source of its current task (i.e. on a setup travel), the assignment can be cancelled, and the task can be re-allocated to another vehicle. In other words, the assignment state can be regarded to be breakable. Where a task assignment is determined to be unbreakable, the vehicle is assumed to be actively delivering a cargo and is required to complete the job.

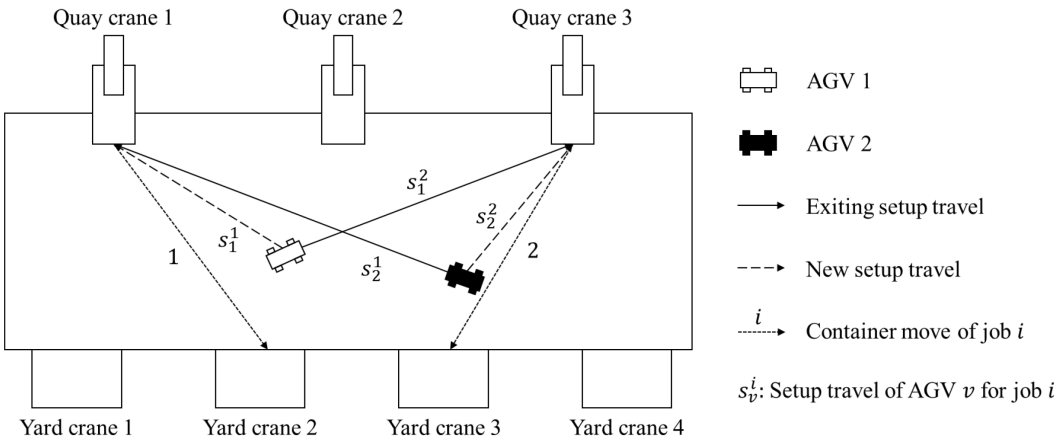


Figure 3-4. Setup travel improvements by reassignments.

Such assignment cancellations could improve AGV operations. As shown in *Figure 3-4*, AGV 1 was going to travel to the origin of job 2, and AGV 2 the source of job 1; both the vehicles needed to travel long distances. By breaking the plans and swapping the tasks, both can have less setup travel distances. The cranes each are also likely to have reduced waiting time.

3.4.6. Load Type

Industrial AGVs can accommodate two 20ft containers at a time (i.e. dual-load capability) as well as one 40ft container. Where an AGV needs to transport a 40ft container, there exists only one possible journey sequence; the vehicle moves from its current position to the task origin and then makes a trip to the destination. However, when an AGV has two 20ft container assignments, there exist six possible journey orders as depicted in *Figure 3-5*.

Assuming that the vehicle should finish one of the tasks before processing any further assignments, only the first and fourth cases in *Figure 3-5* are possible. Therefore, there may be potential optimality loss where the other options can achieve a better objective value.

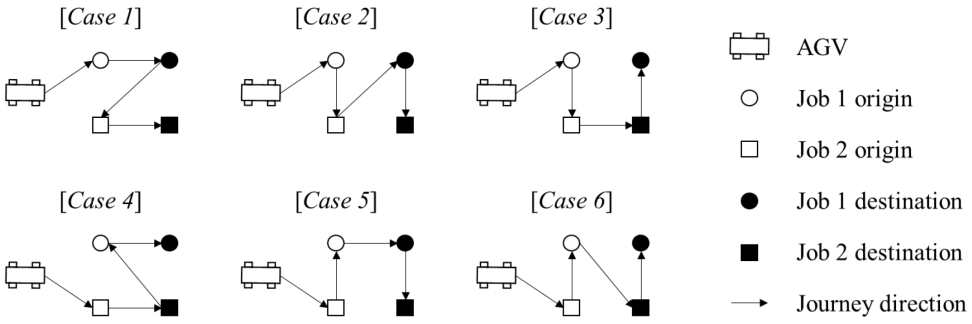


Figure 3-5. Six travel sequences with the dual-load capability.

However, in the research by Grunow et al. (2006), the positive effect of the dual-load capability is likely to decrease as the size of target container terminals increases. Essentially, an AGV's visit at a cargo handling station of a job may be unproductive for the other mission, resulting in its processing delay.

3.5. Strategies for Trajectory Planning

In two-dimensional space, a vehicular trajectory refers to a time-dependent list or a continuous function of time that provides a series of x- and y-coordinates with corresponding headings as additional data. Compared to paths, trajectories suggest more detailed travel information and help to detect potential collisions with other objects over time. They can be either complete or incomplete. Complete trajectories cover the entire journey from an origin to a destination whereas incomplete ones only include a portion of the trip and require further execution of the planning algorithm to complete the journey. Incomplete trajectories generally require shorter computation time, and therefore are preferable in busy settings with multiple interacting agents. Both types can also be used in conjunction with rolling horizons.

Two trajectory planning approaches exist: One is based on vehicular reachable areas, and the other is subject to planned paths. The area-based approach determines the next posture of a vehicle under environmental and kinematic constraints until the vehicle arrives at its destination. In each posture decision step, the approach defines the reachable area of the vehicle and then determines a safe posture considering selected performance indicators, such as path length. In fleet trajectory planning, the approach performs the process to all vehicles.

The second trajectory planning approach generates the path of an agent first and then decides acceleration rates applied on the route. This strategy can work with (1) pre-installed flow path systems (i.e. networks) or (2) a path generator that does not utilise any networks. In the former case, a path planning model for a graph determines a vehicular path. However, in the latter case, a curve generation model, such as Dubins curves, is utilised to create a path.

The first approach has two drawbacks for free-ranging agents. One is the difficulty of choosing the next posture of a vehicle in a reachable space. Any point in the area can be the next position, and the heading at the next position can also vary. Therefore, such next posture decision problems become significantly complicated even when the strategy considers a few vehicles.

The other drawback is that trajectories determined by the first approach contain sequences of angled turns (i.e. zigzags). In practice, most moving agents cannot precisely follow such trajectories that potentially result in significant trajectory tracking error. Also, target vehicles are not maximising their kinematic capability because top curve speed rates tend to be lower than top linear speed rates.

The following mathematical formulation is a simplified version of the models suggested by Xin, Negenborn and Lodewijks (2014) and Corman *et al.* (2016) for free-ranging AGV fleet trajectory planning based on the first approach. It can also be a basic formulation for the fleet trajectory planning problem (FTPP). It regards cargo delivery vehicles as mass points and guides them to their goal locations without collisions by controlling accelerations on the x-axis and the y-axis. The following constraint firstly derives from equations of motion where V is a set of vehicles, and T is a set of time points separated by an interval denoted by Δt.

$$\begin{bmatrix} p_i(k+1) \\ v_i(k+1) \end{bmatrix} = \begin{bmatrix} I_2 & \Delta t I_2 \\ O_2 & I_2 \end{bmatrix} \begin{bmatrix} p_i(k) \\ v_i(k) \end{bmatrix} + \begin{bmatrix} 0.5(\Delta t)^2 I_2 \\ \Delta t I_2 \end{bmatrix} a_i(k) \quad \forall\, i \in V, k \in T \qquad (3\text{-}1)$$

The control parameter $a_i(k)$ refers to the x- and y-acceleration rates of vehicle i at time k. The model represents the velocity and the position as $v_i(k)$ and $p_i(k)$ respectively. It also limits the acceleration and the velocity by (3-2) and (3-3) respectively where a_{max} is the maximum acceleration rate, and v_{max} the maximum velocity (Xin, Negenborn and Lodewijks, 2014).

$$\sqrt{a_i^x + a_i^y} \leq a_{max} \qquad \forall\, i \in V \qquad (3\text{-}2)$$

$$\sqrt{v_i^x + v_i^y} \leq v_{max} \qquad \forall\, i \in V \qquad (3\text{-}3)$$

Solving the two non-linear inequalities can lead to long computation time. As suggested by Richards and How (2002), this simplified model uses the following linear constraints (3-4) and (3-5) instead as the two reference studies. The parameter M describes a set of non-negative

integers, and m is an index used to denote each of the values. Larger M sets usually result in more accurate approximations (Xin, Negenborn and Lodewijks, 2014).

$$a_i^x \sin\left(\frac{2\pi m}{max(M)}\right) + a_i^y \cos\left(\frac{2\pi m}{max(M)}\right) = a_{max} \qquad \forall\, i \in V, m \in M \qquad (3\text{-}4)$$

$$v_i^x \sin\left(\frac{2\pi m}{max(M)}\right) + v_i^y \cos\left(\frac{2\pi m}{max(M)}\right) = v_{max} \qquad \forall\, i \in V \qquad (3\text{-}5)$$

The model should also prevent collisions between the agents. In the model, each vehicle has a square-shaped safety clearance zone at each time point. When the areas of the involved vehicles overlap, the vehicles would be unsafe. Let d be a safety distance applied onto the x-axis and the y-axis. Then, it equals half the width (and height) of the zones (Xin, Negenborn and Lodewijks, 2014).

$$p_i^x(k) \leq p_j^x(k) - 2d + Rc_{i,j,1}(k) \qquad \forall\, i,j \in V, k \in T \qquad (3\text{-}6)$$

$$p_i^x(k) \geq p_j^x(k) + 2d - Rc_{i,j,2}(k) \qquad \forall\, i,j \in V, k \in T \qquad (3\text{-}7)$$

$$p_i^y(k) \leq p_j^y(k) - 2d + Rc_{i,j,3}(k) \qquad \forall\, i,j \in V, k \in T \qquad (3\text{-}8)$$

$$p_i^y(k) \geq p_j^y(k) + 2d - Rc_{i,j,4}(k) \qquad \forall\, i,j \in V, k \in T \qquad (3\text{-}9)$$

$$\sum_{\tau=1}^{4} c_{i,j,\tau}\,(k) \leq 3 \qquad \forall\, i,j \in V, k \in T \qquad (3\text{-}10)$$

The parameter $c_{i,j,\tau}$ is a binary variable, and R is a sufficiently large value. When the value of $c_{i,j,\tau}(k)$ with a specific τ equals 1 among the first four inequalities, the corresponding constraint becomes satisfied regardless of the positional variables. In other words, in the case where the linked inequality is met with the binary variable of 0, the two vehicles i and j are away by over $2d$. Therefore, the positions of the vehicles guarantee safety. (3-10) constrains at least one of $c_{i,j,1}$, $c_{i,j,2}$, $c_{i,j,3}$ and $c_{i,j,4}$ to be 0 (Xin, Negenborn and Lodewijks, 2014).

Finally, the model makes the vehicles arrive at their goal location by adopting the following five constraints. The parameters p_i^{fx} and p_i^{fy} refer to the x-axis and y-axis of the required

final position respectively. The parameter ε is an arrival error range (the reference work does not consider this). If the position of vehicle i at k is in the arrival range of the destination, the binary variable $b_i(k)$ becomes 0. By the constraint (3-15), the arrival of vehicle i happens once only during the given time period T.

$$p_i^x(k) - p_i^{fx} \leq R\big(1 - b_i(k)\big) + \varepsilon \qquad \forall\, i \in V, k \in T \qquad (3\text{-}11)$$

$$p_i^x(k) - p_i^{fx} \geq -R\big(1 - b_i(k)\big) - \varepsilon \qquad \forall\, i \in V, k \in T \qquad (3\text{-}12)$$

$$p_i^y(k) - p_i^{fy} \leq R\big(1 - b_i(k)\big) + \varepsilon \qquad \forall\, i \in V, k \in T \qquad (3\text{-}13)$$

$$p_i^y(k) - p_i^{fy} \geq -R\big(1 - b_i(k)\big) - \varepsilon \qquad \forall\, i \in V, k \in T \qquad (3\text{-}14)$$

$$\sum_{k=1}^{T-1} b_i\,(k) = 1 \qquad \forall\, i \in V, k \in T \qquad (3\text{-}15)$$

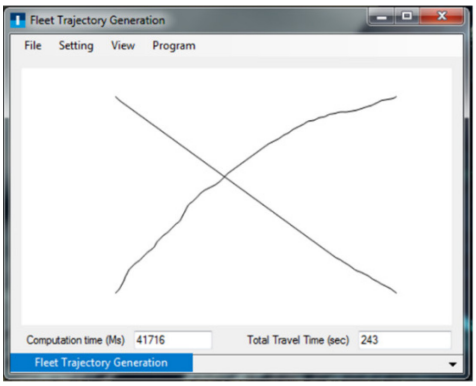
The fleet trajectory planning model tries to minimise the total travel time of the vehicle fleet with the following expression (Xin, Negenborn and Lodewijks, 2014).

$$min \sum_{i \in V} \sum_{k \in T} b_i(k) t(k) \qquad \forall\, i \in V, k \in T \qquad (3\text{-}16)$$

The model is an MIP problem and was tested with a set of FTPP instances different in fleet size (from 2 to 4). It was run in *IBM ILOG CPLEX Optimization Studio 12.6* using a workstation with a 3.40 GHz CPU and 8.00 GB of RAM. The studio utilised branch-and-cut to solve the problems. *Figure 3-6* illustrates two screenshots of the tests.

As seen, the generated paths are not sufficiently smooth and include sudden turns that the vehicles cannot follow in practice. Worthy of mention is also the observed calculation time requirements. In the two-vehicle instance, the model generated trajectories within 44 seconds. However, it required over 130 seconds for three vehicles and failed to determine a valid solution within a 90-minute period in the last scenario.

(a) With two vehicles



(b) With three vehicles

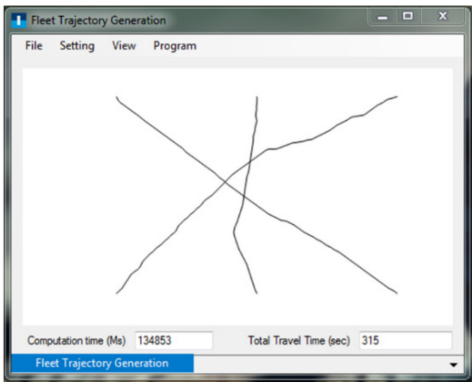


Figure 3-6. Trajectory shapes with two and three vehicles.

Since the presence of many constraints increases time complexity, a fleet trajectory model with the features that were discussed earlier would require substantial computational effort. Also, the presence of multiple vehicles would lead to a disproportionate increase in computational time requirements.

The combination of a distinct path optimiser and an acceleration controller that work in tandem could avoid some of the drawbacks of the first method while having low computational time requirements. To begin with, when a path generation technique is available for an FTPP instance with free-ranging vehicles, lines or curves representing the vehicles' reachable distance ranges replace the reachable areas. Therefore, the search space of the moving agents' positions becomes much smaller than that of the first approach.

At the same time, the path-based strategy does not include any complicated processes for determining vehicular headings because the position of a vehicle on a path results in the corresponding rotation on the route (i.e. the tangent at the point). By utilising Dubins curves or Bézier curves, suggested routes are smooth without zigzag-shaped sections.

3.6. Path Design Methods

As explained, vehicular trajectories need to be follow-able. The follow-ability is subject to path design and acceleration planning, both of which are dependent on the kinematic

limitations of target objects. This section introduces three path generation methods for use in the environment where preinstalled flow paths do not exist.

3.6.1. Virtual Graph-based Algorithms

Imaginary graph-based path designs refer to a path generation strategy that creates and uses a virtual network. This approach includes zone- and edge-based path design schemes. Cell decomposition represents the first group, and roadmap planners the second.

Cell Decomposition

Cell decomposition is a path planning method composed of the following steps (Latombe, 1991): First of all, it decomposes the given space of the problem into multiple areas (i.e. cells) covering the whole domain that can vary in shape (i.e. triangles, rectangles, or polygons). Secondly, the method generates a connectivity graph by linking the areas by bi-directional edges. Finally, it tries to discover the shortest path from the origin to the destination in the problem by using a pathfinder for road networks.

Roadmap

Like the cell-based approach, the core idea of the roadmap planner, including RRTs, is the generation of a road network. Using visibility graphs is one of the earliest path planning approaches based on a road network (Latombe, 1991). Where a group of polygonal obstacles are scattered in a domain with a location origin and a destination, the approach attempts to connect all the possible pair of points (i.e. obstacles' corner points, the origin, and the destination). When a generated edge passes over any obstacle, the method regards the candidate edge as an invisible link and excludes it in the process of connectivity graph generation. A graph-based pathfinding algorithm is then utilised to discover the shortest path.

3.6.2. Bézier Curve Design

Bézier curves belong to a class of parametric curves determined by a set of ordered control points (Marsh, 2005). A Bézier curve with a degree of n is subject to the control points $p_0, p_1 \ldots p_n$. The first two points determine the curve's shape starting from the first point (i.e. the source location), and the last two points do the shape ending at the last control point (i.e. the goal location). (3-17) represents the generalised Bézier curve equation.

$$B(t) = \sum_{i=0}^{n} p_i B_{i,n}(t) \qquad 0 \leq t \leq 1 \qquad (3\text{-}17)$$

$$B_{i,n}(t) = \frac{n!}{(n-1)!\, i!}(1-t)^{n-i} t^i \qquad (3\text{-}18)$$

Based on the curve equation, the three fundamental curves are represented by (3-19), (3-20) and (3-21) respectively with (3-22) as the derivative of the third in the same range of t. Where the number of control points of a Bézier curve is larger than 3, the curve's shape can vary. For instance, it can include a loop or a sharp point.

$$B(t) = (1-t)p_0 + tp_1 \qquad 0 \leq t \leq 1 \qquad (3\text{-}19)$$

$$B(t) = (1-t)^2 p_0 + 2(1-t)tp_1 + t^2 p_2 \qquad 0 \leq t \leq 1 \qquad (3\text{-}20)$$

$$B(t) = (1-t)^3 p_0 + 3(1-t)^2 tp_1 + 3(1-t)t^2 p_2 + t^3 p_3 \qquad 0 \leq t \leq 1 \qquad (3\text{-}21)$$

$$B'(t) = -3(1-t)^2 p_0 + 3(1-4t+3t^2)p_1 + 3t(2-3t)p_2 + 3t^2 p_3 \qquad (3\text{-}22)$$

3.6.3. Dubins Curve Design

A Dubins curve refers to the shortest path of a nonholonomic vehicle that connects the vehicle's two postures by utilising a pair of circles drawn by the maximum steering angle at the source posture and the goal one respectively (Tsourdos, White and Shanmugavel, 2011). Dubins (1957) shows that the shortest path between the two vehicular postures falls into the six path segment combinations: *CSC*, *CSA*, *CAC*, *ASA*, *ASC* and *ACA* (*Figure 3-7*). *C* and *A*

refer to a clockwise rotation and an anticlockwise rotation respectively, and *S* means a straight path segment. In each of the cases, the first and final steering circles are externally located.

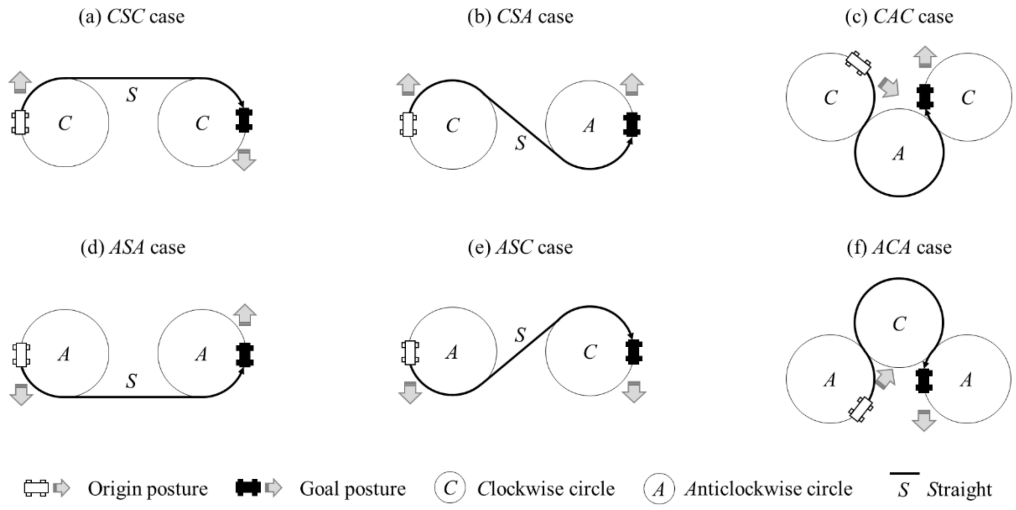


Figure 3-7. Six cases for the shortest path in the Dubins curve design.

Where the middle path segment is *S*, the path design approach utilises one of the four tangent lines (i.e. two internal and two external tangent lines) between the two circles. The selection of one of the tangents depends on the rotational direction of each of the circles.

Figure 3-7 (a) and *Figure 3-7* (d) each show the case that the approach employs an external tangent. In each of the examples, the rotation directions of both circles are identical (i.e. *C*-*C* or *A*-*A*); therefore, the approach needs to select one of the two external tangent lines. Let t_1 and t_2 be two tangent points on the source circle of the two external tangent lines. Where the projection from t_1 towards the vehicular heading on the point meets the destination circle, the tangent line including t_1 is chosen as the middle straight path section of the shortest path; otherwise, the approach chooses the external tangent with t_2.

On the other hand, the approach uses one internal tangent line for the shortest path where the rotation directions are different (*Figure 3-7* (b) and *Figure 3-7* (e)). The curve design method employs the same technique mentioned above to choose one of the internal tangent lines as the middle straight section of the shortest route.

When the middle path section is not *S*, two adjacent circles exist with a specific radius on the sides of the line connecting the origin and destination circle centres. The approach selects one of the adjacent circles resulting in the shorter path. As seen above, the adjacent circles have different steering directions from the source and goal ones (*Figure 3-7* (c) and *Figure 3-7* (f)).

3.6.4. Weaknesses of the Alternatives

The quality of a path design method is dependent on the kinematic characteristics of a target object. Where the vehicle is holonomic, sharp turning points are acceptable. Otherwise, the starting part of its resulting path should be in the direction of the vehicle heading, and the ending part needs to consider the vehicle's required docking posture. Also, the path's curved sections should be follow-able under the vehicle's allowable steering angle range. If the vehicle can travel faster on rectilinear routes than on curves, it will also be preferable to minimise the number of curved sections and their total length.

This research mainly handles container AGVs that have the following features. Firstly, they can move back and forth. Secondly, although they can move sideways (i.e. crab movements), the maximum relevant speed is much lower than those for regular travel settings. According to an AGV brochure[3], their AGVs can travel on straight path sections at a maximum of 6 m/s while their crab movements only allow a top speed rate of 1 m/s. Their maximum steering angle is at least 90° due to the crab movement capability. However, since larger steering decreases their allowable speed range, extreme turns are not preferable. Minimising curved sections is also necessary as they can move faster on straight sections than on curves.

Path design with virtual network layouts is not appropriate for free-ranging AGVs as the resulting paths are not smooth (*Figure 3-8*) and could result in significant trajectory tracking errors. Also, the approach ignores the departure and docking heading directions of moving

[3] Available at: http://www.konecranes.com/equipment/container-handling-equipment/automated-guided-vehicles/agv [Accessed 9 August. 2017]

agents – it is, as such, inappropriate for port environments where container vehicles need to arrive at and depart from their destination in pre-determined postures. Additionally, this approach does not guarantee the shortest paths.

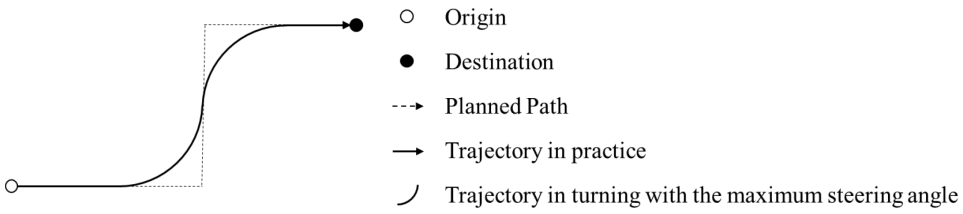


Figure 3-8. Follow-ability of AGVs for the first path generation approach.

The use of Bézier curves to generate paths also has three drawbacks for AGVs. The first is that its paths do not include any straight sections except for the linear version. This feature prevents the vehicles from utilising their maximum speed rate. Secondly, it becomes difficult to calculate the curvature values of the paths when their degree is high. The values need to be known to check the follow-ability of its resulting paths. Thirdly, similar to the first method, this method cannot achieve the shortest journeys of target objects as well.

The curvature on a point of a curve is defined by $\left|\frac{d\theta}{ds}\right|$ where s refers to an arc length, and θ a slope. In two-dimensional space, the curve's x- and y-coordinates can be expressed as $x = x(t)$ and $y = y(t)$ by using a parameter t. Then, the arc length between the points a and b can be calculated by (3-23).

$$s = \int_a^b \sqrt{[x'(t)]^2 + [y'(t)]^2}\, dt \tag{3-23}$$

In the equation above, $\sqrt{[x'(t)]^2 + [y'(t)]^2}$ refers to $v(t)$ which is expressed by $\frac{ds}{dt}$. Also, since $\frac{dy}{dx} = \tan(\theta)$, $\tan^{-1}\left(\frac{y'(t)}{x'(t)}\right)$ can express θ where $x'(t) \neq 0$. (3-24) shows how to calculate the derivative function of θ by letting $f(t)$ and $g(t)$ be $\tan^{-1}(t)$ and $\frac{y'(t)}{x'(t)}$ respectively. The equation is subject to (3-25) and (3-26).

$$\frac{d\theta}{dt} = f'\big(g(t)\big)\, g'(t) = \frac{y''(t)x'(t) - y'(t)x''(t)}{\big(x'(t)\big)^2 + \big(y'(t)\big)^2} \tag{3-24}$$

$$(\tan^{-1}(t))' = \frac{1}{1+t^2} \tag{3-25}$$

$$\frac{y'(t)}{x'(t)} = \frac{y''(t)x'(t) - y'(t)x''(t)}{\big(x'(t)\big)^2} \tag{3-26}$$

So, $\left|\frac{d\theta}{ds}\right|$ is calculated by (3-27).

$$\left|\frac{d\theta}{ds}\right| = \frac{|y''(t)x'(t) - y'(t)x''(t)|}{\left[\big(x'(t)\big)^2 + \big(y'(t)\big)^2\right]^{\frac{3}{2}}} \tag{3-27}$$

Dubins curves are particularly suitable for robust trajectory planning for individual vehicles: Firstly, they provide the minimum requirement of paths (i.e. vehicular postures) and therefore are able to simplify trajectory planning without considering more sophisticated features, such as skidding. Secondly, vehicles can deploy their maximum kinematic limitations since Dubins curves include the least amount of curved sections. Thirdly, as the steering angle of vehicles is considered in these curves, resulting paths are always feasible in terms of curvature. The curvature of a steering circle with a radius of a is always $\frac{1}{a}$ based on (3-27) and the equations: $x = a\cos(t)$ and $y = a\sin(t)$.

However, a few limitations exist for the direct application of Dubins curves to AGVs for container moves. The first movements of the vehicles on Dubins curves are not flexible since the approach only provides the shortest routes. As it utilises the steering circles related to the shortest routes, vehicular turning directions become constant. So, resulting routes are not safe if there exists any possible threat on the paths. Also, the first movements are always turning movements; therefore, when a path re-planning request occurs in operation, the approach cannot provide valid paths in the case where the speed rates of the vehicles are over their maximum curve speed. In practice, vehicles need to travel straight while decelerating first. When their speed reaches the maximum turning speed, they can move in curves.

This study, therefore, seeks to utilise and improve Dubins curves to generate trajectories for a free-ranging AGV fleet that operates in a potential ACT. It employs the following advantages of Dubins curves: (1) the minimum required decision parameters (i.e. x- and y-coordinates and orientation), (2) the least amount of curves and (3) the minimum path length. This study also has less solution search space compared to RRTs which randomly explore operational domains. Also, by considering the three essential parameters, robust path planning can be achieved even for a large-sized vehicle fleet. A couple of techniques are introduced in the following section to overcome the drawbacks of Dubins curves.

3.7. Model Validation Methods

Three approaches exist to proving the validity of a new idea, design and technology, by testing them in real application environments as the first. This option is significantly time-consuming, accompanied by major operational risks and incurred costs. Thus, it is not applicable to large container terminals.

The use of mathematical models would be an alternative approach, focusing on possible changes in target systems. The models are subject to mathematical theories, such as recursive forms. Since it is difficult to mathematically model all the changes in a target system and interactions with its adjacent operational regions, there is a tendency to simplify the formulations by considering a small number of crucial factors. Thus, gaps can be huge between the mathematical results and observations in practice.

The third approach is the use of simulation models. Simulation is an experimental technique that imitates target systems by examining their operations. It is regarded as the most favourable method as it is flexible, intuitive and powerful (Hillier and Lieberman, 2005). This approach is also economical in that it foresees system operations and performance without requiring any associated hardware. Compared to the second approach, simulation can involve more factors with the ability to visualise imitated operations to help system operators' decision-making.

3.8. Summary

This study includes six objectives to achieve the research aim. The literature review is the first objective to cover the recent state of terminal automation and a range of previous work on vehicle dispatching, trajectory planning and vehicle operation simulation. It helped to construct research methodology and other objectives for model development.

The methodology has three model development stages. Two of them are for vehicle dispatching and fleet trajectory planning respectively. Due to the time complexity of the two decision-making problems including many vehicles, this research adopted the framework of SA for real-time applications. The other stage is planned to construct a simulation model to analyse integrated AGV operation planning in potential ACTs with the free-ranging vehicle setting. The model mainly focuses on the transport area of AGVs while providing quay and yard operations to support AGV operations for container handling.

This study also includes a series of simulation experiments and analysis as the final stage. In order to provide reliable results, this stage is planned to select a representative container terminal and gather relevant data from the literature. It also has various scenario settings and performance indicator selection to validate the integrated vehicle operation planning approach in potential ACTs.

Chapter 4. Advanced AGV Dispatching

This chapter starts with a description of container delivery tasks by container vehicles or AGVs. The chapter then introduces a flexible AGV dispatching model with its mathematical formulation in detail. SA is combined with the model to cope with the time complexity of the mathematical form with branch-and-cut.

4.1. AGV Tasks in Container Terminals

Container terminals utilise a diverse range of equipment to perform horizontal and vertical container transfers. Vertical transfers take place at the quay, where quay cranes load and unload containers from visiting vessels, while yard cranes are used to handle containers at storage stacks. AGVs and container vehicles perform horizontal transfers by transporting containers between the quay and yard.

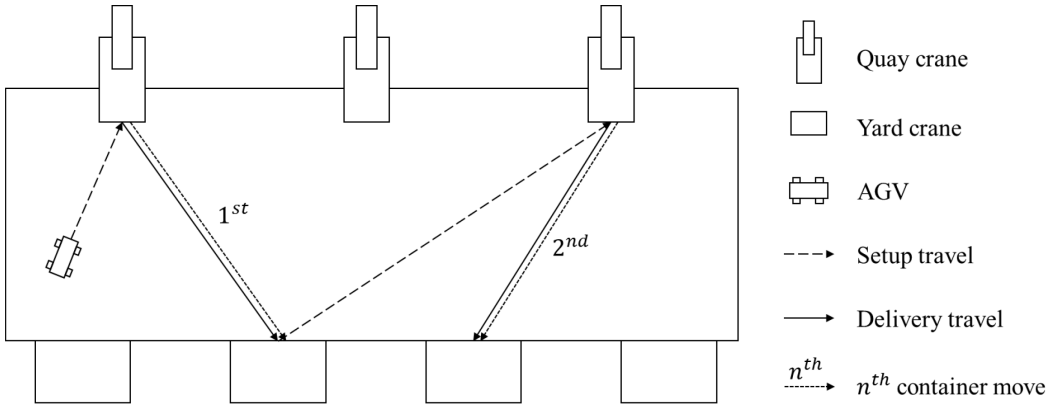


Figure 4-1. Container and AGV moves.

There exist four types of container flows categorised by their location origin and destination. Firstly, inbound and outbound moves include container transfers from the quay to the yard and vice versa in the terminal, while accounting for most of the flows. Secondly, external vehicles, such as trucks, deliver containers from inland depots to the terminal or vice versa. As the third type, there is container relocation between stacks. Shuffling is the fourth in which a yard crane changes the locations of containers in the same block.

The first and third types initiate AGV journeys in the transport area. Where an AGV is assigned to a horizontal container move, it moves to the source crane, receives the involved container and delivers it to the goal crane. Throughout this thesis, the term setup travel is used to describe a journey to a task origin like the first trip, and the term delivery travel refers to a journey to the destination of a container like the second movement (*Figure 4-1*).

The inbound and outbound moves in container terminals are subject to quay cranes whose work plans are determined before vessels' arrivals. Kim and Bae (2004) explain how the plans are generated and what information they include. When a container ship is scheduled to visit a port, the shipping agent sends the terminal operator a bay profile in advance, which is the state of the container placement of the vessel. The profile acts as a guideline on container loading and discharging missions to be processed by a group of quay cranes. Each quay crane is assigned to a bay area and receives a set of ordered container transfer tasks.

As mentioned by Kim and Bae (2004), quay crane schedules cover the sequences of planned container movements along with their intended origin, destination, operation cycle time and earliest possible event time without delay (EPETWD). Loading and discharging operations are likely to occur alternately to minimise the empty travel of the quay crane spreader. The operation cycle time of a transfer mission refers to the time required by the spreader to move from the container slot to the buffer area where AGVs wait to be served. The term EPETWD is used to express the time when the involved container, on its designated AGV, is being picked up by a spreader where the task is a loading operation. In a discharging operation, it is the time when the spreader begins releasing the container onto its assigned AGV.

Table 4-1 presents an example quay crane schedule where the crane spreader takes 20 seconds to pick up or release a container, commencing from a buffer area. The data is adapted from a similar dataset previously used by Kim and Bae (2004). The crane is planned to start the first loading operation when $t = 0$. After 100 seconds, the spreader is planned to be at the container slot. Then, it is likely to start the following discharging mission that requires 110 seconds

including 20 seconds for the spreader to release the container on its designated AGV. So, it is scheduled to start loading the cargo on the vehicle when $t = 190$. After 20 seconds, it is likely to finish the second task and begin the next loading operation. It concludes the transfer at $t = 310$, and the spreader is planned to travel to the buffer for the following discharging operation beginning loading the fourth container on its allocated AGV when $t = 410$.

Table 4-1. Quay crane schedule

Sequence	Type*	Ship location†	Yard location††	Operation cycle time (seconds)	EPETWD (seconds)
1	L	01/02/01	01/03/01/02	100	0
2	D	01/03/06	02/01/05/03	110	190
3	L	01/02/02	04/02/03/05	100	210
4	D	01/03/05	03/05/04/01	120	410

* L: Loading, D: Discharging

† Ship-bay number / row number / tier number

†† Yard block number / yard bay number / row number / tier number

It is preferable that quay cranes follow their job handling sequence since more operations occur when the cranes do not handle the transfers in order. Essentially, when stacking containers, several features, such as weight, are considered - heavy ones are likely to be in lower tiers. Also, terminal operators determine such handling sequences to minimise the total movement of quay crane spreaders. For instance, if two container loading operations occur in a row, the spreader needs to travel from the container slot of the previous task to the crane's buffer without a container. Also, wrong container stacking orders would require additional spreader work to relocate containers.

For quay cranes to perform their tasks in order, it is preferable that designated AGVs to process the container moves arrive at the cranes in the same sequences of the quay crane jobs. Also, on-time or early arrivals need to be achieved to reduce quay crane idle time, which directly affects quay crane throughput.

4.2. A Novel AGV Dispatching Model

This section introduces and describes a newly designed dispatching model for free-ranging AGVs in potential container terminals. It starts with the classification of the model, followed by the explanation of three primary inputs of the model. The mathematical formulation of the model concludes the section.

4.2.1. Model Classification and Key Features

The model has the following key features that are also summarised in *Table 4-2*:

- It assigns two container transport tasks to each AGV and determines the job sequences on the vehicles (i.e. mid-term vehicle dispatching). The dispatcher also adopts a rolling horizon to handle uncertainty in vehicle operation environments.
- It is independent of application settings.
- Since there exists a limit of the number of tasks able to be assigned to each AGV, there is no possibility of a few vehicles processing most tasks (i.e. unbalancing).
- The model tries to minimise the total late arrival time of AGVs at quay and yard cranes while decreasing the job preparation travel time of the vehicles.
- The model also includes a function of reassigning tasks if their container has yet to be picked up by the designated vehicles while considering task-sequencing impacts.
- AGVs are assumed to move a single container at a time.

Table 4-2. Model classification by the key issues

Key issue	Selection or Description
Planning length	Mid-term (two delivery tasks per vehicle at most)
Operation environment	Environment-independent
Workload balancing	Inherently considered
Objective	Minimising vehicular late arrival and setup travel time
Task re-assignability	Considered
Load type	Single-load

Compared to dispatching models and algorithms developed so far, the proposed dispatching model has the following strengths:

- By adopting the mid-term dispatching concept, the new model has the advantages of both short-term planning and long-term planning: short calculation time and less travel caused by the consideration of container handling orders.
- Unlike typical dispatching heuristics, such as the FEFS rule, the model is not subject to application settings. Therefore, it is available in the free-ranging vehicle setting as well as conventional domains with preinstalled guide paths.
- By combining task reassignment with mid-term planning, the proposed dispatching model can consider the influence of sequencing delivery tasks even when cancelling ongoing tasks and reallocating them to AGVs, which has been ignored in the literature.

4.2.2. Model Inputs

The proposed dispatching model requires three groups of inputs: (1) generalised quay cranes schedules, (2) vehicle data with their ongoing assignment and (3) a container delivery task set to be used as a job set per vehicle dispatching process. The following subsections explains all input groups in detail.

Input 1: Generalised Quay Crane Schedules

The dispatching method does not directly employ the quay crane schedule format explained in Kim and Bae (2004) because the format only covers event data on the quayside without considering information on the yard side. Therefore, the proposed model introduces r_i and u_i: The former refers to the AGV arrival reference time at the origin of quay crane job i. The latter means the AGV arrival reference time at the goal location of the task.

Let q (or h) be the spreader service rate of the quay (or yard) crane, and t_i the expected delivery travel time by AGVs. The parameter e_i refers to the EPETWD for the mission. When this is a loading mission, r_i can be calculated by subtracting t_i and h from e_i as shown in (4-1). The parameter u_i is equal to e_i. On the other hand, where it is a discharging operation, r_i is the same as e_i, and u_i can be represented as the sum of e_i, q, and t_i as shown in (4-2).

$$r_i = e_i - t_i - h \tag{4-1}$$

$$u_i = e_i + q + t_i \tag{4-2}$$

As seen above, the two parameters r_i and u_i are subject to e_i. Therefore, the following expressions are proposed for calculating e_i while considering the previous operation type (*Table 4-3*). i indicates the index of a job among the pending missions of a quay crane. The current time (or plan start time) is denoted by t, and a_t is the time based location of the quay crane spreader at t. It should not be negative and it becomes smaller when it is close to the buffer area. The parameter o_i refers to the operation cycle time of job i.

Table 4-3. e_i calculation in a quay crane

Condition	Expression
$i = 1 \,\&\, y_i = 0$	$t + a_t$
$i = 1 \,\&\, y_i = 1$	$t + \lvert o_i - a_t \rvert + o_i - q$
$i > 1 \,\&\, y_{i-1} = 0 \,\&\, y_i = 0$	$e_{i-1} + 2o_{i-1}$
$i > 1 \,\&\, y_{i-1} = 0 \,\&\, y_i = 1$	$e_{i-1} + o_{i-1} + o_i - q$
$i > 1 \,\&\, y_{i-1} = 1 \,\&\, y_i = 0$	$e_{i-1} + q$
$i > 1 \,\&\, y_{i-1} = 1 \,\&\, y_i = 1$	$e_{i-1} + 2o_i$

The first two conditions handle e_i for the first pending quay crane task. If it is a loading operation (i.e. $y_i = 0$), the spreader will be able to start the mission after moving to the buffer by a_t from t. On the other hand, when it is a container discharging duty (i.e. $y_i = 1$), the equipment will need the amount of time to (1) pick up the corresponding container on the vessel by $|o_i - a_t|$ and (2) move to the discharging critical point by $o_i - q$; the point is where the quay crane can begin discharging operations.

The rest cover the e_i calculation of the remaining quay crane missions: The third condition refers to the situation where the i^{th} unhandled transfer is a loading operation following another loading transfer. In this case, the spreader will be able to start job i after (1) finishing the previous loading operation taking o_{i-1} from e_{i-1} and (2) moving back to the buffer by o_{i-1}. If job i is a discharging operation after a container loading operation, it will be able to begin after the quay crane finishes the previous task at $e_{i-1} + o_{i-1}$ and moves its spreader to the discharging critical point by $o_i - q$ (the fourth condition).

In the fifth condition, the quay crane will be able to begin picking up the container involved in the i^{th} job when completing loading of the container linked with the discharging duty on its designated AGV. The sixth condition means the case where job i and the previous transfer each are a container discharging process. For the quay crane spreader to start the i^{th} job, it will require the amount of time for finishing the previous mission at $e_{i-1} + q$, moving to the i^{th} container slot taking o_i, and turning back to the discharging critical point by $o_i - q$.

Input 2: Task Assignment and Vehicle Data

Where an AGV dispatching model cannot cancel current task assignments, it requires data on the destinations of the last missions of delivery vehicles without needing information on the assignment states of tasks on the AGVs (i.e. breakable or unbreakable) and their positional data. This is because the vehicles should process all the tasks with them to handle new delivery duties, and therefore the AGVs will be at the goal locations of the final missions in the task queues before starting their new delivery duty.

However, the proposed uses the assignment state information and vehicular positional data because of its task reassignment function. It perceives the assignment states of being-handled tasks at each plan start time to cancel breakable allocations. Also, it requires the locations of the AGVs to expect setup travel time.

Input 3: Job Set

At each planning stage, the job set refers to a list of quay crane tasks for the dispatcher to assign to AGVs. Where there are V vehicles, the maximum size of the mission set is $2V$ because the dispatching model can allocate two transport tasks per AGV at most. A member of the set is removed when its destination crane finishes discharging the container from its designated AGV. At the same time, the most urgent quay crane job based on EPETWD is added on to the list.

4.2.3. Mathematical Formulation

This section describes the mathematical formulation of the AGV dispatching model. It includes indices (*Table 4-4*), sets (*Table 4-5*), decision variables (*Table 4-6*), parameters (*Table 4-7*), parameter expressions (*Table 4-8* to *Table 4-10*), an objective function ((4-3) to (4-5)) and constraints ((4-6) to (4-13)).

Indices

Table 4-4. Indices in the dispatching model

Indices	Description
i, j	Used for quay crane tasks (regardless of linked cranes)
v	Used for AGVs

Sets

Table 4-5. Sets in the dispatching model

Set	Description
J	Set of quay crane tasks (i.e. job set)
V	Set of AGVs

Decision Variables

Table 4-6. Decision variables in the dispatching model

Decision variable	Description
x_v^i	Connectivity from AGV v to job i as the first mission
x_i^j	Connectivity from job i to job j as the second mission

The decision variables x_v^i and x_i^j are binary. The former represents the first task assignment from AGV v to job i, and the latter the second assignment from job i to job j. For instance, if $x_v^i = 1$ and $x_i^j = 1$, AGV v will process job i first and then job j as its second mission.

Parameters

In *Table 4-7*, c_v is a binary variable to express the travel state of AGV v at ρ. It is set to 0 where the vehicle is on a setup travel; otherwise, 1. m_v refers to the expected remaining delivery travel time of AGV v for the current mission. This value only becomes valid when $c_v = 1$. Where the condition is not met, the value is set to M. f_v^i is a binary variable to show the link from AGV v to job i as the vehicle's current (i.e. first) task at ρ. When the task is currently handled by AGV v, f_v^i is set to 1 and 0 otherwise. The parameter is independent of c_v. y_i refers to the type of job i. When the task is a loading operation, the variable is set to 0; otherwise 1.

s_v^i means the expected setup travel time of AGV v for job i. The setup travel is determined by the values of c_v and f_v^i. Where both the parameters are set to 1, which means that the vehicle is on the delivery for the job, the following s_v^i becomes 0. When $c_v = 1$ and $f_v^i = 0$, the setup travel time is set to M since it means that the vehicle is currently carrying an irrelevant cargo to job i. In the other cases, the expected travel time of s_v^i is calculated by distance. The parameter s_i^j refers to the expected setup travel time from job i to job j. If the index i is equal

to j, the corresponding setup time is set to M. Also, when the two tasks are on the same quay crane, and the sequence number of task i is higher than that of job j, the setup value is set to M.

Table 4-7. Parameters in the dispatching model

Parameter	Description
ρ	Plan start time
q	Required time by a quay crane's spreader to pick up or release a container
h	Required time by a yard crane's spreader to pick up or release a container
c_v	Travel state of AGV v at ρ
m_v	Expected remaining delivery time for the current task of AGV v at ρ
f_v^i	Assignment state from AGV v to job i as the first job at ρ
y_i	Task type of job i
s_v^i	Expected set-up travel time from AGV v to job i
s_i^j	Expected set-up travel time from job i to job j
t_i	Expected travel time from the origin to the destination of job i
p_i	Expected AGV arrival time at the origin of job i
r_i	AGV arrival reference time at the origin of job i
d_i	Expected AGV arrival time at the destination of job i
u_i	AGV arrival reference time at the destination of job i
k_i	Service rate of the origin crane's spreader for job i
l_i	Service rate of the destination crane's spreader for job i
α	Task tardiness weight in the quayside
β	Task tardiness weight in the yard
γ	Setup travel time weight for AGVs
M	Significantly large positive number

p_i is an expected AGV arrival time at the origin crane of job i. The parameter varies depending on which AGV takes the task. How it is calculated is explained in the *Parameter Expressions* section. r_i is the container release reference time of task i. Therefore, the difference from p_i to r_i can be regarded as the lateness of the task at the source crane. When the lateness is positive, it means the tardiness of the involved container pickup.

In a similar way, d_i means an expected vehicle arrival time at the destination crane of job i, which is also dependent on a designated AGV to the job. The way that the parameter is calculated is explained in the *Parameter Expressions* section. u_i refers to the container arrival

reference time of job i. Therefore, the difference between the two parameters is the delivery lateness of the involved container. Also, $max(d_i - u_i, 0)$ refers to the tardiness of the task at the goal crane.

Parameter Expressions

The parameter k_i refers to the service rate of the origin crane's spreader for job i, and l_i the service rate of the destination crane's spreader for the task. The expressions of the parameters are shown in *Table 4-8*. k_i is equal to h when job i is a loading operation. In the case where it is a discharging one, the parameter equals q. In a similar way, l_i is determined by the type of job i (i.e. y_i).

Table 4-8. k_i and l_i expressions

Parameter	Expression
k_i	$(1 - y_i)h + y_i q$
l_i	$y_i h + (1 - y_i)q$

Unlike the two parameters, p_i and d_i each are dependent on the assignment result of job i (i.e. x_v^i and x_i^j). This task can firstly or secondly be handled by an AGV, meaning conditional situations, and therefore one or more binary variables and parameters should be included in the expressions of p_i and d_i. *Table 4-9* and *Table 4-10* show the expressions of p_i and d_i respectively.

Table 4-9. p_i calculation

Condition	Expression
As the first	$\sum_{v \in V} x_v^i \{(1 - c_v)(\rho + s_v^i) + c_v \rho\}$
As the second ($c_v = 0$)	$\sum_{v \in V} \sum_{j \in J} x_v^j x_j^i \{(1 - c_v)(\rho + s_v^j + k_j + t_j + l_j + s_j^i)\}$
As the second ($c_v = 1$)	$\sum_{v \in V} \sum_{j \in J} x_v^j x_j^i \{c_v(\rho + m_v + l_j + s_j^i)\}$

In the case where task i is allocated to vehicle v as the first mission, there exist two cases depending on the value of c_v. Where $c_v = 0$, meaning that the vehicle is on a setup travel, it is expected to arrive at the source crane of the mission when finishing its current journey to the resource location. The vehicle would arrive at the goal crane after being handled at the origin crane by k_i and travelling to the goal location by t_i. If $c_v = 1$ (i.e. on an execution travel), p_i is unavailable, and therefore it will be set to the smallest possible value (ρ). The vehicle is expected to arrive at the destination crane when travelling the remaining distance taking m_v.

Table 4-10. d_i calculation

Condition	Expression
As the first	$\sum_{v \in V} x_v^i\{(1-c_v)(\rho + s_v^i + k_i + t_i) + c_v(\rho + m_v)\}$
As the second ($c_v = 0$)	$\sum_{v \in V}\sum_{j \in J} x_v^j x_j^i \{(1-c_v)(\rho + s_v^j + k_j + t_j + l_j + s_j^i + k_i + t_i)\}$
As the second ($c_v = 1$)	$\sum_{v \in V}\sum_{j \in J} x_v^j x_j^i \{c_v(\rho + m_v + l_j + s_j^i + k_i + t_i)\}$

When task i is the second mission of vehicle v on a setup travel, the vehicle would arrive at the source crane of the task after (1) travelling to the source crane of the first task (i.e. job j), (2) being served by the crane taking k_j, (3) moving to the goal crane of the first job requiring t_j, (4) being handled by the crane by l_j and (5) travelling to the origin crane of the second job by s_j^i. The vehicle is expected to be processed by k_i and then move to the goal crane of the second task by t_i. Thus, the arrival time can be expressed by the sum of p_i, k_i and t_i.

In the third condition, m_v is utilised instead of s_v^j, k_j and t_j in the second expressions. This is because where the vehicle is on a delivery, there do not exist the setup travel of the first mission (i.e. s_v^j) and the service of the first origin crane (i.e. k_j). Where job i is on the second queue, the multiplication of the two decision variables happens, which results in the situation

that the dispatching model becomes unsolvable in many commercial solvers, such as *CPLEX*. Therefore, $x_v^j x_j^i$ can be replaced with $min(x_v^j, x_j^i)$, which acts as the multiplication term.

Objective Function

The AGV dispatching model pursues the minimisation of total cost about the arrival delay of AGVs in the quay and the yard as well as the setup travel time of the vehicles. (4-3), (4-4) and (4-5) express the penalty terms respectively.

$$\alpha \sum_{i \in J} ((1 - y_i) * max(d_i - u_i, 0) + y_i * max(p_i - r_i, 0)) \quad (4\text{-}3)$$

$$\beta \sum_{i \in J} ((1 - y_i) * max(p_i - r_i, 0) + y_i * max(d_i - u_i, 0)) \quad (4\text{-}4)$$

$$\gamma \left(\sum_{v \in V} \sum_{i \in J} s_v^i x_v^i + \sum_{i \in J} \sum_{j \in J} s_i^j x_i^j \right) \quad (4\text{-}5)$$

(4-3) and (4-4) refer to the quay and yard tardiness penalties of all tasks in J respectively. Depending on y_i, specific terms are activated. Potential AGV arrival tardiness at container source cranes is activated in (4-3) for discharging operations and (4-4) for loading operations. Potential AGV arrival tardiness at container goal cranes is considered in (4-3) for container loading jobs and (4-4) for cargo discharging missions. (4-5) represents the summation of the setup travel time values of all the vehicles in V. By adding on the three penalty terms, a total cost function is generated that is used as the objective function of the model.

Constraints

The model includes the following constraints.

$$\sum\nolimits_{v \in V} x_v^i \leq 1 \qquad \forall\, i \in J \quad (4\text{-}6)$$

$$\sum_{i \in J} x_v^i \leq 1 \qquad \forall\, v \in V \qquad (4\text{-}7)$$

$$\sum_{i \in J} x_i^j \leq 1 \qquad \forall\, j \in J \qquad (4\text{-}8)$$

$$\sum_{j \in J} x_i^j \leq 1 \qquad \forall\, i \in J \qquad (4\text{-}9)$$

$$\sum_{v \in V} x_v^i + \sum_{j \in J} x_j^i \leq 1 \qquad \forall\, i \in J \qquad (4\text{-}10)$$

$$0 \leq \sum_{v \in V} x_v^i - \sum_{j \in J} x_i^j \leq 1 \qquad \forall\, i \in J \qquad (4\text{-}11)$$

$$c_v\left(f_v^i + x_v^i\right) \neq 1 \qquad \forall\, v \in V, i \in J \qquad (4\text{-}12)$$

$$x_v^i, x_i^j \in \{0, 1\} \qquad \forall\, v \in V, i, j \in J \qquad (4\text{-}13)$$

Each transport task can be allocated to at most one AGV as the first mission, which is constrained by (4-6). The inequality (4-7) renders each AGV able to take one transport job as its first duty or none. Each task can be in the second queue of an AGV's job list, which is constrained by (4-8). The constraint (4-9) shows that each task can initiate another. The constraint (4-10) prohibits each task from being the first task and the second job of any of the AGVs at the same time.

In (4-11), when job i is used as the first task of an AGV (i.e. the first term is 1), the job can initiate another, which means the second term can be either 0 or 1. Otherwise (i.e. the first term is 0), the mission should not initiate another.

The constraint (4-12) is for the reassignment of transport jobs on their setup. Since all the variables and parameters are binary, there exist eight possible combinations as described in *Table 4-11*. Among the possible combinations, there exist two impossible situations that should not happen. As shown in *Table 4-11*, the constraint successfully prohibits the two cases. The constraint (4-13) refers to the binary values of the decision variables x_v^i and x_i^j.

Table 4-11. Descriptions of the reassignment constraint

Parameters		Decision Variable	Possible?	Description
c_v	f_v^i	x_v^i		
1	1	0	No	If AGV v is on a delivery for job i, the job should be the first task of the vehicle.
1	1	1	Yes	
1	0	0	Yes	If AGV v is on a delivery not for job i, the vehicle should not take the job as its first duty.
1	0	1	No	
0	1	0	Yes	If AGV v is on a setup travel, the vehicle can take any jobs as its first mission.
0	1	1	Yes	
0	0	0	Yes	
0	0	1	Yes	

In summary, the defined mathematical form falls under MIP with some decision variables as integers. Therefore, branch-and-cut is available and the next section shows its application and limitation.

4.3. Application of Simulated Annealing

Branch-and-cut is widely used to MIP problem instances, including the AGV dispatching problem defined in *Section* 4.2.3. Therefore, *IBM ILOG CPLEX Optimization Studio 12.6* was utilised to solve the AGV dispatching problem's instances. It is a commercial software package that provides a range of functions of solving various types of mathematical problems, and it can adopt branch-and-cut to tackle MIP.

Table 4-12. Dispatching problem parameter settings

Parameter	Value
Job delay weight in the quay (α)	0.7
Job delay weight in the yard (β)	0.2
AGV setup travel weight (γ)	0.1
M	1,000,000

A series of simulation tests were implemented by using the workstation described in *Section* 3.5, and it was observed that *CPLEX* requires significant calculation time to handle the

dispatching problem although it can provide high-quality solutions. To be more specific, the size of a problem instance of the AGV dispatching problem is subject to the number of AGVs to be used. When four vehicles are involved, calculation time rapidly risen with high variations as shown in *Table 4-13*.

Table 4-13. Averages of the calculation times (300 samples)

Calculation time	Number of AGVs		
	2	3	4
Average	0.309	0.492	26.702
Standard deviation	0.049	0.277	151.077

* unit: seconds

Since there exist numerous factors that affect the computational complexity of algorithms (including time complexity), it is also rather difficult to identify the reasons in the observed case. However, from the mathematical model it is clear that the complexity of the model is not linearly expressed. This is because the formulation includes multiplications of the decision variables and some of the parameters that cause non-linearity. Also, the not-equal constraint (4-12) exponentially increases the time complexity of the dispatching model in the studio.

Therefore, another solver needs to be developed that can provide satisfactory solutions within a few seconds for use in real container terminal environments. In such cases, a metaheuristic is a widely-accepted option.

4.3.1. Alternative Simulated Annealing Based AGV Dispatcher

In this thesis, SA is adopted instead of the others (i.e. TS and GAs). The reasons are as follows: Firstly, it requires smaller memory space compared to the others. TS needs a data field to store its tabu list, and GAs require even more since they utilise solution populations. Secondly, the selected metaheuristic does not require additional operations when it handles the defined vehicle dispatching problem. In the case where a GA framework applies to the problem, its crossover operations should include a chromosome adjusting function. This is because

offspring chromosomes would lose cargo delivery missions or have duplicated jobs during the mating processes as shown in *Figure 4-2*.

AGV 1	Job 1	Job 3
AGV 2	Job 4	Job 7
AGV 3	Job 2	Job 5
AGV 4	Job 6	Job 8

AGV 1	Job 2	Job 8
AGV 2	Job 1	Job 5
AGV 3	Job 4	Job 7
AGV 4	Job 3	Job 6

AGV 1	Job 1	Job 3
AGV 2	Job 4	Job 7
AGV 3	Job 4	Job 7
AGV 4	Job 3	Job 6

AGV 1	Job 2	Job 8
AGV 2	Job 1	Job 5
AGV 3	Job 2	Job 5
AGV 4	Job 6	Job 8

Figure 4-2. Crossover operation example.

Due to the drawbacks of TS and GAs, SA is more appropriate for the defined AGV dispatching problem, and therefore this research includes the development of an SA-based vehicle dispatcher. *Figure 4-3* depicts the process of the model that follows the framework of SA while maintaining simplicity.

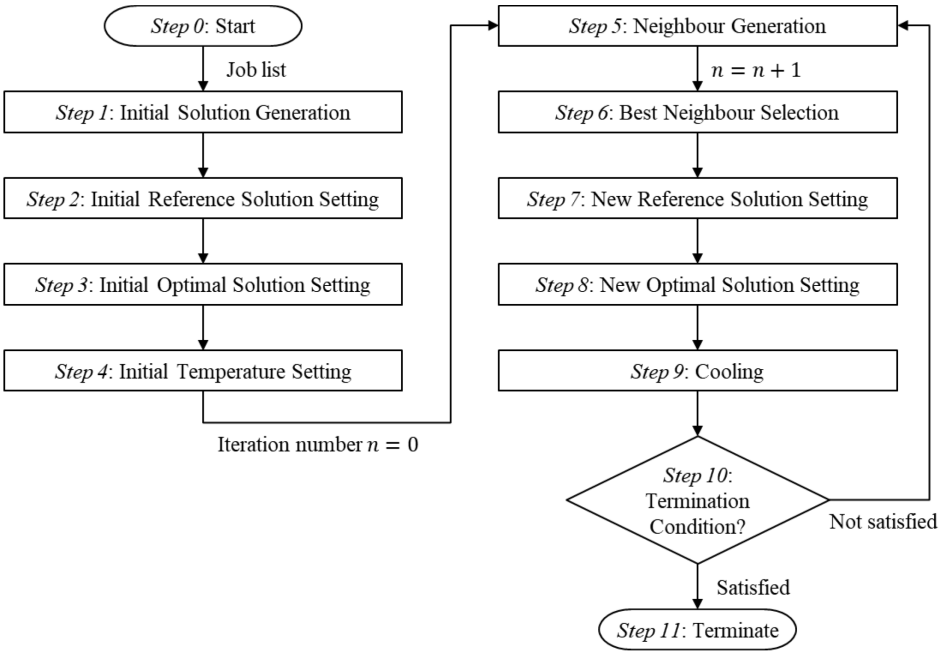


Figure 4-3. Logical structure of the proposed AGV dispatcher.

As the first step, the solver creates an initial dispatch plan (i.e. *Initial Solution*) that satisfies all the constraints explained in *Section* 4.2.3. Based on the tasks in the job list, the dispatcher randomly allocates at most two tasks to each AGV. The algorithm then copies the initial solution onto two data fields named *Reference Solution* and *Optimal Solution* respectively.

The value in the former is a reference for the dispatcher to generate neighbour solutions, and the value in the latter means the optimal solution found in all past iterations. Before the first iteration process, it is set to the initial value.

In the next step (i.e. *Step 4*), the dispatcher sets its initial temperature (i.e. *Initial Temperature*). This value is closely related to a defined terminate temperature (i.e. the lowest temperature in the annealing process) and a cooling schedule to apply. For instance, although the difference between the initial and the terminate values is rather significant, sufficient iterations cannot be performed with a rapid cooling schedule. Therefore, the three factors need to be considered as a set while examining that the dispatcher experiences sufficient iterations. As a default setting, the initial temperature is 10,000.

After finishing the initial configuration, the algorithm starts its main iteration process. Firstly, it generates several neighbour solutions by slightly amending the reference solution. The AGV dispatcher uses five ways in generating neighbours as depicted in *Figure 4-4*. When the travel state of an involved vehicle is a delivery travel, task exchanges including the first mission of the vehicle are unable; for instance, in the case where vehicle w is delivering a container, the second and the fifth scenarios are infeasible. The neighbour generation process can be implemented multiple times depending on a user setting. Where the dispatcher performs the process k times, it can generate $5k$ neighbour solutions as the maximum.

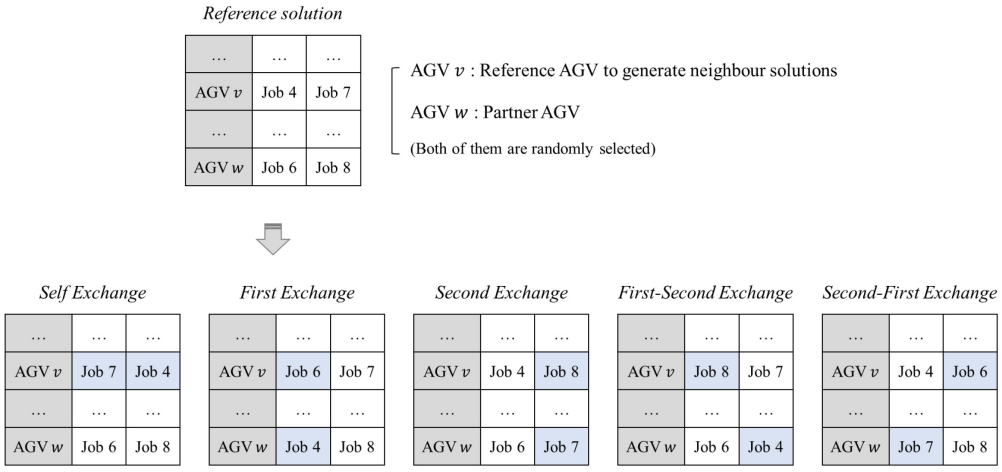


Figure 4-4. Neighbour solution generation.

Among them, the solver selects the best solution and stores it in the data field *Best Neighbour Solution* for updating the solutions in *Reference Solution* and *Optimal Solution*. It compares the best neighbour with the solution in *Reference Solution*; where the neighbour is superior to the reference, the dispatcher removes the previous value and stores the neighbour in *Reference Solution*. Otherwise, the algorithm determines whether it utilises the neighbour for generating dispatching solutions in the next iteration based on the probability function (4-14).

$$P = e^{\frac{s_o - s_n}{T}} \tag{4-14}$$

In the equation, T is the current temperature, and s_o and s_n mean the current optimal solution and the best neighbour solution respectively. P refers to the acceptance probability of s_n as the next reference solution. Therefore, in the given problem, the result of the probability function is always larger than 1 when the reference's objective value is over the objective score of the neighbour. In the opposite situation, the equation provides a positive value less than 1. This value becomes smaller by a lower temperature and a bigger gap between the two solutions; therefore, the acceptance level decreases by iteration numbers. The algorithm determines a new value of *Optimal Solution* by comparing its current value and the best neighbour solution; it chooses the better one.

$$T_{k+1} \geq \frac{T_0}{\ln(1+k)} \tag{4-15}$$

$$T_{k+1} = \alpha T_k \tag{4-16}$$

Then, the AGV dispatcher lowers the temperature subject to a predetermined cooling function. Essentially, SA increasingly decreases temperature reduction degree. Nourani and Andresen (1998) show various types of cooling strategies, such as (4-15) suggested by Geman and Geman (1984). The proposed dispatcher employs (4-16); this is a basic version with the core idea of the cooling in SA frameworks with a single control parameter. It is widely used to discover a suboptimal solution (Du and Swamy, 2016). In the two cooling functions, T_k is the temperature at the k^{th} iteration, and α is a control parameter larger than 0 and less than 1.

After finishing the current iteration cycle, the dispatcher needs to determine whether it continues such an iteration process again. Similar to most SA frameworks, it terminates and returns the value in *Optimal Solution* when it has reached the lowest temperature. There is no chance for the final solution to be infeasible since the algorithm always provides and handles valid solutions.

4.3.2. Computational Performance

Before detailed investigations of the SA-based AGV dispatching algorithm, this subsection provides a series of simulation experiment results to identify computational power in the SA-based solver. The used experiment settings are as follows: There were 6 and 16 units of quay and yard cranes respectively. At the initial stage, each vehicle had a randomly generated mission list that included two container transport tasks. In each experiment, the two dispatchers provided their AGV dispatch plan with their calculation time. *Table 4-14* summaries the used default settings for the SA-based dispatcher.

Table 4-14. SA parameter settings

Parameter	Value
Initial temperature	10,000
Final temperature	1
Coefficient of the cooling function	0.95 (180 iterations)
Iterations for neighbour generation	3

Table 4-15 shows the simulation results with a small-sized vehicle fleet; V refers to the number of AGVs used in the tests, and each unit test includes 300 AGV dispatching problem instances. Regarding solution quality, the SA-based model showed almost the equivalent performance as the *CPLEX*-based one. However, its calculation time was significantly low with negligible variations compared to the *CPLEX*-based solver that showed unstable computation time with high variations.

Table 4-15. Computation performance comparison ($V = 4$)

	CPLEX-based	SA-based
Calculation time average (seconds)	26.702	0.057
Calculation time standard deviation	151.077	0.020
Objective value average	1300.936	1302.162
Objective value standard deviation	202.298	202.777

Since the *CPLEX*-based solver requires significant computation time with high variations, the following tests only included the SA-based dispatcher including 5, 10, 15, 20, 25 and 30 AGVs respectively. To begin with, as shown in *Figure 4-5*, the SA-based dispatcher quickly provided AGV allocation plans. Although there was an upward trend in calculation time by the fleet size increase, it only required around 11 seconds in the 30-AGV case; the calculation period is less than the half of the time required by the *CPLEX*-based dispatcher with the 4-AGV case (i.e. 26.702 seconds). The SA-based model also shown insignificant variations in computation time with 300 randomly-created problem instances. Also, computation time can even decrease under high-performance computing (HPC) in real container terminal environments.

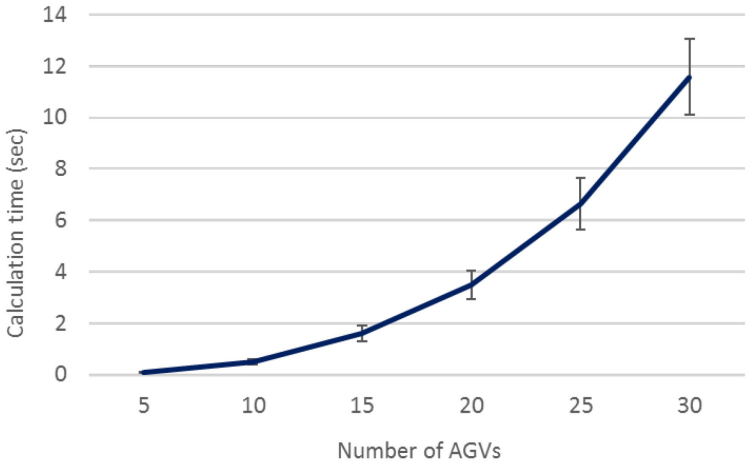


Figure 4-5. Calculation time change (in seconds).

Figure 4-6 depicts objective value changes observed in the tests by different cooling parameters applied, 0.85, 0.90 and 0.95 respectively. In the first two cases (i.e. 10- and 20-AGV cases), enough iterations seemed to be performed with 0.85 as the value of the cooling

parameter, resulting in 57 iterative processes. Although better AGV assignment plans can be obtained by more iterations, the degree of objective value improvements would be minor. In the 30-AGV case, 0.90 or 0.95 seemed to be appropriate for experiencing convergence in solution quality. Since the number of iterations is directly related to calculation time, the application of 0.90 to the cooling parameter would require around half of the observed computation time in *Figure 4-5*.

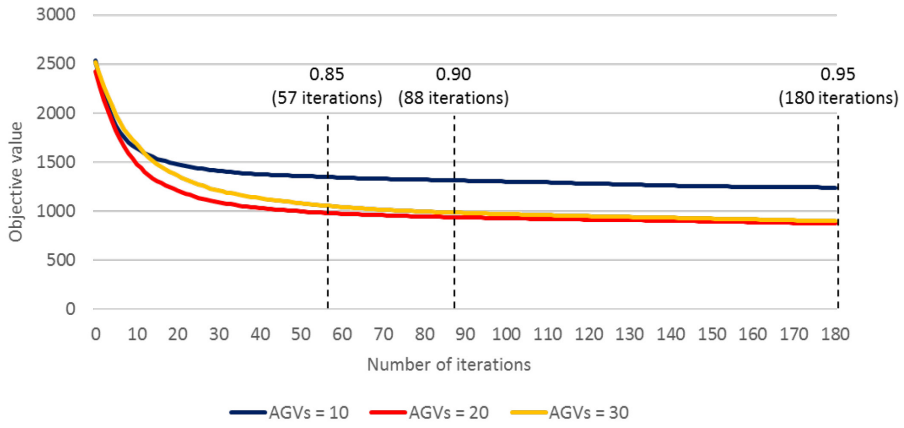


Figure 4-6. Objective value changes by iterations.

4.4. Summary

This chapter defines several key issues in AGV dispatching and provides a mathematical formulation of the defined AGV dispatching problem for use in container terminals, especially in the free-ranging AGV setting. As mentioned, in the literature there is a tendency to ignore task-reassignment although the involved transporters are on their setup travel. Therefore, this research suggests a flexible AGV dispatching technique with the ability to perform task-reassignment to reduce AGV arrival delay time in the quay and the yard and the setup travel time of AGVs. This technique can also work with a rolling horizon to provide AGV allocation plans periodically and handle errors from traffic congestion.

Due to inherent time complexity in the mathematical model, the branch-and-cut technique cannot be feasible for real-time applications in container terminals. Therefore, the SA-based

AGV dispatcher was developed to solve the defined problem within a limited time span. A series of pre-tests were performed to identify the computational power of the SA-based AGV dispatching algorithm, and the validity of the model for real-time applications was shown.

Chapter 5. Advanced Trajectory Planning Framework

This chapter suggests and describes an advanced fleet trajectory planning algorithm. This algorithm includes (1) a path planner based on the idea of Dubins curves, (2) a strategy-based acceleration generator and (3) a fleet trajectory planner (i.e. coordinator) constructed in the framework of simulated annealing. The algorithm periodically updates vehicular trajectories based on a rolling horizon to achieve a high level of vehicular journey flexibility.

The path planner diversifies early vehicular manoeuvres by adding a straight section or curve onto Dubins curves. Such path adjustments help vehicles to avoid collisions that could be inevitable in Dubins curves. Also, the planner provides feasible paths for normal car-like vehicles travelling at over their maximum turning speed. This is because resulting paths can also start with straight sections if required, which is not allowed in path planning based on Dubins curves.

The strategy-based acceleration generator defines several travel strategies, such as for cruising and stopping, instead of simply utilising a constant speed rate or the maximum speed of vehicles. This feature makes it possible to suggest multiple trajectories per path. The suggested generator also guarantees less acceleration changes compared to random-based acceleration generation. Therefore, in real applications, the strategy-based planner is likely to be favourable due to control simplicity.

The path planner and the acceleration generator are joined with the framework of simulated annealing to achieve AGV fleet trajectory planning. The fleet trajectory planner combines vehicular trajectories and finds the optimal trajectory set for a vehicle fleet based on a defined cost function. This cost function does not simply utilise conventional factors only, such as the total required travel distance. Instead, it tries to reduce the total required path length while preventing vehicles from staying for a long time in operation to avoid deadlocks.

5.1. Novel Path Generation Algorithm for Free-ranging AGVs

This section introduces and describes a novel path generation algorithm based on Dubins curves. The model improves the weaknesses of Dubins curves mentioned in *Section* 3.6.4 while increasing the flexibility of path shapes to avoid possible conflicts between agents. Also, it considers the speed rates of the agents at each planning start time, as well as their posture, to create follow-able paths when the speeds are over their maximum curve speed. It can also work with a rolling horizon, and therefore it allows the vehicles to reuse their currently-used path instead of employing new ones. It also provides imaginary paths for vehicles being served by other resources, such as cranes, to make them hold at their current location.

5.1.1. Path Definition and Data

In this research, trajectories are discrete by time; therefore, an edge list can represent a path. An edge refers to a path segment including its source, destination, distance and type. The first two elements each are an x- and y-coordinate pair, and the distance is subject to the Euclidean distance. The edge type is either a curve or straight. The path has the following additional data based on its edge list: a cumulative edge length list, the path length, a section type list, a section length list and a cumulative section length list.

Figure 5-1 shows the structure of a path with its fundamental data.

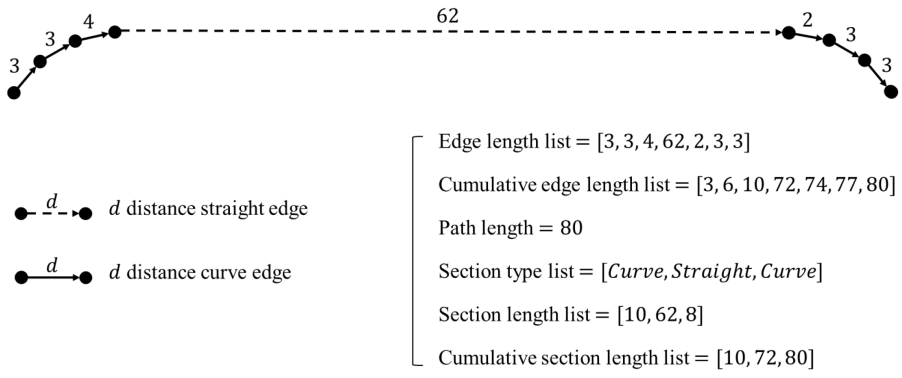


Figure 5-1. Path structure with principal data

5.1.2. Key Features

This subsection introduces and explains two core ideas for adjusting Dubins curves in shape. The first idea is the generation of straight path segments from vehicular starting postures, and the second allows vehicles to perform different and additional steering from their first turn in Dubins curves. The first idea helps to suggest feasible paths for nonholonomic vehicles travelling at over their maximum turning speed. The combination of the two ideas diversifies vehicular paths per origin-destination pair, which is especially required in fleet trajectory planning for collision avoidance.

Straight Addition

The presented path generation model creates a vehicular path that can begin with a straight section including the minimum deceleration distance to the maximum curve speed of a vehicle. Since moving straight as the first movement causes a longer path as shown in *Figure 5-2*, Dubins curves are designed to begin with curved sections. However, the feature makes container delivery AGVs impossible to always implement resulting paths based on Dubins curves due to their higher linear speed.

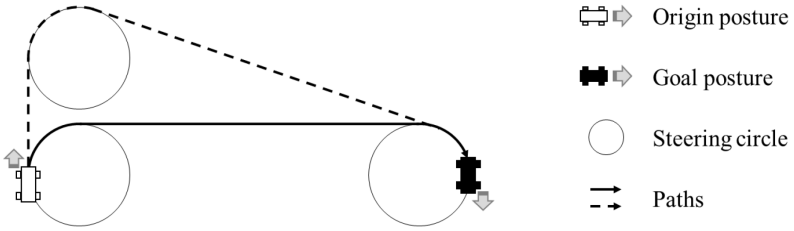


Figure 5-2. Impact of the first straight movement on path length.

Additionally, the model allows each vehicle to move straight more than its minimum required deceleration distance. As mentioned, there is a possibility of its shortest path being infeasible due to expected conflicts with other agents' paths. The straight addition technique allows the path generator to suggest various routes per origin-destination pair.

The path planner creates such an additional straight segment for a vehicle based on a user-defined probability value less than or equal to 1. Where a random value is less than or equal to the defined probability value, the model adds a straight path segment on the essential one. The length of the added part is a random-based value less than a predetermined maximum length.

Detour Curve Generation

Although the straight addition technique diversifies possible links between two vehicular postures, unavoidable collisions exist when the threats are on the straight path segments as depicted in *Figure 5-3*.

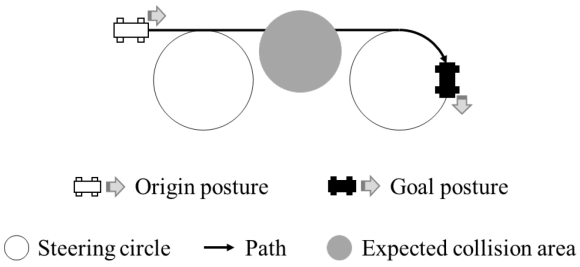


Figure 5-3. Expected collision in a linear path section.

The vehicle in the figure could use different steering circles located at the first turning point and the destination; however, it still cannot avoid the expected conflict with its possible turning movements as shown in *Figure 5-4*. To resolve the problem, the model has a function that employs an additional steering movement after the first steering movement (*Figure 5-5*).

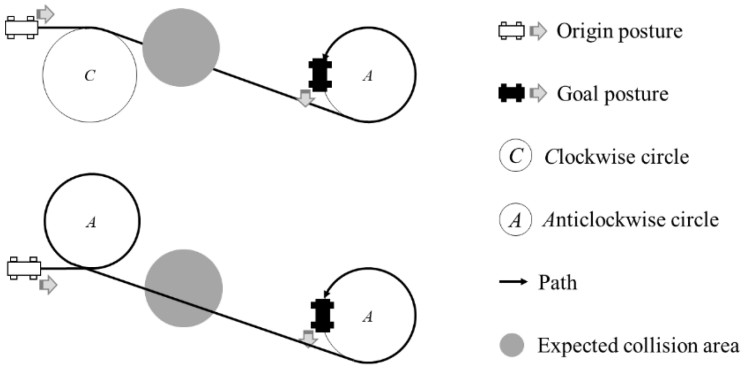


Figure 5-4. Steering circle infeasibility.

Figure 5-5 contains four possible cases in which the path generator adds a steering circle on one of the near-source circles while connecting one of the near-goal circles with a tangent line. In all the cases, the vehicle avoids the obstacle although it needs to travel longer. The second straight lines connected to the near-goal circles can also be replaced with curves when relevant steering circles are close.

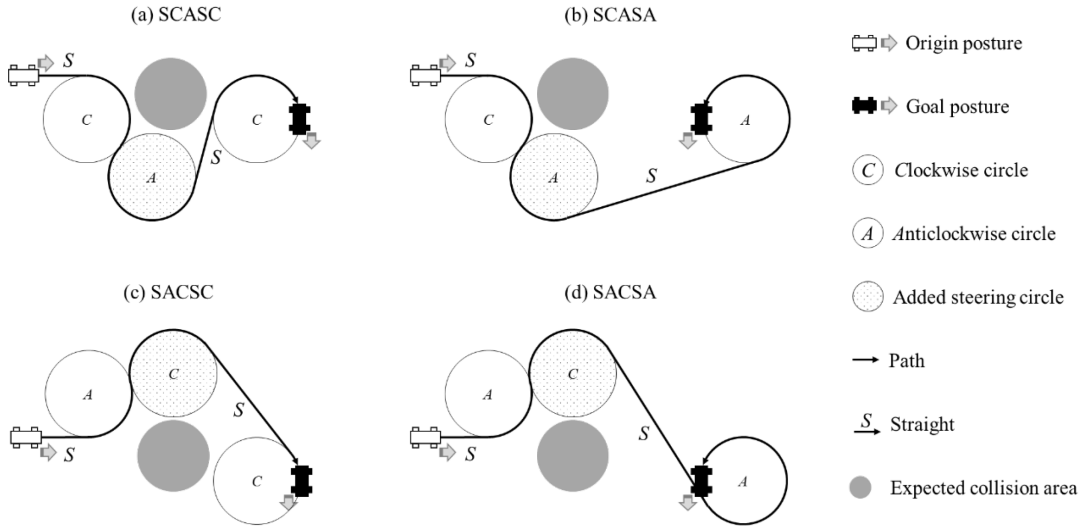


Figure 5-5. Detour curves.

The model does not fix the positions and the sizes of the optional circles for suggesting various paths. *Figure 5-6* describes the graphical expression of drawing the new circles subject to the user-defined parameter u referring to a directional range.

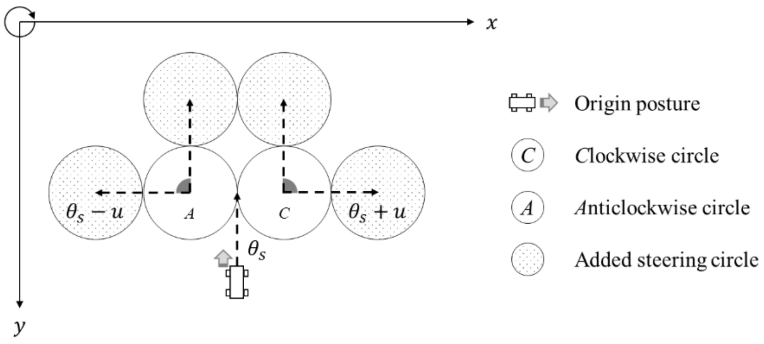


Figure 5-6. Graphical expression for the steering circle addition.

5.1.3. Path Classifications

Similar to Dubins curves, the paths provided by the proposed path planner can be expressed by the combination of path segment types, each of which fills a slot. The model introduces and uses N meaning no additional steering as well as utilising the existing three segment types (i.e. C, A and S). It also provides two more path segment slots compared to the basic version. One is the first straight movement, and the other is the third for representing the additional steering undefined in the fundamental model.

As a result of slot expansion, the model covers 12 path types. Each of them starts with S to realise the first rectilinear movement. The length can be 0. After the straight section, C or A exists as the first turning movement. The model also determines whether it takes an additional steering action after the first turn. Since a steering circle cannot externally meet circles with the same steering direction, the second steering direction at the third slot becomes C (or A) where the first is A (or C). The fourth slot has one of *S*, *C* and *A*, and its element connects to one of the final steering circles. When the model selects S, it can use any of the final circles; otherwise, it should choose one with the opposite steering direction.

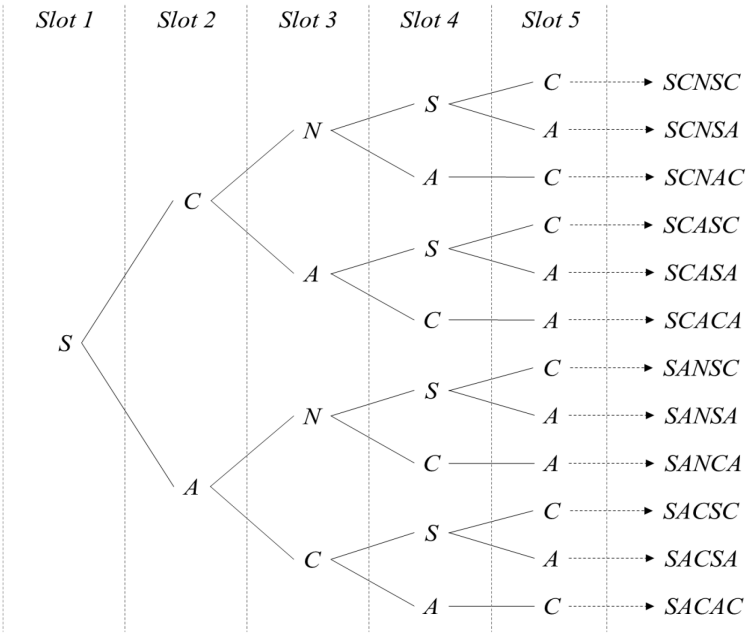


Figure 5-7. Path types in the path generator (logical).

Figure 5-7 shows the summary of all the possible path structures in the algorithm, and *Figure 5-8* visually describes all the path types.

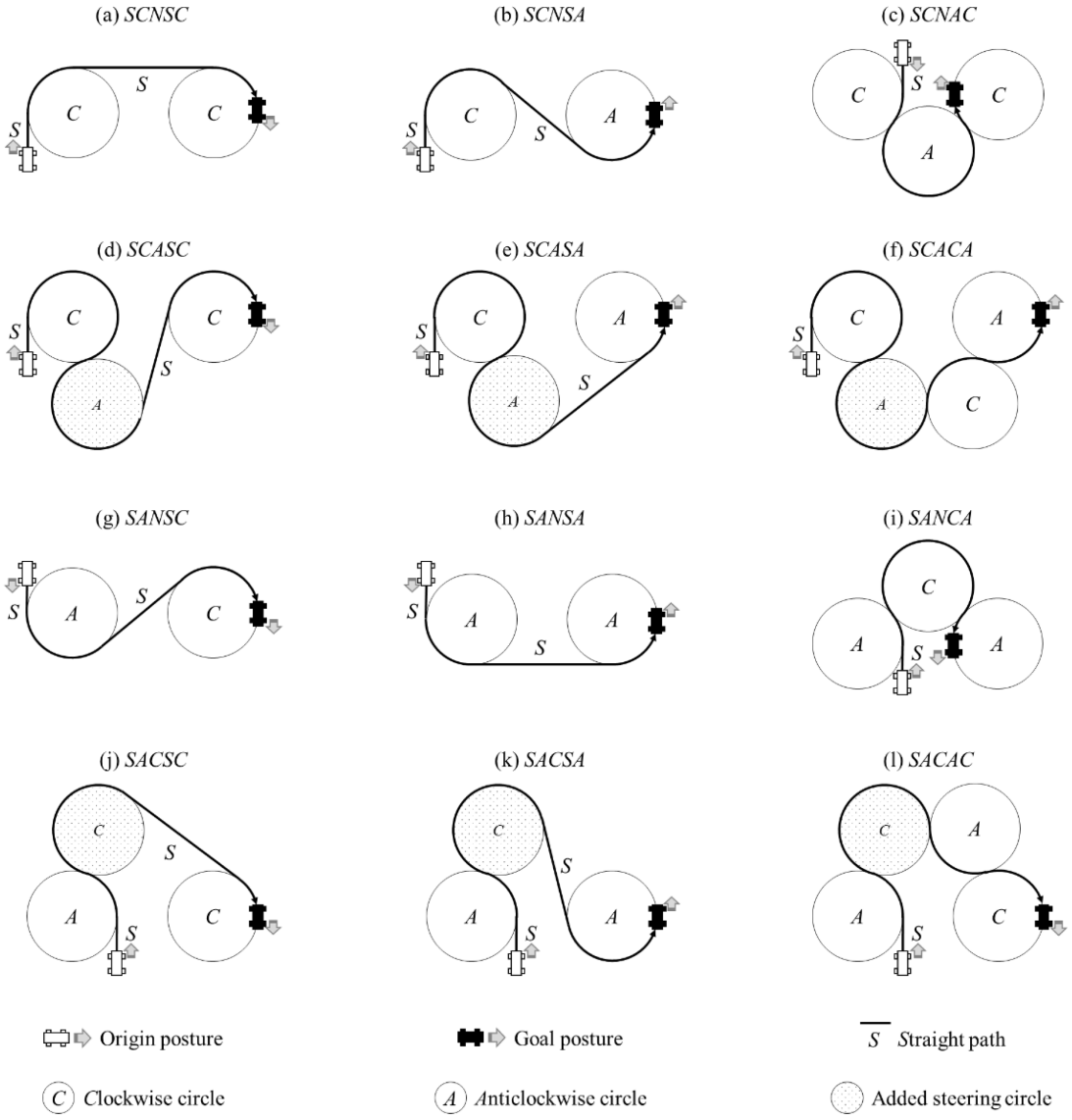


Figure 5-8. Path types in the path generator (graphical).

5.1.4. Path Generation Inputs and Setting Parameters

In addition to a source-goal location pair of a vehicle, the path creator requires additional input data the basic model does not utilise. In each path generation process, the speed at planning start time needs to be known to check that the vehicle can turn immediately. Since the technique can work with a rolling horizon, it allows the vehicle to reuse its current path,

instead of suggesting a new one. So, the model requires the path in use and the cumulative travelled distance on the route as well. It uses the aggregate value to discard the travelled range on the road. Also, as the shortest path may not be the best option in fleet trajectory planning, whether it tries to provide the shortest path for the origin-destination pair is also an input parameter as a binary value (i.e. true or false). The last input parameter can be the load state of the AGV for adjusting its maximum speed by its cargo weight (optional).

Table 5-1. Input parameters for the path generation algorithm

Group	Input parameters	Type
Used in the basic version	Source posture at a given planning start time	Posture
	Required posture at the goal	Posture
Not used in the basic version	Speed at the planning start time	Number
	Current path	Path
	Cumulative travelled distance on the used path	Number
	Shortest	Boolean
	Loaded	Boolean

As well as the input parameters in *Table 5-1*, the algorithm requires setting parameters (*Table 5-2*). The main difference between the two parameter types is that the former is dependent on each agent whereas the latter is not. Some of the setting parameters are subject to a given vehicle operation domain. In other words, all paths should exist in the field. Since the model employs a rectangular map as default, the minimum and the maximum coordinate sets are domain setting parameters.

Some are related to the essential straight segments starting from the source postures of vehicles. Firstly, the path generator should know their maximum curve speed rate. Moreover, if the model distinguishes between the loaded and the empty states of the AGVs for controlling speeds, the setting value will be split for the two cases. The algorithm also needs to know the minimum acceleration (i.e. the maximum deceleration) to calculate required deceleration distances to the vehicle's target speeds for turning and stopping.

In addition to the required straight segments, the path planner can generate additional rectilinear movements for various path shapes requiring some relevant setting parameters. It requires a predetermined probability value between 0 and 1 to decide whether the model generates additional linear path segments. Also, the model has a setting value as the maximum additional straight length.

Table 5-2. Setting parameters for the path generation algorithm

Group	Setting parameters	Type
Environment	Minimum point on a given domain	Point
	Maximum point on the domain	Point
Required straight	Maximum curve speed in empty states	Number
	Maximum curve speed in loaded states	Number
	Minimum acceleration	Number
Additional straight	Existence probability of additional straights	Number
	Maximum additional straight length	Number
Curves	Minimum steering angle (in degrees or radians)	Number
	Maximum steering angle (in degrees or radians)	Number
	Curve approximation angle size	Number
	Directional range of the additional steering circles	Number
Others	Detour generation trial number	Number
	Path reuse rate	Number
	Arrival range	Number

The minimum and maximum allowable steering angles are considered as setting parameters for curved sections. Additionally, the algorithm utilises an angle size to approximate curves. To be more specific, the parameter helps to convert a curve into a set of edges as shown in *Figure* 5-9. The figure depicts four approximated curves by different curve smoothing angle sizes: 45 (in black), 30 (in red), 15 (in blue), and 0.1 (in green) in degrees. The converted edge lists become smoother by decreasing the angle size. Additionally, the model needs to have a setting parameter to create the detour steering circles explained in *Section* 5.1.2. Thus, it employs a directional range parameter for limiting the locations of the new steering circles.

Figure 5-9. Curve smoothing.

The final setting parameters are the number of detour generation trials, a path reuse rate and an arrival range. In some cases, the path generator cannot quickly discover a new detour due to the closeness of the two input postures. This situation would occur when the algorithm works with rolling horizons because vehicles become closer to their destination over time while requiring new paths. Therefore, the model needs to limit the number of detour creation trials. Where it fails to find a candidate route for a vehicle, it returns the shortest path if possible, or reuses its current path. Also, the model allows the vehicles to keep using their current path based on a probability value (i.e. the path reuse rate). Finally, the arrival range identifies when a vehicle arrives at its destination.

5.1.5. Algorithm Descriptions

The proposed path generation technique includes three sub-algorithms to generate vehicular paths. The first creates a new path without employing the current path of a vehicle. The second reuses and trims the current path to maintain the planned travel. The third realises the case where vehicles need to stay at specific service locations, such as cargo handling stations.

Main Logic

The main logic of the algorithm consists of three parts. Given a path generation problem instance with a vehicle, the first part provides a new candidate path for the vehicle. It can select one among the 12 path types in *Section* 5.1.3 as well as the shortest path. The second part trims the current path by abandoning the vehicle's travelled path range to make the path

begin from the vehicular posture in the input set. The third creates a significantly short path to make the vehicle stop, only used in arrival cases.

```
1.  Algorithm GeneratePath is
2.    Input: being-used path p_b
3.           travelled distance on being-used path h
4.           source posture P_s(x_s, y_s, θ_s)
5.           goal posture P_g(x_g, y_g, θ_g)
6.           speed s
7.           shortest path required r
8.           arrival range α
9.           path reuse rate β
10.          direction range u
11.          maximum detour generation trial number m
12.   Output: path p
13.
14.   distance d ← GetDistance(P_s, P_g)
15.   if d ≤ α do
16.     p ← GeneratePathToStay(P_s, P_g)
17.   else
18.     shortest path p_s ← GenerateNewPath(P_s, P_g, s, true, u, m)
19.     if p_b == null & p_s ≠ null do
20.       p ← GenerateNewPath(P_s, P_g, s, r, u, m)
21.     else p_b ≠ null & p_s ≠ null do
22.       if RandomRealNumber(0, 1) ≤ β do
23.         p ← TrimBeingUsedPath(p_b, h)
24.       else
25.         p ← GenerateNewPath(P_s, P_g, s, r, u, m)
26.     else p_b ≠ null & p_s == null do
27.       p ← TrimBeingUsedPath(p_b, h)
28.     else
29.       p ← null
30.   if p == null & p_s ≠ null do
31.     p ← p_s
32.   return p
33. End GeneratePath
```

Figure 5-10. Main logic of the path generator.

Figure 5-10 shows the main logic of the proposed path planning model. The last four input parameters (i.e. α, β, u and m) are external setting parameters, and therefore the actual relevant code does not take them as inputs. Also, the data structure of vehicular postures includes three values: x- and y-coordinates and orientation.

The logic works as follows: In the case where the vehicle can be regarded as being at its destination (Line 15), the path generator makes the AGV stay (Line 16). Otherwise, it tries creating a new path or trims the current one by checking the following two conditions: (1) whether an input path exists and (2) whether the shortest path can be generated. When the first condition is false, and the second is true (Line 19), the path generation algorithm employs the first method (Line 20). However, where both are true (Line 21), it chooses one of the two methods based on a probability value (Line 22 to 25). When only the first is true (Line 26), it selects the second method (Line 27). In the case where both are false (Line 28), the algorithm sets p to a null reference (Line 29). Since the first strategy may fail to find a detour in Line 20 and Line 25, the algorithm utilises the shortest path p_b instead if possible (Line 30 to 31).

Sub-Logic 1: New Path Generation

The first sub-logic produces a new route for a vehicle without utilising its current path. It first attempts to generate a detour several times based on a predetermined number of detour generation trials. If it fails, it returns a null reference.

A path suggested by the model possesses five slots, each of which contains an edge list. This is denoted as E_i where i refers to the corresponding slot index (from 1 to 5). The sizes of the lists can vary. The first set can have only straight edges whereas E_2, E_3 and E_5 can include curves but cannot have linear edges. On the other hand, the fourth can possess either. In the process of producing the edge groups, the model does not consider zero-length edges when merging all of them to form a path.

The planner generates the five edge lists in different ways depending on whether it needs to return the shortest path of a given problem instance. When it creates the optimal route in length, E_1 only includes the minimum required deceleration distance for turning, and E_3 is empty. Also, the vehicle utilises the maximum steering angle for E_2, E_4 and E_5. The model suggests six path types (i.e. *SCNSC*, *SCNSA*, *SCNAC*, *SANSC*, *SANSA* and *SANCA*) and selects the shortest option. When the algorithm provides a detour, E_1 can include an additional straight path segment, and E_3 can have curve edge elements. The steering circles' radii can also vary in E_2, E_3, E_4 and E_5 depending on vehicular kinematic conditions.

Except for E_1, the rest sets (i.e. E_2, E_3, E_4 and E_5) are dependent on relevant steering circles. Since the first edge only handles a line from the source point, its creation can be independent of turning movements whereas the other edge lists cannot. Even in the case where E_4 contains a single straight edge, its generation should be subject to two steering circles in (1) E_2 or E_3 and (2) E_5 respectively.

The method generates steering circles in the sequence $'C_2 \rightarrow C_5 \rightarrow C_3\ (\rightarrow C_4)'$ where C_i refers to a steering circle set for E_i. C_2 includes two circles that meet at the destination of the last element in E_1. Similarly, C_5 has two steering rings on the goal posture. After setting the two groups, it draws two circles for C_3, each of which meets one of the members in C_2. Unlike the other circle sets, C_4 possesses four steering circles. Each of them meets one of the steering rings in C_2 or C_3 and one in the last circle set at the same time.

The circles in the sets are expressed in two ways. Let i and k be slot indices and j and l be steering direction indices that can be either a (i.e. anticlockwise) or c (i.e. clockwise). Those in C_2, C_3 and C_5 are denoted as $c_{i,j}$. On the other hand, the members in C_4 are denoted as $c_{i,j}^{k,l}$. This refers to the steering circle in the i^{th} slot with the steering direction j, meeting with the steering circle in the k^{th} slot with the steering direction l as the previous turning movement (i.e. $k < i$). So, the following four circles are in the set: $c_{4,c}^{2,a}$, $c_{4,c}^{3,a}$, $c_{4,a}^{2,c}$ and $c_{4,a}^{3,c}$.

Figure 5-11 describes the logic of creating a new path. In the logic, the function GetRadius takes a binary value that represents whether it utilises the minimum steering radius of the vehicle. When the binary variable is 0 (i.e. false), the function returns a radius within a range of its allowable steering radii; otherwise, it does the minimum steering radius by adopting the maximum steering angle.

```
1.  Algorithm GenerateNewPath is
2.     Input: source posture P_s(x_s, y_s, θ_s)
3.            goal posture P_g(x_g, y_g, θ_g)
4.            speed s
5.            shortest path required r
6.            direction range u
7.            maximum detour generation trial number m
8.     Output: path p
9.
10.    trial number t ← 1
11.    while t ≤ m do
12.       P_f ← GetPostureForFirstTurning(P_s, s, r)
13.       C_2 ← GenerateC_2(P_f, GetRadius(r))
14.       C_5 ← GenerateC_5(P_g, GetRadius(r))
15.       if r ≠ true do
16.          C_3 ← GenerateC_3(θ_f, u, C_2, GetRadius(r))
17.       C_4 ← GenerateC_4(C_2, C_3, C_5, GetRadius(r))
18.       feasible path set F ← GetFeasiblePaths(P_s, P_f, C_2, C_3, C_4, C_5, P_g)
19.
20.       if r == true or F = empty do
21.          return null as p
22.       if r == true or F ≠ empty do
23.          return the shortest in F as p
24.       if r ≠ true or F ≠ empty do
25.          return an element in F as p
26.       else
27.          t ← t + 1
28. End GenerateNewPath
```

Figure 5-11. Logic for generating a new path.

The two sub-methods GenerateC_2 and GenerateC_5 work in similar techniques. Each creates two circles that meet at the point included in its input posture. The centres of the steering circles are away by the input radius in the directions of the input heading $\pm$ 0.5π respectively. GenerateC_3 utilises the method described in *Figure 5-12*. The third parameter C_p is either $c_{2,c}$ or $c_{2,a}$. The function Translate(x, y, d, h) implements a translation of (x, y) by the distance d in the direction of h (Line 15). The function GenerateC_4 in *Figure 5-11* works similarly to Dubins curves for the middle curve section; however, the suggested algorithm utilises the steering circles in C_3 (i.e. for detour steering movements) as well as those in C_2.

```
1.   Algorithm GenerateDetourSteeringCircle is
3.      Input: heading at the first turning posture θ_f
4.             direction range u
5.             previous steering circle C_p(x_p, y_p, r_p, d_p)
6.             detour steering circle radius r
7.      Output: steering circle C(x, y, r, d)
8.
9.      if d_p == Clockwise do
10.        d ← Anticlockwuse
11.        direction h ← θ_f + u RandomRealNumber(0, 1)
12.     else
13.        d ← Clockwise
14.        h ← θ_f − u RandomRealNumber(0, 1)
15.     (x, y) ← Translate(x_p, y_p, r_p + r, h)
16.     C ← CreateCircle(x, y, r, d)
17.     return C
18.  End GenerateDetourSteeringCircle

1.   Structure Circle C(x, y, r, d) is
2.      Data: x-coordinate of the centre x
3.            y-coordinate of the centre y
4.            radius r
5.            rotation direction d
6.   End Circle
```

Figure 5-12. Logic for the detour steering circle generation.

The method GetFeasiblePaths has a role in identifying the locational feasibility of the generated steering circles for each path type by checking each resulting route is in the domain. *Table 5-3* summarises target steering circles per path structure.

Table 5-3. Involved steering circles per path type

Path type	Circle sets			
	C_2	C_3	C_4	C_5
SCNSC	$c_{2,c}$	-	-	$c_{5,c}$
SCNSA	$c_{2,c}$	-	-	$c_{5,a}$
SCNAC	$c_{2,c}$	-	$c_{4,a}^{2,c}$	$c_{5,c}$
SCASC	$c_{2,c}$	$c_{3,a}$	-	$c_{5,c}$
SCASA	$c_{2,c}$	$c_{3,a}$	-	$c_{5,a}$
SCACA	$c_{2,c}$	$c_{3,a}$	$c_{4,c}^{3,a}$	$c_{5,a}$
SANSC	$c_{2,a}$	-	-	$c_{5,c}$
SANSA	$c_{2,a}$	-	-	$c_{5,a}$
SANCA	$c_{2,a}$	-	$c_{4,c}^{2,a}$	$c_{5,a}$
SACSC	$c_{2,a}$	$c_{3,c}$	-	$c_{5,c}$
SACSA	$c_{2,a}$	$c_{3,c}$	-	$c_{5,a}$
SACAC	$c_{2,a}$	$c_{3,c}$	$c_{4,a}^{3,c}$	$c_{5,c}$

Sub-Logic 2: Path Trim and Reuse

Three edge groups exist in the second sub-logic. The first group contains all path segments passed by the target vehicle already. The second has the edge the object is on. The third includes the edges that the vehicle has not visited yet.

The logic categorises each edge as follows: Let the edge to be investigated be e_i where i refers to the edge's index, the model identifies the cumulative path length by e_{i-1}, denoted by p_i. Next it checks that the vehicle has not visited e_i by the statement '$h \leq p_i$' where h is the travelled distance of the vehicle on the given path. A binary variable q_i denotes the condition: where the vehicle has not visited the edge, the value is 1 (i.e. $true$) and 0 (i.e. $false$) otherwise. It also identifies that the vehicle is on e_i, which is denoted by a binary variable o_i. When the

condition $p_i < h < c_i$ is met where c_i is the cumulative length by e_i, the value is set to 1, otherwise 0.

```
1.  Algorithm TrimBeingUsedPath is
2.    Input: being-used path p_b
3.           travelled distance on being-used path h
4.    Output: path p
5.
6.    edges of p_b E
7.    new edge list N
8.    for each e_i in E do
9.      length by previous edge p_i ← GetPreviousLength(e_i, E)
10.     length by current edge c_i ← GetCurrentLength(e_i, E)
11.     not visited q_i ← h ≤ p_i
12.     in the edge o_i ← p_i < h < c_i
13.     if q_i == true do
14.       N ← AddOnList(e_i)
15.     else o_i == true
16.       source s ← SetNewSource(p_b, h)
17.       goal g ← GetGoal(e_i)
18.       type t ← GetType(e_i)
19.       trimmed edge e ← CreateEdge(s, g, t)
20.       N ← AddOnList(e)
21.   p ← ConstructPath(N)
22.   return p
23. End TrimBeingUsedPath
```

Figure 5-13. Logic for path trim and reuse.

Based on the identified data (i.e. p_i, q_i and o_i), the model processes each element on the edge list of the path as shown in *Figure 5-13*. In the case where the vehicle has not experienced e_i as described in Line 13, the algorithm adds the element on the edge list of a result path as they are (Line 14). On the other hand, when the vehicle is on the edge (Line 15), the path generator replaces the corresponding edge with a new one. It firstly discards the travelled range from the given path and identifies the source location of the new edge (Line 16), and the model reuses the goal and the type of the reference edge (Line 17 to 19). When the target edge is not

included in any of the two conditions, the logic does not consider the edge since it means that the vehicle has already passed over the edge. The logic constructs a path with the resulting edge set as the last step.

Sub-Logic 3: Path Generation for Staying

The third sub-logic (i.e. GeneratePathToStay in *Figure 5-10*) renders a target vehicle remaining its current position in the case where the AGV is waiting for a service or being handled by a resource, such as a crane.

If the model creates an edge that its origin and the destination are identical, it cannot be used for the target vehicle. In this case, the edge is not a line segment, but a point. It means that the model cannot identify the heading of the vehicle with the edge because it cannot compute the derivative at the point.

Therefore, the model provides an imaginary origin point to produce a non-zero edge that helps to maintain the vehicular heading as required. The imaginary origin is placed away from the destination by a significantly short distance in the opposite direction of the required heading at the goal. The logic then creates an edge by connecting the imaginary source and the goal point. The edge represents a resulting path and does not initiate any movements of the vehicle.

5.2. Strategy-based Acceleration Determination

Acceleration rates on planned paths can be determined by a constant speed strategy. This primitive method controls vehicles to cruise by applying an acceleration of 0 m/sec^2. In this case, it produces one trajectory per path and therefore decreases the variety of trajectories. Also, vehicles would experience adverse situations that could have been handled by the deceleration or the stop of some of the vehicles.

Instead, the proposed trajectory planner allows flexible acceleration rates while determining the values successively. For example, when it determines four acceleration rates to be applied from time 0 to 3 for a vehicle, it sets an acceleration at time 0 and then for time 1, 2 and 3 in a sequence. In each acceleration decision stage, one or more available acceleration rates exist subject to the path section's type, the vehicle's kinematic limitations and its load state.

One possible acceleration decision rule is the selection of the highest possible acceleration value. Since it leads to the highest speed rate, it guarantees the quickest travel. However, like the constant speed strategy, it limits the number of trajectories on a path to one, and therefore unavoidable collisions would frequently happen.

Another simple approach is a random selection method. It randomly chooses one allowable acceleration, and therefore it can suggest many trajectories per path, including the quickest one. However, when it works in a metaheuristic-based trajectory planning algorithm, this strategy may not be appropriate as it would take more iterations to find a quick trajectory.

As an alternative, the trajectory planner randomly chooses one of the following travel strategies to be applied for a given period, such as 0 to 5, to handle the mentioned issues: *Quickest*, *Cruising*, *Staying* and *Randomness*. The first rule makes the planner choose the highest acceleration rates to guarantee the fastest movement during the time. When the model works with *Cruising*, it controls the target vehicle to adjust its speed rate at the maximum curve speed for cruising regardless of path section types. The acceleration planner can make the vehicle stop by using *Staying* or randomly determines the vehicle's acceleration rates (*Randomness*).

5.3. Fleet Trajectory Planning

The FTPP refers to a decision-making problem of providing a set of trajectories for a given vehicle fleet. It is almost the same as single agent trajectory planning except for the feature that the fleet trajectory set should guarantee conflict-free movements whilst achieving an

objective, such as the minimisation of the total travelled distance or time of the fleet. There may be situations where a sub-optimal trajectory for a single vehicle may be chosen if that trajectory were to improve the performance of the fleet of vehicles.

The application of a metaheuristic to the FTPP is an appropriate option because of the inherent complexity of the problem. As described in *Section* 3.5, the mathematically-defined FTPP falls under MIP; therefore, a solver firstly can utilise a branch-and-cut algorithm. However, although the problem maintains simplicity without including any sophisticated constraints, such as path smoothness, employing more agents significantly increases the time required to provide a valid solution. Therefore, metaheuristics are predominant in the field, especially GAs.

Such population-based metaheuristics, however, would not be appropriate. As the time to create a vehicular trajectory (i.e. a gene) based on *Section* 5.1 and *Section* 5.2 is not negligible, a trajectory generator based on GAs would require significant computational effort even when providing an initial population as shown in *Figure 5-14*. It used considerable time compared to a given period (i.e. 10 seconds) in which the trajectory generator should provide trajectories. This means that the generator cannot implement enough iterative processes. Therefore, real-time vehicle control would not be achievable without an HPC facility.

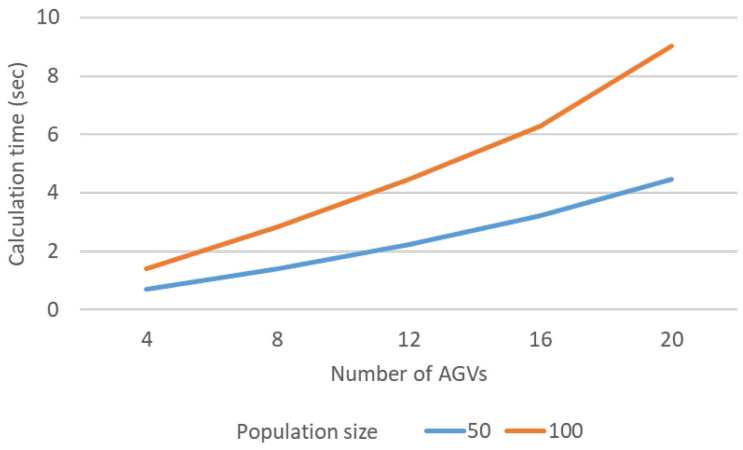


Figure 5-14. Average time of initial population generation (300 trials).

Although TS does not have the drawback in the population-based technique, it would hardly define a tabu list while requiring more memory compared to SA. In the fundamental TSP with n cities, a partial visiting sequence can be an element of the tabu list. Therefore, each of the sub-elements is a value in a finite set (i.e. one of the city indices in the city set). However, in the FTPP, a tabu list's element is a group of trajectories for several vehicles in the fleet. Unlike the TSP, a sub-element in the FTPP's tabu list is not a value or member in a finite set. Therefore, this feature causes difficulty in defining such a tabu list.

5.3.1. Alternative Simulated Annealing Based Fleet Trajectory Planner

Because of the drawbacks in TS and GA, the proposed fleet trajectory planning algorithm adopts the framework of SA. With an FTPP instance, it first creates an initial solution that includes a trajectory set for the fleet by utilising the path and acceleration generation techniques explained in *Section* 5.1 and *Section* 5.2 respectively. It also informs the initial solution's quality and collision-causing trajectory pairs. The algorithm then improves the solution in the annealing process. However, unlike the SA-based AGV dispatcher, solutions handled by the trajectory planner are not always feasible and potentially contain collision events. Therefore, the model terminates when at least it finds a collision-free trajectory set.

Also, the model uses a rolling horizon approach. Thus, every solution covers a predetermined planning horizon, such as four seconds (*Figure 5-15*). While the AGVs are tracking their reference trajectory, the algorithm calculates the next trajectories starting in the middle of the currently-used trajectories regarding time.

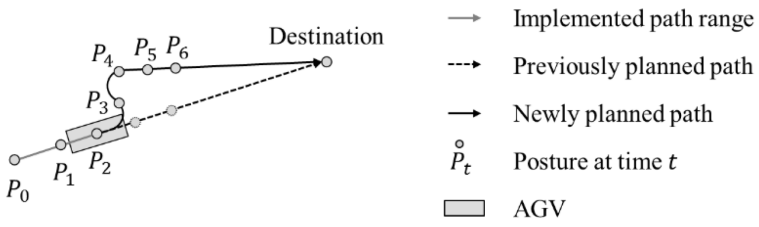


Figure 5-15. Trajectory planning with a rolling horizon.

Figure 5-16 shows the flowchart of the proposed SA-based trajectory planner. With reference to the figure, the algorithm is described as follows.

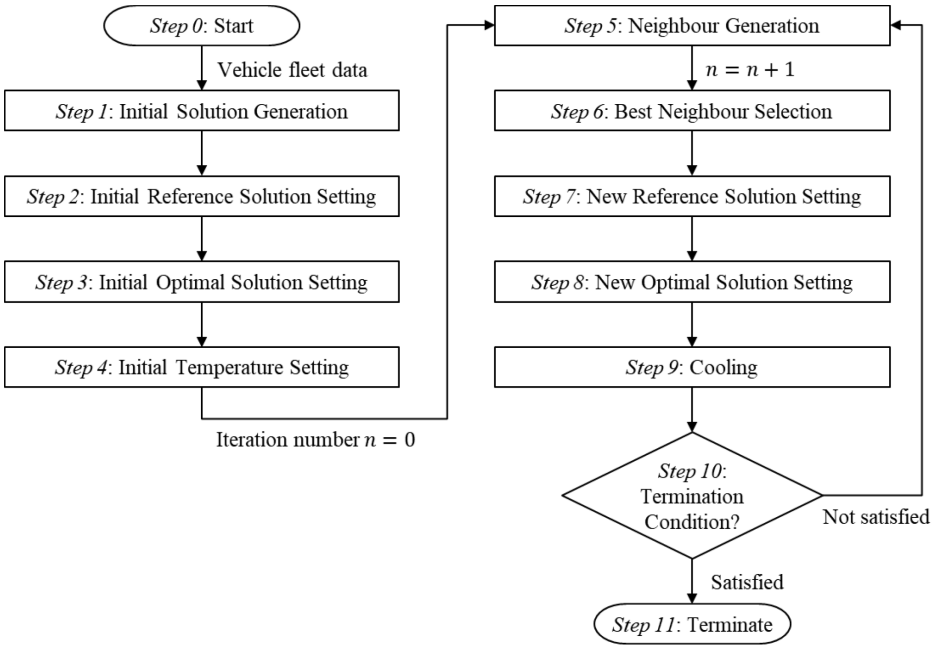


Figure 5-16. Overview of the SA-based fleet trajectory planning algorithm.

Given a required planning horizon, a trajectory planning problem instance initiates the model (*Step 0*). The instance includes a target AGV fleet with the vehicles' source and goal postures and current states, such as the speed rates at the starting time of the horizon. Such initiations occur at regular intervals, each of which refers to a trajectory execution period.

The planner creates an initial solution (*Step 1*). Since the quickest trajectories would be ideal, the initial solution contains the optimal travel plan for each vehicle in path length and travel time. Therefore, the first-suggested option may include collision events between some of the agents. The other two types of data required (i.e. level of solution quality and a set of vehicular trajectory pairs resulting in collisions) are determined as follows:

The distance the vehicles will travel towards their destination and the total path length are the primary criteria for evaluating a trajectory set while considering safety. However, their use is likely not to be appropriate for the algorithm since they do not have resistance to deadlock situations. For instance, when two vehicles are rather close, and each is moving to the other's

locational direction, their central controller would suggest a trajectory set as shown in *Figure 5-17* (a) and *Figure 5-17* (b). The vehicles would take the shortest paths because the detours are inferior with respect to the mentioned performance measures. After an execution period, the planner would make the vehicles stay at their final location rather than provide detours that increase the total path length and guide the vehicles away from their goal location (*Figure 5-17* (c)).

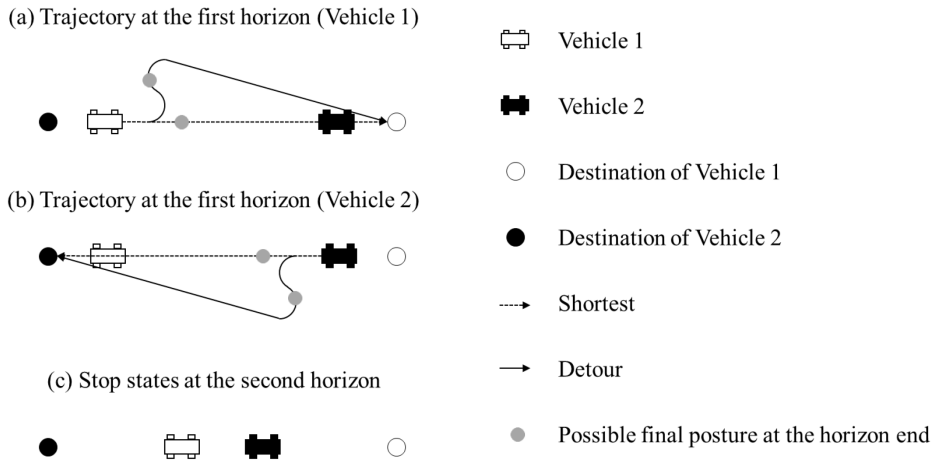


Figure 5-17. Trajectories with the basic evaluation factors.

Therefore, the algorithm employs a new trajectory-set evaluation function while considering the drawback. The function has three terms as described in (5-1). In the equation, F refers to the score of a trajectory set, and V is the size of the vehicle fleet. e_k^p refers to the travelled distance of vehicle k during the planning horizon p. f_k denotes the subtraction from the path length where vehicle k uses its current path to the newly computed path length. When the new path is shorter than the current path starting from the vehicle position at the planning start time, the corresponding term becomes positive.

The coefficient α adjusts the weights of e_k^p and f_k. M is a positive number that is sufficiently larger than any feasible fitness values, and c^p is the number of vehicular trajectory pairs causing collision events in the solution. The algorithm performs a series of collision tests for each trajectory pair at all time points. The following subsection will explain the threat detections in detail.

$$F = \sum_{k=1}^{V}\left(\alpha e_k^p + (1-\alpha) f_k\right) - Mc^p \quad (5\text{-}1)$$

Compared to the primary performance measures for trajectory planning, the equation has a different criterion (i.e. e_k^p) to escape deadlock situations. The sum of the term evaluates the total travelled distance of the vehicles regardless of their moving direction. So, when α is set to larger than 0.5, the model controls the vehicles to move in any direction although the resulting paths become longer. Such control happens in crowded areas since the algorithm essentially tries to improve trajectories in both path length and travelled distance.

After defining the initial solution, the algorithm sets the values of *Reference Solution* and *Optimal Solution* to the generated trajectory set (*Step 2 to 3*). It then determines the initial temperature. As explained in *Chapter 4*, the value closely works with the algorithm's cooling function and termination temperature. These need to allow the model to experience enough iterations during the execution horizon. The two temperatures and the linked parameter of the cooling scheduler can be determined by trial and error.

By using the defined reference option, the SA-based fleet trajectory planner creates a list of neighbour solutions (*Step 5*) by using the following two techniques: Firstly, a neighbour of a reference solution is a solution generated by amending a small part of the base. The proposed model has two strategies for determining the part to be changed (i.e. one or two target vehicles' trajectories) to provide a new option. The choice of the methods depends on whether the reference contains collision events. When the source is not a safe option, the algorithm selects a pair of vehicles in a state of danger as target vehicles. Otherwise, the trajectory planner randomly selects one vehicle. Then, the algorithm adjusts the applied acceleration rates of each of the target vehicles while employing the being-used path or provides a trajectory with a new route. The process continues until the model creates n neighbours where n refers to the neighbour list size.

Secondly, the SA-based fleet trajectory planner defines a term *Bizarre Neighbour* in which all vehicles in this neighbour decelerate and stop for safety. In many cases, a resolution of a

collision event between two vehicles results in another conflict with other agents. Thus, there is a possibility of solutions not being improved by the neighbour-generation technique. In other words, improving trajectories from a safe state would be better in finding a satisfactory trajectory set. The algorithm creates this solution as the first option only in the first iteration.

The next two steps are as follows: the model selects the best among the suggestions as the element for *Best Neighbour* (*Step 6*). In *Step 7*, it sets a new reference solution for use in the coming iteration by utilising the acceptance function (5-2) where T is the current temperature, s_n refers to the objective value of the best neighbour, and s_o means the optimal value in *Optimal Solution* before the solution update process. When s_n is larger than s_o, the equation always returns a value over 1, and therefore the algorithm updates the data field with the new solution. On the other hand, where $s_n < s_o$, the value is greater than 0 and less than 1. Thus, the algorithm chooses the best neighbour as the next reference based on probability.

$$P = e^{\frac{s_n - s_o}{T}} \tag{5-2}$$

In *Step 8*, the planner updates the value of *Optimal Solution*. It replaces the element in the data field with the best neighbour when this is superior to the existing option. Otherwise, it keeps its current solution. The algorithm then adjusts the temperature by employing the cooling function adopted by the SA-based AGV dispatcher in *Chapter 4* and identifies whether the procedure continues based on the following two conditions: (1) whether the model has reached the lowest temperature and (2) whether the current optimal option is safe. When both are satisfied, the algorithm returns the optimal trajectory set and terminates (*Step 11*). Otherwise, it starts the next iteration process by generating a new neighbour solution set (Go to *Step 5*).

5.3.2. Trajectory Safety

A safety clearance zone refers to an unshareable imaginary space around a vehicle that needs to be kept clear of other agents or obstacles to prevent possible collisions. An overlap of the

two zones is likely to lead to conflicts between the associated objects although it does not always mean a physical collision event will occur.

If all factors in a domain were controllable, the zone would be equal to a region of inevitable collision (RIC) regardless of control (LaValle, 2006; Karumanchi, Iagnemma and Scheding, 2013). This region is a swept area generated by a vehicle decelerating at its maximum deceleration to stop while considering all the allowable steering angles.

However, in practice, various types of errors exist forming the total system error (TSE) that needs to be considered in safety clearance zone generation, such as map definition error (MDE), navigation system error (NSE) and vehicle control error (VCE) (Schuster and Ochieng, 2011). The first refers to the difference between a defined map and the real environment that causes discrepancies between planned paths and expected routes in practice. The second covers positional errors between the real and estimated locations coming from positioning equipment. The third is the difference between planned trajectories and implemented ones occurring when controlling vehicles in practice. Under the assumption that there is no correlation between the error types, the TSE is the square root of the sum of each error type's square, as shown in (5-3). As well as the RIC, the total error is a primary input parameter to construct vehicular safety clearance zones.

$$TSE^2 = MDE^2 + NSE^2 + VCE^2 \tag{5-3}$$

The errors are technically-observed values, and each error type forms a probability distribution. Thus, a representative value for the TSE needs to be defined. This value is dependent on the safety level the users require; the higher level they want to achieve, the larger value they utilise to have a wider confidence interval. It consequently leads to larger safety clearance zones.

Such a safety clearance field is likely to be bounded by a simple shape, such as a circle or a polygon, to achieve the simplicity of the area shape itself and collision tests. In literature, circles, axis-aligned bounding boxes (AABB), and oriented bounding boxes (OBB) are predominant. The first bounding strategy is the simplest but causes redundant space on the

sides of a long and narrow vehicular swept area. This disadvantage also occurs in AABB which are changeable in size by rotation. However, OBB are rotatable rectangles that remain constant in size and shape regardless of the headings of the objects. Thus, redundant space on the sides is always smaller than or equal to those based on the others. OBB are therefore more suitable for the vehicles that operate in a crowded area, such as container terminals.

Determining a safety zone for a vehicle moving on a trajectory can be subject to a tube that covers a time window. This approach first identifies safety clearance zones on the trajectory in the period and bounds them altogether. It guarantees trajectory safety in the interval; however, it tends to be difficult to draw such tubes due to the continuity of time.

However, when a trajectory planner works with discrete trajectories, it can simply perform collision tests by checking that every pair of two clearance zones from different trajectories do not intersect at the time. Although this method is easy to apply without generating tubes between the time points, it could cause undetected collisions between the time steps without additional separation distance. The level of trajectory safety increases by higher time resolution and slower objects velocity rates.

The algorithm adopts the latter approach, as most of the previous studies do, despite the inherent drawback. The AGVs for use in container terminals can travel straight at around 6 m/sec at most while having a maximum turning speed of approximately 3 m/sec. The speed rates are significantly low compared to other automated transporters, such as aircraft, UAVs and cars on the road.

Additionally, in the simulation experiments in *Chapter 7*, time intervals were set to a second, considered as a short period. Since the target vehicles are around 15 metres in length, the postures at two consecutive time points cover all postures in the corresponding time window. Therefore, trajectory safety is guaranteed when the vehicles travel straight. The approach should add more space to the safety clearance zones to handle undetected possible collision areas that occur when the vehicles move on curves as shown in *Figure 5-18*.

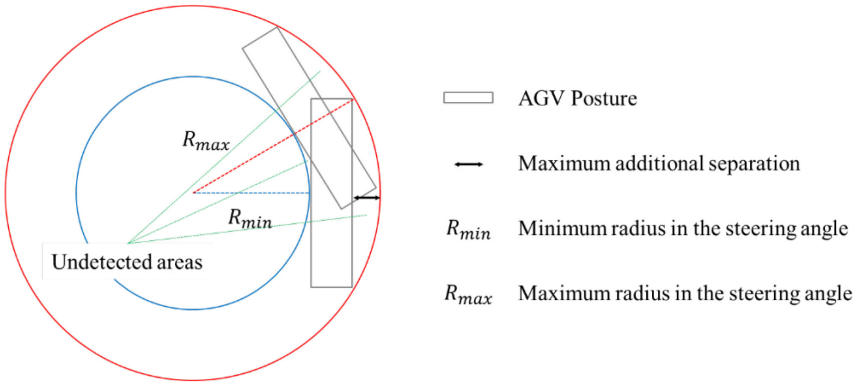


Figure 5-18. Undetected possible collision areas in a turning case.

Where the maximum steering angle of target industrial AGVs is 30°, the maximum additional separation is around 2.75 metres. In the 45° case, it increases to approximately 3.70 metres.

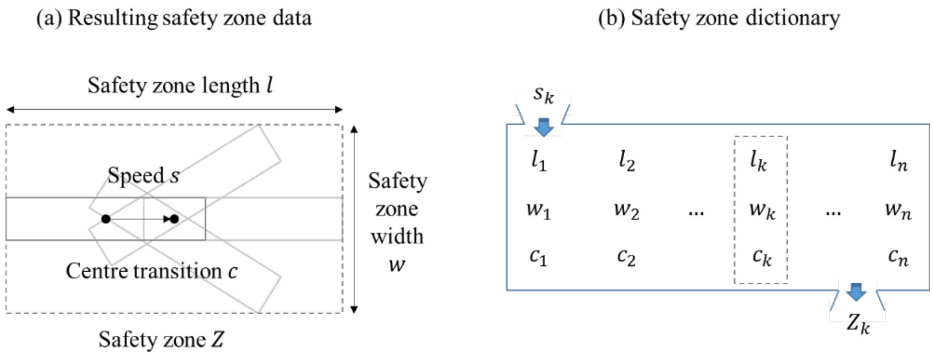


Figure 5-19. Rapid safety zone generation.

For the rapid generation of such safety regions, the fleet trajectory planner has a link-type data set. This is a type of dictionary in which each allowable speed rate of the AGVs maps to the resulting safety zone's data: its dimension and the central transition from the vehicular centre to the corresponding safety zone's centre (*Figure 5-19*). Thus, the planner does not need to calculate the RIC per zone generation request since the values of l, w and c are determined in advance. The next section will show its resulting calculation time reduction by simulation.

5.3.3. Computational Performance

Before testing the algorithm in a container terminal environment, a series of simulation tests were implemented in generic scenarios. The following tests included up to 20 AGVs, which

forms a larger vehicle fleet, when compared to literature. The primary objectives of the simulation experiments are to identify the following performance issues: (1) the impacts of fleet size increases on solution quality and calculation time, (2) the superiority of the movement coefficient e_p^k in the objective function described in *Section* 5.3.1, (3) calculation time reduction by the application of the safety zone dictionary explained in *Section* 5.3.2 and (4) performance differences between the SA- and the GA-based fleet trajectory planners.

The unit simulator generates vehicle stations on the north and the south side. Each station can hold at most one vehicle and has a pair of required vehicular docking and departure postures. When each vehicle arrives at its destination, it randomly chooses the next goal location and starts the next journey (*Touch-and-Go*). Therefore, resulting travel networks are rather complicated with many potential collision threats. The specifications of the simulated vehicles are based on industrial AGVs for use in container terminals.

Table 5-4 summarises the default setting values of the SA-based fleet trajectory planner used in the simulation experiments.

Table 5-4. Default settings for evaluating the fleet trajectory planner

Group	Name	Value
Path	Maximum additional straight	50 (m)
	Path reuse rate	0.30
	Detour trial number	5
	Additional straight probability	0.50
	Maximum detour circle addition direction	π
Acceleration	Time interval	1 (sec)
	Planning horizon	20 (sec)
	Execution horizon	10 (sec)
	Travel strategy application horizon	5 (sec)
SA	Initial temperature	10,000
	Lowest temperature	1
	Cooling function parameter (α)	0.90
	Number of neighbours	10
	Movement coefficient	0.6

Figure 5-20 shows observed solution quality changes in path length and travel time by different sized fleets with resulting calculation time data. In each of the fleet size scenarios, each vehicle performed around a hundred journeys and the algorithm solved over a thousand FTPP instances. Since more vehicles cause higher traffic congestion, the performance of the fleet trajectory planner was inversely proportional to the applied fleet size in the tests. However, resulting trajectories were always shorter than their corresponding Manhattan distance (*Figure 5-20* (a)); they are approximately 80% of the counterparts in path length.

One remarkable point is that travel time was more sensitive to the fleet size change compared to travel distance as described in *Figure 5-20* (b) and (c). The reason is that the vehicles were increasingly exposed to situations where they needed to decelerate to avoid collisions that are difficult to be solved by path structure changes. The ratio of travelled distance to corresponding Dubins path length slowly increased with the number of used vehicles. It is below 1.07 even in the 20-vehicle case. On the other hand, the ratio of travel time to theoretical quickest travel time rapidly went up while showing increasingly high variations.

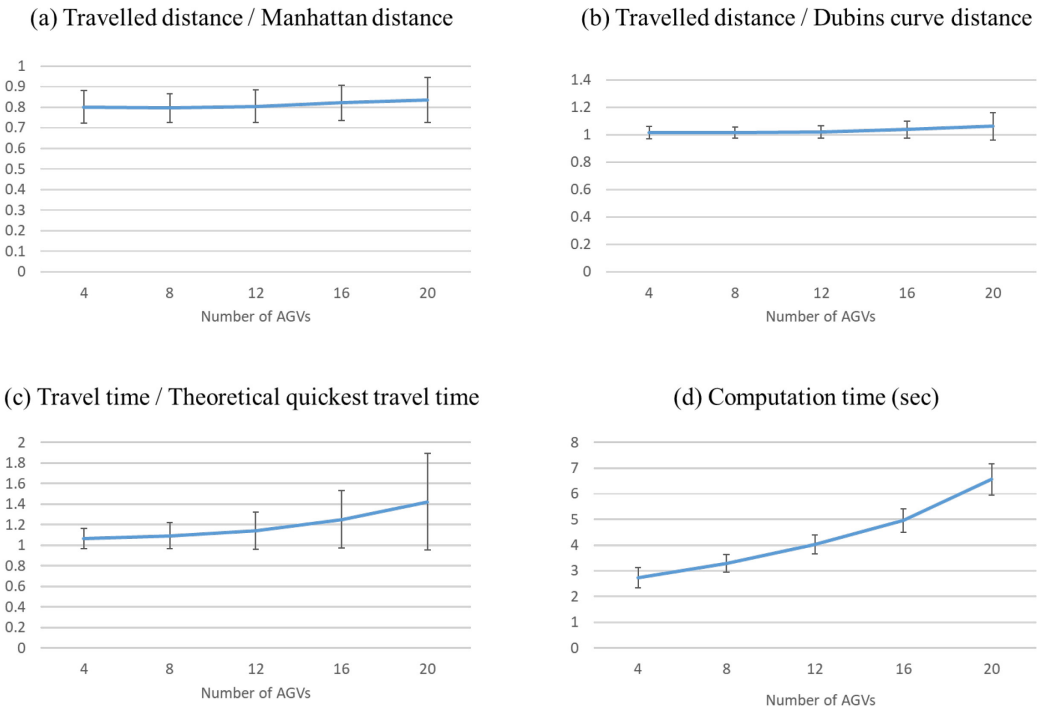


Figure 5-20. Performance changes by different fleet sizes with standard deviations.

Figure 5-20 (d) depicts average calculation time values observed in the tests. As seen, the time required to provide vehicular trajectories rose by fleet size change. The algorithm computed collision- and deadlock-free solutions within the determined execution period (i.e. 10 seconds) without any exception.

Figure 5-21 shows objective value convergences by iterations. Since the size of an FTPP instance is determined by the number of the vehicles in the problem, it was observed that the algorithm required more iterations when it handled more AGVs. For example, in the 12-vehicle case, the objective value remained stable after 30 iterations; however, in the 20-vehicle case, the algorithm needed over 80 iterations for objective value convergence.

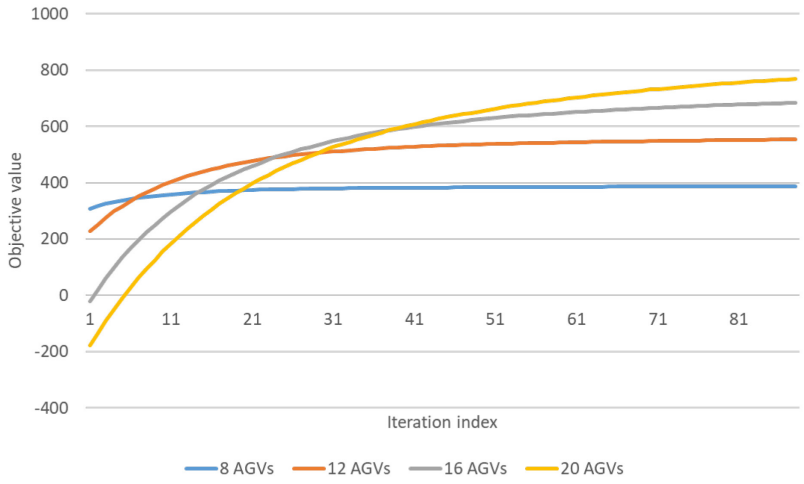


Figure 5-21. Objective value convergence by iterations.

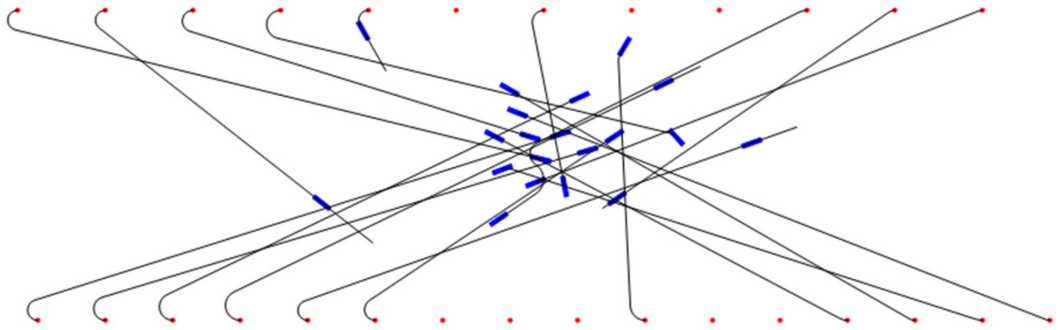


Figure 5-22. Detected deadlock situation without the movement coefficient.

Figure 5-22 depicts a deadlock situation that can happen without the application of the movement coefficient in the objective function explained in *Section* 5.3.1. Since the vehicles

in the experiments were only controlled to move towards their destination, they became stuck and experienced deadlock situations easily.

The third test result shows the superiority of a safety zone dictionary application regarding calculation time. When a safety zone generator adopts the dictionary, it can compute a vehicular safety field within a millisecond; however, as summarised in *Table 5-5*, in the test, safety region calculation time increased until the maximum turning speed rate. Additionally, there were rapid calculation increases from a speed rate of 2 m/sec to 3 m/sec in all the maximum steering angle cases (i.e. 15°, 30°, and 45°). This is because the number of turning movement scenarios exponentially increases with increased required stopping distance.

Table 5-5. Safety clearance zone calculation time in seconds.

Maximum Steering Angle	Speed rate (m/sec)						
	0	1	2	3	4	5	6
15°	-	-	0.003	0.13	0.131	0.129	0.13
30°	-	-	0.015	0.996	1.025	1	1.013
45°	-	-	0.034	3.33	3.326	3.352	3.33

* -: less than a millisecond
* maximum turning speed: 3 m/sec

As the final test in this chapter, a performance comparison was performed between the SA- and the GA-based fleet trajectory planners with 20 AGVs. Different population sizes were applied and each test included over 1,000 FTPP instances. The GA-based model has the following rules with the default settings summarised in *Table 5-6*:

- The first trajectory set (i.e. the first chromosome) in an initial population controls all the AGVs involved to decelerate and stop. This solution is the same as *Bizarre Neighbour* in the SA-based fleet trajectory planner.
- The second trajectory set in the initial population includes the quickest trajectories for the vehicle fleet.
- Mutation occurs in every gene (i.e. a trajectory) if the gene causes a collision; otherwise, a mutation rate determines whether the algorithm performs a mutation operation.

Table 5-6. Default settings of the GA-based planner

Setting field	Value
Elitism	Used
Parent selection method	Rank-based Roulette wheel selection
Crossover rate	0.85
Crossover strategy	Uniform crossover
Mutation rate	0.01
Termination condition	Convergence
Objective value convergence range	±10
Convergence limit	5 times

As seen in *Figure 5-23* (a), (b), and (c), the GA-based fleet trajectory planner produced better trajectory sets regarding the three performance indicators by adopting larger populations. However, as compensation for better performance, it required longer computation time with bigger variations. On the other hand, the number of required iterations slowly decreased as seen in *Figure 5-24*.

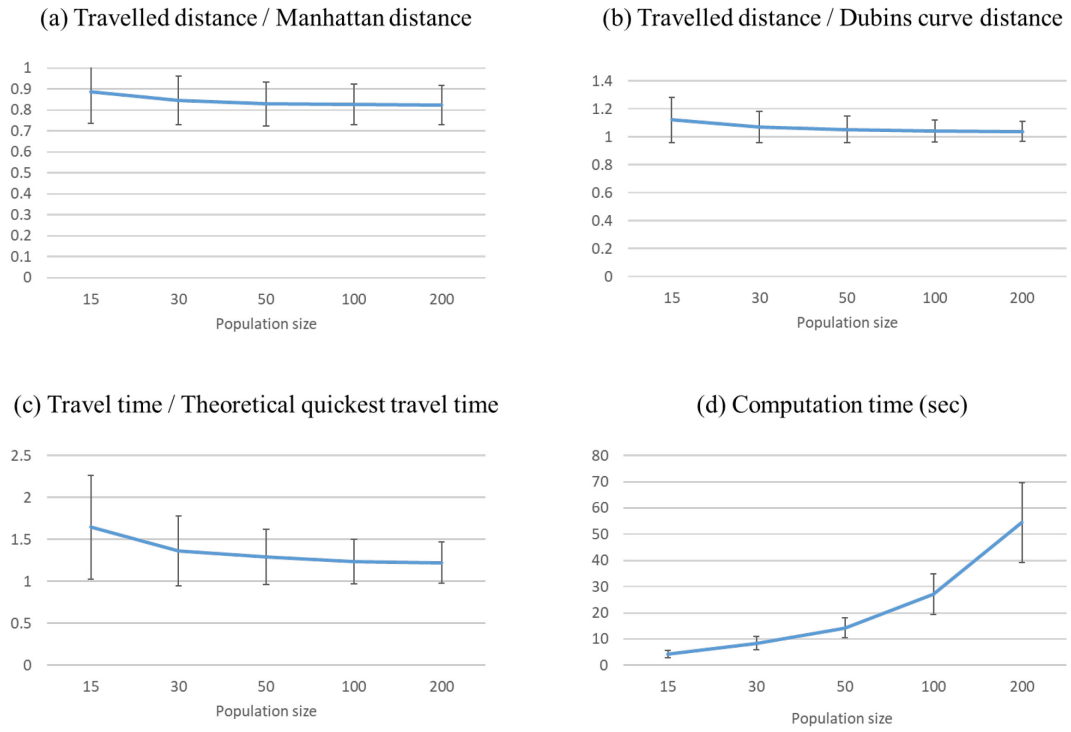


Figure 5-23. Performance changes by different population sizes (20 AGVs).

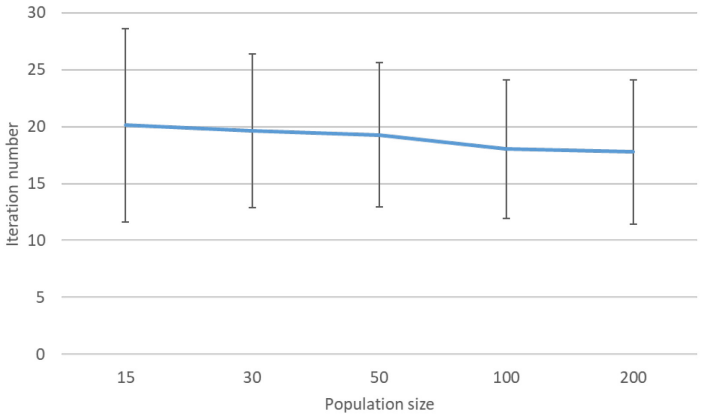


Figure 5-24. Average iterations with standard deviations.

Table 5-7 compares (1) the result in *Figure 5-20* (i.e. the SA-based model) and (2) the performance of the GA-based planner with 30 as its population size. As seen in the table, there is no significant difference in the first three performance indicators. However, the SA-based model provided collision- and deadlock-free trajectories more quickly and the observed computation time of the SA-based planner was four times more stable than the counterpart.

Table 5-7. Comparison of the two metaheuristic applications

	SA-based planner	GA-based planner
vs. Manhattan distance (in length)	0.836	0.845
vs. Dubins curves (in length)	1.061	1.069
vs. Theoretical quickest (in length)	1.423	1.367
Average calculation time (sec)	6.557	8.439
Calculation time standard deviation	0.612	2.586

5.4. Summary

This chapter explains two paradigms of trajectory planning: the reachable area based approach and the path-based methods. In the test environment with the *CPLEX* solver, the former required significant computational effort, even with a few vehicles. Additionally, the resulting paths were not smooth enough for industrial vehicles to follow. Therefore, additional constraints are essential to generate follow-able routes, resulting in more computation time.

For the adoption of the latter approach, this research includes an investigation of several existing path generation techniques for use in the free-ranging vehicle environment. The reviewed imaginary graph based methods require an additional path smoothing process for industrial AGVs. Bézier curves are not appropriate as they do not contain any straight lines where vehicles can travel faster than in curve sections. Although Dubins curves guarantee optimality in path length, they do not possess a function of diversifying their route shapes to avoid potential conflicts between agents.

Therefore, a path generation technique was developed by utilising the idea of Dubins curves with a strategy-based acceleration generator. The path creation method lets paths contain minimal curve sections while designing various first movements. Additionally, due to the application of a rolling horizon, paths can be subject to continuous change as well as being able to remain constant. The developed acceleration generator provides a set of acceleration rates per path, covering a predetermined time span (i.e. a planning horizon). Instead of randomly setting the values, it determines the rates based on several travel strategies.

Due to the inherent time complexity of the FTPP, the suggested fleet trajectory planner works with the framework of SA. It iteratively improves a trajectory set for the AGV fleet while resolving its detected collision events by amending a trajectory of the solution. The algorithm also adopts the OBB bounding policy while using the idea of a safety clearance zone dictionary to reduce the time required to check trajectory safety. Compared to a GA-based fleet trajectory planner, it can show significantly better performance in calculation time. Therefore, it seems more appropriate for real-time application in port environments.

Chapter 6. Simulation of AGV Operations

Container terminals are large and complex with various types of equipment, such as cranes and container vehicles. Each equipment agent forms a dependent operation block that cooperates with other operation blocks by exchanging data, information and containers. The operating system of container vehicles or ACVs interacts with the quay and the yard sides to synchronise vertical and horizontal container handling operations.

Terminal operators and equipment providers have focused on the improvement of container handling and the reduction of vessel turnaround time since the introduction of containers in the 1950s. Some of the most significant changes have occurred in container handling hardware with others in control systems and software. For instance, advanced types of yard cranes, using twin and dual configurations, and various container vehicles have been introduced into sea container terminals with the idea of port automation. At the same time, significant research efforts continue to improve conventional and newly-adopted equipment by applying advanced control logic.

Although new technologies and suggestions seem to be appropriate and helpful, their applications to real terminal environments may not guarantee port performance improvement. The main reason is that it is difficult to prove that such new approaches can be superior to existing systems without causing any operational risk. Additionally, terminal operators are rather conservative in adopting new technologies since they handle significantly expensive equipment units.

Simulation is one of the most competitive testing approaches in such situations where new ideas or strategies need to be evaluated in application environments with expensive equipment, including container terminals. It does not require any associated hardware, and therefore it is significantly economical. Also, this testing approach is capable of including more parameters, variables and constraints.

6.1. Development of a Bespoke Simulation Model for Integrated AGV Operations

As described in *Section* 2.6, commercial simulation packages provide a high level of user-friendliness without requiring uses to have a high level of programming knowledge and skills. They allow users to simply construct their approaches by dragging-and-dropping predefined function blocks and drawing operational flowcharts. Since these function blocks are generic, the packages are easily extensible to various industrial settings, especially manufacturing processes and able to cover a wide analysis scope.

As compensation for a high degree of user-friendliness and application extensibility, commercial simulation packages have a limited range of functions. So, they would not be appropriate for use in specific environments, such as container terminals. In addition, their limited functions make it difficult to express complex models and algorithms, including the developed ones for sophisticated vehicle dispatching and fleet trajectory planning detailed in *Chapter 4* and *Chapter 5* respectively.

Table 6-1. Comparison of simulation model classes

	Commercial	In-house
Price	High	-
Analysis coverage	Wide	Narrow
Extensibility	High	Low (Case-specific)
Algorithmic flexibility	Low	Very high*
User-friendliness	High	Low
Key Feature(s)	Drag-and-drop	A high level of programming

* Only when users develop their own simulation model

Instead, researchers and port operators have also developed and utilised in-house simulation models, such as Limen developed by Angeloudis and Bell (2010). These models are case- and objective-specific, and therefore users can consider detailed, customised, sophisticated functions in the process of model development. However, these models have distinctive target domains; for example, some were specifically designed for oil terminals, but some for AGV systems with predetermined flow-path networks. In addition, each simulation model focuses

on specific, narrow analysis scope, such as only handling vehicular collisions and deadlocks. Therefore, additional programming is mostly required when being applied to a different type of analysis.

This doctoral research focuses on vehicle dispatching and fleet trajectory planning for use in potential ACTs where free-ranging AGVs operate. Both decision-making problems (i.e. combinatorial optimisation) have a high degree of complexity, and the resulting algorithms described in the previous two chapters possess sophisticated features and planning techniques. Commercial simulation packages are inappropriate in expressing such sophisticated functions, and existing in-house models are difficult to be introduced to this research due to their low extensibility.

Therefore, this study includes simulation model development that focuses on testing and evaluating different vehicle operation settings for free-ranging container AGVs. The model does not cover the whole container terminal domain; instead, it expresses highly detailed vehicle operations (i.e. high fidelity) whereas the program relatively simplifies quay and yard crane operations (i.e. low fidelity). It excludes the remaining parts of the terminal (i.e. unconsidered). *Figure 6-1* depicts the fidelity level of each operation block in a typical container terminal on a side view. The program is constructed with Visual Studio 2017 (C#).

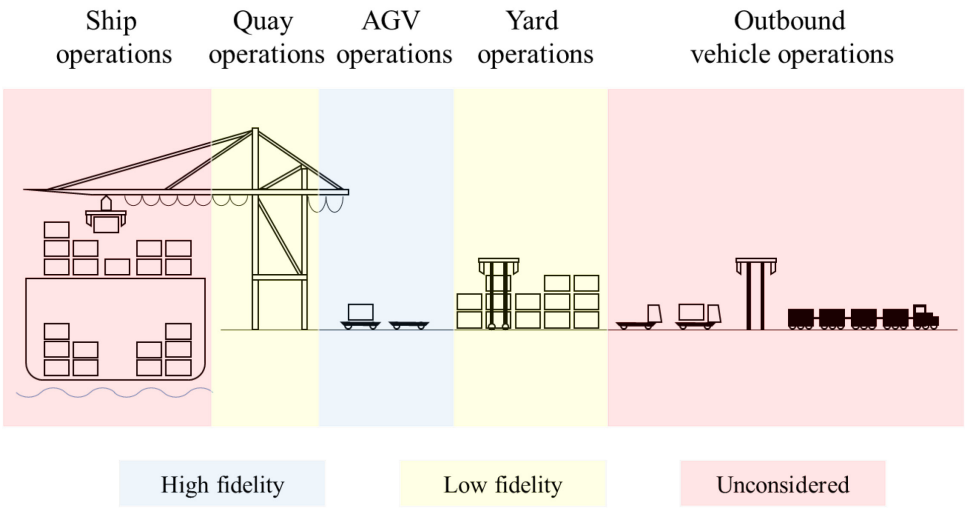


Figure 6-1. Fidelity levels of container terminal operation blocks.

6.1.1. Integration of the AGV Operations

AGV dispatching controllers are expected to interact with AGV trajectory planning and execution while being a decision-making problem on the upper layer. It has a role in determining the vehicles' destinations that are key inputs in the following planning. On the other hand, the latter influences the postures of the delivery vehicles and the expected setup travel time of the AGVs for their first assignment that are parameters in the upper problem. *Figure 6-2* depicts the relationship between the two vehicle operation problems in container terminals. This research covers the integration of the vehicle operation types to let them exchange information.

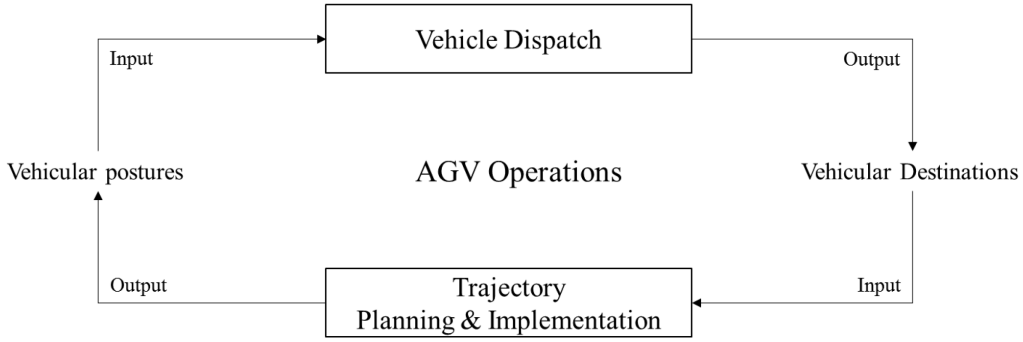


Figure 6-2. Relationship between the two AGV operations.

6.1.2. Simulator Structure

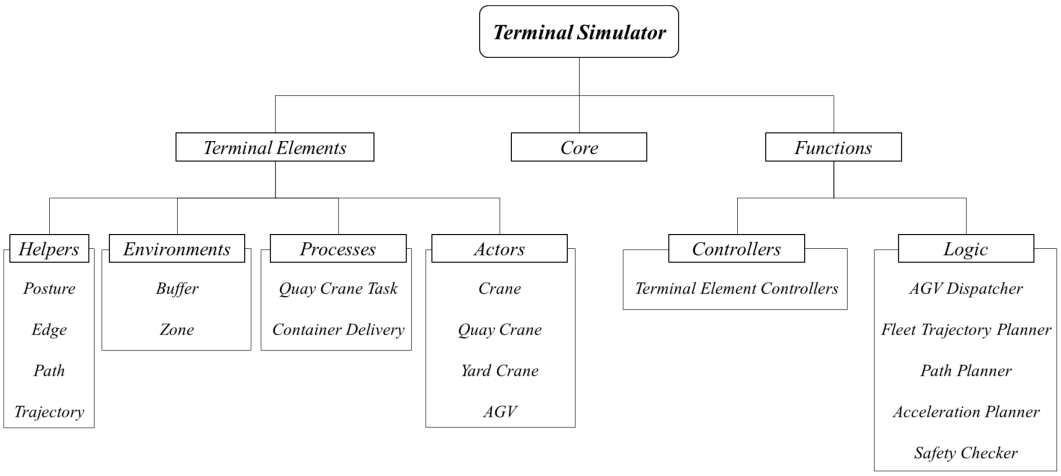


Figure 6-3. Simulator structure.

The simulator has four types of port elements (named *Helpers*, *Processes*, *Actors* and *Environments*) and a group of functions to (1) control the terminal elements and (2) generate AGV dispatch plans and vehicular trajectories. *Figure 6-3* depicts the overall structure of the simulator. The following subsections explain the members in the illustration.

Core

The *Core* module is a functional block for controlling the developed simulator. It includes several functions. Firstly, it has a time controller that runs the simulation timer based on a predetermined time interval. As a default value, the interval is one second. Secondly, visualisation functions are in the module to display vehicle operation animations; a group of trajectories are planned at each time point, and simulated AGVs update their posture subject to the information. In the process of model development, these functions helped to identify defective function blocks that could be difficult to discover without visualisation. Thirdly, it contains some user interface windows with performance evaluation metrics to provide the convenience of accessing the animation window and the result data blocks of the simulator.

Helpers

The *Helpers* group elements facilitate AGV movements in a given domain. It includes the following four members: *Posture*, *Edge*, *Path* and *Trajectory*. *Posture* refers to the structural data of vehicular postures in a generic sense; it has an x- and y-coordinate pair with a heading angle in degrees. The other three members represent edges, paths and trajectories respectively.

Environments

The *Environments* group has two environmental elements: *Buffer* and *Zone*. This first (i.e. *Buffer*) is a data structure to generate buffer instances used as waiting areas for AGVs. As

basic information, it possesses a numerical index (i.e. identification number) and locational information expressed by x- and y-coordinates. It includes two data fields to express vehicular docking and departure heading angles in degrees. Additionally, it contains a variable that represents the type of *Buffer* instance. In the simulator, there exists a group of buffers in a vehicle depot area where AGVs stay at the initial stage of each simulation test. Also, there are buffers in the quay and the yard areas for AGVs to wait or be served.

Like *Buffer*, *Zone* contains an index and a locational data field expressed as x- and y-coordinates. Additionally, it has numeric data to show the size of its instances, such as their dimension.

Processes

The group of *Processes* consists of *Quay Crane Task* and *Container Delivery*. *Quay Crane Tasks* represent the respective tasks defined in *Section* 4.2.2. Therefore, its instances are members of those of *Quay Crane*. *Table 6-2* summarises the information units of the data structure.

The value in 'Sequence' of a *Quay Crane Task* instance is the sequence number in the allocated quay crane. 'Source' and 'Goal' represent the origin crane and the goal crane of the task instance respectively; each of them can be a quay crane or a yard crane. The service cycle data of the two relevant resources are available from 'Source cycle time' and 'Goal cycle time' respectively. 'Type' can be either a container loading operation or a discharging task.

The data field 'State' expresses the state of the container currently involved in the instance. It can be one of the following seven states: 'on the source stack', 'with the source crane spreader', 'being loaded on an AGV', 'on an AGV', 'being discharged from an AGV', 'with the goal crane spreader' and 'on the goal stack'.

Table 6-2. Data fields of Quay Crane Task

Name	Type
Sequence	Number
Source	Crane
Goal	Crane
Source cycle time	Number
Goal cycle time	Number
Type	Types
State	States
Expected travel time	Number
EPETWD	Number
AGV arrival reference time at source crane	Number
AGV arrival reference time at goal crane	Number
Observed AGV arrival time at source crane	Number
Observed AGV arrival time at goal crane	Number
Allocated AGV	AGV

'Expected travel time' captures the expected travel time from the origin crane to the destination of the instance calculated at the initial stage. The following three are explained in *Section* 4.2.2. The two observation travel times are updated when its designated AGV arrives at the task origin and the destination respectively. 'Allocated AGV' refers to the AGV on the task. Due to the idea of task re-assignment, its value can be changeable.

Container Delivery refers to a data structure that expresses horizontal container moves performed by AGVs. Therefore, each of its instances has data for both the source crane and the goal one. The instance also includes journey-related data fields to identify the distances based on the Euclidean and Manhattan distances respectively. Also, it can save its resulting travelled distance and time to compare with their theoretical optimal value.

Actors

The *Actors* group has the following four data structures: *Crane*, *Quay Crane*, *Yard Crane* and *AGV*. *Crane* is a shared data structure by *Quay Crane* and *Yard Crane* meaning that it contains

a set of data fields used in both (*Table 6-3*). The simulator does not directly create its instances but employs those of the two specific cranes instead (i.e. *Quay Crane* and *Yard Crane*).

Table 6-3. Data fields of Crane

Name	Type
Index	Number
Type	Types
Centre	Point (x, y)
Spreader	Spreader
Previous task	*Quay Crane Task*
Current task	*Quay Crane Task*

'Index' stands for informing the identification number of each *Crane* instance. 'Type' expresses whether the crane instance represents a quay crane or a yard crane. The data field 'Centre' means the central coordinates of the resource instance by x- and y-coordinates. 'Spreader' is a nested data structure containing two variables. One represents the location of each 'Spreader' instance, and the other shows its current state determined by the moving direction and whether it is transferring a cargo. Also, users can identify whether the equipment instance is currently idle or waiting for a delivery vehicle. In addition to the data structure, *Quay Crane* has an additional field named 'Quay crane task list'. Such lists are scheduled by *Quay Crane Controller* at the initial stage of a simulation test.

AGV instances imitate industrial AGVs for use in potential container terminals without any predetermined guide-path systems. The structure contains the following data fields summarised in *Table 6-4*.

Similar to the index data in the other actors, 'Index' stands for representing the identification number of each *AGV* instance. 'Initial Posture' gives the information on the instance's location and heading at the initial stage of a simulation test. 'Container' is one-dimensional numeric array with two values: one refers to the linked quay crane index of its current task, and the other means the task sequence in the quay crane. It informs whether the vehicle instance has a container with it and helps to identify the travel type of the vehicle (i.e. the setup travel or

the delivery one). ‘Movable’ is set to 1 (i.e. true) when the vehicle can move and 0 otherwise. To be more specific, when the vehicle is waiting or being served by a crane, the delivery vehicle is unable to move.

Table 6-4. Data fields of AGV

Name	Type
Index	Number
Initial Posture	Posture
Container	Numeric array
Movable	Binary
Complete container delivery set	*Container Delivery* set
Current task list	*Quay Crane Task* set
Dwell time at cranes	Number

‘Complete container delivery set’ includes the delivery journeys completed by the vehicle instance, and ‘Current task list’ is a set of *Quay Crane Task* instances updated by *AGV Dispatcher* (explained in *Chapter 4*), *Quay Crane* and *Yard Crane*. The first has a function of renewing a task list on the vehicle whereas the last two can remove a member of the list whenever they finish the corresponding task.

‘Dwell time at cranes’ reveals the sum of the waiting and service times at cranes; therefore, the total setup travel time is the subtraction of this variable’s value and the sum of observed travel time values in ‘Complete container delivery set’ from the simulation execution time.

Functions

The *Functions* group consists of *Logic* and *Controller*: The former performs vehicle dispatch and fleet trajectory planning whereas the latter has a direct role in creating and controlling container terminal elements explained in the previous subsections.

The *AGV Dispatcher* module is the first main module of *Logic*. It implements AGV dispatch with the SA-based solver introduced in *Section* 4.3. Whenever a vehicle-dispatch request occurs, the vehicle operation simulator generates an input data instance, as described in

Section 4.2.2. The simulator transmits the input to the solver, and this provides an output named 'DispatchPlan'. The result contains a vehicle dispatch schedule to assign at most two tasks per vehicle.

The *Fleet Trajectory Planner* module implements trajectory planning for a free-ranging vehicle fleet based on the previous chapter. It utilises three submodules: *Path Planner*, *Acceleration Planner* and *Safety Checker*. The first sub-function provides vehicular paths described in *Section* 5.1, and the second determines acceleration rates to be used on the generated routes (*Section* 5.2). The third stands for guaranteeing trajectory safety with a role in checking that vehicles do not collide with each other. It also has a function of generating vehicular safety clearance zones based on OBB, as explained in *Section* 5.3.2.

The members of *Controllers* manage sea container terminal elements instantiated by *Quay Crane*, *Yard Crane*, *AGV*, *Buffer*, *Zone* and *Quay Crane Task*. Each of them creates relevant instances based on setting values and updates the instances' states at each time step in the simulator.

6.2. Summary

This chapter explains and depicts the directional influences between vehicle dispatching and fleet trajectory planning and implementation in an integrated vehicle operation setting for use in container terminals. Also, it focuses on explaining the constructed simulator for use in the thesis. The program development resulted from the absence of a simulator to test the integrated free-ranging vehicle operation setting in potential ACTs. The provided simulation model can realise each of the two vehicle operation problems (i.e. vehicle dispatching and fleet trajectory planning). Additionally, it allows the two model blocks to exchange data and provides multiple variables for evaluating resulting vehicle dispatch plans and trajectories.

Chapter 7. Simulation Experiments

This chapter presents and discusses simulation results obtained to evaluate the AGV dispatching and fleet trajectory planning algorithms detailed in *Chapter 4* and *Chapter 5* respectively. A schematic simulation domain is firstly described, which is followed by an explanation of terminal element settings for quay crane tasks and container terminal resources (i.e. cranes and AGVs). This chapter then provides used performance indicators in simulation and evaluates the vehicle operation models based on the indicators.

7.1. Simulation Domain

There are three types of container handling resources in simulation environments used for the study: quay cranes, yard cranes and free-ranging AGVs. Based on the structure defined in *Section* 6.1.2, all of the terminal resources fall under *Actors*. In a rectangular simulation domain, quay cranes are located on the north side in a row, and each has a list of container loading and discharging missions. Yard cranes are on the south side, and AGVs travel between the quay and the yard while performing their container delivery missions.

Buffers in the simulator fall into two groups. Some buffers are used as vehicular parking slots in a depot area at the initial stage of each test, falling under *Depot Buffer*. The others each (i.e. *Crane Buffer*) belong to either a quay crane or a yard crane. They are used as waiting places and service points for container loading and discharging operations.

The simulator can also define three types of *Zone* instances that can be applicable in a potential ACT where free-ranging AGVs operate. The first is a zone including all the *Depot Buffer* instances in the terminal environment; there is only one instance in the group, and therefore this zone can represent the depot area. The second includes the *Crane Buffer* instances that belong to cranes. The third represents the transport area of an ACT, which is the main scope of the fleet trajectory planning algorithm.

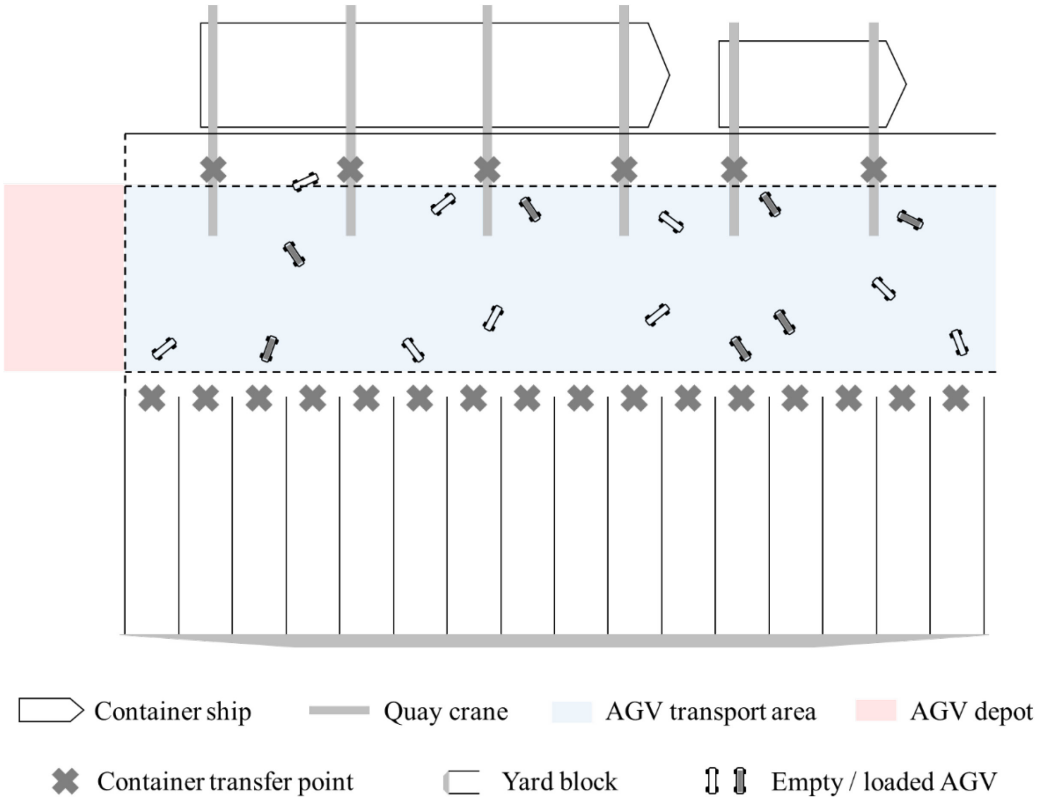


Figure 7-1. Schematic arrangement of container terminal resources.

Figure 7-1 depicts a simulation domain based on the descriptions above. It does not contain a flow-path network for guiding AGVs. The depot area provides space for the vehicles to dwell at the start of each simulation test.

7.2. Terminal Element Settings

All possible settings would ideally be utilised in simulation experiments if the values were able to be perfectly obtained. However, terminal-related data are rather confidential. In addition, there exist security and safety issues when observing terminal operations in person to gather relevant data, such as container loading and discharging operations by quay cranes.

Instead, most researchers define potential application domains or select representative operational environments among existing cases while trying to gather real data and assuming parameters. In this case, the representative domain should carefully be determined. For instance, it should be generic in terms of the layout of terminal resources, especially cranes

and yard blocks. When determining parameter values, researchers need to refer to relevant papers and reports or calculate the values by using other parameters available in the literature. At the same time, they should be aware that a level of algorithm (or model) performance in representative environments may differ from that in selected domains by other users.

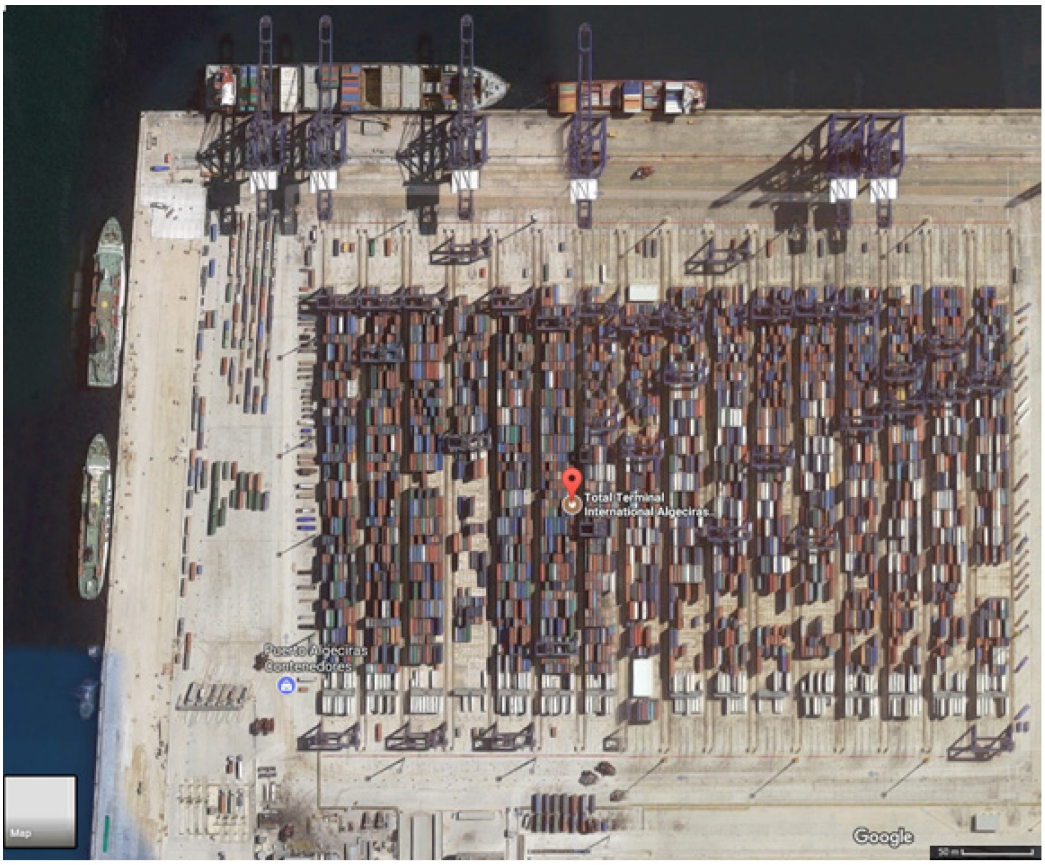


Figure 7-2. Bird's-eye view of TTI Algeciras. Source: Google Maps (2017)

During the study, a considerable amount of effort went to the development of a simulation domain; however, it still has been rather difficult to obtain real data from container terminal operators. Therefore, a port-like simulation domain was defined for simulation experiments based on a real container terminal, named Total Terminal International Algeciras (TTI Algeciras) in Spain (*Figure 7-2*), which is the second biggest terminal in the port of Algeciras (Louppova, 2017). The terminal can represent container terminals where yard blocks are vertically arranged. Such terminals are potential application environments in which free-ranging AGVs operate for container deliveries between the quay and the yard. The layout of quay and yard cranes is also similar to that of most container terminals.

7.2.1. Cranes

The number of quay cranes in use can vary whereas that of yard cranes is rather constant in a container terminal. Generally, one or two quay cranes serve a small- or mid-sized container ship, and three or four a large vessel. In the following experiments, three scenarios were defined: (1) one small vessel, (2) one large vessel and (3) one small vessel and one large vessel. Therefore, two, four and six quay cranes were adopted in the scenarios respectively. The number of yard cranes was determined based on TTI Algeciras shown in *Figure 7-2*. Thus, 16 units of yard cranes were employed to interact with AGVs.

The separation between cranes can also vary; however, a deterministic value was used. Since the length of container vessels is around from 100 to 300 metres, 80 metres can be an available option as intervals between quay cranes. Therefore, two quay cranes can be allocated to a small container ship, and up to four to a large one. Intervals between yard cranes were set to 35 metres from TTI Algeciras.

The service time of quay and yard crane spreaders in serving AGVs needs to be identified as well. Kim and Bae (2004) utilise a deterministic period (i.e. 20 seconds per operation) required for a quay crane spreader to lift or load a container from or onto an AGV. Their assumption would be reasonable since relevant operations are rather similar, and therefore their service time variations would not be significant. The service time of a yard crane spreader was set to 10 seconds based on the research by (Saanen and Valkengoed, 2005).

7.2.2. AGVs

Unlike the cranes, a wide range of data are available on industrial AGVs for container transport. An AGV brochure by Konecranes Gottwald (2017) provides the specifications of their vehicles that cover most vehicular settings required in the research (*Table 7-1*). However, the maximum steering angle is not available from the brochure because in fact there is no limitation of vehicular steering due to the crab-movement capability. Therefore, any setting

value can be allowable for the cargo delivery vehicles. In the following tests, the maximum steering angle was set to 30 in degrees as the default value.

The number of AGVs used was differently set depending on the ratio of the number of quay cranes to the vehicle fleet size. Essentially, a level of traffic congestion rises by a fleet size increase. However, the application of fewer AGVs would cause AGV arrival delays in the quay and the yard. The following tests applied two, three, four and five to the terminal resource ratio respectively.

Table 7-1. AGV specifications. Source: Konecranes Gottwald (2017)

AGV Specifications		
Dimensions	Length	14.8 m
	Width	3.0 m
Speed rates	Forwards / reverse max.	6 m/s
	Curves max.	3 m/s
	Crab steering max.	1 m/s
Other(s)	Positioning accuracy	+/- 25mm

7.2.3. Quay Crane Tasks

Data on container loading and discharging cycle time at the quay and the yard could not be obtained from the real directly; instead, information in previous work was utilised in the thesis. Phan-Thi, Ryu and Kim (2013) refer to a real terminal and set quay operation cycle time to a uniform distribution of $U(70,130)$ in seconds. Saanen and Valkengoed (2005) provide various yard crane specifications in real container terminal environments. The horizontal movement speed of yard crane spreaders is around 3 m/sec, and the equipment takes approximately 10 seconds for serving an AGV with a vertical movement speed of 1 m/sec. Since the vertical length of yard blocks in TTI Algeciras is about 300 metres, a uniform distribution of $U(50,140)$ was adopted as a default setting for yard operations with AGVs.

The linked yard crane of each quay crane task was not available from the field and the literature. However, several previous studies include a description of container storage

principles in the yard. This thesis employed the following four rules to generate quay crane missions:

- It is preferable for workload to be balanced in yard blocks (Zhang *et al.*, 2002).
- Minimising the expected total travel time of AGVs is a main objective of container storage planning (Zhang *et al.*, 2003).
- Different types of containers are likely not to be stored in the same yard bay as well as the same ship bay (Kim, Park and Ryu, 2000).
- There exist several yard blocks for storing special containers, such as empty ones or those with hazardous goods (Günther and Kim, 2006).

7.3. Performance Indicators

Since container terminals are a complex system where various types of equipment are employed, they cannot be evaluated by a single performance measure. Thus, multiple factors were adopted to observe operational gains by the presented AGV fleet operation planner for AGV dispatching and trajectory planning (*Table 7-2*). Three performance indicator groups exist. One is for evaluating performance of the AGV fleet controller regarding terminal resource productivity. Another is specifically utilised to observe operational changes by different dispatching strategies. The other is for the side of AGV fleet trajectory planning.

Table 7-2. Performance indicators

Group	Performance measure	Unit
Resource productivity	Quay crane productivity	Containers/hour
	AGV fleet productivity	Deliveries/hour
AGV dispatching	AGV arrival delay in the quay	Seconds
	AGV arrival delay in the yard	Seconds
	Setup travel time	Seconds
Trajectory planning	Ratio of travel distance to the Manhattan distance	Ratio
	Ratio of travel distance to the Dubins curve	Ratio
	Ratio of travel time to the quickest trajectory	Ratio

In each simulation test, two types of resource productivity were revealed: quay crane productivity and AGV fleet productivity. The former refers to average quay crane productivity per crane per hour, representing the efficiency of quay crane use. It helps port operators and shipping liners to estimate the duration of a vessel's stay. The latter means the average throughput of a used AGV fleet per hour in container deliveries, as part of a container terminal. It can be used by terminal operators when determining an appropriate AGV fleet size.

On the side of AGV dispatching, three performance indicators were selected: two AGV arrival delay time indicators in the quay and the yard respectively and AGV setup travel time, each of which is a term of the suggested AGV dispatching model's objective function. The delay indicators represent the tardiness of the missions in the two vertical container handling blocks, and the other shows the efficiency of vehicle operation regarding setup travel.

The other three indicators pertain to fleet trajectory planning for container moves. Path length and travel time were used to evaluate vehicular movements. Based on each journey, three types of distance data were generated: (1) the Manhattan distance, (2) the Dubins curve based distance and (3) the observed travelled distance by the proposed algorithm. The comparison between (1) and (3) refers to the minimum performance improvement from a conventional ACT subject to a predetermined guide path network. On the other hand, the comparison between (2) and (3) means the excess rate from the theoretical maximum performance. Also, the simulator produced two types of travel time data: (1) the quickest travel time without considering collisions and (2) the observed travel time. By comparing the two data types, the excess rate from the theoretical maximum performance in travel time could also be obtained.

7.4. Model Evaluation

The proposed vehicle operation model was evaluated by rigorous testing. Three scenarios were defined by the number of quay cranes used in the simulated container terminal, and each scenario included four cases determined by the number of AGVs employed (*Table 7-3*). Each

case was run with 20 randomly-generated seed values (i.e. quay cranes' schedules), and the duration of simulation was 20,000 seconds.

Table 7-3. Number of AGVs used in each case

Scenario (Number of quay cranes)	Ratios of quay cranes to AGVs used			
	1:2	1:3	1:4	1:5
2 quay cranes	4	6	8	10
4 quay cranes	8	12	16	20
6 quay cranes	12	18	24	30

As benchmarking techniques for vehicle dispatching, the nearest station and the most urgent task rules were adopted. Among pending tasks, the former assigns an available vehicle to a job whose origin is the closest to the vehicle. Therefore, this rule is likely to decrease vehicular setup travel distances and time. On the other hand, the latter allocates an idle container transporter to the most urgent delivery mission based on EPETWD. This policy mainly focuses on reducing AGV arrival delays in the quay and the yard.

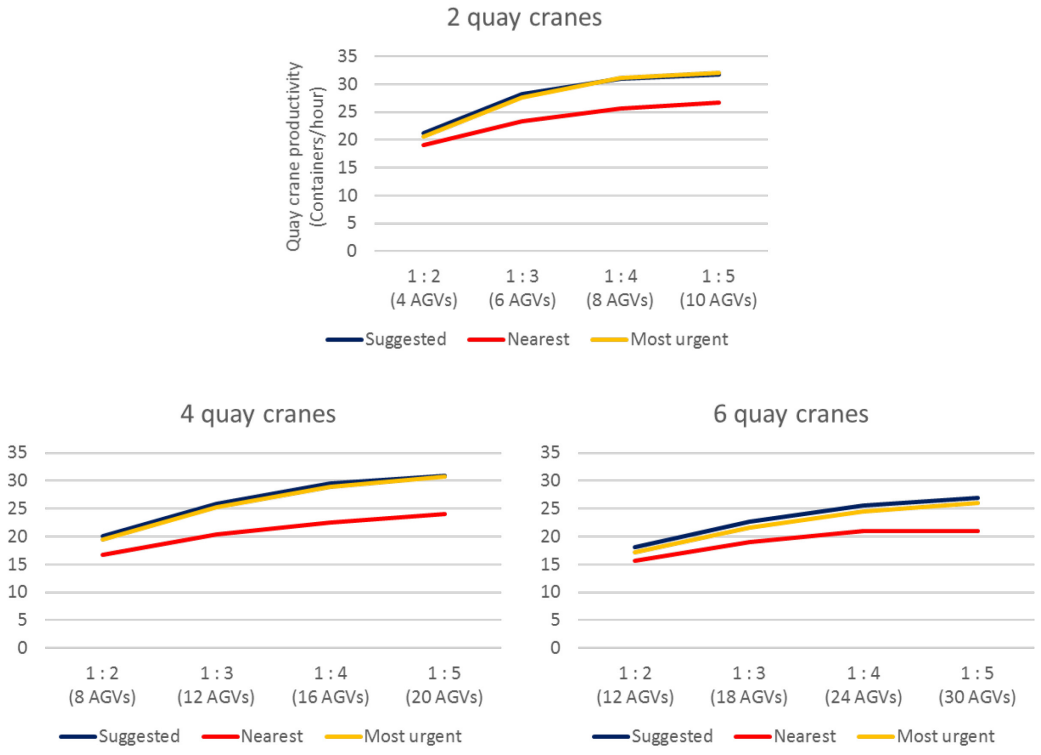


Figure 7-3. Average quay crane productivity per crane per hour.

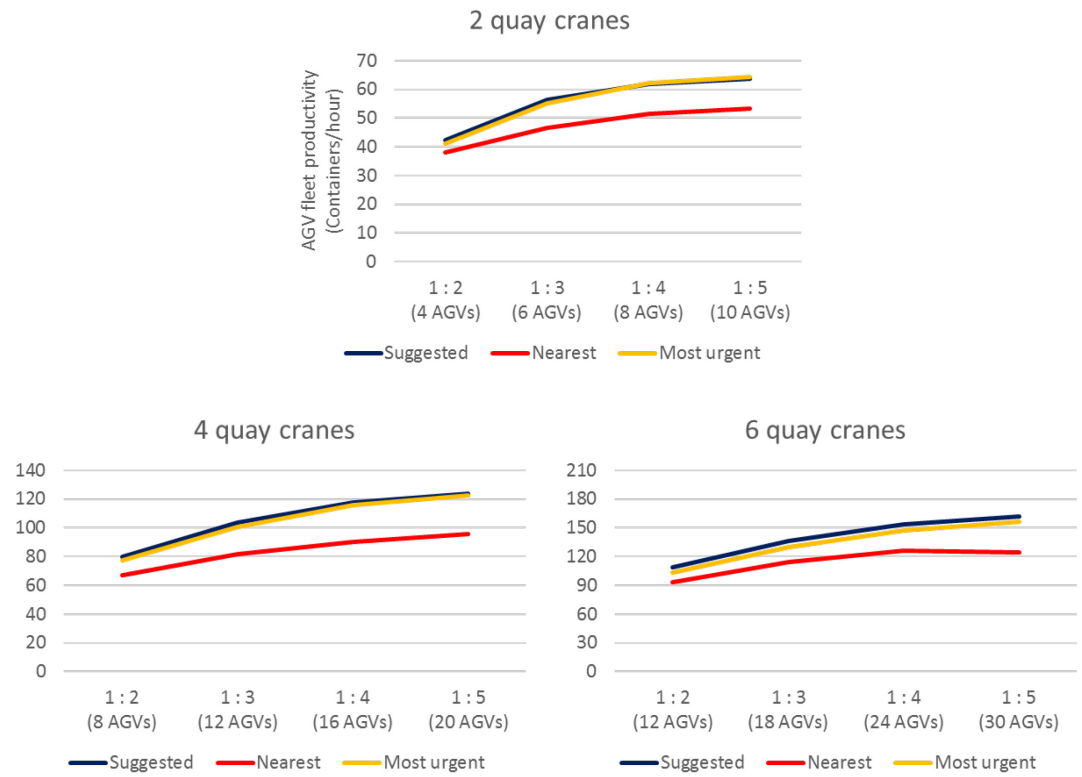


Figure 7-4. Average AGV fleet productivity per hour.

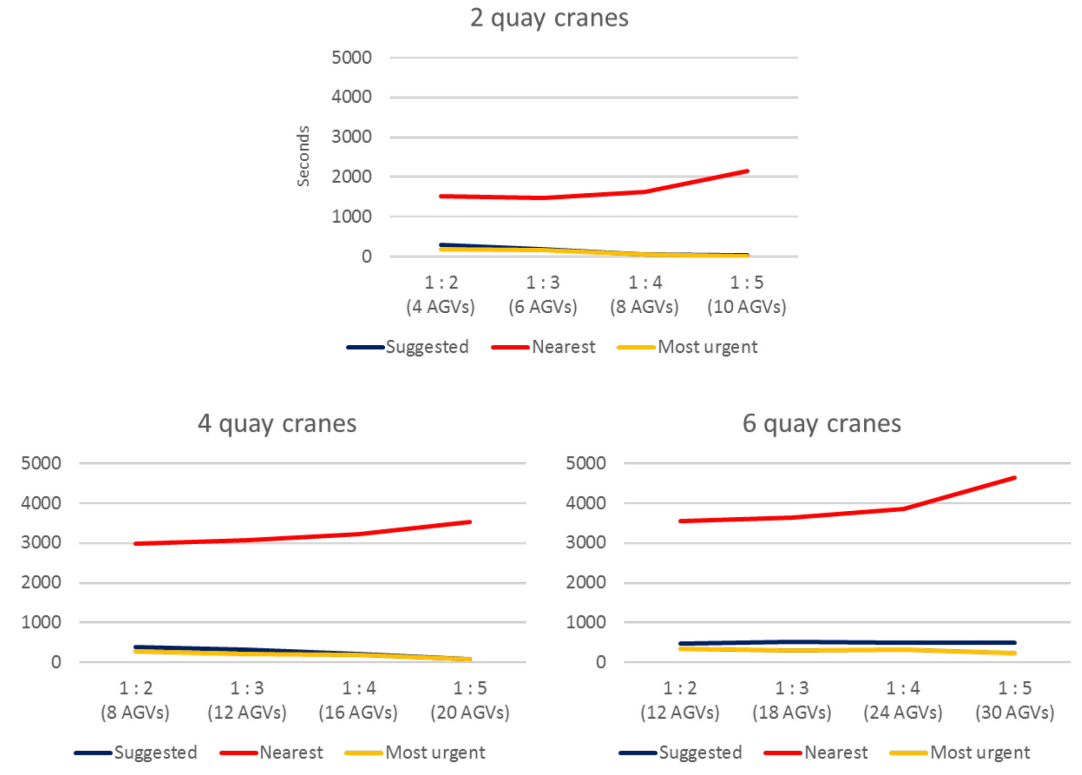


Figure 7-5. Standard deviations of quay crane idle time.

Figure 7-3 and *Figure 7-4* shows quay crane and AGV fleet productivity changes respectively while including the three scenarios in *Table 7-3*. The x-axis refers to the ratio of the number of quay cranes and the number of AGVs used, from 2 to 5. The y-axis means the number of containers processed per crane per hour in *Figure 7-3*, and the number of containers delivered by the AGV fleet per hour in *Figure 7-4*.

In the results, the two types of productivity showed the same feature, revealing several findings. Firstly, in all the cases, terminal resource productivity increases were not significant after the AGV fleet size reached a ratio of four to the number of quay cranes. Secondly, the most urgent task rule and the suggested dispatching model showed almost equivalent performance whereas the nearest station rule was inferior to the other dispatching techniques. The result means that the quay cranes were not evenly utilised during the simulation as summarised in *Figure 7-5*.

Thirdly, under the most urgent task rule and the suggested dispatching model, the proposed fleet trajectory planning algorithm helped to achieve a quay crane service rate of 30 containers per hour, in the first two scenarios with 4 as the AGV fleet size ratio to the number of quay cranes used. This productivity level is regarded as the maximum quay crane productivity by skilled operators and reference productivity by quay crane automation in practice (Mongelluzzo, 2015). However, in the last scenario, the busiest traffic case with 30 AGVs managed to allow each quay crane to process 27 containers per hour. Thus, the relative fleet size of AGVs used needs to increase to achieve the reference level of quay crane productivity.

The following three figures (*Figure 7-6*, *Figure 7-7* and *Figure 7-8*) display average AGV arrival delay time in the quay and the yard and average setup travel time observed in the tests. To begin with, AGV arrival delay time in both the vertical container handling blocks was reduced by the increases in employed AGVs. Like quay and AGV fleet productivity, the effect of an AGV fleet size increase became smaller.

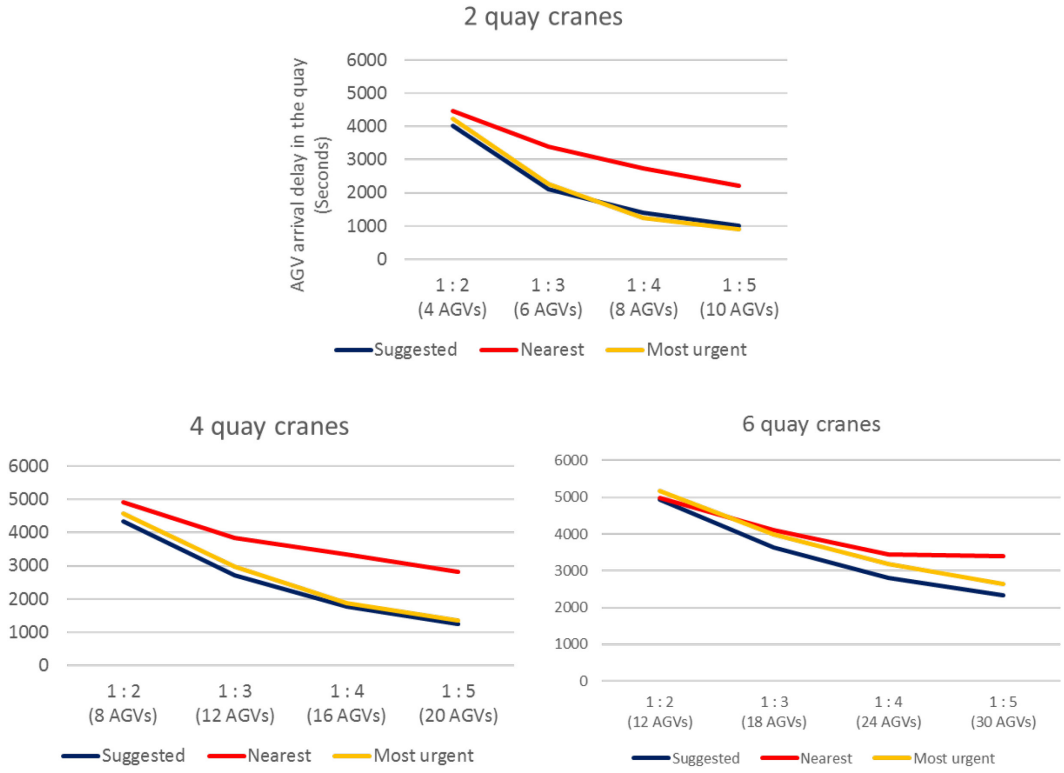


Figure 7-6. Average AGV arrival delay per vehicle in the quayside.

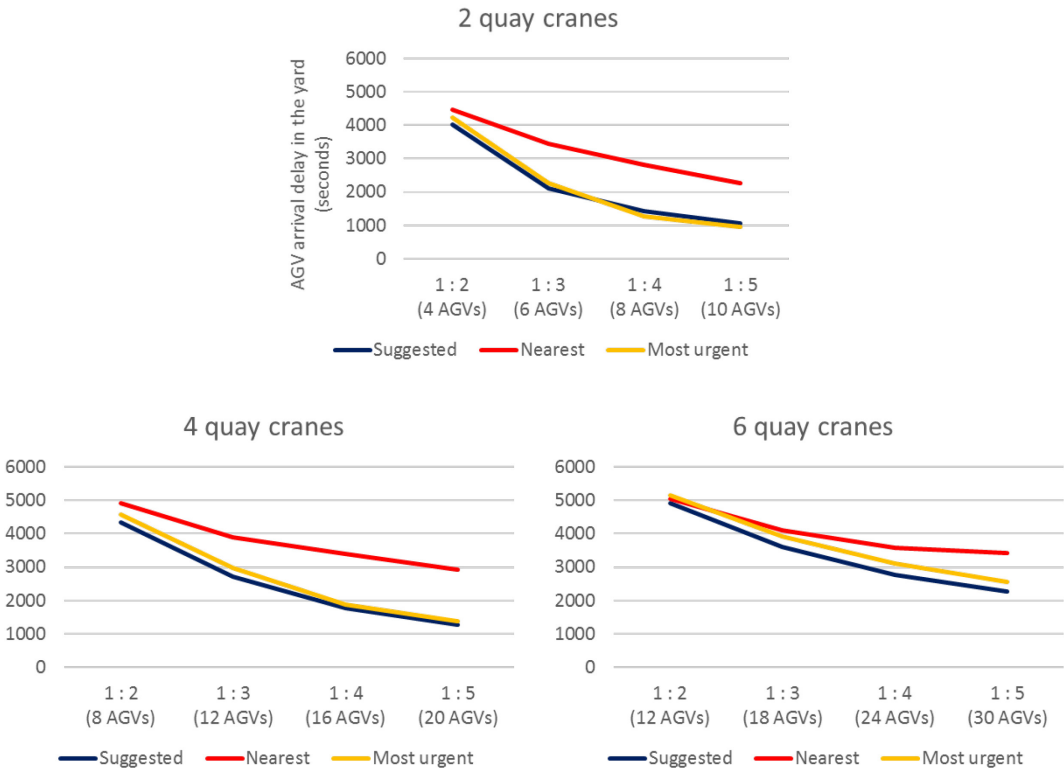


Figure 7-7. Average AGV arrival delay per vehicle in the yard side.

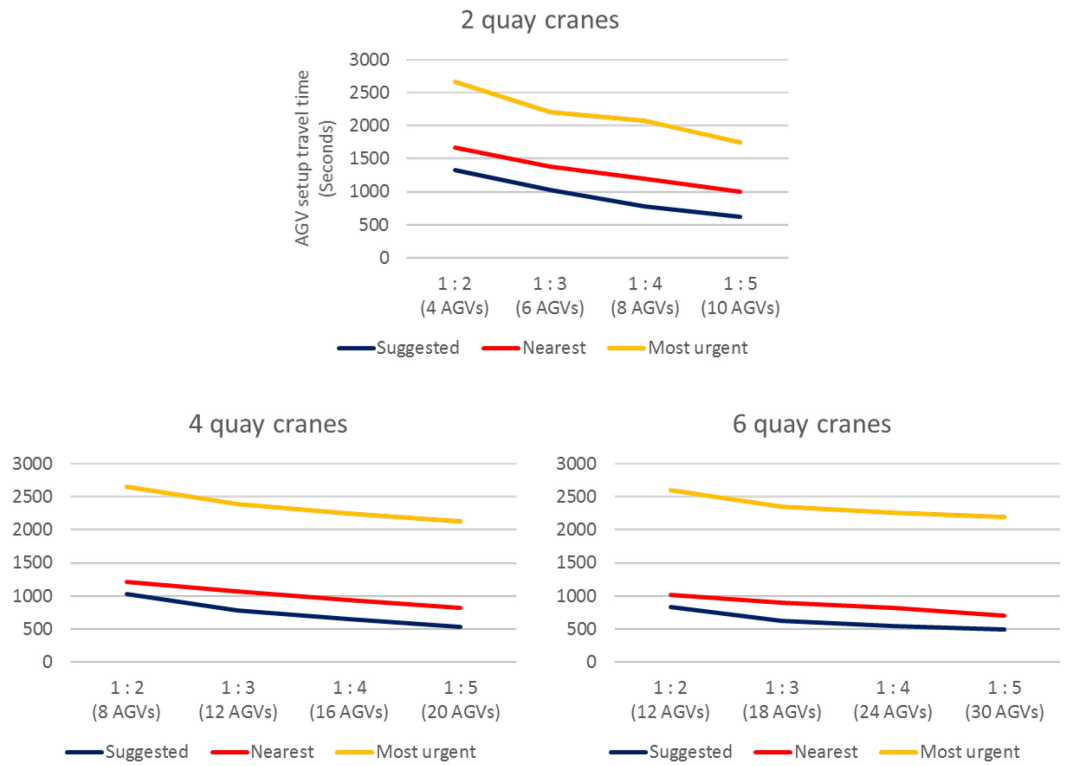


Figure 7-8. Average setup travel time per AGV.

The nearest station rule also showed the worst performance regarding vehicular arrival tardiness; especially, as the AGV fleet size increased, performance differences tended to become larger. The other AGV dispatching strategies showed almost similar performance in the first two scenarios including 2 and 4 quay cranes respectively. In the third scenario, the suggested model slightly outperformed the most urgent task policy.

In addition, it was observed that more vehicles guaranteed less setup travel time in all the scenarios. Unlike the two AGV arrival delay indicators, AGV setup travel time gently decreased, especially in the third scenario. The nearest station rule showed 1.5 to 3 times better performance compared to the most urgent task policy. The demonstrated strength is rather predictable due to its conceptual objective. As a compensation for less setup travel time, the nearest station rule could not achieve a high level of job delay minimisation.

Although the suggested model seemed not to outperform the most urgent task rule in AGV arrival delay time for container delivery jobs, it showed rather better performance in vehicular setup travel time, even compared to the nearest station policy. It outperformed the nearest

station rule 1.2 to 1.8 times, depending on the number of quay cranes and the AGV fleet size used. When it is compared to the most urgent task rule, the proposed dispatcher overwhelmed the counterpart while showing about 1,000 to 1,800 seconds less setup travel time. *Table 7-4* summaries the competitiveness of the three dispatching ideas based on the reviewed results.

Table 7-4. Relative performance of the three dispatching strategies

Relative Performance	Quay crane / AGV fleet productivity	AGV arrival delay time in the quay and the yard	AGV setup travel time
Suggested	+	+	++
Nearest	-	-	+
Most urgent	+	+	--

* + / -: Positive / negative

Figure 7-9, *Figure 7-10* and *Figure 7-11* pertain to observed AGV journeys implemented by the fleet trajectory planning algorithm explained in *Chapter 5*. Although the figures showed similar features under the different dispatching strategies, the suggested dispatching model slightly outperformed the basic heuristics regarding travel distances and time.

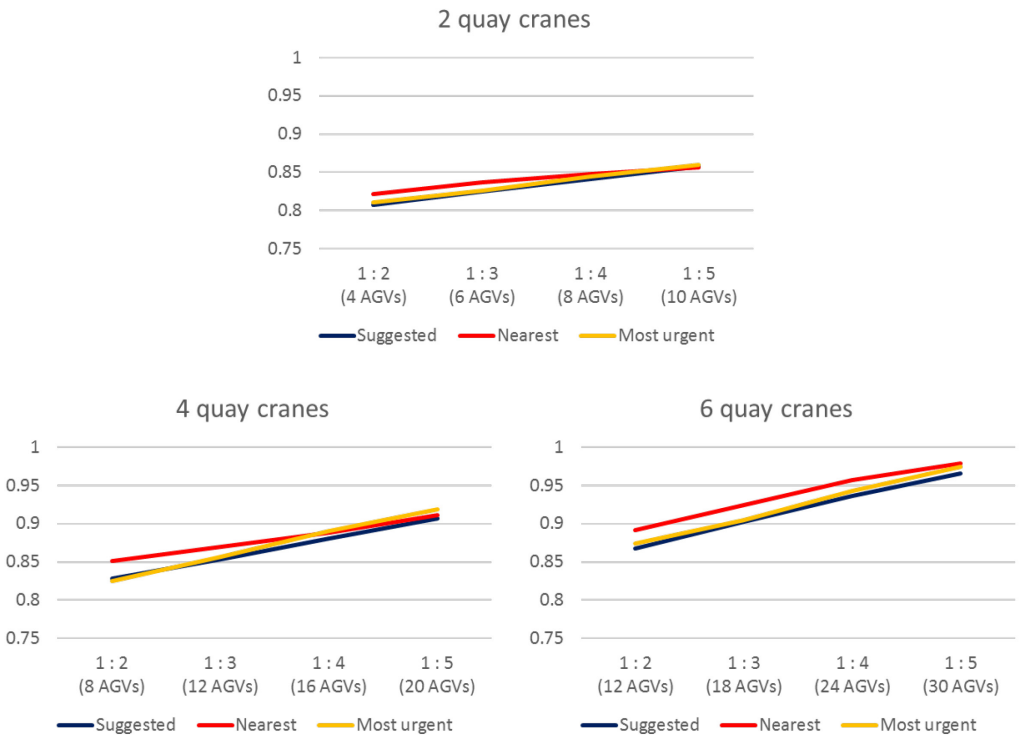


Figure 7-9. Average travelled distance ratio to the Manhattan distance.

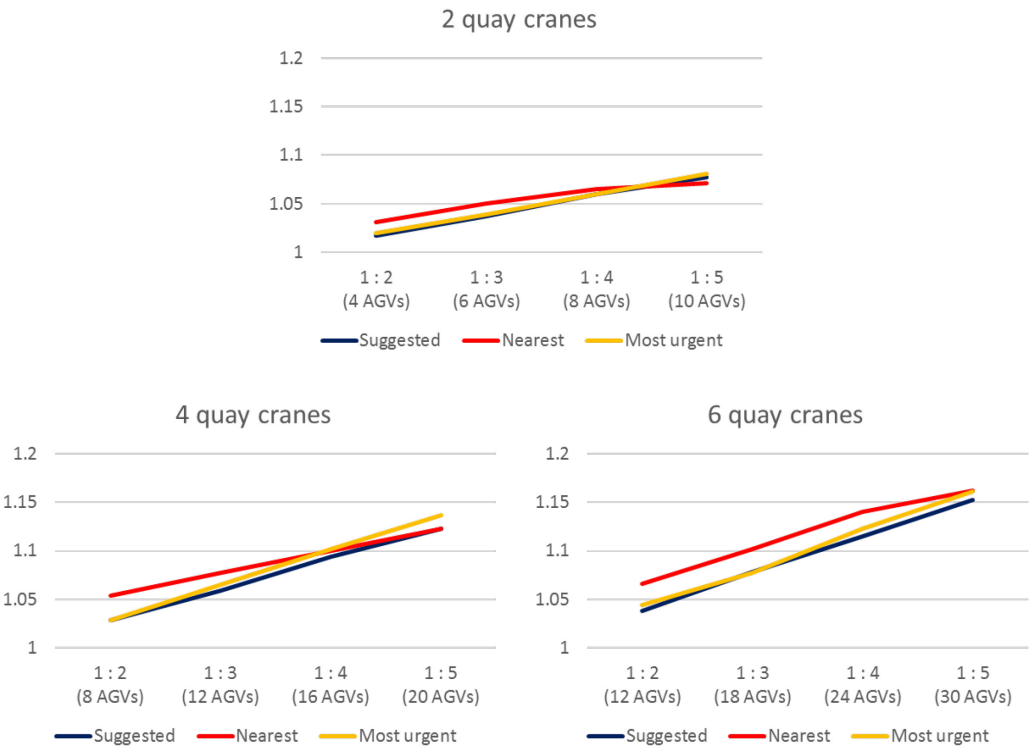


Figure 7-10. Average travelled distance ratio to the Dubins shortest paths.

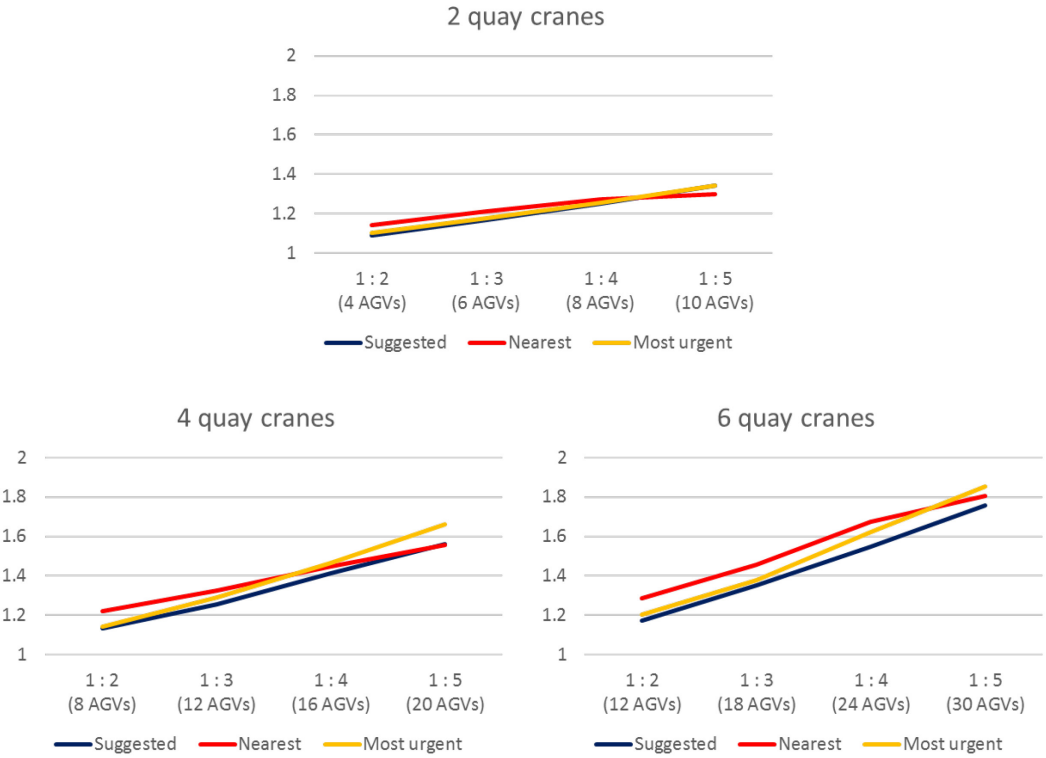


Figure 7-11. Average travel time ratio to the theoretical quickest travel time.

As *Figure 7-9* depicts, on average, the proposed fleet trajectory planning algorithm allowed the simulated AGVs to travel along shorter routes than the Manhattan distances, which are the theoretical minimum distances in most predetermined guide-path networks. Therefore, it would outperform any trajectory planning algorithms for use in the path-based systems regarding vehicular travel distances. Over 30 AGVs are likely to result in longer paths than the reference minimum distances under the default settings summarised in *Appendix A*.

Compared to the Dubins shortest paths (*Figure 7-10*), the observed vehicular travelled distances were inferior by 15% in the most crowded case whereas there was only around a 5% difference in the cases where the AGV fleet size was relatively small (i.e. 8 AGVs or below). When the fleet size reached around 20, the average observed travelled distance was 10% larger than the theoretical shortest path length.

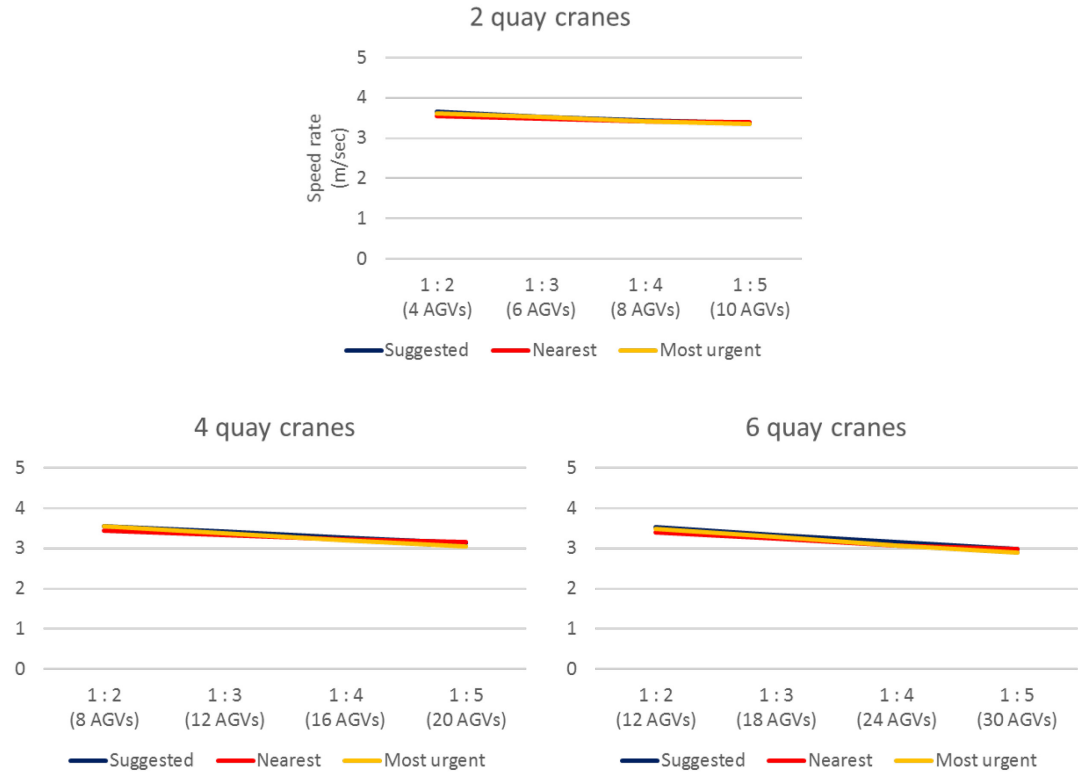


Figure 7-12. Average fleet speed rate with containers.

On the other hand, as depicted in *Figure 7-11*, the simulated AGVs required 1.75 times more travel time than the theoretical quickest travel time to complete their deliveries in the worst

case, including 30 AGVs. Performance degradation of travel time by AGV fleet size increases was rather rapid compared to the distance-based performance indicators. An employment of an additional AGV in the vehicle system led to a 3 to 4% increase in the ratio of the theoretical quickest travel time to the observed travel time whereas it did a less than 1% increase in the ratio of the shortest travel distance to the observed travelled distances.

Figure 7-12 shows average AGV fleet speed rates observed in the simulation tests. In each scenario, the difference between the highest and lowest was around 0.5 m/sec with the maximum speed rate of 5 m/sec. This amount is relatively significant while considering the observed average speed of a fleet of 4 AGVs - approximately 14% of the observed speed.

Additional tests were performed to identify the impact of iterations in the quality of AGV allocation and AGV fleet trajectory planning solutions by changing the value of a cooling parameter in the suggested dispatching and trajectory planning models respectively. The tests included 4 quay cranes and 20 AGVs with 20 different seed values (i.e. quay crane schedules). 0.85, 0.90 and 0.95 were applied to the cooling parameter in the dispatching model first, and *Table 7-5* summaries the results.

Table 7-5. Performance changes by different cooling parameters (dispatching)

Performance indicator	AGV dispatcher cooling parameter values (iterations)		
	0.85 (57)	0.90 (88)	0.95 (180)
Quay crane productivity	30.566	30.544	30.994
AGV fleet productivity	122.82	122.64	124.02
AGV arrival delay in the quay	1351.095	1310.383	1269.501
AGV arrival delay in the yard	1379.941	1319.467	1279.737
Setup travel time	569.656	542.881	536.906
Travel distance ratio to the Manhattan distance	0.908	0.907	0.906
Travel distance ratio to the Dubins shortest	1.126	1.125	1.124
Travel time ratio to the theoretical quickest	1.565	1.575	1.563

* The units of the indicators in this table are the same in Table 7-2.
* The shaded area includes performance indicators that show remarkable changes.

To being with, performance in the two types of terminal resource productivity was not significantly improved by the parameter value increase from 0.85 to 0.95; quay crane productivity increased by 0.43 (i.e. 1.4%), and AGV fleet productivity rose by 1.2 (i.e. 1%). Also, no difference was found in the last three performance indicators, all of which are directly related to trajectory planning and implementation. Average fleet speed also remained stable at around 3.13 m/sec.

However, more iteration processes by parameter adjustments yielded remarkable performance improvements in the remaining three indicators that are considered in the AGV dispatching model's objective function. AGV arrival delay time in the quay dropped by 6%, and that in the yard decreased by 7.3%. 5.7% less setup travel time was also observed in the tests. Thus, it can be said that more iterations in the AGV dispatching algorithm bring less container handling delay time and AGV setup travel time.

Figure 7-13 depicts objective value changes observed while solving several AGV dispatching problem instances during the tests. The x-axis refers to the number of iterations, and the y-axis is an objective value. In each of the examples, the algorithm experienced a rapid solution quality improvement before the 30th iteration, and it slowly lowered the objective value by its iterative procedure thereafter.

Figure 7-13. Objective value changes (AGV dispatching).

Table 7-6. Performance changes by different cooling parameters (Trajectory planning)

Performance indicator	Fleet trajectory planning cooling parameter values		
	0.85	0.90	0.95
Quay crane productivity	30.791	30.848	30.994
AGV fleet productivity	123.24	123.78	124.02
AGV arrival delay in the quay	1371.899	1314.605	1269.501
AGV arrival delay in the yard	1387.967	1324.429	1279.737
Setup travel time	597.481	562.375	536.906
Travel distance ratio to the Manhattan distance	0.968	0.945	0.906
Travel distance ratio to the Dubins shortest	1.197	1.169	1.124
Travel time ratio to the theoretical quickest	2.067	1.844	1.563
Fleet speed rate	2.828	2.966	3.141

Table 7-6 shows that more iterations in the fleet trajectory planning algorithm advanced vehicle operation in container handling regarding several performance indicators, except for the two resource productivity areas. One remarkable point is that the parameter changes affected the performance areas of AGV dispatching as well as those of trajectory planning whereas AGV allocation plan improvements by more iterations did not significantly influence vehicular speed, travel time and distances.

Simply put, increasing the value of the cooling parameter from 0.85 to 0.95 brought decreases in AGV arrival delay time in the quay and the yard and vehicular setup travel time by 7.5%, 7.8% and 10.1% respectively. These improvements are even larger than those achieved by the parameter value changes summarised in *Table 7-5*.

In addition to the dispatching-related performance areas, it was observed that by more iterations the simulated AGVs travelled more quickly on shorter paths while requiring less travel time. Especially, the ratio of the theoretical quickest time to observed travel time drastically dropped by 0.504, compared to those of the other ratio indicators, 0.062 and 0.073 respectively. Vehicular speed also rose by 0.313 m/sec, which is over a 10% increase. *Figure 7-14* shows solution quality improvements by iterations observed in the fleet trajectory planning algorithm.

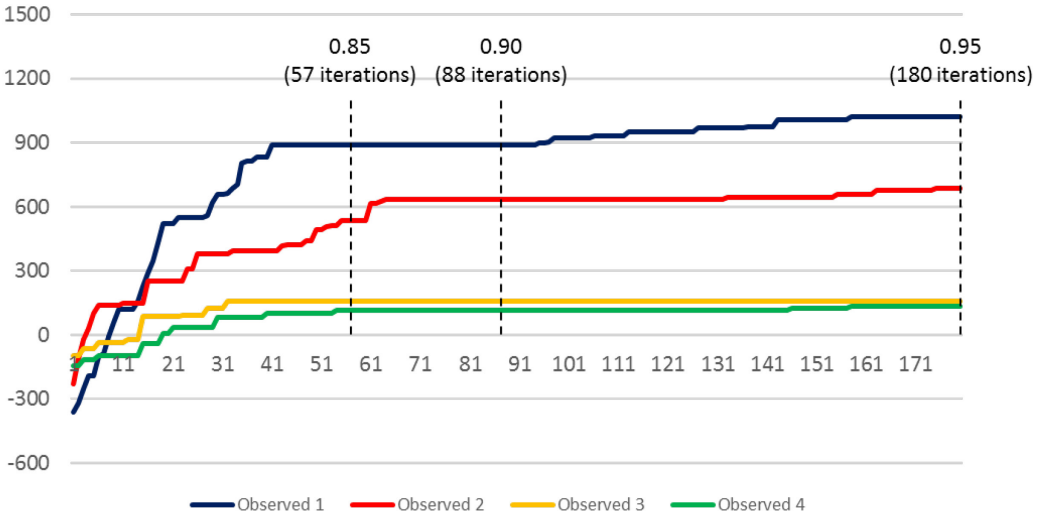


Figure 7-14. Objective value changes (fleet trajectory planning).

Finally, the impact of flexible safety clearance zones was identified. Like the parameter adjustment tests, 4 quay cranes and 20 AGVs were adopted in the used simulation environment, and a series of tests were implemented based on 20 different seed values. Two different settings were utilised in safety clearance zone generation for each AGV: one for providing the maximum size zone that covers all possible situations and the other for generating a safety zone based on the load state of the vehicle and its speed rate.

Table 7-7 summarises the simulation results, and it was revealed that allowing the flexibility of safety clearance zones in size significantly increased performance in all the indicators listed in *Table 7-2*. To be clearer, the two terminal resource productivity areas experienced approximately a 10% performance increase, and the safety zone flexibility led to a 14.5% decrease in vehicular setup travel time.

Drastic changes were especially observed in AGV arrival delay time in the quay and yard: AGV operation was improved by over 40% in the relevant performance indicators, both of which can be key terms in vehicle dispatching models. It was also identified that the simulated AGVs with their flexible safety zones could travel 50% faster than those with fixed safety fields. This is because the former AGVs each required the less safety space in its journeys, causing the capability of higher speed in the same separation case between two AGVs. Due to

the travelled distance decrease and the fleet speed increase, the travel time ratio rapidly dropped to a third in the tests. Thus, it can be said that the free-ranging capability of AGVs should work with compact safety clearance zones to achieve advanced vehicle operation.

Table 7-7. Performance changes by safety clearance zone flexibility in size

Performance indicator	Safety clearance zone	
	Fixed	Flexible
Quay crane productivity	27.353	30.893
AGV fleet productivity	109.413	123.57
AGV arrival delay in the quay	2262.339	1287.496
AGV arrival delay in the yard	2224.065	1304.445
Setup travel time	621.563	531.64
Travel distance ratio to the Manhattan distance	2.082	3.136
Travel distance ratio to the Dubins shortest	1.283	0.907
Travel time ratio to the theoretical quickest	1.574	1.124
Fleet speed rate	4.881	1.563

7.5. Summary

This chapter describes the simulation domain defined from a real container terminal, TTI Algeciras, and provides the results obtained by a series of simulation experiments to evaluate the proposed AGV dispatching and fleet trajectory planning algorithms. Various scenarios, cases and performance indicators were carefully chosen while adjusting the values of the cooling parameters in the proposed algorithms.

Port authorities and terminal operators would be interested in the potential application of the algorithms. In the simulation results, the dispatching algorithm showed better performance compared to the nearest station rule and the most urgent job rule, both of which are conventional, popular dispatching heuristics. The proposed algorithm outperformed the nearest station rule in quay crane and AGV productivity areas, arrival delay time and vehicular setup travel time. At the same time, it overwhelmed the most urgent task rule in AGV setup travel time although they showed almost equivalent performance in the other performance

indicators. Nonetheless, the proposed algorithm is still more favourable because less setup travel time increases the operational efficiency of AGVs, and therefore improves the lifespan of the vehicles.

On the side of trajectory planning, the free-ranging vehicle setting showed favourable features in AGV journeys compared to conventional environments that include pre-defined flow-paths, especially with a relatively small fleet size of AGVs. Resulting vehicular trajectories are shorter than the corresponding Manhattan distances, which are the minimum in predetermined guide-path systems. Although the observed speed rates of the simulated AGVs were over their maximum turning speed without significant variations, discrepancies from the theoretical quickest travel time to observed travel time increasingly became large by increases in the number of AGVs employed. Thus, the suggested free-ranging vehicle algorithm would be more appropriate with less AGVs on use.

The impact of flexible safety clearance zones for AGVs is also worth of mention. Although fixed-sized safety zones provide the easiness of applications, they are likely to bring about performance degradation in both vehicle dispatching and fleet trajectory planning. This supports the necessity of compact safety clearance zones dynamically changed by vehicular states, such as speed.

Before real applications, it would be necessary to carefully discuss a couple of implementation challenges. Firstly, due to simplicity this research does not consider the dual-load capability of container AGVs. However, the vehicular feature would influence performance of the proposed dispatching algorithm. Secondly, more rigorous experiments need to be performed to check that deadlocks do not happen in target application environments. Although deadlocks did not occur in the implemented simulation tests, it does not guarantee that the fleet trajectory algorithm always achieves deadlock-free vehicular journeys in any case.

Chapter 8. Conclusions and Further Research

Without advanced control software, the benefits of highly sophisticated handling equipment cannot fully be utilised in any automated system, including ACTs. In other words, ACTs would be inferior to their conventional form if appropriate control software for automated container handling equipment was not applied. In current ACTs, AGVs have yet to operate in an advanced way as they are designed. Thus, container terminal operators have been reluctant to embrace the new form of terminal automation or make physical changes in their domains that will bring about significant investment costs. This research, therefore, has focused on a couple of main issues in current AGV systems that can be resolved by advanced AGV operation planning.

8.1. Conclusions

In order to achieve the aim of the research (i.e. the development of an integrated AGV operation planner and a simulation model that focuses on the transport area of ACTs under the free-ranging vehicle setting), this study has handled the several objectives explained at the beginning of the thesis. Each objective has been achieved and explained chapter-by-chapter. The summaries are as follows:

- *Chapter 2* includes a comprehensive review on existing vehicle dispatching, trajectory planning and simulation models for vehicle operation analysis. It introduces a description of terminal automation in vertical and horizontal container handling and shows the mainstream of unmanned vehicle operation planning and simulation in vehicle dispatching and trajectory planning. It emphasises the gap between the application settings of the existing models and potential ACTs where free-ranging AGVs operate in an advanced manner (i.e. the target domain of this research).

- *Chapter 3* suggests and clearly explains the research methodology of this thesis with a process diagram that shows the research flow and relationships between performed research activities.
- A flexible AGV dispatching model has been developed that allows AGVs on their setup travel to abandon their current mission and take new container delivery jobs with the use of SA for potential real-time applications. The model also considers task-sequencing under the idea of mid-term dispatching to reduce vehicular setup travel as an objective of AGV dispatching (*0*).
- A fleet trajectory planning algorithm based on SA has been developed for nonholonomic AGVs capable of operating without preinstalled guide paths. This model achieves trajectory followability by considering curve smoothness and the higher linear speed rate than the turning speed of container AGVs. In conjunction with rolling horizons, the model improves Dubins curves by two additional techniques to change the early manoeuvres of vehicles to avoid detected collision threats (*Chapter 5*).
- *Chapter 6* introduces and identifies a simulation model programmed to test and evaluate the vehicle dispatching algorithm and the AGV fleet trajectory planner in various container terminal settings without using predetermined flow-paths. This bespoke simulation model mainly covers the transport area of container terminals where AGVs operate to deliver containers between the quay and the yard. Vertical container handling by quay and yard cranes is briefly included to support AGV operation analysis.
- A series of simulation experiments were implemented with carefully-selected performance indicators subject to terminal resource productivity (related to quay cranes and AGVs), vehicle dispatching and fleet trajectory planning. *Chapter 7* displays their results on the proposed vehicle operation techniques through several container terminal settings: different resource group sizes (i.e. quay cranes and AGVs) and dispatching strategies. The new dispatcher showed balanced performance compared to the nearest station rule and the most urgent task rule, and the trajectory planning approach

guaranteed operational gains in required travelled distance and time compared to conventional ACTs subject to predetermined flow-paths, especially with the application of a small fleet of AGVs. Consequently, the developed algorithms by this research can be used as alternatives in existing AGV systems and control software.

8.2. Further Research

Potential further studies are here provided that surmount several limitations the thesis has. They are not confined to container terminals but cover various application settings and domains for automated vehicles.

- The AGV dispatching algorithm only allows a single unit of cargo per journey although AGVs for container transport can transport two 20ft containers at any time. Further research could include scenarios on the utilisation of the dual-load capability.
- As introduced at the beginning of this thesis, there are several types of automated container delivery vehicles (such as ALVs), and AGVs are only one of the types. Nevertheless, all the vehicle settings share the core function, represented by horizontal container handling. Thus, extended work on vehicle dispatching and trajectory planning could be carried out with these types of equipment.
- Parametric sensitivity analysis in the proposed vehicle dispatching and trajectory planning models could be performed to identify operational changes in detail, especially in industrial sectors. It could focus on either vehicle dispatching or AGV fleet trajectory planning, or cover the whole vehicle operation scope.
- Traffic control in crane buffer areas was not considered in the implemented simulation experiments. Therefore, further studies could include AGV traffic management techniques near the crane buffer areas and combine the methods with the presented fleet trajectory planning framework for free-ranging AGVs.

- Container terminals are a complex system with a range of dependent operational blocks for container handling. This research, however, provided a simulation model that mainly focuses on the transport areas of AGVs while simplifying the operations of quay and yard cranes. With an expansion of research scope for terminal operation, more advanced simulation models could be required and developed in further research.

References

Al-Taharwa, I., Sheta, A. and Al-Weshah, M. (2008) 'A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment', *Journal of Computer Science*, 4(4), pp. 341–344.

Angeloudis, P. and Bell, M. G. H. (2010) 'An Uncertainty-aware AGV Assignment Algorithm for Automated Container Terminals', *Transportation Research Part E: Logistics and Transportation Review*, 46(3), pp. 354–366.

Askari, A. *et al.* (2016) 'A New Approach in UAV Path Planning Using Bezier-Dubins Continuous Curvature Path', *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 230(6), pp. 1103–1113.

Bartholdi III, J. J. and Platzman, L. K. (1989) 'Decentralized Control of Automated Guided Vehicles on a Simple Loop', *IIE Transactions*, 21(1), pp. 76–81.

Berger, J. *et al.* (2012) 'A New Mixed-integer Linear Programming Model for Rescue Path Planning in Uncertain Adversarial Environment', *Computers & Operations Research*, 39(12), pp. 3420–3430.

Bish, E. K. *et al.* (2005) 'Dispatching Vehicles in a Mega Container Terminal', *OR Spectrum*, 27(4), pp. 491–506.

Briskorn, D., Drexl, A. and Hartmann, S. (2006) 'Inventory-based Dispatching of Automated Guided Vehicles on Container Terminals', *OR Spectrum*, 28(4), pp. 611–630.

Cai, Z. and Peng, Z. (2002) 'Cooperative Coevolutionary Adaptive Genetic Algorithm in Path Planning of Cooperative Multi-mobile Robot Systems', *Journal of Intelligent and Robotic Systems*, 33(1), pp. 61–71.

Carlo, H. J. and Martínez-Acevedo, F. L. (2015) 'Priority Rules for Twin Automated Stacking Cranes that Collaborate', *Computers & Industrial Engineering*, 89, pp. 23–33.

Castalia Strategic Advisors (2012) *The Effect of Wages on Australian Port Costs and their*

Competitiveness in an International Context. Available at: http://www.castalia-advisors.com/files/Port_Wage_Cost_report_20120604.pdf (Accessed: 15 November 2017).

Cetin, B., Bikdash, M. and Hadaegh, F. Y. (2007) 'Hybrid Mixed-logical Linear Programming Algorithm for Collision-free Optimal Path Planning', *IET Control Theory & Applications*, 1(2), pp. 522–531.

Cetin, O. and Zagli, I. (2012) 'Continuous Airborne Communication Relay Approach Using Unmanned Aerial Vehicles', *Journal of Intelligent & Robotic Systems*, 65(1–4), pp. 549–562.

Cetin, O., Zagli, I. and Yilmaz, G. (2013) 'Establishing Obstacle and Collision Free Communication Relay for UAVs with Artificial Potential Fields', *Journal of Intelligent & Robotic Systems*, 69(1–4), pp. 361–372.

Chen, X. *et al.* (2013) 'A Fast Two-stage ACO Algorithm for Robotic Path Planning', *Neural Computing and Applications*, 22(2), pp. 313–319.

Chen, Y., Han, J. and Zhao, X. (2012) 'Three-dimensional Path Planning for Unmanned Aerial Vehicle Based on Linear Programming', *Robotica*, 30(5), pp. 773–781.

Chiew, K. (2012) 'Scheduling and Routing of Autonomous Moving Objects on a Mesh Topology', *Operational Research*, 12(3), pp. 385–397.

Chiew, K. and Qin, S. (2009) 'Scheduling and Routing of AMOs in an Intelligent Transport System', *IEEE Transactions on Intelligent Transportation Systems*, 10(3), pp. 547–552.

Chiu, M.-C. (2010) 'Numerical Assessment of Path Planning for an Autonomous Robot Passing through Multi-layer Barrier Systems Using a Genetic Algorithm', *Information Technology Journal*, 9(7), pp. 1483–1489.

Chung, J.-H., Yoon, H. and Maeng, S. R. (1994) 'A New Deadlock Prevention Scheme for Nonminimal Adaptive Wormhole Routing', *Microprocessing and Microprogramming*, 40(7), pp. 465–486.

Coffman, E. G., Elphick, M. and Shoshani, A. (1971) 'System Deadlocks', *ACM Computing*

Surveys (CSUR), 3(2), pp. 67–78.

Conde, R. *et al.* (2012) 'Conflict Detection and Resolution Method for Cooperating Unmanned Aerial Vehicles', *Journal of Intelligent & Robotic Systems*, 65(1–4), pp. 495–505.

Corke, P. (2011) *Robotics, Vision and Control : Fundamental Algorithms in MATLAB*. Berlin, Germany: Springer.

Corman, F. *et al.* (2016) 'Optimal Scheduling and Routing of Free-range AGVs at Large Scale Automated Container Terminals', *Periodica Polytechnica. Transportation Engineering*, 44(3), pp. 145–154.

Corréa, A. I., Langevin, A. and Rousseau, L.-M. (2007) 'Scheduling and Routing of Automated Guided Vehicles: A Hybrid Approach', *Computers & Operations Research*, 34(6), pp. 1688–1707.

Cortés, P. *et al.* (2007) 'Simulation of freight traffic in the Seville inland port', *Simulation Modelling Practice and Theory*, 15(3), pp. 256–271.

Cui, R., Li, Y. and Yan, W. (2016) 'Mutual Information-Based Multi-AUV Path Planning for Scalar Field Sampling Using Multidimensional RRT*', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(7), pp. 993–1004.

Dantzig, G. B. and Ramser, J. H. (1959) 'The Truck Dispatching Problem', *Management Science*, 6(1), pp. 80–91.

Deepak, B. B. V. L., Parhi, D. R. and Raju, B. M. V. A. (2014) 'Advance Particle Swarm Optimization-based Navigational Controller for Mobile Robot', *Arabian Journal for Science and Engineering*, 39(8), pp. 6477–6487.

Deepak, B. and Parhi, D. R. (2012) 'PSO Based Path Planner of an Autonomous Mobile Robot', *Central European Journal of Computer Science*, 2(2), pp. 152–168.

Ding, X. C., Rahmani, A. R. and Egerstedt, M. (2010) 'Multi-UAV Convoy Protection: An Optimal Approach to Path Planning and Coordination', *IEEE Transactions on Robotics*, 26(2),

pp. 256–268.

Du, K.-L. and Swamy, M. N. S. (2016) *Search and Optimization by Metaheuristics*. Springer International Publishing.

Dubins, L. E. (1957) 'On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents', *American Journal of Mathematics*, 79(3), pp. 497–516.

Dudek, G. and Jenkin, M. (2010) *Computational Principles of Mobile Robotics*. Second Edi. Cambridge University Press.

Dugarjav, B. *et al.* (2013) 'Scan Matching Online Cell Decomposition for Coverage Path Planning in an Unknown Environment', *International Journal of Precision Engineering and Manufacturing*, 14(9), pp. 1551–1558.

Earl, M. G. and D'Andrea, R. (2005) 'Iterative MILP Methods for Vehicle-control Problems', *IEEE Transactions on Robotics*, 21(6), pp. 1158–1167.

Egbelu, P. J. and Tanchoco, J. M. A. (1984) 'Characterization of Automatic Guided Vehicle Dispatching Rules', *International Journal of Production Research*, 22(3), pp. 359–374.

De Filippis, L., Guglieri, G. and Quagliotti, F. (2011) 'A Minimum Risk Approach for Path Planning of UAVs', *Journal of Intelligent & Robotic Systems*, 61(1–4), pp. 203–219.

Flood, M. M. (1956) 'The Traveling-salesman Problem', *Operations Research*, 4(1), pp. 61–75.

Galceran, E. and Carreras, M. (2013) 'A Survey on Coverage Path Planning for Robotics', *Robotics and Autonomous Systems*, 61(12), pp. 1258–1276.

Gasparetto, A. *et al.* (2015) 'Path Planning and Trajectory Planning Algorithms: A General Overview', in. Springer International Publishing, pp. 3–27. doi: 10.1007/978-3-319-14705-5_1.

Ge, S. S. and Cui, Y. J. (2000) 'New Potential Functions for Mobile Robot Path Planning',

IEEE Transactions on Robotics and Automation, 16(5), pp. 615–620.

Ge, S. S. and Cui, Y. J. (2002) 'Dynamic Motion Planning for Mobile Robots Using Potential Field Method', *Autonomous Robots*, 13(3), pp. 207–222.

Geman, S. and Geman, D. (1984) 'Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6), pp. 721–741.

Gharehgozli, A. H. *et al.* (2015) 'Scheduling Twin Yard Cranes in a Container Block', *Transportation Science*, 49(3), pp. 686–705.

Ghasemzadeh, H., Behrangi, E. and Azgomi, M. A. (2009) 'Conflict-free Scheduling and Routing of Automated Guided Vehicles in Mesh Topologies', *Robotics and Autonomous Systems*, 57(6–7), pp. 738–748.

Ghita, N. and Kloetzer, M. (2012) 'Trajectory Planning for a Car-like Robot by Environment Abstraction', *Robotics and Autonomous Systems*, 60(4), pp. 609–619.

Gigras, Y. and Gupta, K. (2012) 'Ant Colony Based Path Planning Algorithm for Autonomous Robotic Vehicles', *International Journal of Artificial Intelligence & Applications (IJAIA)*, 3(6), pp. 31–38.

Gilman, S. (1999) 'The Size Economies and Network Efficiency of Large Containerships', *International Journal of Maritime Economics*, 1(1), pp. 39–59.

Google Maps (2017) *Map of Total Terminal International Algeciras SA*. Available at: https://www.google.co.uk/maps/place/Total+Terminal+International+Algeciras+SA/@36.1305816,-5.4249952,856a,35y,88.5h/data=!3m1!1e3!4m5!3m4!1s0xd0c952509ea06a5:0xf316a60e25150759!8m2!3d36.1299839!4d-5.4258029 (Accessed: 4 October 2017).

Grunow, M., Günther, H.-O. and Lehmann, M. (2006) 'Strategies for Dispatching AGVs at Automated Seaport Container Terminals', *OR Spectrum*, 28(4), pp. 587–610.

Gudelj, A., Kezić, D. and Vidačić, S. (2012) 'Planning and Optimization of AGV Jobs by Petri Net and Genetic Algorithm', *Journal of Information and Organizational Sciences*, 36(2), pp. 99–122.

Günther, H.-O. and Kim, K.-H. (2006) 'Container Terminals and Terminal Operations', *OR Spectrum*, 28(4), pp. 437–445.

Habib, D., Jamal, H. and Khan, S. A. (2013) 'Employing Multiple Unmanned Aerial Vehicles for Co-operative Path Planning', *International Journal of Advanced Robotic Systems*, 10(5), pp. 1–9.

Henesey, L. (2004) 'A multi agent based simulator for managing a container terminal', in *Proceedings of the 2nd European Workshop on Multi-Agent Systems (EUMAS 2004)*. Barcelona, Spain, pp. 291–302.

Hillier, F. S. and Lieberman, G. J. (2005) *Introduction to Operations Research*. 8th edn. Singapore: McGrawHill.

Huang, W. and Chung, P. W. H. (2005) 'Integrating Routing and Scheduling for Pipeless Plants in Different Layouts', *Computers & Chemical Engineering*, 29(5), pp. 1069–1081.

Hussein, A. *et al.* (2012) 'Metaheuristic Optimization Approach to Mobile Robot Path Planning', in *2012 International Conference on Engineering and Technology (ICET)*. Cairo, Egypt: IEEE, pp. 1–6.

Jeon, S. M., Kim, K. H. and Kopfer, H. (2011) 'Routing Automated Guided Vehicles in Container Terminals through the Q-learning Technique', *Logistics Research*, 3(1), pp. 19–27.

Jolly, K. G., Sreerama Kumar, R. and Vijayakumar, R. (2009) 'A Bezier Curve Based Path Planning in a Multi-agent Robot Soccer System without Violating the Acceleration Limits', *Robotics and Autonomous Systems*, 57(1), pp. 23–33.

Karaman, S. and Frazzoli, E. (2011) 'Sampling-based algorithms for optimal motion planning', *The International Journal of Robotics Research*, 30(7), pp. 846–894.

Karumanchi, S., Iagnemma, K. and Scheding, S. (2013) 'Mobility Erosion: High Speed Motion Safety for Mobile Robots Operating in Off-road Terrain', in *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany: IEEE, pp. 397–402.

Khaksar, W. *et al.* (2012) 'Sampling-based Tabu Search Approach for Online Path Planning', *Advanced Robotics*, 26(8–9), pp. 1013–1034.

Kim, B.-I. *et al.* (2007) 'Effectiveness of vehicle reassignment in a large-scale overhead hoist transport system', *International Journal of Production Research*, 45(4), pp. 789–802.

Kim, B.-I. *et al.* (2009) 'Effective overhead hoist transport dispatching based on the Hungarian algorithm for a large semiconductor FAB', *International Journal of Production Research*, 47(10), pp. 2823–2834.

Kim, C. W., Tanchoco, J. M. A. and Koo, P.-H. (1999) 'AGV Dispatching Based on Workload Balancing', *International Journal of Production Research*, 37(17), pp. 4053–4066.

Kim, D.-H. *et al.* (2015) 'Adaptive rapidly-exploring random tree for efficient path planning of high-degree-of-freedom articulated robots', *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 229(18), pp. 3361–3367.

Kim, D. H. and Shin, S. (2006) 'Local Path Planning Using a New Artificial Potential Function Composition and Its Analytical Design Guidelines', *Advanced Robotics*, 20(1), pp. 115–135.

Kim, J., Kim, M. and Kim, D. (2011) 'Variants of the Quantized Visibility Graph for Efficient Path Planning', *Advanced Robotics*, 25(18), pp. 2341–2360.

Kim, K.-H. and Bae, J.-W. (1999) 'A Dispatching Method for Automated Guided Vehicles to Minimize Delays of Containership Operations', *International Journal of Management Science*, 5(1), pp. 1–25.

Kim, K. H. and Bae, J. W. (2004) 'A Look-ahead Dispatching Method for Automated Guided Vehicles in Automated Port Container Terminals', *Transportation Science*, 38(2), pp. 224–234.

Kim, K. H., Jeon, S. M. and Ryu, K. R. (2006) 'Deadlock Prevention for Automated Guided Vehicles in Automated Container Terminals', *OR Spectrum*, 28(4), pp. 659–679.

Kim, K. H., Park, Y. M. and Ryu, K.-R. (2000) 'Deriving decision rules to locate export containers in container yards', *European Journal of Operational Research*, 124(1), pp. 89–101.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983) 'Optimization by Simulated Annealing', *Science, New Series*, 220(4598), pp. 671–680.

Konecranes Gottwald (2017) *AGV*. Available at: http://www.konecranes.com/equipment/container-handling-equipment/automated-guided-vehicles/agv (Accessed: 13 September 2017).

Lam, S. K. and Srikanthan, T. (2001) 'High-speed Environment Representation Scheme for Dynamic Path Planning', *Journal of Intelligent and Robotic Systems*, 32(3), pp. 307–319.

Latombe, J.-C. (1991) *Robot Motion Planning*. Springer Science & Business Media.

Lauffenburger, J. P. *et al.* (2003) 'Driver-aid system using path-planning for lateral vehicle control', *Control Engineering Practice*, 11(2), pp. 217–231.

LaValle, S. M. (2006) *Planning Algorithms*. Cambridge University Press.

Lavalle, S. M. and Kuffner, J. J. (2001) 'Randomized Kinodynamic Planning', *The International Journal of Robotics Research*, 20(5), pp. 378–400.

Le-Anh, T. and De Koster, M. B. M. (2006) 'A Review of Design and Control of Automated Guided Vehicle Systems', *European Journal of Operational Research*, 171(1), pp. 1–23.

Le, H. M., Yassine, A. and Moussi, R. (2012) 'DCA for Solving the Scheduling of Lifting Vehicle in an Automated Port Container Terminal', *Computational Management Science*, 9(2), pp. 273–286.

Lee, S. and Cho, G. (2007) 'A simulation study for the operations analysis of dynamic planning in container terminals considering RTLS', in *Innovative Computing, Information and*

Control, 2007. ICICIC'07. Second International Conference on. Kumamoto, Japan: IEEE.

Lehmann, M., Grunow, M. and Günther, H.-O. (2006) 'Deadlock Handling for Real-time Control of AGVs at Automated Container Terminals', *OR Spectrum*, 28(4), pp. 631–657.

Levinson, M. (2016) *The Box : How the Shipping Container Made the World Smaller and the World Economy Bigger*. Second Edi. Princeton University Press.

Li, Q. *et al.* (2006) 'An Improved Genetic Algorithm of Optimum Path Planning for Mobile Robots', in *Sixth International Conference on Intelligent Systems Design and Applications*. Jinan, China: IEEE, pp. 637–642.

Li, X. *et al.* (2017) 'Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles', *Mechanical Systems and Signal Processing*, 87, pp. 118–137.

Lim, J. K. *et al.* (2003) 'A Dispatching Method for Automated Guided Vehicles by Using a Bidding Concept', *OR Spectrum*, 25(1), pp. 25–44.

Lin, J. T., Chang, C. C. K. and Liu, W.-C. (1994) 'A Load-routeing Problem in a Tandem-configuration Automated Guided-vehicle System', *International Journal of Production Research*, 32(2), pp. 411–427.

Lin, J. T. and Dgen, P.-K. (1994) 'An Algorithm for Routeing Control of a Tandem Automated Guided Vehicle System', *International Journal of Production Research*, 32(12), pp. 2735–2750.

Liu, C.-I., Jula, H. and Ioannou, P. A. (2002) 'Design, Simulation, and Evaluation of Automated Container Terminals', *IEEE Transactions on Intelligent Transportation Systems*, 3(1), pp. 12–26.

Liu, C., Liu, H. and Yang, J. (2011) 'A Path Planning Method Based on Adaptive Genetic Algorithm for Mobile Robot', *Journal of Information & Computational Science*, 8(5), pp. 808–814.

Louppova, J. (2017) *HMM visited Algeciras to discuss TTI sale*, *port.today*. Available at: https://port.today/hmm-visited-algeciras-discuss-tti-sale/ (Accessed: 4 October 2017).

Marsh, D. (2005) *Applied Geometry for Computer Graphics and CAD*. Second Edi. Springer.

Meisel, F. (2009) *Seaside Operations Planning in Container Terminals*. Berlin, Germany: Physica-Verlag.

Merkuryev, Y., Kamperman, F. and Visipkov, V. (2000) 'ARENA-based simulation of logistics processes at the Baltic container terminal', in *Proceedings of the 14th European Simulation Multiconference on Simulation and Modelling: Enablers for a Better Quality of Life*, pp. 433–437.

Miao, H. and Tian, Y.-C. (2013) 'Dynamic Robot Path Planning Using an Enhanced Simulated Annealing Approach', *Applied Mathematics and Computation*, 222, pp. 420–437.

Mongelluzzo, B. (2015) *New automated Rotterdam container terminal shows how far US lags*, *The Journal of Commerce*. Available at: https://www.joc.com/port-news/terminal-operators/apm-terminals/new-automated-rotterdam-container-terminal-shows-just-how-far-us-lags_20150502.html (Accessed: 23 November 2017).

Moorthy, R. L. *et al.* (2003) 'Cyclic Deadlock Prediction and Avoidance for Zone-controlled AGV System', *International Journal of Production Economics*, 83(3), pp. 309–324.

Di Natale, M. *et al.* (2010) 'Synthesis of Multitask Implementations of Simulink Models With Minimum Delays', *IEEE Transactions on Industrial Informatics*, 6(4), pp. 637–651.

Nearchou, A. C. (1998) 'Path Planning of a Mobile Robot Using Genetic Heuristics', *Robotica*, 16(5), pp. 575–588.

Nguyen, V. D. and Kim, K. H. (2009) 'A Dispatching Method for Automated Lifting Vehicles in Automated Port Container Terminals', *Computers & Industrial Engineering*, 56(3), pp. 1002–1020.

Nourani, Y. and Andresen, B. (1998) 'A comparison of simulated annealing cooling

strategies', *Journal of Physics A: Mathematical and General*, 31(41), pp. 8373–8385.

Otani, T. *et al.* (2009) 'Applying a path planner based on RRT to cooperative multirobot box-pushing', *Artificial Life and Robotics*, 13(2), pp. 418–422.

Pallottino, L., Feron, E. M. and Bicchi, A. (2002) 'Conflict Resolution Problems for Air Traffic Management Systems Solved with Mixed Integer Programming', *IEEE Transactions on Intelligent Transportation Systems*, 3(1), pp. 3–11.

Park, J. H., Kim, H. J. and Lee, C. (2009) 'Ubiquitous Software Controller to Prevent Deadlocks for Automated Guided Vehicle Systems in a Container Port Terminal Environment', *Journal of Intelligent Manufacturing*, 20(3), pp. 321–325.

Park, M. G. and Lee, M. C. (2003) 'A New Technique to Escape Local Minimum in Artificial Potential Field Based Path Planning', *KSME International Journal*, 17(12), pp. 1876–1885.

Paull, L. *et al.* (2013) 'Sensor-driven Online Coverage Planning for Autonomous Underwater Vehicles', *IEEE/ASME Transactions on Mechatronics*, 18(6), pp. 1827–1838.

Phan-Thi, M.-H., Ryu, K. and Kim, K. H. (2013) 'Comparing Cycle Times of Advanced Quay Cranes in Container Terminals', *Industrial Engineering & Management Systems*, 12(4).

Pharpatara, P., Herisse, B. and Bestaoui, Y. (2017) '3-D Trajectory Planning of Aerial Vehicles Using RRT∗', *IEEE Transactions on Control Systems Technology*, 25(3), pp. 1116–1123. doi: 10.1109/TCST.2016.2582144.

Qu, H., Xing, K. and Alexander, T. (2013) 'An Improved Genetic Algorithm with Co-evolutionary Strategy for Global Path Planning of Multiple Mobile Robots', *Neurocomputing*, 120, pp. 509–517.

Rashidi, H. and Tsang, E. P. K. (2011) 'A Complete and an Incomplete Algorithm for Automated Guided Vehicle Scheduling in Container Terminals', *Computers & Mathematics with Applications*, 61(3), pp. 630–641.

Richards, A. *et al.* (2002) 'Spacecraft Trajectory Planning with Avoidance Constraints Using

Mixed-integer Linear Programming', *Journal of Guidance, Control, and Dynamics*, 25(4), pp. 755–764.

Richards, A. and How, J. P. (2002) 'Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming', in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*. Anchorage, AK, USA: IEEE, pp. 1936–1941.

Saanen, Y. (2016) *AGV versus Lift AGV versus ALV: A Qualitative and Quantitative Comparison*. Delft, Netherlands. Available at: https://www.tba.nl/resources/press%2Bsection/publications/PTI70_AGV_vs_ALV (Saanen).pdf (Accessed: 6 June 2017).

Saanen, Y. A. and Valkengoed, M. V. (2005) 'Comparison of Three Automated Stacking Alternatives by Means of Simulation', in *Proceedings of the 2005 Winter Simulation Conference* . Orlando, FL, USA: IEEE, pp. 1567–1576.

Sahingoz, O. K. (2014) 'Generation of Bezier Curve-based Flyable Trajectories for Multi-UAV Systems with Parallel Genetic Algorithm', *Journal of Intelligent & Robotic Systems*, 74(1–2), pp. 499–511.

Savla, K., Frazzoli, E. and Bullo, F. (2008) 'Traveling Salesperson Problems for the Dubins Vehicle', *IEEE Transactions on Automatic Control*, 53(6), pp. 1378–1391.

Schuster, W. and Ochieng, W. (2011) 'Airport Surface Movement – Critical Analysis of Navigation System Performance Requirements', *The Journal of Navigation*, 64(2), pp. 281–294.

Seneviratne, L. D., Ko, W.-S. and Earles, S. W. E. (1997) 'Triangulation-based Path Planning for a Mobile Robot', *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 211(5), pp. 365–371.

Shanmugavel, M. *et al.* (2010) 'Co-operative Path Planning of Multiple UAVs Using Dubins Paths with Clothoid Arcs', *Control Engineering Practice*, 18(9), pp. 1084–1092.

Shiltagh, N. A. and Jalal, L. D. (2013) 'Path Planning of Intelligent Mobile Robot Using Modified Genetic Algorithm', *International Journal of Soft Computing and Engineering (IJSCE)*, 3(2), pp. 31–36.

Singh, N., Sarngadharan, P. V. and Pal, P. K. (2011) 'AGV Scheduling for Automated Material Distribution: A Case Study', *Journal of Intelligent Manufacturing*, 22(2), pp. 219–228.

Sivanandam, S. N. and Deepa, S. N. (2008) *Introduction to Genetic Algorithms*. Berlin, Germany: Springer-Verlag.

Skinner, B. *et al.* (2013) 'Optimisation for Job Scheduling at Automated Container Terminals Using Genetic Algorithm', *Computers & Industrial Engineering*, 64(1), pp. 511–523.

Škrjanc, I. and Klančar, G. (2010) 'Optimal Cooperative Collision Avoidance between Multiple Robots Based on Bernstein–Bézier Curves', *Robotics and Autonomous Systems*, 58(1), pp. 1–9.

Song, X., Cao, H. and Huang, J. (2013) 'Vehicle path planning in various driving situations based on the elastic band theory for highway collision avoidance', *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 227(12), pp. 1706–1722.

Stahlbock, R. and Voß, S. (2008) 'Operations Research at Container Terminals: A Literature Update', *OR Spectrum*, 30(1), pp. 1–52.

Tanchoco, J. M. A. and Sinriech, D. (1992) 'OSL—Optimal Single-loop Guide Paths for AGVs', *International Journal of Production Research*, 30(3), pp. 665–681.

Tsourdos, A., White, B. and Shanmugavel, M. (2011) *Cooperative Path Planning of Unmanned Aerial Vehicles*. Wiley. doi: 10.1002/9780470974636.

Tuncer, A. and Yildirim, M. (2012) 'Dynamic Path Planning of Mobile Robots with Improved Genetic Algorithm', *Computers & Electrical Engineering*, 38(6), pp. 1564–1572.

UNCTAD (2017) *Review of Maritime Transport 2017*. Geneva.

Wang, G. and Ge, S. S. (2014) 'General Fight Rule-based Trajectory Planning for Pairwise Collision Avoidance in a Known Environment', *International Journal of Control, Automation and Systems*, 12(4), pp. 813–822.

Wehbe, B., Bazzi, S. and Shammas, E. (2017) 'Novel three-dimensional optimal path planning method for vehicles with constrained pitch and yaw', *Robotica*, 35(11), pp. 2157–2176.

Wei, X. (2013) 'Robot Path Planning Based on Simulated Annealing and Artificial Neural Networks', *Research Journal of Applied Sciences, Engineering and Technology*, 6(1), pp. 149–155.

Williams, M. and Jones, D. I. (2001) 'A Rapid Method for Planning Paths in Three Dimensions for a Small Aerial Robot', *Robotica*, 19(2), pp. 125–135.

Wu, N. and Zeng, W. (2002) 'Deadlock Avoidance in an Automated Guidance Vehicle System Using a Coloured Petri Net Model', *International Journal of Production Research*, 40(1), pp. 223–238.

Xin, J., Negenborn, R. R. and Lodewijks, G. (2014) 'Trajectory Planning for AGVs in Automated Container Terminals Using Avoidance Constraints: A Case Study', *IFAC Proceedings Volumes*, 47(3), pp. 9828–9833.

Yang, D., Li, D. and Sun, H. (2013) '2D Dubins Path in Environments with Obstacle', *Mathematical Problems in Engineering*, 2013, pp. 1–6.

Yang, Y.-C. and Shen, K.-Y. (2013) 'Comparison of the Operating Performance of Automated and Traditional Container Terminals', *International Journal of Logistics: Research and Applications*, 16(2), pp. 158–173.

Yilmaz, N. K. *et al.* (2008) 'Path Planning of Autonomous Underwater Vehicles for Adaptive Sampling Using Mixed Integer Linear Programming', *IEEE Journal of Oceanic Engineering*, 33(4), pp. 522–537.

Yoo, J. *et al.* (2005) 'An Algorithm for Deadlock Avoidance in an AGV System', *The International Journal of Advanced Manufacturing Technology*, 26(5–6), pp. 659–668.

Zeng, J. and Hsu, W.-J. (2008) 'Conflict-free Container Routing in Mesh Yard Layouts', *Robotics and Autonomous Systems*, 56(5), pp. 451–460.

Zhang, C. *et al.* (2002) 'Dynamic crane deployment in container storage yards', *Transportation Research Part B: Methodological*, 36(6), pp. 537–555.

Zhang, C. *et al.* (2003) 'Storage space allocation in container terminals', *Transportation Research Part B: Methodological*, 37(10), pp. 883–903.

Zhang, P.-Y., Lü, T.-S. and Song, L.-B. (2004) 'Soccer Robot Path Planning Based on the Artificial Potential Field Approach with Simulated Annealing', *Robotica*, 22(5), pp. 563–566.

Zhang, Y., Gong, D. and Zhang, J. (2013) 'Robot Path Planning in Uncertain Environment Using Multi-objective Particle Swarm Optimization', *Neurocomputing*, 103, pp. 172–185.

Zheng, K. *et al.* (2013) 'Distributed Control of Multi-AGV System Based on Regional Control Model', *Production Engineering*, 7(4), pp. 433–441.

Zhou, L. and Jiang, J. (2012) 'An Approach to Safe Path Planning for Mobile Robot in the Dynamic Environment Based on Compact Maps', *Journal of Computers*, 7(2), pp. 405–410.

Zhu, Z. *et al.* (2015) 'Global Path Planning of Mobile Robots Using a Memetic Algorithm', *International Journal of Systems Science*, 46(11), pp. 1982–1993.

Appendix A: Simulation Settings

Category	Description	Value	Unit
AGVs	Width	3	m
	Length	15	m
	Distance between wheel axes	8.5	m
	Max linear speed (unloaded)	6	m/sec
	Max linear speed (loaded)	5	m/sec
	Max turning speed (unloaded)	3	m/sec
	Max turning speed (loaded)	2	m/sec
	Max acceleration	1	m/sec^2
	Min acceleration	-1	m/sec^2
	Max steering degree	30	°
Quay cranes	Service time for AGVs	20	sec
	Interval	80	m
	Y-coordinate	15	m
Yard cranes	Number of cranes	16	-
	Service time for AGVs	10	sec
	Interval	35	m
	Y-coordinate	150	m
Quay crane tasks	Min quay crane cycle time	70	sec
	Max quay crane cycle time	130	sec
	Min yard crane cycle time	50	sec
	Max yard crane cycle time	140	sec
AGV dispatching	Big-M	1,000,000	-
	α in the objective function	0.7	-
	β in the objective function	0.2	-
	γ in the objective function	0.1	-
	Initial temperature	10,000	-
	Ending temperature	1	-
	Cooling parameter	0.95	-
	Neighbour generation iterations	10	-
Trajectory planning	Max additional straight	50	m
	Additional straight probability	0.5	-
	Path reuse rate	0.3	-
	Detour trial number	5	-

	Movement coefficient	0.6	-
	Decreased path coefficient	0.4	-
	Collision test range	50	m
	Collision penalty	100,000	-
	AGV bidirectionality	True	-
	Initial temperature	10,000	-
	Ending temperature	1	-
	Cooling parameter	0.95	-
	Neighbours per iteration	10	-
Rolling horizon	Trajectory execution period	10	sec
	Trajectory planning period	20	sec
	Dispatching horizon	120	sec