Mathematical Modelling and Efficient Algorithms for Autonomous Straddle Carriers Planning at Automated Container Terminals

by

Shuai Yuan

A thesis submitted in fulfilment
of the requirements for the degree of
Doctor of Philosophy

University of Technology, Sydney
Faculty of Engineering

October 2013

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of a requirement for a degree except as fully acknowledged within the text.

I certify that the thesis has been written by me. Any help that I have received in my research work and preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

_____

(Shuai Yuan)

Sydney, October 2013

Abstract

In the past several decades, automation of handling equipment has been a worldwide trend in seaport container terminals. Increasing automation of yard handling vehicles not only reduces the cost of terminal operation, but also increases the efficiency of container transport. However, the primary loss of performance in the transhipment process is caused by the uncoordinated allocation and scheduling of quay cranes, yard vehicles and land-side operations. Hence, integrating transhipment processes is imperative for a fully automated container terminal. This thesis aims to study an integrated process and develop practically deployable strategies and algorithms, with the practical example of the Patrick AutoStrad container terminal, located in Brisbane, Australia.

The thesis first formulates two mathematical models: The *Comprehensive Model* is an analytical optimisation model which integrates the quay-side, yard and land-side operational sub-problems of the Patrick AutoStrad container terminal. Derived from the comprehensive model, the *Job Scheduling Model* is formulated to focus on the optimisation of job scheduling, as job scheduling plays a more important role than path planning, and resource utilisation and port operation are more dependent on job scheduling.

To solve the *Comprehensive Model*, a job grouping approach is proposed for solving the integrated problem, and experimental results show that the job grouping approach can effectively improve the time related performance of planning container transfers. Solving the *Job Scheduling Model* using a global optimisation approach is expected to provide higher productivity in automated container terminals. Hence, a modified genetic algorithm is proposed for solving the job scheduling problem derived from the integrated mathematical model of container transfers. Moreover, the live testing results show that the proposed algorithm can effectively reduce the overall time-related cost of container transfers at the automated container terminal.

Last but not least, a new crossover approach is proposed in order to further improve the solution quality based on the modified genetic algorithm, and it can also be directly

applied in solving the generic multiple travelling salesmen problem using the two-part chromosome genetic algorithm. The experimental results also show that the proposed crossover approach statistically outperforms the existing approaches when solving the job scheduling problem and the standard multiple travelling salesmen problem.

Acknowledgements

This thesis would never have been written without the support of the people who have enriched me through their wisdom, friendship and love.

I would like to express my deepest appreciation to my supervisors: Professor Dikai Liu and Associate Professor Shoudong Huang. I really appreciate their guidance, encouragement and constructive advice on modelling and algorithms. They have provided me with invaluable help throughout the course of my research.

I would like to thank my Patrick project mentors: Dr Bradley Skinner and Dr Haye Lau for all the hours of collaboration, insightful ideas and research inspiration.

I would like to thank all the other team members from UTS and PTS: Professor Gamini Dissanayake, Dr Binghuang Cai, Daniel Pagac, Timothy Pratley and Andrew Bott for their helpful insight along with their patience and passion for autonomous straddle carriers.

Last, but not the least, I am deeply indebted to my parents, Xijia and Shuhua, for the years of education, encouragement and care; my beloved wife Momo, for her unwavering support; and my dearest son Shengbo for imbuing my life with wonderful expectations.

To the people who helped

List of Abbreviations

ASC:	Autonomous Straddle Carrier
QC:	Quay Crane
TK:	Truck
TC:	Transfer Crane
TIA:	Truck Import Area
B2Y:	Buffer-to-Yard
Y2B:	Yard-to-Buffer
T2Y:	Truck-to-Yard
Y2T:	Yard-to-Truck
Y2Y:	Yard-to-Yard
VRP:	Vehicle Routing Problem
CVRP:	Capacitated Vehicle Routing Problem
PDP:	Pickup and Delivery Problem
QCSP:	Quay Crane Scheduling Problem
QCAP:	Quay Crane Allocation Problem
BAP:	Berth Allocation Problem

AGV:	Automated Guided Vehicle
ALV:	Automated Lifting Vehicle
ASCSP:	Autonomous Straddle Carrier Scheduling Problem
MTSP:	Multiple Travel Salesmen Problem
GA:	Genetic Algorithm
TCX:	Two-part Chromosome Crossover
ORX:	Ordered Crossover
PMX:	Partially Matched Crossover
CYX:	Cycle Crossover
ORX+A:	Combination of Ordered Crossover and Asexual Crossover
PMX+A:	Combination of Partially Matched Crossover and Asexual Crossover
CYX+A:	Combination of Cycle Crossover and Asexual Crossover

Nomenclature

Indices:

v	index of the ASC.
q	index of the QC.
g	index of the TK.
j	index of the job.
n	index of the positional node which indicates a position on the map.
l	index of the link between positional nodes on the map.
s	index of the schedule. Each schedule contains a list of jobs which need to be finished in order, and each schedule can be assigned to any ASC, but one schedule cannot be shared by multiple ASCs. It is not a complete solution, but it can be a possible sub-solution for some jobs.

Sets:

V	set of all ASCs, $\|V\|$ is the total number of ASCs in the fleet.
Q	set of all QCs, $\|Q\|$ is the total number of QCs.
G	set of all TKs, $\|G\|$ is the total number of TKs.
J	set of all jobs, $\|J\|$ is the total number of jobs.
J^H	set of jobs with high priority for scheduling and these jobs usually are time critical or considered as urgent tasks and $J^H \subset J$.
S	set of all schedules.
J^Y	set of yard-to-yard jobs.
δ_v	planned trajectory of ASC (v) including the travelling nodes and associated times. Please refer to Section 3.4.1 for more detail.
N	set of all positional nodes, $\|N\|$ is the total number of nodes in the map.
L	set of all links, $\|L\|$ is the total number of links on the map.

Vectors:

J_s	vector of jobs in schedule s. That is, a list of jobs that will be done by an ASC in the defined order.
J_q^D	vector of jobs associated with discharging QC q.
J_q^U	vector of jobs associated with uploading QC q.
J_g^E	vector of jobs associated with exporting TK g.
J_g^I	vector of jobs associated with importing TK g.
J_v^G	vector of potential jobs grouped by (v).
$N^{lock}(n)$	vector of locked nodes associated with the node (n) of an ASC to prevent all vehicle collisions.

Parameters:

m_q	the total number of discharging containers for QC q.
n_q	the total number of uploading containers for QC q.
e_g	the total number of containers for exporting TK g.
r_g	the total number of containers for importing TK g.
p_v^I	the initial position of ASC v.
α_{ab}	parameter for defining pick-up job sequence. $\alpha_{ab} = \{0,1\} : \forall(a,b)$. Let $\alpha_{ab} = 1$ if and only if job a needs to be picked up before job b is picked up, else $\alpha_{ab} = 0$.
β_{ab}	parameter for defining set-down job sequence. $\beta_{ab} = \{0,1\} : \forall(a,b)$. Let $\beta_{ab} = 1$ if and only if job a needs to be set-down before job b is set-down, else $\beta_{ab} = 0$.
u_j	the pick-up position of job j. The position is generated by the yard management system.
d_j	the set-down position of job j. The position is generated by the yard management system.

y_{js} a binary parameter. $y_{js} = 1$ if and only if the candidate schedule *s* has job *j*, otherwise $y_{js} = 0$.

t_0 plan starting time.

Δt_{pickup} time required for an ASC to pick up a container.

$\Delta t_{setdown}$ time required for an ASC to set down a container.

Δt_{acc} time associated with acceleration of an ASC moving from zero velocity to maximum (constant) velocity.

Δt_{dec} time associated with deceleration of an ASC from its maximum (constant) velocity to zero velocity.

Δt^{QC} the turnaround time of each QC unloading/uploading a single container.

Δt^{TK} the turnaround time of an ASC performing TK importing/exporting for a single container within the TK area.

t_q^D the starting time of QC *q* operations (discharging/unloading) which is predefined a priori as part of the QC schedule. For each discharging QC we assume that no container from QC *q* can be picked up by an ASC at the buffer before the time ($t_q^D + \Delta t^{QC}$).

t_q^U the starting time of QC *q* operations (uploading) which is predefined a priori as part of the QC schedule. For each uploading QC we assume that no container can be set-down at buffer by an ASC before the time ($t_q^U + \Delta t^{QC}$).

t_g^E the starting time of TK *g* operations (exporting) which is predefined a priori as part of the TK schedule. For each exporting TK we assume that no container from TK *g* can be picked up by an ASC at the TK gate before the time ($t_g^E + \Delta t^{TK}$).

t_g^I the starting time of TK *g* operations (importing) which is predefined a priori as part of the TK schedule. For each importing TK we assume that no container from TK *g* can be set-down by an ASC at the TK gate before the time ($t_g^I + \Delta t^{TK}$).

w_{oz} the minimum theoretical travel time between node o and node z. A look-up table built based on Dijkstra's algorithm is used for all positional nodes.

σ_j^R the predefined parameter which specifies the container door alignment at the setdown node (d_j) for job (j). Let σ_j^R=1 if the container door is aligned, otherwise σ_j^R=0. Here, $\sigma_j^R \in \{0,1\}$, depending on the actual job type.

σ_j^S the predefined parameter which specifies the container door's initial alignment at the pickup node (u_j) for job (j).Let σ_j^S=1 if the container door is aligned, otherwise σ_j^S=0. Here, $\sigma_j^S \in \{0,1\}$, depending on the actual job type.

f_{ij} a flip flag of each physical link which relates the container alignment during traversal of the link. Let $f_{ij}=1$ if container alignment is changed when an ASC carrying a container from node ($i \in N$) to node ($j \in N$), otherwise $f_{ij}=0$. The flip flag of each link is encoded within the map.

ψ_{abc} Patrick yard predefined traversal information between ($a \to b \to c$). Let $\psi_{abc}=1$ if an ASC needs to decelerate and adjust direction and then accelerate for traversing from node a to node c via node b, otherwise, $\psi_{abc}=0$. This information is encoded within the map.

ε_j the minimum theoretical processing time based on Dijkstra's algorithm (from pickup to setdown) for job (j).

Decision variables:

X_{vs} binary decision variable. $X_{vs}=1$ if and only if the candidate schedule s is selected for ASC v, otherwise $X_{vs}=0$.

T_j^S the planned start time of job j.

T_j^F the planned finish time of job j.

T_q^{DS} the planned time to pick-up the final container for discharging QC q.

T_q^{UF} the planned time to set-down the final container for uploading QC q.

T_g^{ES} the planned time to pick-up the final container for exporting TK g.

T_g^{IF} the planned time to set-down the final container for importing TK g.

Dependent variables

$P_v(t)$	position of ASC (v) at time ($t \geq t_0$), according to the trajectory planning.
σ_j^F	dependent variable which is the final box alignment associated with the trajectory planning of job (j), and $\sigma_j^F \in \{0,1\}$.

Table of Contents

CERTIFICATE OF AUTHORSHIP/ORIGINALITY 2
Abstract 3
Acknowledgements 5
List of Abbreviations 6
Nomenclature 8
Table of Contents 13
List of Figures 16
List of Tables 18
1. Introduction 20
1.1 Background 21
1.2 Motivation 22
1.3 Aims 24
1.4 Contributions 25
1.5 Publications 25
1.6 Thesis outline 27
2. Literature Review 30
2.1 Container Handling in Transhipment 30
2.1.1 Quay-side Container Handling Schemes 30
2.1.2 Yard Container Handling Schemes 32
2.1.3 Land-Side Container Handling Schemes 34
2.1.4 Integrated Container Handling Schemes 35
2.1.5 Scheduling of Automated Vehicles 37
2.2 Relevant Problem Models for Scheduling 38
2.2.1 The Multiple Travelling Salesman Problem 38
2.2.2 Pick-up and Delivery Problem 40
2.2.3 Capacitated Vehicle Routing Problem 42
2.2.4 Discussion 43
2.3 Optimisation with Genetic Algorithms 44
2.4 Summary 47
3. Mathematical Modelling for Automated Container Transfers 49
3.1 Introduction 49
3.2 Overview of the Patrick AutoStrad Container Terminal 49
3.3 Job Definition 51

3.3.1 Quayside Transfers 51
3.3.2 Yard Transfers 53
3.3.3 Land-side Transfers 54
3.4 Model 1: Comprehensive Model 55
3.4.1 Description of Container Alignment and ASC Trajectory 55
3.4.2 Objective Function 57
3.4.3 Constraints 59
3.5 Model 2: Job Scheduling Model 65
3.6 Matlab Seaport Simulation 67
3.6.1 Matlab Seaport Simulator 67
3.6.2 Replanning in the Seaport Simulator 68
3.6.3 Model Validation within the Seaport Simulator 69
3.7 Summary 70
4. Job Grouping Approach for Planning Container Transfers 72
4.1 Motivation 72
4.2 Job Grouping Approach 73
4.2.1 Guiding Function of Job Grouping 75
4.2.2 Grouping Algorithm 77
4.3 Sequential Job Planning Approach 79
4.4 Experimental Results 80
4.5 Summary 85
5. Modified Genetic Algorithm for Scheduling Optimisation 86
5.1 The Modified GA Approach 86
5.1.1 Two-part Chromosome Representation 87
5.1.2 Chromosome Validation and Repair Operation 88
5.1.3 Fitness Calculation Strategy 89
5.1.4 Selection and Crossover 93
5.1.5 Mutation 94
5.1.6 Replacement 94
5.2 Experimental Results 94
5.2.1 Simulation Testings 95
5.2.2 Live Testing Results at Patrick AutoStrad Terminal 99
5.3 Summary 100
6. A New GA Crossover Approach for Further Improvement 102
6.1 Overview and Analysis of the Existing Strategy 102

6.1.1 The Two-part Chromosome Encoding Technique 102
6.1.2 Existing Crossover Method for Two-part Chromosomes 103
6.1.3 Limitations of the Existing Method 104
6.2 A New Crossover Approach 106
6.3 Experiments for Job Scheduling Optimisation 110
6.4 Computational Testing Methodology for the MTSP 113
6.5 Computational Results for the MTSP 115
6.5.1 Experiments for Minimising Total Travel Distance 116
6.5.2 Experiments for Minimising Longest Tour 118
6.5.3 The Robustness of the TCX 120
6.5.4 Optimality 122
6.5.5 Examination of TCX Operator with an Additional Dataset 123
6.6 Summary 124
7. Conclusions and Future Research 126
7.1 Summary 126
7.1.1 Two Optimisation Models of Container Transfers by ASCs 126
7.1.2 A Job Grouping Approach for the Comprehensive Model 126
7.1.3 A Modified Genetic Algorithm for Optimising Job Scheduling 127
7.1.4 A Novel Crossover Method 127
7.2 Future Work 128
Appendix: Methodology of *t*-test 130
Bibliography 132

List of Figures

Figure 1-1: Satellite view of the Patrick AutoStrad container terminal within the Port of Brisbane at Fisherman Islands Australia [Google Earth]. 20

Figure 1-2: ASC transporting a 40-foot container from the quay side as part of a buffer-to-yard task. 21

Figure 1-3: The structure of the thesis 27

Figure 2-1: Typical QCs employed at the container Terminal in Brisbane, Australia. . 31

Figure 2-2: ASC servicing a TK in the TIA at the Patrick AutoStrad container terminal located in Brisbane, Australia. 34

Figure 3-1: Schematic diagram of the static seaport environment showing berth, QCs, bays, special nodes (QC and TK gates), TIA, and nodes in the yard. 50

Figure 3-2: Quay Side Operations. A trajectory for the QC jib can be calculated from the associated nodes and links between the ship and wharf. (TLR refers to twist lock release.)..... 52

Figure 3-3: TK Area Operations showing associated nodes and links between the yard and TK area. TKs can import or export any combination of 20-ft and 40-ft containers (up to a total of 4TEU (Twenty-foot Equivalent Unit) in each direction) into the seaport, as shown by the different combinations. 54

Figure 3-4: Container orientations (a,b) and flip movements (c). 56

Figure 3-5: Typical event timings and timeline for an ASC performing a job. 57

Figure 3-6: A planned path from pickup node to setdown node consiting of 56 nodes. A 3-point turn occurs at the 16th node (nodes are not uniformly spaced in the map)..... 64

Figure 3-7: Example of a motion profile, which is applied to a planning path for an ASC (nodes are not uniformly spaced in the map). 64

Figure 3-8: Classification of a 3-point turn (case 1) and regular turning motion (case 2) for motion planning. 65

Figure 3-9: Flow diagramming for sequential job allocation and job injection approach based on a greedy nearest-vehicle-first heuristic. 68

Figure 3-10: Flow diagramming which illustrates the replanning processes within the seaport simulator. 69

Figure 3-11: Model Validation Method – Hierarchical assertion checking within the Matlab Seaport Simulator ensures model constraints are not violated. 70

Figure 4-1: An example of job grouping. 74

Figure 4-2: Plot of Eq(4.2) for job starting times (sec). 76

Figure 4-3: Plot of Eq(4.2) for job finishing times (sec). 76

Figure 4-4: Flowchart of job grouping method 77

Figure 4-5: Flowchart of sequential job allocation method 80

Figure 4-6: Performance comparison for the high QC load scenario. 82

Figure 4-7: Performance comparison for the medium QC load scenario. 82

Figure 4-8: Performance comparison for the low QC load scenario. 83

Figure 5-1: Example of two-part chromosome representation for a 9-job schedule with 3 ASCs 88

Figure 5-2: An example of calculating ASC and QC waiting times and updating related timings 91

Figure 5-3: Ordered crossover (ORX) method for first part chromosome 93

Figure 5-4: Single point asexual crossover for second part chromosome 93

Figure 5-5: Comparison on scheduling for 24 mixed jobs with 8 ASCs 97

Figure 5-6: Comparison on scheduling for 80 mixed jobs with 20 ASCs 98

Figure 6-1: Example of using the ORX+A crossover method (Carter and Ragsdale, 2006) for two-part chromosomes 103

Figure 6-2: The modified GA flowchart for solving the job scheduling problem 110

Figure 6-3: Comparison on scheduling for 24 mixed jobs with 8 ASCs 111

Figure 6-4: Comparison on scheduling for 80 mixed jobs with 20 ASCs 112

List of Tables

Table 4-1: Parametric Configuration 81

Table 4-2: Number of Jobs for Three Different Scenarios..... 81

Table 4-3: Comparison of the Job Grouping Algorithm VS Sequential Job Allocation Algorithm 83

Table 4-4: Performance of the Job Grouping Algorithm Relative to the Sequential Job Allocation Algorithm 84

Table 5-1: Parametric configuration for scheduling 24 mixed jobs 95

Table 5-2: Parametric configurations for scheduling 80 mixed jobs..... 97

Table 5-3: Comparison of GA vs Sequential Job Scheduling Algorithm for Total Cost 98

Table 5-4: GA settings 99

Table 5-5: Various testing results from Patrick AutoStrad scheduling system..... 100

Table 6-1: The number of fundamental combinations in the second part of the chromosome for each job scheduling problem (number of jobs_number of ASCs) showing the exponentially increasing number of combinations with increasing *m* 106

Table 6-2: Limited search space of the existing method for the second part of the chromosome..... 106

Table 6-3: Comparison of ORX GA vs TCX GA for Total Cost..... 113

Table 6-4: Computational test conditions 114

Table 6-5: Parametric configuration for GA 114

Table 6-6: Experimental results for minimising total travel distance. 116

Table 6-7: *t*-test results of TCX versus (ORX, CYX and PMX) crossover approaches for minimising total travel distance. 117

Table 6-8: Comparison of the mean results between two approaches using GA with seeding enabled for minimising total travel distance. 117

Table 6-9: Experimental results for minimising longest tour. 118

Table 6-10: *t*-test results of TCX versus (ORX, CYX and PMX) crossover approaches for minimising longest tour. 119

Table 6-11: Comparison of crossovers with GA seeding for minimising longest tour. 119

Table 6-12: TCX Robustness to varying GA parameters for minimising total travel distance 120

Table 6-13: TCX Robustness to varying GA parameters for minimising longest tour 122

Table 6-14: Optimality Gap for small-sized symmetric MTSP problems for minimising total travel distance 123

Table 6-15: Optimality Gap for small-sized symmetric MTSP problems for minimising longest tour 123

Table 6-16: Comparison of crossover methods for the sgb128 MTSP test problem for minimising the total travel distance 124

Table 6-17: Comparison of crossover methods for the sgb128 MTSP test problem for minimising longest tour 124

1. Introduction

With the recent advances in autonomous vehicle development, autonomous materials handling in complex and dynamic environments has become a reality, as evidenced by the opening of the world's first fully automated container terminal by Patrick Stevedores Holdings in 2005. The Patrick AutoStrad container terminal is shown in Figure 1-1.

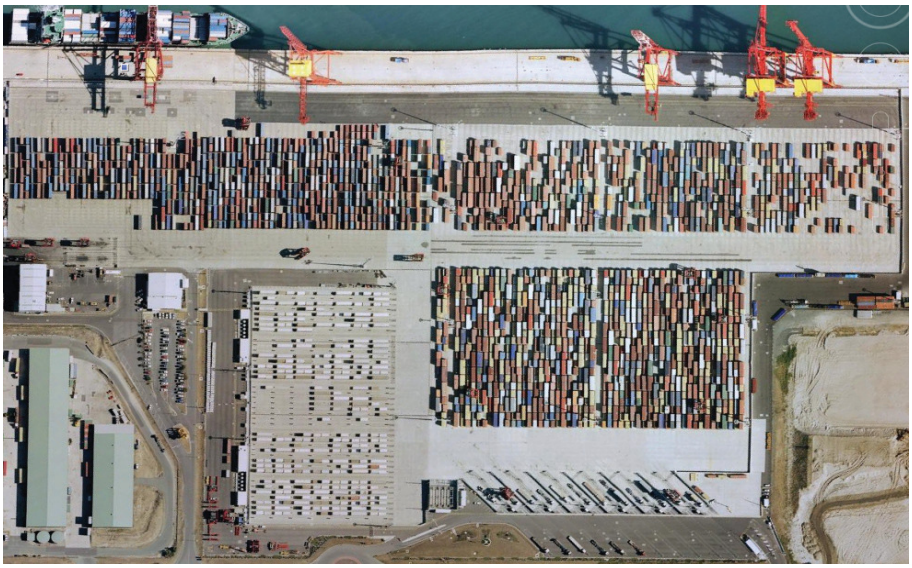


Figure 1-1: Satellite view of the Patrick AutoStrad container terminal within the Port of Brisbane at Fisherman Islands Australia [Google Earth].

Unlike many seaport terminals, which utilise yard vehicles that are either manned or fixed upon a circuit of tracks (Vis and Harika, 2004), the Patrick AutoStrad container terminal environment has been designed for the use of a fleet of fully autonomous straddle carriers (ASCs) (Figure 1-2). Each ASC with dimensions of 10m × 5m × 13m and costing over one million dollars, is deployed to transport containers between dynamically-specified work points within the terminal. However, the efficient strategies and algorithms for planning container transfers at automated container terminals are necessarily important to improve the performance of the port operation. This is crucial to guarantee that the terminal system can react in the most cost-effective way to meet the continuous growth of container transfers. This thesis focuses on problem modelling and efficient algorithms for planning container transfers using the practical example of the Patrick AutoStrad container terminal.

Figure 1-2: ASC transporting a 40-foot container from the quay side as part of a buffer-to-yard task.

This chapter introduces the research work presented in the thesis. It commences with a background of the container transfers in an automated container terminal and details key research issues. The remaining sections of this chapter describe the objectives of the thesis and its main contributions followed by an outline of the thesis.

1.1 Background

At seaport container terminals, both the capacity and frequency of arriving container carrying ships have increased steadily during the past several decades (Steenken et al., 2004, Günther and Kim, 2006). To reduce costs to terminal operators the berthing time of these large ships must be as small as possible. This requires efficient use of yard vehicles to load, unload and transfer containers during the transportation process.

In recent years, several seaport terminals have been planning or actively working towards automating the terminal to some extent (Kim et al., 2004b, Liu et al., 2004). Increasing automation of yard vehicles not only reduces the labouring cost of terminal operators, but also has the potential to increase the efficiency of the container transport. However, as compared with the human operated yard vehicles, there is an ongoing requirement to ensure a high degree of coordination and efficiency for all material handling equipment participating in the transportation process.

Current seaport container terminals around the world employ a multitude of container handling equipment, which can be categorised into quay-side, yard and land-side resources. Although quay cranes (QCs) are commonly used in seaports for uploading/unloading containers to/from ships on the quay-side, different seaports use different yard vehicles for transfer containers within the ports. In Patrick AutoStrad container terminal, the more than twenty ASCs are issued high level commands from a central computer system. They can pick up/set down containers autonomously without a human operator, both in the yard and on/off the truck (TK). The navigation system allows the ASCs to travel along any planned trajectory while performing collision detection and emergency stops. These ASCs are the only vehicles to transfer containers amongst the QC area, yard area, and the TK area. Compared to current seaports that use human operated ASCs for container transporting, this terminal is unique.

At the Patrick AutoStrad container terminal, like many other seaport terminals around the world, one of the main tasks of a fleet of ASCs is to service the QCs such that the maximal QC turn-around rate can be achieved (Nelmes, 2005, Yuan et al., 2009, Liu and Kulatunga, 2007). This will reduce the ship berthing time. Moreover, the ASCs also need to service the TKs, which are used to transfer containers between customers and the seaport terminal, such that the TK waiting time is minimised. Finally, the seaport operator would like ASCs to perform less urgent yard-to-yard container transportation jobs as part of their yard management strategy.

Because ASCs do not require a human driver, the planner not only needs to allocate jobs and assign paths to the vehicle, but also needs to plan the paths of each vehicle such that collision will not occur. In addition, the planned path should aim at reducing the travelling and waiting time of ASCs performing the transportation process, while finding a feasible allocation and schedule to complete all necessary jobs.

1.2 **Motivation**

Within any seaport environment, an effective schedule for the transhipment of containers requires efficient allocation and scheduling of quay-side, yard and land-side resources. Although the physical operation of these resources is typically decoupled

from each other, the overall efficiency of the integrated transhipment process must consider their interdependent processes, particularly if there is a high level of automation involved. At the Patrick AutoStrad container terminal, uncoordinated operations can introduce significant waiting times for QC, ASC idle time, buffer contention and extensive waiting time at the TK import. Recently, several researchers have suggested that integrating decisions at different levels may improve the efficiency of schedules (Meersmans and Wagelmans, 2001, Kozan and Preston, 2007, Kim et al., 2004a).

The problem of scheduling and path planning of ASCs contains subtle but significant differences to the Vehicle Routing Problem (VRP) and Pickup and Delivery Problem (PDP) (Savelsbergh and Sol, 1995) widely discussed in the operations research literature. In the VRP and PDP, the path networks (or the environments in general) are usually of metropolitan scale, and a vehicle is considered as a moving point. Vehicle kinematics and dynamics, for example acceleration times and breaking distances, are insignificant and are therefore ignored. Multiple vehicles can occupy the same road at the same time and thus vehicle collision is not a necessary concern. On the other hand, ASCs operate in port environments where a large number of vehicles are concentrated into a confined area. The transportation paths are organised such that one section of this network is restricted to one vehicle at a given time and thus the vehicles are not able to pass or overtake each other. Thus collision avoidance, as well as traffic congestion and the dynamic characteristics of the vehicles are very important aspects that need careful consideration during planning and scheduling for multiple ASCs. As such, a wide range of algorithms have been proposed in the extensive literature related to the VRP (Choi and Tcha, 2007, Pisinger and Ropke, 2007, Mester et al., 2007) and PDP (Fu, 2002, Bent and Hentenryck, 2006, Nanry and Barnes, 2000), and general optimisation software packages that are commercially available (IBM) cannot be directly used to the model and solve the scheduling and routing problem in the Patrick container terminal.

Generally, there are three major features which make the problem unique and challenging, compared to those well-studied operational issues (Choi and Tcha, 2007, Pisinger and Ropke, 2007, Mester et al., 2007, Fu, 2002, Bent and Hentenryck, 2006, Nanry and Barnes, 2000). First of all, the Patrick AutoStrad terminal is fairly confined and complex. There are more than twenty ASCs in a 0.2 square kilometre area occupied

mostly by storage space of multi-level stacked containers, which greatly restricts the number of possible paths all ASCs can travel at any one time. Secondly, ASCs are free-ranging within the container terminal and must take into account collision avoidance between each other when travelling within the yard. Thirdly, coordination of resources is not only required for ASCs but also needed for QC and TK. Thus, an effective coordination strategy should take into account other port resources which may affect the overall productivity of the terminal. Scheduling container transfers must consider time-critical jobs, such as QC and TK related jobs that are required to be conducted within the pre-defined time. Otherwise this could result in expensive waiting costs of QC or TK.

Therefore, it is essential to develop an integrated problem model and efficient algorithms for the Patrick AutoStrad terminal.

1.3 **Aims**

To address the issues of ineffective or inefficient planning of resources in the transshipment process, it is necessary to investigate and develop an analytical model which integrates quay-side, yard and land-side operations. In addition, strategies and algorithms are required so as to enhance the operational performance. These initiatives will, in turn, provide the tools to intelligently schedule jobs to ASCs by taking into account the current state of ASCs, the environment, traffic conditions and the jobs' requirements. They will also serve to maximise productivity.

Therefore, the objectives of this research are to mathematically model a complicated planning problem and develop efficient algorithms for improving the operational performance of the automated container terminal. Specifically, the research work will:

- Develop an analytical optimisation model for planning container transfers at an automated container terminal.
- Investigate efficient algorithms for optimising the planning operations.
- Test and evaluate the developed mathematical models and algorithms.

1.4 Contributions

The major contributions of this thesis are:

- It formulates two mathematical models of container transfers. With the features of using ASCs, our formulated models are different from the existing terminal models. The first one is an analytical optimisation model of container transfers, which integrates the scheduling of ASCs and the collision free path planning of ASCs. The model also considers many of the practical constraints present in a dynamic container terminal. The second formulated model focuses on job scheduling optimisation, which is derived from the integrated model.

- It proposes a job grouping approach for planning container transfers with the integrated model. The proposed job grouping approach is an enhancement based on the existing sequential approach, and it can effectively improve the efficiency of the schedule for yard jobs, while reducing the waiting time of QCs, TKs and ASCs.

- It proposes a modified genetic algorithm (GA) for optimising job scheduling with the job scheduling model. A practical contribution is that the proposed approach has been fully implemented on a trial basis in the live scheduling system at the Patrick container terminal, and it effectively improves the performance of the seaport container terminal.

- It comes up with a novel crossover method based on the modified GA for improving solution quality. Compared to the existing crossover approaches for the two-part chromosome GA, the proposed crossover method can effectively enhance the search ability and performance. Also, the proposed method can be easily used in solving the classic MTSP with performance improvements as well.

1.5 Publications

Parts of the research work have been published in the following papers:

Journal articles:

1. **Shuai Yuan,** Bradley Skinner, Shoudong Huang and Dikai Liu. "A new crossover approach for solving the multiple travel salesmen problem using genetic algorithms", *European Journal of Operational Research*, Vol. 228(1), 2013, pp. 72-82.

2. Bradley Skinner, **Shuai Yuan,** Shoudong Huang, Dikai Liu, Binghuang Cai, Gamini Dissanayake, Haye Lau, Andrew Bott and Daniel Pagac. "Optimisation for job scheduling at automated container terminals using genetic algorithm", *Computers & Industrial Engineering*, Vol. 64(1), 2013, pp. 511-523.

3. **Shuai Yuan,** Bradley Skinner, Shoudong Huang, Dikai Liu, Gamini Dissanayake, Haye Lau and Daniel Pagac. "A job grouping approach for planning container transfers at automated seaport container terminals", *Advanced Engineering Informatics*, Vol. 25(3), 2011, pp. 413-426.

Peer reviewed conference papers:

4. **Shuai Yuan**, Bradley Skinner, Shoudong Huang, Dikai Liu, Gamini Dissanayake, Haye Lau, Daniel Pagac and Timothy Pratley. "Mathematical Modelling of Container Transfers for a Fleet of Autonomous Straddle Carriers", *Proceedings of 2010 IEEE International Conference on Robotics and Automation (ICRA 2010)*, Anchorage, Alaska, USA, May 3-8, 2010, pp. 1261-1266.

5. **Shuai Yuan,** Haye Lau, Dikai Liu, Shoudong Huang, Gamini Dissanayake, Daniel Pagac and Timothy Pratley. "Simultaneous Dynamic Scheduling and Collision-Free Path Planning for Multiple Autonomous Vehicles", *Proceedings of 2009 IEEE International Conference on Information and Automation (ICIA 2009)*, Zhuhai, China, June 22 - 25, 2009, pp.522-527.

1.6 Thesis outline

This thesis consists of seven chapters. The thesis is organized as follows, which is shown in Figure 1-3.

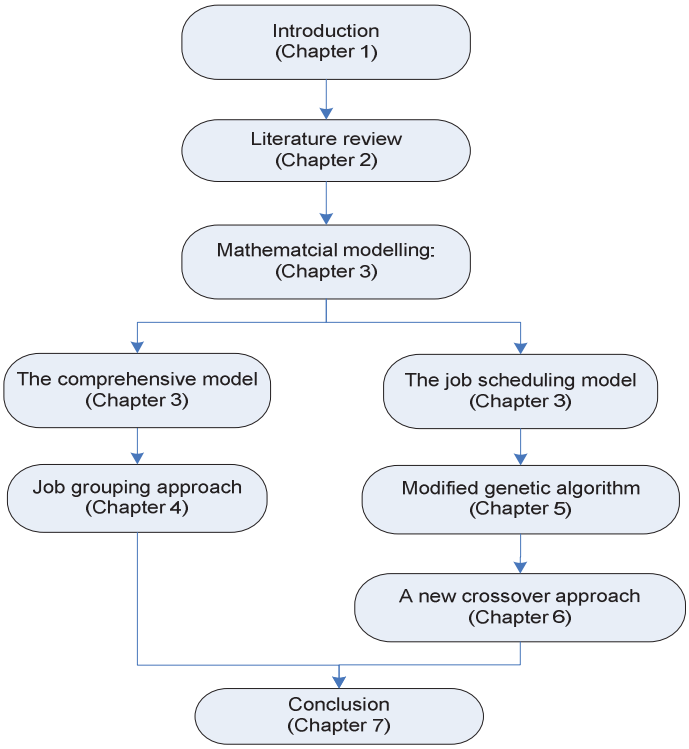


Figure 1-3: The structure of the thesis

The thesis is structured so that the first two chapters outline the research and provide background for the research. Chapter 3 gives two mathematical models. Chapter 4 presents a job grouping approach for solving the integrated problem, which takes into account path planning and job scheduling. Chapter 5 focuses on solving the job scheduling problem, and presents a modified genetic algorithm. Chapter 6 presents a new crossover approach and the improvements on scheduling performance. Conclusions and future directions are presented in Chapter 7. Detailed outlines of each chapter appear below:

Chapter 2 provides a literature review of previous works on container transhipment, classic problem models related to scheduling, and solution methods. Some typical approaches in different container handling processes are presented, and the scheduling for

automated vehicles and solution techniques are introduced, and also three classic problem models which are similar to the scheduling problem are reviewed.

Chapter 3 provides two mathematical models of automated container transfers in a complex environment. The comprehensive model not only covers the essential consideration of collision avoidance in relation to a fleet of large vehicles in a confined area, the model, but also deals with many other difficult practical constraints, such as multiple levels of container stacking and sequencing, variable container orientations, and vehicular dynamics that require finite acceleration and deceleration times. The job scheduling model is derived from the comprehensive model with the focus on optimisation of job scheduling.

Chapter 4 proposes a job grouping approach for improving the existing sequential planning. It is combined with a guiding function that encourages early starting and finishing for schedule robustness. The performance of the existing sequential planning method and the proposed job grouping approach are evaluated and compared statistically using a pooled *t*-test for 30 randomly generated yard configurations. The experimental results show that the job grouping approach can effectively improve the overall performance.

Chapter 5 introduces a GA-based optimisation approach to solve the job scheduling problem. This chapter focuses on scheduling for container transfers and encodes the problem using a two-part chromosome approach which is then solved using a modified GA. The modified GA-based approach has been fully implemented on a trial basis in the live scheduling system at Patrick container terminal and it effectively improves the performance of the seaport container terminal.

Chapter 6 proposes a new crossover approach for further improvement of the solution quality. It adopts the two-part chromosome representation technique which has been proven to minimise the size of the problem search space. The existing crossover methods for the two-part chromosome representation have two limitations, but they have been overcome by the proposed approach. Also, the proposed method can be directly used to solve the standard MTSP. The experimental results show that TCX can improve the

solution quality of the GA compared to the existing crossover approaches for both the scheduling problem and the MTSP.

Chapter 7 summarises the research work presented in this thesis. Conclusions are then drawn about the research and avenues for future work are proposed.

2. Literature Review

This chapter provides a literature review of previous works on container transhipment and classic problem models related to scheduling, and reviews solution methods. This chapter is organised as follows: Section 2.1 reviews typical approaches used in different handling processes; Section 2.2 reviews three classic problem models which are related to the scheduling problem. Section 2.3 introduces some aspects of GAs for the optimisation problems considered in this thesis, and Section 2.4 summarises the chapter.

2.1 Container Handling in Transhipment

Current seaport container terminals around the world employ a multitude of container handling equipment, which can be categorised into quay-side, yard and land-side resources. There are several notable surveys which provide comprehensive overviews of the operational research applications and optimisation models related to the transhipment of containers within seaport container terminals (Steenken et al., 2005, Vis and Koster, 2006, Vis and Harika, 2005, Bierwirth and Meisel, 2009, Crainic and Kim, 2006). This section presents container handling schemes according to quay-side, yard, land-side and the integrated situation.

2.1.1 Quay-side Container Handling Schemes

Quay-side container handling is performed using specialised QCs, which are mounted on tracks located orthogonal to the quay. Crossover of QCs during motion is not possible. Typically, QCs are able to move along the quay into locations called bays, where they are parked prior to performing container movement tasks. Dividing the workload of a ship into non-overlapping bay areas, provides each bay exclusive access to a single QC and associated quay buffers. This avoids interference between individual cranes, but increases the need for providing a sufficient workload for each QC. Identical QCs are normally employed to load or discharge containers between berthed ships and predefined buffer areas located on the quay as shown in Figure 2-1.

Figure 2-1: Typical QCs employed at the container Terminal in Brisbane, Australia.

Containers are transported one or two (in case of a twin lift of 20-foot containers) at a time, usually in a fixed sequence and processed without pre-emption. Such precedence constraints arise due to a vessel's predefined *stowage plan* provided by the shipping operators. According to the stowage plan, there is usually a loading list for each assigned QC. The load sequencing problem has been studied in detail (Gambardella et al., 2001). An unloading plan, which indicates which container should be unloaded and in which area it is situated in the ship, is given before the arrival of the ship. Within a defined area the QC driver can freely determine the order in which the containers are unloaded. The unload sequencing problem has also been examined in detail (Gambardella et al., 2001, Kim et al., 2004a).

The Quay Crane Scheduling Problem (QCSP) considers the starting and ending time of each job in a set of jobs assigned to a set of QCs servicing a vessel. Minimising the berthing or handling time of a ship is achieved by minimising the makespan of the QC schedule. Computing the minimum makespan for the QCSP with parallel identical machines and constraints is considered to be *NP*-hard for two or more QCs. Further, no job pre-emption and non-uniform processing times are employed (Bierwirth and Meisel, 2009).

One of the first discussions on the QCSP was provided in the late 20th century (Daganzo, 1989). The simple static QCSP was formulated as a mixed integer program

that was solved exactly for a small number of ships and QCs. The author of the study, Daganzo, was able to minimise crane idle time and increase berthing throughput, while minimising the berthing time of vessels. In another later study (Kim and Park, 2004), the authors proposed an analytical formulation of the QCSP. Using a meta-heuristic search algorithm called GRASP, near optimal solutions were found. In addition, a lower bound on the optimal performance was proposed using the branch and bound method. However, both these studies assume complete availability of yard resources when transporting containers from the yard to the QC loading area.

In order to reduce complexity of the integrated problem, this research does not consider the berth allocation problem (BAP) or the quay crane allocation problem (QCAP). These sub-problems have been suitably solved at the Patrick AutoStrad container terminal. As such, the integrated approach presented in this study incorporates the QCSP sub-problem and not the BAP or QCAP.

Imai et al. (2003) introduced the service priority of the ship in dynamic berth allocation circumstances. They first attempted to utilise the Lagrangian relaxation; however, they found that the relaxed problem is reduced to the quadratic assignment problem (QAP), which is difficult to solve in polynomially bounded computation time. Consequently, they applied a genetic algorithm for this problem. Imai et al. (2007) solved the berth allocation problem at indented berths for mega-containerships by a GA.

2.1.2 Yard Container Handling Schemes

When the container terminal is designed, the type of material handling equipment that carries out the transport of containers between the quayside and the storage yard should be determined at the strategy level. Vehicles like forklift TKs, yard TKs or straddle carriers can be used at a manually operated terminal; while at an automated terminal, automated guided vehicles (AGV) or automated lifting vehicles (ALV) are the most common equipment. Different types of container transport equipment have also been studied (Baker, 1998, Vis and Harika, 2004, Duinkerken et al., 2006, Yang et al., 2004).

Unlike many seaport terminals, which utilise yard vehicles that are either manned or fixed upon a circuit of tracks (Vis and Harika, 2004), the Patrick AutoStrad container terminal environment has been designed for the use of a fleet of fully ASCs. ASCs are classified as ALVs. They are capable of independently lifting a container, transporting the container from the pickup location and setting down the container at a destination. The pickup and setdown of containers can occur at ground level or from/to a multiple level stack of containers. The use of ASC effectively decouples the unloading/loading operations performed by the QC from the transportation of the container within the yard stack. However, the operations performed by the ASC and QC intersect at a buffer area. The buffer area provides the temporary storage of containers during the transhipment process by acting as an operational interface between QCs and ASCs.

One of the decisions at the tactical level is the determination of the necessary number of transport vehicles. The fleet sizing problem has been examined in depth (Steenken, 1992, Vis et al., 2001, Koo et al., 2004). An important decision at the operational level is to determine which vehicle transports which container by which route. The general VRP has been studied at length (Bish et al., 2001, Narasimhan and Palekar, 2002, Li and Vairaktarakis, 2004).

The fleet sizing and vehicle routing procedure by Koo et al. (2004) uses a two-phase heuristic Tabu Search algorithm. This system is used for container ports with several yards. The goal of the procedure is to find the smallest required fleet size and a route for each vehicle to fulfill all transportation requirements within a static planning horizon. A computational study shows solutions of good quality in comparison with two other existing methods. It also assumes that the vehicles' travel time is constant between two points.

The Autonomous Straddle Carrier Scheduling Problem (ASCSP) considers the starting and ending time of each yard job in a set of yard jobs assigned to a fleet of ASCs servicing a set of QCs. The issue is to minimise the *usage* of ASCs in the transhipment process, while finding a feasible and efficient schedule. This requirement aims to reduce costs associated with performing yard-side operations, while solving the allocation and scheduling sub-problem for yard resources, namely a fleet of ASCs.

2.1.3 Land-Side Container Handling Schemes

Land-side operations are the phase of container handling operations to transport from/to customers or other terminals. As such, containers need to be moved from/to the yard stack to/from other modes of transportation like road, rail or waterways (Vis and Koster, 2006). The amount of delay time of external TKs for receiving and delivery operations is the most important performance measure for the customer service level. The external TK sequencing problem was studied in the early 21st century (Kim and Park, 2003).

The Patrick AutoStrad container terminal has been designed to support road transportation in its land-side operations. Similar to berthed ships, TKs can either be importing or exporting containers between the yard and the Truck Import Area (TIA). Transporting containers between the yard and the TIA is a land-side operation which requires the use of ASC. ASCs are capable of performing pickup and setdown operations directly upon TKs located in a bay of the TIA as illustrated in Figure 2-2. Minimising the turnaround time of TKs at the TIA can provide greater throughput of the overall process provided there are waiting TKs to fill any free TIA bays. Since TK movements are booked using the Vehicle Booking System (VBS), and free TIA bays are common at the Patrick AutoStrad container terminal, the allocation of TKs to TIA bays is greatly simplified to a first-come-first-served (FCFS) algorithm.

Figure 2-2: ASC servicing a TK in the TIA at the Patrick AutoStrad container terminal located in Brisbane, Australia.

Zhao and Goodchild (2010) used simulation to evaluate the use of truck arrival information to reduce container rehandles during the import container retrieval process by improving terminal operations. A variety of scenarios with different levels of truck information and various container bay configurations were modelled to explore how the information quality and bay configuration affect the magnitude of benefit.

In a timely study (Ballis and Abacoumkin, 1996), the authors presented a simulation model of the port of Piraeus, Greece for performance evaluation of land-side operations using TKs. Here, the TKs are serviced by a fleet of ASCs. By incorporating expert knowledge from operators into the model, the simulator assesses the different configuration of five terminal characteristics. The resulting simulations suggest that the 'observed strategy leads to shorter TK service time but increases the traffic conflicts in the terminal's internal transport networks'.

2.1.4 Integrated Container Handling Schemes

Probably the first attempt at performing integrated scheduling of different resources at a seaport container terminal was performed at the turn of the 3rd millennium (Meersmans and Wagelmans, 2001). The authors used a static and a dynamic variation of the beam search algorithm to model the problem. A form of rescheduling, variable job horizons (i.e. the number of jobs for each handling vehicle to consider ahead) and different vehicle dispatching algorithms were also investigated in this extensive study. In summary, longer planning horizons (up to 50 containers/QC) provided better performance on average. Furthermore, the static beam search performed better than the dynamic variation and it was also better than a number of typical dispatching rules in that it avoided deadlock situations.

In a rigorous study (Kim et al., 2004a), the authors investigated the integration of two decision-making sub-problems. In the first sub-problem, a pickup schedule was constructed which determined both the route of a transfer crane (TC), as well as the number of containers it must pick up at each yard-bay. In the second sub-problem, the load sequence for individual containers was determined. A beam search algorithm was applied to solve the load-sequencing problem in the seaport container terminal. The algorithm was used to maximise the operational efficiency of TCs and QCs while

satisfying various constraints on stacking containers onto vessels. It was shown that the beam search algorithm outperforms an ant colony algorithm in the solution values of the objective functions and computational time.

In another study (Hartmann, 2004), the author proposed a generalised optimisation model that could be simultaneously applied to ASCs, AGVs, stacking cranes and reefer workers. A priority rule based heuristic and GA were used to find adequate solutions to the optimisation problem. However, this study was performed offline on artificial data, which is unable to effectively capture the dynamics of a real seaport container terminal. The author suggested the use of a simulation environment as an important direction for future research in relation to integrated optimisation problems.

The authors of another significant study (Lau and Zhao, 2007), recognised that most of the relevant literature considers the allocation and scheduling for only a single type of handling equipment. The authors proposed that optimising just one type of handling equipment in the transhipment process discounts the tight coupling between the schedules and allocation schemes of the remaining handling equipment. As a result, this can produce inefficiencies and sub-optimality in the entire transhipment operation. A mathematical programming model and multi-layer (hierarchical) GA were developed and used in numerical simulation studies. This multi-objective formulation attempted to minimise QC delay time, the total travelling time of AGVs between the quay-side and storage blocks, and the total travelling time of AGVs and automated yard cranes (AYCs). Static travelling times of AGVs were used in the simulation negating the requirement for any path planning algorithm. The fleet size of AGVs varied from 8-to-16 vehicles, the planning horizon for QCs varied from 8-to-16 tasks per QC and the final factor injected uncertainty into the operation times of QCs. With four QCs, simulation results were compared with different AGV dispatching rules including, earliest due date, fixed AGV-to-QC assignment and nearest-vehicle-first. A multi-layered genetic algorithm (MLGA) consistently found lower objective function values for all simulation experiments.

To address the issues of ineffective or inefficient scheduling of resources in the transhipment process, this thesis defines an integrated problem and proposes an analytical model which integrates quay-side, yard and land-side operations.

2.1.5 Scheduling of Automated Vehicles

Briefly, the scheduling of AGVs is to dispatch a set of AGVs to complete a batch of pickup/drop-off jobs to achieve certain goals (e.g. shortest completion time, minimum AGV idle time, etc.) under given constraints. As one of the enabling technologies, the scheduling of AGVs has attracted considerable attention. In addition, many algorithms about the scheduling of AGVs have been proposed (Daniels, 1988, Grunow et al., 2004, Kim and Bae, 2004, Kim and Tanchoco, 1991). Recently, more container terminals have utilised automated vehicles, like AGVs. As a consequence, the research on the scheduling of AGVs becomes important.

In a compelling study (Grunow et al., 2004), the authors proposed a novel heuristic dispatching algorithm for a fleet of multi-load AGVs. The analysis performed by the authors is based on the distinction of different degrees of AGV availability and a definition of assignment patterns which promotes alternative AGV tours to be generated with very limited computational effort. Furthermore, the problem is also formulated as a mixed integer linear programming (MILP) model. The performance of the MILP model is compared to the heuristic dispatching algorithm for different AGV fleet sizes. Due to the port layout, this study only investigated the assignment problem for a small fleet of AGVs (i.e. a maximum of 6 vehicles). The assignment problem is solved while minimising the total lateness of the AGVs. Additional objectives related to the QC or land-side operations were not considered. In a similar study (Kim and Bae, 2004), the authors examined AGV dispatching methods by utilising information about locations and times of future delivery tasks. An MILP model is provided for assigning optimal delivery tasks to a small fleet of AGVs. A heuristic algorithm is suggested for overcoming the excessive computational time needed for solving the MILP model. In this study, the dispatching problem is reduced to an assignment problem by defining the exact pickup and delivery times of each container for a single QC.

Also of note, a recent study, (Kim and Nguyen, 2009) discussed a method for dispatching a small fleet of ALVs for supporting the efficient operations of QCs and automated yard cranes (AYC). The problem was introduced as a scheduling problem in terms of precedence and buffer constraints. These constraints arise owing to the

variable buffer space allocated in the apron and yard upon which QCs, AYCs, and ALVs can all release containers. An MILP model and a heuristic algorithm were proposed for the ALV scheduling problem, with the aim of minimising both the total travel time of ALVs and the total delay in QC operations. The heuristic algorithm was used to find solutions to this *NP*-hard scheduling problem after the authors applied a procedure to convert buffer constraints into time-window constraints. Interestingly, the effect of the dual cycling at buffers of varying sizes was also analysed, which suggested that as the proportion of dual cycling operations increased, the ALV total travel time, QC total delay time, and vessel completion (berthing or handling) time decreased. However, the authors suggest the performance of the algorithms proposed in their study needed to be assessed in a simulated and dynamic environment. Furthermore, this study only addressed the dispatching of ALVs and not the integrated scheduling problem which includes ALV path planning, automated yard cranes and QC operations.

2.2 Relevant Problem Models for Scheduling

Scheduling plays a central role in logistics management. A wide variety of scheduling problems have been studied in the literature. Different scheduling problems address different practical situations but focus on a common problem—the efficient use of a group of vehicles/machines for executing a set of jobs. There are three models that have been studied most widely in the literature: the Multiple Travelling Salesman Problem (MTSP), PDP, and Capacitated Vehicle Routing Problem (CVRP).

2.2.1 The Multiple Travelling Salesman Problem

MTSP is a generation of the well-known travelling salesman problem (TSP), which is an NP-hard problem. The MTSP is even more difficult than TSP because it aims to find a group of Hamiltonian circuits without sub-tours for m (m>1) salesmen to serve a set of n (n>m) cities. This leads to the optimum solution of MTSP becoming more computationally infeasible as the problem size increases. Compared to the TSP, the MTSP is more suitable for modelling real world problems, because it is capable of handling more than one salesman. As such, many practical problems have been modelled as MTSP, such as print press scheduling (Gorenstein, 1970), crew scheduling

(Svestka and Huckfeldt, 1973), hot rolling scheduling (Tang et al., 2000), mission planning (Ryan et al., 1998) and vehicle scheduling (Park, 2001). Additionally, the MTSP can also be extended to many variations. Based on the number of depots, it can have single depot or multiple depots. It can also have open or closed tours, where the difference is whether the salesmen need to return to their depot(s). Added to this, if the salesmen need to pick up loads, it can be cast as the MTSP pick up and delivery problem (MTSPPDP) (Wang and Regan, 2002). If the tasks have time-related constraints, it arises as the MTSP with time windows (MTSPTW) (Kim and Park, 2004). There are other combinations of the above forms coming from real world applications. In all of the problems, job planning and vehicle scheduling are the most commonly researched areas (Bektas, 2006).

Due to the combinatorial complexity of the MTSP, many researchers have tried to relax the MTSP to the TSP and use exact algorithms to solve it, yet the results have always been unsatisfactory (Bektas, 2006). It is therefore preferable to utilise heuristics to obtain a near-optimum solution to the problem, particularly when its search space is extremely large. Most of the research on using GAs for the vehicle scheduling problem have focused on using two different chromosome designs (i.e. one chromosome representation and two chromosome representations) for the MTSP. Both of these chromosome designs can be manipulated using classic GA operators developed for the TSP; however, they are also prone to produce redundant solutions to the problem. The authors in one study (Carter and Ragsdale, 2006) have proposed a GA for the MTSP that uses a two-part chromosome representation which effectively reduces the redundancy in the search space. The crossover operation for the two-part chromosome is separated into two sections. The first section uses an ordered crossover operator, while the second section uses an asexual crossover operator so as to ensure that the second part of the chromosome remains feasible (with the sum of the values in the chromosome equalling n). However, due to the nature of the two-part chromosome, Carter has suggested in his doctoral thesis (Carter, 2003) that further research on more effective crossover operators will become increasingly important.

Although the TSP has received a great deal of attention, the research on the MTSP is relatively limited, and most of the work is related to MTSP applications (Bektas, 2006). Apart from GAs, other bio-inspired optimisation algorithms such as ant colony

optimisation (ACO), artificial neural network (ANN) and particle swarm optimisation (PSO) have been used to solve the TSP/MTSP (Kulkarni and Tai, 2010,). These algorithms are used in conjunction with various local improvement/heuristic techniques to jump out of local minima and also to reduce the computational cost. For example, authors in one study (Liu et al., 2009) have proposed an ACO algorithm for solving the MTSP. In the algorithm, the pheromone trail updating and limits followed the MAX-MIN Ant System scheme, and a local search procedure was used to improve the performance of the algorithm. They compared the results of the algorithm with GAs on some benchmark instances in the relevant literature, and the computational results show that their algorithm was competitive over two objective functions. Some of the fundamental ways of solving the MTSP are the expansion of the problem by converting it into the standard TSP and the simplification of the problem by using the approach of cluster-first-route-second. The work in an early study (Bellmore and Hong, 1974) represents one of the first attempts to expand the MTSP by converting it into the standard TSP through introducing m-1 imaginary bases/depots. This makes the total number of cities to be $n + m$ - 1. This increases the problem size to almost double (Bellmore and Hong, 1974, Kulkarni and Tai, 2010,). It also extends the cost matrix to $(n + m - 1) \times (n + m - 1)$ (Kulkarni and Tai, 2010,, Christofides et al., 1981), making the problem computationally tedious as compared to the standard TSP with the same number of cities. This becomes worse when the number of cities is very large.

2.2.2 Pick-up and Delivery Problem

In the general PDP (Savelsbergh and Sol, 1995), a set of routes has to be constructed in order to satisfy transportation requests. A fleet of vehicles is available to operate the routes. Each vehicle has a given capacity, a start location and an end location. Each transportation request specifies the size of the load to be transported, the location where it is to be picked up (the origin) and the location where it is to be delivered (the destination). Each load has to be transported by one vehicle from its set of origins to its set of destinations without any transshipment at other locations.

A lot of work has been done in PDP which initially started in the 1980s (Psarafis, 1980). The work can be found in some of the comprehensive literature surveys done over the years (Parragh et al., 2008).

With regard to exact methods, Psaraftis (Psarafis, 1980, Psarafis, 1983) and Desrosiers (Desrosiers et al., 1986) used dynamic programming techniques to solve the PDP and the PDP with time windows, all of them defining the concept of the state of the system for each particular problem. Dumas *et al.* (Dumas et al., 1991) proposed a column generation procedure to solve the PDP with time windows, using the Dantzig–Wolfe decomposition. Their method was able to solve problems of size up to 50 customers. Ruland and Rodin (Ruland and Rodin, 1997) formulated the PDP as a mixed-integer programming problem (MIP) and proposed four kinds of valid inequalities for the problem, based on the travelling salesman problem with precedence constraints. They used a branch-and-cut procedure to solve instances up to 15 customers. Lu and Dessouky (Lu and Dessouky, 2004) solved the PDP with time windows using a branch-and-cut technique, based on new valid inequalities proposed by the authors. They solved instances up to 17 customers. More recently, Dumitrescu (Dumitrescu, 2005) provided several valid inequalities for solving the travelling salesman problem with pickups and deliveries (TSPPD), establishing those among all known inequalities that define facets of the TSPPD polytope.

Although many exact methods have been developed for solving variants of the PDP, none of them actually avoid the complexity of the problem, limiting their solution power to small size problems. This evident drawback motivated the development of good heuristics to solve medium and large scale systems. For example, Sexton and Bodin (Sexton and Bodin, 1985a, Sexton and Bodin, 1985b) proposed an heuristic method to solve the pick-up and delivery problem, based on Benders decomposition. They decomposed the problem into a routing problem (hard) and a scheduling problem (easy).

In the recent years, some sophisticated approximate techniques have been used for solving dynamic PDP instances, mainly GAs (Haghani and Jung, 2005, Osman et al., 2005), and various metaheuristics (Li et al., 2005, Tarantilis et al., 2005, Bianchessi and Righini, 2007). Cordeau and Laporte (Cordeau and Laporte, 2003) developed a tabu

search heuristics for the DARP with time windows, where the objective was to minimise routing costs. Moreover, Cordeau (Cordeau, 2006) developed a branch and cut algorithm to solve the same type of problem.

2.2.3 Capacitated Vehicle Routing Problem

The CVRP is a combinatorial optimisation and integer programming problem seeking to service a number of customers with a fleet of vehicles. In the CVRP, one has to deliver goods to a set of customers with known demands on minimum-cost vehicle routes originating and terminating at a depot. The vehicles are assumed to be homogeneous and have a certain capacity. The CVRP literature is vast. Classic heuristics for the problem have been surveyed by Laporte and Semet (Laporte and Semet, 2002), and metaheuristics have been surveyed by Gendreau *et al.* (Gendreau et al., 2002) and more recently by Cordeau *et al.* (Cordeau et al., 2004). CVRP heuristics have typically been tested on 14 instances containing between 50 and 199 customers. In the early 1990s very good metaheuristics for the CVRP were developed such as the parallel tabu search by Taillard (Taillard, 1993). Significantly, most of the solutions to the 14 classic instances found in these studies have to date not been improved upon. More recently, some larger instances have been introduced containing between 240 and 1200 customers (Li et al., 2005, Golden et al., 1998). These new instances seem to have spurred a new interest into metaheuristics for the CVRP as indicated in the survey by Cordeau *et al* (Cordeau et al., 2004). Until recently, exact methods for the CVRP were dominated by branch-and-cut methods. One of the best branch-and-cut algorithms for the CVRP was developed by Lysgaard et al. (Lysgaard et al., 2004). Recent research results indicate that branch-and-cut-and-price algorithms are viewed as a more promising approach (Fukasawa et al., 2004). For the CVRP, the largest problem that has been solved to optimality contains 135 customers.

Furthermore, the CVRP can be extended to VRPs with Time windows (VRPTW) by associating time windows with the customers. The time window defines an interval during which the customer must be visited. Solving the VRPTW to optimality has also received much attention. The current state of the art exact methods are proposed by Kallehauge *et al.* (Kallehauge et al., 2001), Irnich and Villeneuve (Irnich and

Villeneuve, 2003) and Chabrier (Chabrier, 2006), and all follow the branch-and-price framework. The two first mentioned approaches also strengthen the obtained lower bound by adding valid inequalities to the LP formulation. The size of the instances that consistently can be solved to optimality is rather limited as unsolved instances with 50 customers exist, but some large-scale instances can be solved. For example, Kallehauge *et al.* (Kallehauge et al., 2001) report that a 1000 customer instance has been solved. Solving problems of this size using exact methods is possible only if the instance has a certain structure and the time constraints are very tight. These observations justify the research into heuristics for the VRPTW because industrial routing problems demand robust algorithms for large-sized instances. Additionally, the VRPTW has been the target of extensive research and almost every type of meta-heuristic has been applied to the problem. For recent surveys on state of the art VRPTW research it is worthwhile to recommend the survey by Cordeau *et al.* (Cordeau et al., 2002) that describes both exact and heuristic methods, and the survey by Bräysy and Gendreau (Bräysy and Gendreau, 2005) that focuses on meta-heuristics. It is hard to single out a few VRPTW metaheuristics as no particular heuristics dominates all the other heuristics in all areas.

2.2.4 Discussion

Basically, CVRP is similar to the path planning problem in ASCs, since both of them seek to service a number of customers/containers with a fleet of vehicles, and with the time window they do not want to block or collide with each other. MTSP and PDP are very similar to the job scheduling of ASCs, because their common objective is to efficiently allocate the jobs to resources (i.e. salesman or vehicle). In this research, the problem of scheduling ASCs has subtle but significant differences to the MTSP, PDP and CVRP.

(1). In general, MTSP, PDP and CVRP do not have sequence constraints on visiting cities or nodes, and salesmen or vehicles are normally allowed to conduct jobs in any order. In the particular problem that this thesis addresses (i.e the operation requirements at a container terminal), the job conducting sequence has to be taken into account, particularly for coordinating QC and TK related jobs.

(2). MTSP, PDP and CVRP do not have to deal with constraints such as servicing QCs and TKs, which adds complex timing dependence for related jobs. Due to the timing dependence of all resources (i.e. ASCs, QCs and TKs), it is not suitable to use time-window based approaches (e.g. MTSP with time-window and PDP with time-window) to deal with the correlated job sequence and timing constraints.

(3). The performance metrics for ASC scheduling are based on practical operations at Patrick AutoStrad terminal, which are different from the general MTSP, PDP and VRP. The objectives in this research are not only related to ASCs (e.g. minimising total travel time, which is a linear function), but also include the performances of QCs and TKs (e.g. QC waiting time and TK waiting time which are non-linear functions). Since those studied approaches (Bektas, 2006, Dumas et al., 1991, Li and Lim, 2001, Ruland and Rodin, 1997, Savelsbergh and Sol, 1995, Park, 2001) to solving MTSP, PDP and CVRP are often for linear objective functions with linear constraints, they cannot be directly used to the model and solve the problem at the Patrick AutoStrad Terminal.

2.3 **Optimisation with Genetic Algorithms**

GA is an optimisation method which is very easy to understand and is easily transferred to existing simulations and models, and various modified GAs have been utilised to solve many container handling problems. Imai et al. (2006) solved a multi-objective simultaneous stowage and load planning problem by GA. Imai et al. (2007) solved the berth allocation problem at indented berths for mega-containerships by GA. Lee *et al* (Lee et al., 2008) used GA to solve the quay crane scheduling problem. Golias *et al*. (2009) proposed the dynamic BAP with customer service differentiation based on respective agreements. They formulated their BAP as a multi objective problem and developed a GA-based heuristic.

GAs have been already viewed as a relatively new optimisation technique that can also be applied to solving many problems such as MTSP, PDP and VRP. The basic ideas behind GAs evolved in the mind of John Holland at the University of Michigan in the early 1970s (Holland, 1975). GAs were not originally designed for highly constrained optimisation problems but were soon adapted to order based problems like TSP (Carter,

2003, Goldberg and Lingle, 1985). The development of effective GA operators for TSPs led to a great deal of interest and research to improve GAs' performance for solving this type of problem.

In a nutshell, GAs work by generating a population of numeric vectors (called chromosomes), each representing a possible solution to a problem. The individual components (numeric values) within a chromosome are called genes. New chromosomes are created by crossover (the probabilistic exchange of values between vectors) or mutation (the random replacement of values in a vector). Mutation provides randomness within the chromosomes to increase coverage of the search space and help prevent premature convergence on a local optimum. Chromosomes are then evaluated according to a fitness (or objective) function, with the fittest surviving and the less fit being eliminated. The result is a gene pool that evolves over time to produce better and better solutions to a problem (Carter, 2003). The GA's search process typically continues until a pre-specified fitness value is reached, a set amount of computing time passes, or until no significant improvement occurs in the population for a given number of iterations.

The key to finding a good solution using a GA lies in developing a good chromosome representation of candidate solutions to the problem. A good GA chromosome should reduce or eliminate redundant chromosomes from the population. Redundancy in the chromosome representation refers to a solution being able to be represented in more than one way and appearing in the population multiple times. These multiple representations increase the search space and slow the search. So far, the two-part chromosome technique (Carter and Ragsdale, 2006) has been viewed as the more suitable representation with minimum redundancy for MTSP.

In (Singh and Baghel, 2009), the authors proposed a new grouping GA based approach for the MTSP and compared their results with other approaches available in the literature. Their results showed that the approach outperformed the other approaches on two objectives, particularly for the computation time. However, in the two proposed approaches (Carter and Ragsdale, 2006, Singh and Baghel, 2009), they both adopted some local heuristic search techniques, which are designed to provide a very good starting point for GAs (as a seed).

In general, GAs are global optimisation techniques (Bo et al., 2006, Cus and Balic, 2003, Smith and Smith, 2002, Goldberg, 1989, Holland, 1992) that avoid many of the shortcomings that exist in classical local search techniques. In particular, crossover is a very powerful mechanism for introducing new genetic material and maintaining genetic diversity, but with the defining property that good parents also produce well-performing offspring or even better offspring. Although using a local optimisation technique can speed up the overall search process, it is difficult to tell the real search ability of a crossover method. Hence, in this thesis, crossover operators are compared without any local optimisation so as to show differences of search ability with different crossovers.

Generally, one-point crossover and two-point crossover have been used as two basic genetic crossover operators for many problems (Chatterjee et al., 1996, Kellegöz et al., 2008). One-point crossover is the most basic crossover operator proposed by Holland (Holland, 1975), and the specific operation process is as follows: select a crossover point in the individual string stochastically, the two point before and after the crossover point exchange their structure, then two new individuals are generated. One-point crossover tends to treat some loci preferentially as the segments exchanged between two parental chromosomes always contain the endpoints of the gene string. To reduce such positional bias, two-point crossover can be used. Two-point crossover (Holland, 1975) can also be viewed as the improvement to the one-point crossover. The difference between them is that, two-point crossover chooses two points in the two individual strings which are mated with each other, and then the gene segments are exchanged between the selected two crossover points. In an early study (Spears and De Jong, 1991), Spears and De Jong found that two-point and in particular multi-point crossover encourages exploration of the search space due to the increase in gene disruption. This helped prevent premature convergence to highly fit individuals thereby making the search more robust.

There have also been many attempts to discover appropriate crossover operators for TSP, and the most widely used crossover operators are: the ordered crossover operator (ORX), the cycle crossover operator (CYX) and the partially-matched crossover operator (PMX). The ORX (Davis, 1985, Gen and Cheng, 1997) was proposed by

Davis, which constructs an offspring by choosing a subsequence of one parent and preserving the relative order of genes of the other parent. This crossover operator has been commonly used in GAs for solving many problems, particularly for TSP, MTSP (Bektas, 2006, Kulkarni and Tai, 2010,). The CYX (Oliver et al., 1987) was proposed by Oliver *et al.*, which attempts to create offspring from the parents where every position is occupied by a corresponding element from one of the parents. The PMX (Goldberg and Lingle, 1985) was suggested by Goldberg and Lingle, which builds offspring by choosing a subsequence of a chromosome from one parent and preserving the order and position of as many genes as possible from the other parent. However, these crossover methods cannot be directly applied in the GA with the two-part chromosome representation. In (Carter and Ragsdale, 2006), Carter and Ragsdale proposed a combined crossover approach ORX+A (ORX combined with an asexual crossover (Chatterjee et al., 1996)) for the two-part chromosome representation. They employed the ORX for the first part of the chromosome, and used an asexual crossover for the second part of chromosome. Chapter 6 provides an overview of the existing approach, and we propose a new crossover approach, and compare the performance with the three crossover methods (ORX+A, CYX+A and PMX+A) for the two-part chromosome.

2.4 **Summary**

This chapter provides a literature review on previous works which are closely related to this research. Three key container handling schemes have been reviewed, which are associated with quay-side, yard, land-side operations and the integrated process. Subsequently, the scheduling of AGVs has also been discussed with the related work. Furthermore, three classic problem models: MTSP, PDP and CVRP have been reviewed, and the related solution techniques have been presented. These approaches have proved to be applicable in some cases, but cannot be directly applied to container scheduling directly. Last, optimisation with GAs has been reviewed.

Most achievements on the port automation are related to the optimisation of the container handling process, such as using modified GA for solving the QC scheduling/allocation problem and container storage problem. For the general studied

problems, such as MTSP, PDP and CVRP, these can be cast into many practical problems and the solution methods correspondingly vary to a strong degree.

This thesis studies an integrated process and develops practical deployable strategies and algorithms for scheduling container transfers. A comprehensive mathematical model is presented in Chapter 3. Chapter 4 presents a job grouping approach for solving the integrated problem. Chapter 5 presents a modified GA for solving the scheduling problem. Chapter 6 presents a new crossover approach for further improvement.

3. Mathematical Modelling for Automated Container Transfers

3.1 Introduction

This chapter formulates two mathematical models for automated container transfers: a comprehensive model and a job scheduling model. There may be multiple formats for the problem formulation, yet our formulation is based on the mixed integer programming with extra specific constraints from the container terminal.

Apart from the essential consideration on collision avoidance of a fleet of large vehicles in a confined area, the comprehensive model also deals with many other practical constraints, such as the presence of multiple levels of container stacking and sequencing, variable container orientations, and vehicular dynamics that require finite acceleration and deceleration times. Based on the comprehensive model, a job scheduling model is formulated to focus on the optimisation of ASC job scheduling. Due to the expensive computational cost for solving a large problem in the comprehensive model, production managers of the automated container terminal are usually more concerned with the job scheduling part, as resource utilisation and port cost are more dependent on the job scheduling.

3.2 Overview of the Patrick AutoStrad Container Terminal

A yard environment map has been developed to model the actual Patrick AutoStrad container terminal, which is at the container terminal located within the Port of Brisbane at Fisherman Islands, Australia. The model formulation includes all the container handling subsystems of an automated container terminal. Figure 3-1 illustrates the static seaport environment from which a map was developed consisting of 18380 positional nodes and 83155 predefined links. It includes a set of QCs located within bays ready to perform either uploading or dispatching operations between a berthed container ship and associated buffer nodes located in lanes on the quay-side of the yard. The yard stack and TIA are also illustrated. The areas labelled MXA and

MXB are used for the temporary storage of containers being transported between the yard and the TIA.

Currently, there is a fleet of 30 ASCs operating within the yard environment. The working area is strictly confined and ASCs can travel freely from position to position, along paths amongst the defined links. Thus, the problem of optimising the assignment of tasks to ASCs (task allocation) is complicated by the additional requirement of ensuring safety through collision-free path planning, while attempting to meet the overall objective of minimising the turnaround time of berthed ships and TKs docked at the TIA.

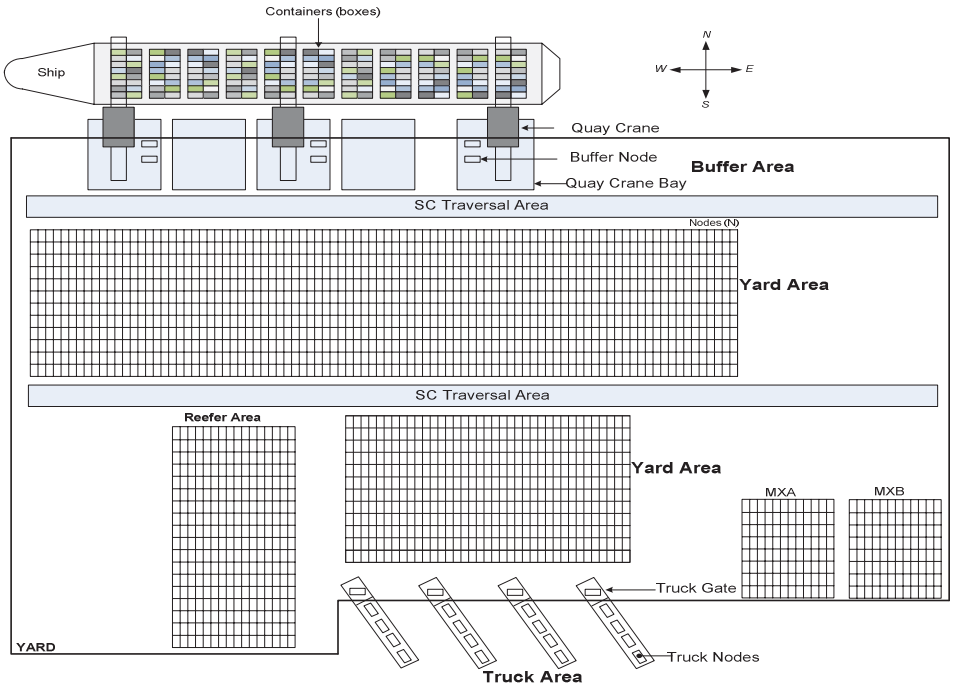


Figure 3-1: Schematic diagram of the static seaport environment showing berth, QCs, bays, special nodes (QC and TK gates), TIA, and nodes in the yard.

Considering the requirements, let the map be represented by a graph (*N, L*), where all nodes are contained in a set *N* and all feasible links upon which ASCs can travel are contained in the set *L*. Nodes are not uniformly spaced within the map, hence links are not equal. All links in *L* are bi-directional and each connects two neighbouring nodes. For each node in the map, there is a two-level stack where a container can be stored. That is, each node can be occupied by two containers vertically, which adds significant complexity to the problem as both setdown and pickup sequencing must be considered.

Using a graph to represent the seaport map allows for the accurate determination of position and trajectory information at any time.

3.3 Job Definition

In general, container transfers refer to three categories of jobs: quayside transfers, yard transfers and hinterland (landside) transfers. It is necessary that all job specifications be described using valid locations within the seaport environment map and that all job dependencies are defined completely and correctly.

A list of container transfers jobs needs to be performed by a fleet of identical ASCs. For each job, the Yard Management System (YMS) provides both an initial node (i.e. pick-up position) and a destination node (i.e. set-down position), and also specifies the container stack level and container alignment for both the pick-up location and set-down location. In this thesis, the container transferring jobs are divided into five different categories based on their initial and destination nodes in the yard:

- Buffer-to-Yard (B2Y): transport a container from a buffer node to a yard node for QC ship discharging;
- Yard-to-Buffer (Y2B): transport a container from a yard node to a buffer node for QC ship uploading;
- Truck-to-Yard (T2Y): transport a container from a truck area node to a yard node TK exporting;
- Yard-to-Truck (Y2T): transport a container from a yard node to a truck area node for TK importing;
- Yard-to-Yard (Y2Y): transport a container from a yard node to another yard node for yard management.

3.3.1 Quayside Transfers

In this study, QCs are fully-programmable and able to perform discharging (i.e. ship-to-buffer) jobs (J^D) and uploading (i.e. buffer-to-ship) jobs (J^U) according to a predefined job sequence. Each QC jib is associated with four *special* positions. These include, two buffer positions (*P1* and *P2*), one QC jib (i.e. the long part of the crane)

mid-point node (*P3*) and one ship node (*P4*) as illustrated in Figure 3-2. Quayside transfers are usually conducted by specific QCs for container discharging from the ship or container uploading onto the ship. J^D is a set of ship-to-buffer jobs which must be performed using a set of discharging QCs. A discharging QC is associated with a specific set of jobs (J_q^D) which must be performed by the QC (q) between the ship and allocated buffer positions. In addition, all discharging jobs (J_q^D) need to be performed sequentially and in a fixed order that is decided *a priori* by the particular shipping company and port management system. Usually, there is a predefined order for the containers to be placed onto the buffer area, so that the system can check if any boxes are out of sequence or if there is any delay incident. If the order cannot be followed due to some delay or other issue, the system will update the status, and the related containers can be placed in the buffer as long as the buffer node is available.

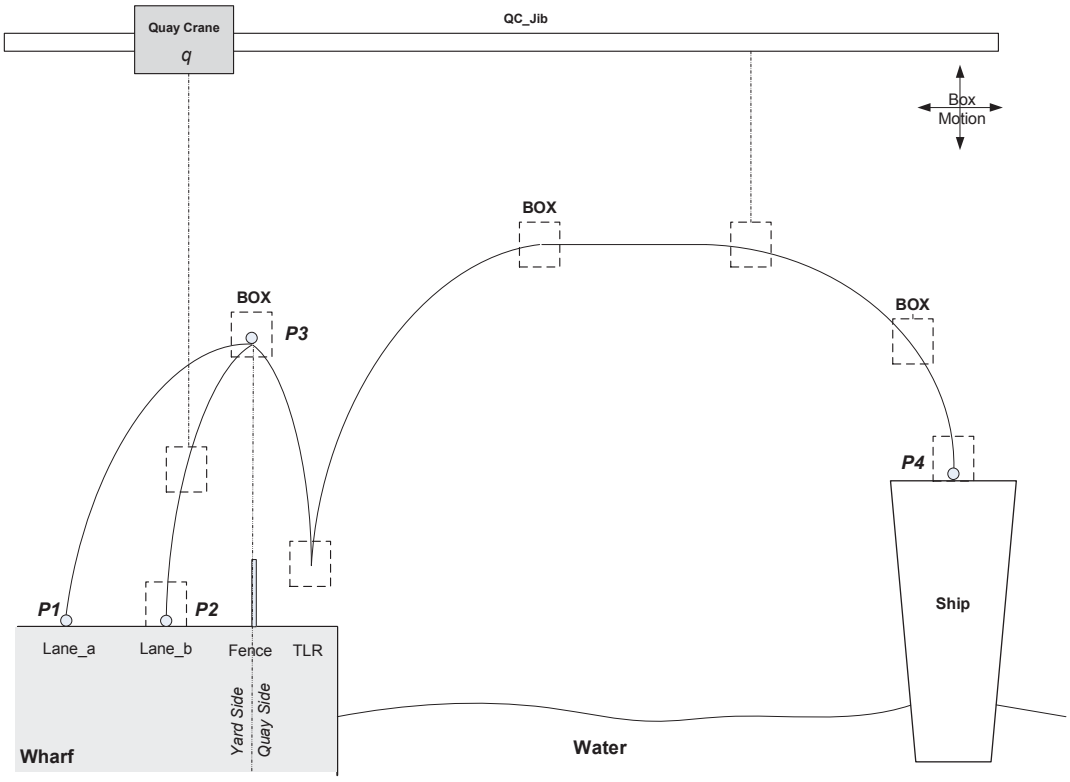


Figure 3-2: Quay Side Operations. A trajectory for the QC jib can be calculated from the associated nodes and links between the ship and wharf. (TLR refers to twist lock release.)

For any discharging job, a box identification number, initial position (*P4*), destination positions (*P1* or *P2*) and associated discharging QC are required. The final destination node can be one of two buffer nodes (*P1* or *P2*) determined by the planner. The buffer

positions are single stack level, and the alignment of boxes placed on the buffer nodes is determined by the direction of the berthed ship. In general, if a container door faces north (*N*) or east (*E*), then the container alignment is *aligned*. If, however, a container door faces south (*S*) or west (*W*), then the container alignment is *opposite*.

Similarly, for any uploading job, a box identification number, initial positions (*P1* or *P2*), destination position (*P4*) and associated uploading QC are required. The initial position can be one of two buffer positions (*P1* or *P2*) determined by the planner. There is a single stack level for the buffer positions and the alignment of boxes placed in the buffer nodes is determined by the direction of the berthed ship. All uploading jobs (J^U) need to be performed sequentially and in a fixed order that is planned *a priori* by the particular shipping company and Patrick. When ASCs service QCs by transferring containers, ship-to-buffer jobs (J^D) become B2Y jobs, and buffer-to-ship jobs (J^U) become Y2B jobs. When the same QCs do the uploading job as well as the discharging job, it is called dual-cycling. However, due to the complexity of introducing dual-cycling, the situation in the model will not be specifically addressed in this thesis.

3.3.2 Yard Transfers

A set of Y2Y jobs (J^Y) must be performed using a set of identical ASCs (V). Each Y2Y job (j) must be conducted by one and only one ASC (v). That is, when ASC (v) picks up a box as part of an assigned job (j) the box must be transported to the destination by the same ASC (v). For all Y2Y jobs the Yard Management System (YMS) provides both an initial and destination node.

The initial and destination nodes are located within the yard. An initial yard node ($i \in N$) together with current stack level ($S_i \in \{0,1\}$) and alignment ($A_{S_i} \in \{0,1\}$) are required. Similarly, the destination yard node ($k \in N$) together with stack level ($S_k \in \{0,1\}$) and alignment ($A_{S_k} \in \{0,1\}$) are required. Typically, in the Patrick AutoStrad terminal, the YMS provides a destination set containing between 1:20 candidate destination nodes. However, this model formulation only considers a single destination node for Y2Y jobs. For example, the YMS could randomly select a single destination node *a priori* from the set of candidate nodes. A single destination node provided by the YMS would greatly reduce the computational complexity of the problem.

3.3.3 Land-side Transfers

In this study, export jobs (i.e. T2Y) (J^E) and import (i.e. Y2T) (J^I) jobs can be performed entirely by ASCs, including set-down and pick-up actions within the TIA. Consider an ASC (v) servicing a waiting TK. The TK is required to reverse into the TK import area assigned to the TK gate (g) and come to a complete stop, effectively locating itself upon four nodes (a,b,c,d) in the TK area, as illustrated in Figure 3-3.

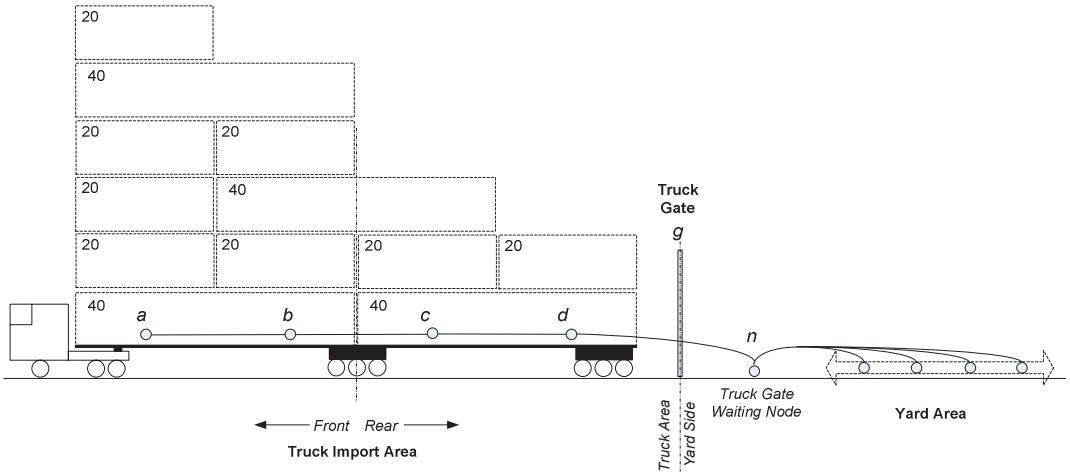


Figure 3-3: TK Area Operations showing associated nodes and links between the yard and TK area. TKs can import or export any combination of 20-ft and 40-ft containers (up to a total of 4TEU (Twenty-foot Equivalent Unit) in each direction) into the seaport, as shown by the different combinations.

For all export jobs, the YMS provides both a destination and initial node. The initial node can be one of four special nodes in the TIA, such that $i \in \{a,b,c,d\}$. There is no directional or stack level information for these special nodes. All TKs in the TK gate area have their boxes facing to the rear of the TK and are a single level stack. The destination yard node together with stack level and direction alignment is required as well.

For any import job, a box identification number, initial yard position, destination position (TK node) and associated TK gate are required. The destination node can be one of four special nodes in the TK import area, such that $i \in \{a,b,c,d\}$. There is no directional or stack level information for these special nodes. An initial yard node together with current stack level and alignment are required.

In addition, all import jobs and export jobs being serviced at the TK gate (g) must be arranged in a fixed order. For import jobs (J^I) the front containers are set-down (loaded) before the rear containers if the number of containers to be set-down onto a TK is greater than one. Similarly, for export jobs (J^E) the rear containers are picked (unloaded) before the front containers if the number of containers to be picked from a TK is greater than one.

3.4 Model 1: Comprehensive Model

This section presents a comprehensive mathematical model for container transfers by ASCs. The comprehensive model deals not only with the essential consideration of the collision avoidance of a fleet of large vehicles in a confined area, but also many other practical constraints, such as the presence of multiple levels of container stacking and sequencing, variable container orientations, and vehicular dynamics that require finite acceleration and deceleration times. The overall objective function and constraints will be presented as well.

3.4.1 Description of Container Alignment and ASC Trajectory

Considering that the orientation of a container is fixed in a single direction for both QC and TK related jobs, path planning must consider the orientation at the initial and destination nodes. To guarantee the orientation at the destination node, we model the alignment and changes in alignment during transportation (flip movements) as a container is transported from its initial node to the destination node.

Figure 3-4 depicts the door alignment and situations of changing direction via flip movements.

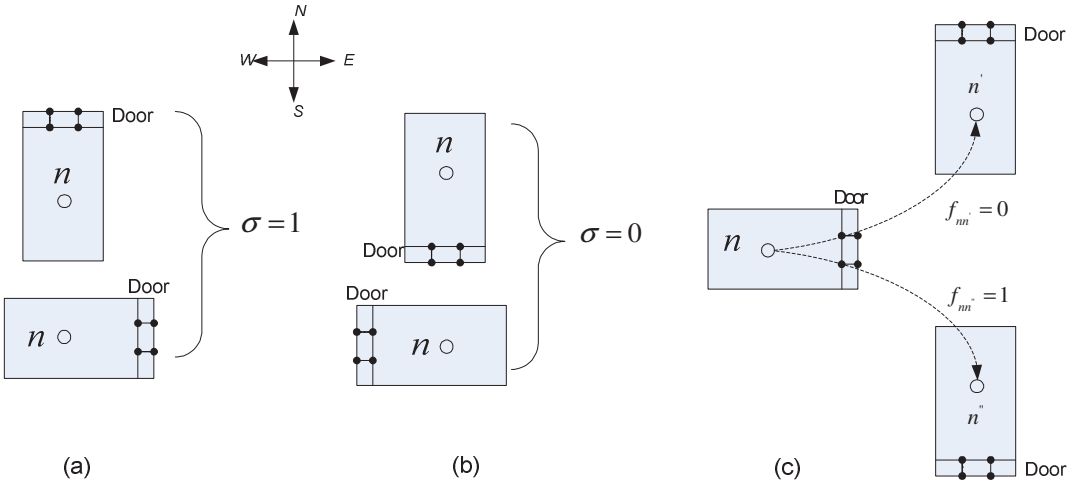


Figure 3-4: Container orientations (a,b) and flip movements (c).

where σ is the alignment of a container at node n. Due to the overall orientation and design of the Patrick AutoStrad terminal, if a container door faces *North* or *East*, then the container alignment is defined as *aligned* and $\sigma = 1$. However, if a container door faces *South* or *West*, then the container alignment is defined as the *opposite,* and $\sigma = 0$. This feature is determined by the current Patrick container terminal.

Let $R_v = \{p_{v1}, p_{v2}, ..., p_{vz}\}$ be the set of all position nodes of v trajectory and $R_v \subset N$. z is the total number of nodes which are planned for v. Let the timings for arrival and departure at each node in the trajectory of ASC (v) be given by the ordered sets $T_v^{Arrive} = \{t_{v1}^{Arrive}, ..., t_{vz}^{Arrive}\}$ and $T_v^{Depart} = \{t_{v1}^{Depart}, ..., t_{vz}^{Depart}\}$ respectively. t_{v1}^{Arrive} is equal to t_0 which is the plan starting time. The following constraints must be met:

$$\forall v \in V, p_{v1} = p_v^I \tag{3.1}$$

$$\forall k, 1 \le k \le z : t_k^{Depart} \ge t_k^{Arrive} \tag{3.2}$$

Eq(3.1) ensures that the initial position of the ASC (v) is the first node in its planned trajectory. Eq(3.2) requires that the departure time is not earlier than the arrival time at a node for v. Furthermore, the node, arrival and departure time ordered sets can be combined into the following general representation:

$$\delta_v = \left[R_v, T_v^{Arrive_i}, T_v^{Depart}\right] = \left[\{p_{v1}, ..., p_{vz}\}, \{t_{v1}^{Arrive}, ..., t_{vz}^{Arrive}\}, \{t_{v1}^{Depart}, ..., t_{vz}^{Depart}\}\right] \tag{3.3}$$

Eq(3.3) represents a planned trajectory of an ASC (v) and associated timings for arrival (t_k^{Arrive}) and departure (t_k^{Depart}) at each node $p_{vk} \in R_v$ in the path, and $1 \le k \le z$.

In order to achieve efficient task allocation of ASCs at anytime, we must be able to accurately determine the relative timings for all jobs. Figure 3-5 illustrates the relationship between the ASC position (relative to nodes) and the associated timing information for a job. Basically, when an ASC arrives at the pickup node u_j or the setdown node d_j, it requires a constant pickup time (Δt_{pickup}) or setdown time ($\Delta t_{setdown}$). T_j^S and T_j^F are the planned times for ASC to pick up the container at node u_j and set down the container at node d_j respectively.

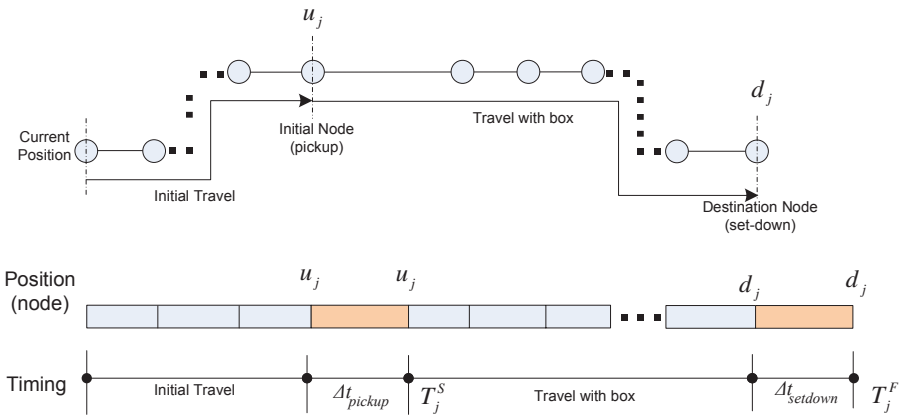


Figure 3-5: Typical event timings and timeline for an ASC performing a job.

It is this predefined time-critical job constraint which might cause the sequential job allocation method to insert unnecessary and unwanted ASC waiting times. Thus, it is necessary to avoid unnecessary ASC waiting times when planning jobs for ASCs.

3.4.2 Objective Function

This section presents the objective function which is used to calculate the overall cost of a particular schedule. The notations used for the indices, sets, parameters and variables in the mathematical formulation are defined in the Nomenclature. The overall objective function can be defined as:

$$\textit{Minimise} \qquad \sum_{v\in V}\sum_{s\in S}(\lambda_1 C_s^{TravelTime} + \lambda_2 C_s^{SC\_waiting} + \lambda_3 C_s^{QC\_waiting} + \lambda_4 C_s^{TK\_waiting} + \lambda_5 C_s^{HP\_finishing})X_{vs} \tag{3.4}$$

where $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$ are predefined parameters used to normalise the contributions from each function. These parameters provide a means to allocate the relative importance to each of the individual costs. Typically, λ_3 and λ_4 will have larger values since QC and TK waiting are of high importance and their contribution to the overall objective function must be amplified.

The cost of travel time ($C_s^{TravelTime}$) is used to reflect the utilisation of ASCs for a schedule and the cost is calculated based on Dijkstra's algorithm, which is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms. Basically the cost function sums up all the travel time of all the visited nodes by the ASC.

$$C_s^{TravelTime} = w_{p_v u_1} + w_{u_1 d_1} + w_{d_1 u_2} + \ldots + w_{d_{f-1} u_f} + w_{u_f d_f} \tag{3.5}$$

where p_v is the initial position of the ASC (v) which is assigned with the schedule s. f is the final job in the selected schedule s.

Vehicle waiting time ($C_s^{SC\_waiting}$) is an important performance metric which represents the efficiency of vehicle usage. In general, less waiting time indicates a better ASC usage and less fragmentation of the schedule. The calculation is based on the difference between planned timings (starting and finishing) and the theoretical shortest travel time.

$$\begin{aligned} C_s^{SC\_waiting} &= (T_1^S - w_{p_v u_1}) + (T_1^F - T_1^S - w_{u_1 d_1}) + (T_2^S - T_1^F - w_{d_1 u_2}) + \\ &(T_2^F - T_2^S - w_{u_2 d_2}) + \ldots + (T_f^S - T_{f-1}^F - w_{d_{f-1} u_f}) + (T_f^F - T_f^S - w_{u_f d_f}) \end{aligned} \tag{3.6}$$

QC waiting time ($C_s^{QC\_waiting}$) is a critical cost at container terminals and the following equation minimises the total waiting time. The QC waiting time can be calculated based on the last job of the QC. It is assumed that all QCs have a constant turnaround time for each job.

$$C_s^{QC\_waiting} = \sum_{j \in J_q^D, q \in Q} y_{js} \times (T_q^{DS} - t_q^D - m_q \times \Delta t^{QC}) + \sum_{j \in J_q^U, q \in Q} y_{js} \times (T_q^{UF} - t_q^U - n_q \times \Delta t^{QC}) \quad (3.7)$$

Similarly, TK waiting time ($C_s^{TK\_waiting}$) is an important consideration at container terminals and the following equation minimises the total waiting time. The TK waiting time can be calculated based on the last job in the TK gate. It is assumed that a constant turnaround time is applied for each job at the TK gate.

$$C_s^{TK\_waiting} = \sum_{j \in J_g^E, g \in G} y_{js} \times (T_g^{ES} - t_g^E - e_g \times \Delta t^{TK}) + \sum_{j \in J_g^I, g \in G} y_{js} \times (T_g^{IF} - t_g^I - r_g \times \Delta t^{TK}) \quad (3.8)$$

Occasionally there is a requirement for a particular job to be finished as early as possible. As such, it is regarded as a high priority job. The finishing time ($C_s^{HP\_finishing}$) of high priority jobs can be defined as:

$$C_s^{HP\_finishing} = \sum_{j \in J^H} T_j^F \times y_{js} \quad (3.9)$$

3.4.3 Constraints

This section describes the various constraints that need to be enforced as part of the analytical model.

3.4.3.1 Job Scheduling Constraints

For assigning jobs to ASCs within schedules, Eq(3.10) ensures that each job *j* is included in one and only one selected schedule *s* with the assigned ASC (*v*) while Eq(3.11) and Eq(3.12) ensure that each ASC (*v*) is assigned to only one selected schedule *s*.

$$\sum_{v \in V, s \in S} y_{js} X_{vs} = 1, \forall j \in J \quad (3.10)$$

$$\sum_{s \in S} X_{vs} = 1, \forall v \in V \quad (3.11)$$

$$X_{vs} \in \{0,1\}, \forall v \in V, \forall s \in S \quad (3.12)$$

For each scheduled job, its planned start time must precede the corresponding job finish time as shown in Eq(3.13).

$$T_j^S < T_j^F, \forall j \in J \tag{3.13}$$

Pickup and setdown job sequencing is required for buffer-to-ship jobs, some truck-to-yard jobs and some yard jobs that require multi-tiered stacking. Pickup sequencing requirements are expressed by Eq(3.14). Here, α_{ab} is a predefined parameter that indicates if two jobs (a,b) have a pickup sequencing requirement. Such that, $\alpha_{ab} \in \{0,1\} : \forall (a,b) \in J$. Let $\alpha_{ab} = 1$ if job (a) *must* be picked up before job (b), otherwise $\alpha_{ab} = 0$. Similarly, set-down sequencing operations are expressed in Eq(3.15). Here, β_{ab} indicates if two jobs (a,b) have a setdown sequencing requirement. Such that, $\beta_{ab} \in \{0,1\} : \forall (a,b) \in J$. Let $\beta_{ab} = 1$ if job (a) *must* be setdown before job (b), otherwise $\beta_{ab} = 0$.

$$\alpha_{ab} = 1, \forall (a,b) \in J \rightarrow T_a^S < T_b^S \tag{3.14}$$

$$\beta_{ab} = 1, \forall (a,b) \in J \rightarrow T_a^F < T_b^F \tag{3.15}$$

Eq(3.16) ensures that job starting times would not be less than the travel time from the initial position of assigned ASC (v) to the pick-up position of job *j*. Here, $\forall v \in V, \forall j \in J, \forall s \in S$.

$$y_{js} X_{vs} = 1, T_j^S \geq w_{p_v u_j} + t_0 \tag{3.16}$$

For each scheduled job, its planned start time must precede the corresponding job finish time as shown in Eq(3.17).

$$T_j^F - T_j^S \geq w_{u_j d_j}, \forall j \in J \tag{3.17}$$

Eq(3.18) ensures that the time difference between starting and finishing a job is not less than travel time from the job pick-up to set-down position. Here,
$T_a^S < T_b^S, \forall (a,b) \in J, \forall s \in S$.

$$y_{as} y_{bs} = 1, T_b^S - T_a^F \geq w_{d_a u_b} \tag{3.18}$$

Each ASC cannot conduct other jobs when performing an assigned job, so Eq(3.19) ensures that an ASC can only transfer one container at the same time. Here, $\forall(a,b)\in J, \forall s\in S$.

$$y_{as}y_{bs}=1, T_a^F < T_b^S \vee T_b^F < T_a^S \tag{3.19}$$

Eq(3.20)-Eq(3.23) ensure that any two jobs associated with the same QC or TK should take into account the related turnaround time of QC or TK and any QC or TK job should not start or finish earlier than the related starting time of QC or TK.

$$\left|T_a^S - T_b^S\right| \geq \Delta t^{QC}, T_a^S \geq t_q^D + \Delta t^{QC}, \forall(a,b)\in J_q^D, \forall q\in Q \tag{3.20}$$

$$\left|T_a^F - T_b^F\right| \geq \Delta t^{QC}, T_a^F \geq t_q^U + \Delta t^{QC}, \forall(a,b)\in J_q^U, \forall q\in Q \tag{3.21}$$

$$\left|T_a^S - T_b^S\right| \geq \Delta t^{TK}, T_a^S \geq t_g^E + \Delta t^{TK}, \forall(a,b)\in J_g^E, \forall g\in G \tag{3.22}$$

$$\left|T_a^F - T_b^F\right| \geq \Delta t^{TK}, T_a^F \geq t_g^I + \Delta t^{TK}, \forall(a,b)\in J_g^I, \forall g\in G \tag{3.23}$$

Eq(3.24) is applicable to ASCs and ensures that the assigned job's pickup node (u_j) and setdown node (d_j) are specified in the planned trajectory. In addition, that the position $P_v(t)$ of ASC (v) corresponds to the pickup node at (T_j^S) and the destination node at (T_j^F) for job (j). If a box associated with job (j) has been picked up by an ASC then u_j is the ASC's initial position.

$$\forall v\in V, \forall j\in J, \forall s\in S: y_{js}X_{vs}=1 \rightarrow \left(P_v\left(T_j^S\right)=u_j \wedge P_v\left(T_j^F\right)=d_j\right) \tag{3.24}$$

3.4.3.2 Container Alignment Constraints

Container alignment (or orientation) is important to ensure that the door of the container is correctly aligned during transport, loading and storage. As such, the door alignment (σ) at the destination node must be as same as the required door alignment. Eq(3.25) ensures that the required setdown door direction is consistent with the final setdown door direction of the trajectory planning.

$$\forall j \in J, \sigma_j^F = \sigma_j^R \tag{3.25}$$

To guarantee container alignment at the destination node, changes in alignment during transportation are modelled using a concept called *flip movements* as a container is transported from its initial node to destination node. Eq(3.26) and Eq(3.27) ensure that container alignment changes are valid within trajectories:

$$\begin{aligned} &\forall j \in J, \forall v \in V, \forall s \in S, y_{js} X_{vs} = 1 : \ \sigma_j^S = \sigma_j^F \rightarrow \\ &\exists \{(p_{vx}, t_{vx}^{Arrive}, t_{vx}^{Depart}), (p_{vy}, t_{vy}^{Arrive}, t_{vy}^{Depart})\} \subset \delta_v, \\ &T_j^S \in (t_{vx}^{Arrive}, t_{vx}^{Depart}), T_j^F \in (t_{vy}^{Arrive}, t_{vy}^{Depart}), \\ &x \le y-1, (\sum_{i=x}^{i=y-1} f_{p_{vx} p_{vy}}) MOD\ 2 = 0 \end{aligned} \tag{3.26}$$

$$\begin{aligned} &\forall j \in J, \forall v \in V, \forall s \in S, y_{js} X_{vs} = 1 : \ \sigma_j^S \neq \sigma_j^F \rightarrow \\ &\exists \{(p_{vx}, t_{vx}^{Arrive}, t_{vx}^{Depart}), (p_{vy}, t_{vy}^{Arrive}, t_{vy}^{Depart})\} \subset \delta_v, \\ &T_j^S \in (t_{vx}^{Arrive}, t_{vx}^{Depart}), T_j^F \in (t_{vy}^{Arrive}, t_{vy}^{Depart}), \\ &x \le y-1, (\sum_{i=x}^{i=y-1} f_{p_{vx} p_{vy}}) MOD\ 2 = 1 \end{aligned} \tag{3.27}$$

where, σ_j^S is a parameter which is the initial alignment associated with the destination of job (j). δ_v is the planned trajectory of ASC (v) and associated timings for arrival t_k^{Arrive} and departure t_k^{Depart} at each node (p_{vk}) in the trajectory. f_{ij} is the flip flag associated with each physical link. The flip flag is used to track the alignment of a container during traversal of the link. The flip flag of each link is encoded into the map. Specifically, Eq(3.26) ensures that, when the container's door alignments at initial position and destination are the same, the total number of flip movements in the planned path should be even. Likewise for Eq(3.27), when the container's door alignments at initial position and destination are different, then the total number of flip movements in the planned path should be odd.

3.4.3.3 Vehicle Motion Constraints

Eq(3.28) ensures that collision avoidance is guaranteed during the path planning of all ASCs in the fleet:

$$\forall t \ge t_0 : P_i(t) \notin \bigcup_{j=1; i \neq j}^{|V|} N^{lock}(P_j(t)) \tag{3.28}$$

where, $P_i(t)$ is the position of ASC ($i \in V$) at time (t), and $N^{lock}(P_j(t))$ represents sets of locked nodes and links at time (t) by ASC ($j \in V$) based on its position $P_j(t)$ so as to prevent any ASC collisions.

Eq(3.29) and Eq(3.30) ensure that ASCs perform feasible actions, such as pickup and setdown on a node.

$$\forall j \in J, \forall v \in V, \forall s \in S, y_{js} X_{vs} = 1 \rightarrow t^{Depart}_{P_v(T_j^S)} - t^{Arrive}_{P_v(T_j^S)} \geq \Delta t_{pickup} + \Delta t_{acc} + \Delta t_{dec} \tag{3.29}$$

$$\forall j \in J, \forall v \in V, \forall s \in S, y_{js} X_{vs} = 1 \rightarrow t^{Depart}_{P_v(T_j^F)} - t^{Arrive}_{P_v(T_j^F)} \geq \Delta t_{setdown} + \Delta t_{acc} + \Delta t_{dec} \tag{3.30}$$

where, Δt_{pickup} and $\Delta t_{setdown}$ are the time required for an ASC (v) to pick up and set down a container respectively. Also, Δt_{acc} and Δt_{dec} are the time associated with acceleration and deceleration respectively. Specifically, Eq(3.29) and Eq(3.30) requires that, if an ASC is allocated a job, then the ASC have to spend the necessary amount of time picking up or setting down a container at a position within the planned trajectory. Eq(3.31) handles the situation when an ASC needs to perform a reversing or 3-point turning manoeuvre.

$$\begin{aligned} &\forall v \in V, \forall (p_{vm}, p_{v(m+1)}, p_{v(m+2)}) \subset \delta_v : \psi_{m(m+1)(m+2)} = 1 \\ &\rightarrow t^{Arrive}_{P_{v(m+2)}} - t^{Depart}_{P_{vm}} \geq w_{P_{vm}P_{v(m+1)}} + w_{P_{v(m+1)}P_{v(m+2)}} + \Delta t_{dec} + \Delta t_{acc} \end{aligned} \tag{3.31}$$

where $w_{P_{vm}P_{v(m+1)}}$ is the predefined minimum travel time between node p_m and node p_{m+1}. Likewise, $w_{P_{v(m+1)}P_{v(m+2)}}$ is for node p_{m+1} and node p_{m+2}. This constraint ensures that the necessary time for acceleration and deceleration is taken into account when the ASC is performing a reversing or 3-point turning manoeuvre.

As an example, Figure 3-6 illustrates an ASC path from the pickup node to the setdown node.

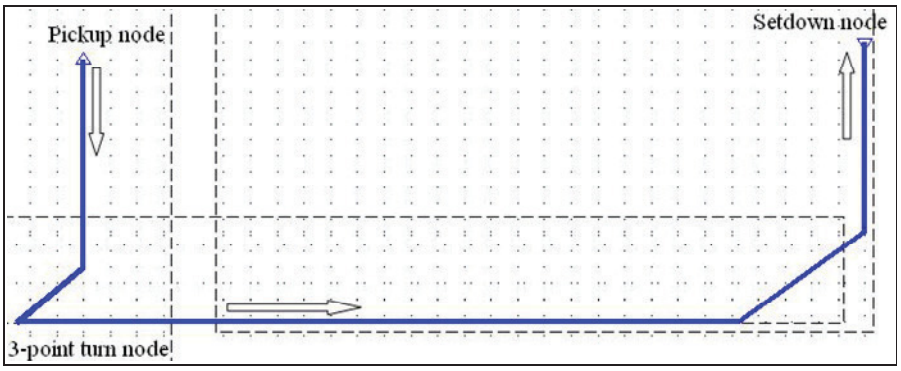


Figure 3-6: A planned path from pickup node to setdown node consiting of 56 nodes. A 3-point turn occurs at the 16th node (nodes are not uniformly spaced in the map).

Now, Figure 3-7 shows the motion profile applied to the example path in Figure 3-6. Here, we can see that during the first 7 nodes the velocity increases from 0ms^{-1} to the maximum velocity (8.3ms^{-1}). Then, deceleration occurs as the ASC approaches the pickup node and performs a three-point turn at node 16, where the velocity is 0ms^{-1}. The ASC then accelerates to maximum velocity and travels to the setdown node where it begins deceleration at node 49 and stops at node 56 to perform the box setdown action.

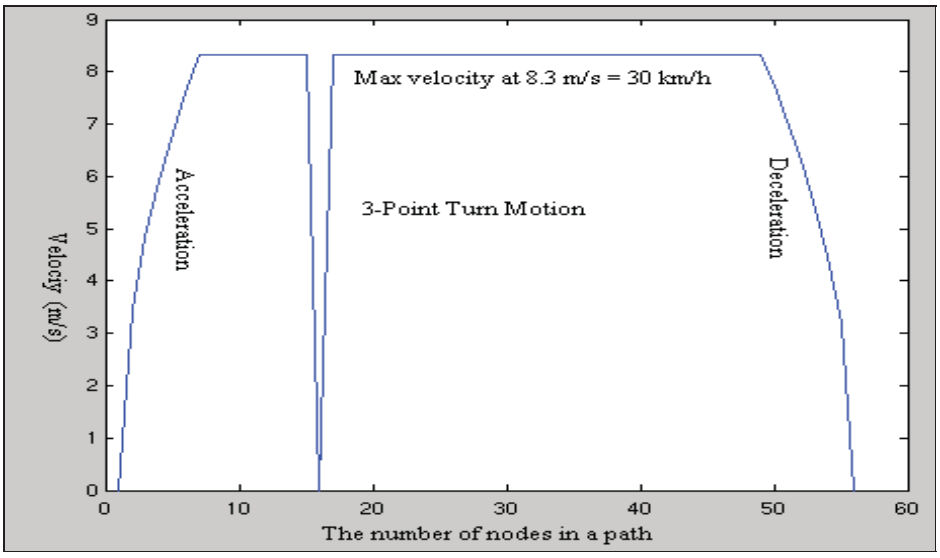


Figure 3-7: Example of a motion profile, which is applied to a planning path for an ASC (nodes are not uniformly spaced in the map).

There are two situations where the velocity of the ASC changes during the turning motion, which are associated with the definition of ψ_{abc} in constraint Eq(3.31). Figure 3-8 shows the two situations: 3-point turn motion ($0^o \le \theta < 90^o$) and regular turn motion ($90^o \le \theta \le 180^o$). In the first case, a straddle must reduce its velocity between node n_1 to

node n_2. At n_2 the velocity must be zero. The straddle then accelerates from node n_2 onwards. In the second case, the straddle can traverse the turn without making any changes to its maximum velocity, because the turning angle (θ) is greater than 90^o.

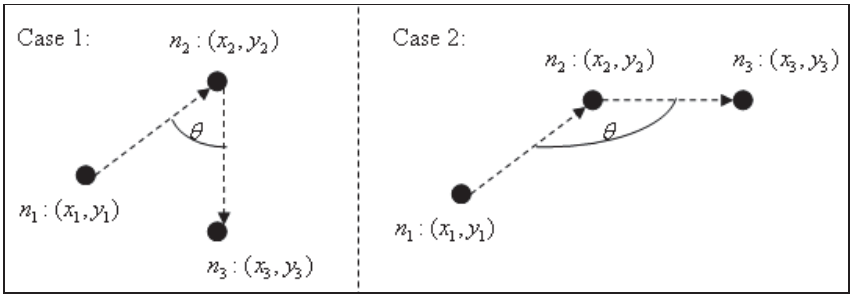


Figure 3-8: Classification of a 3-point turn (case 1) and regular turning motion (case 2) for motion planning.

Finally, Eq(3.32) ensures that the ASC's total travel time is met when travelling between the nodes in a trajectory.

$$\forall v \in V, \forall (p_{vm}, p_{v(m+1)}) \subset \delta_v : t^{Arrive}_{v(m+1)} - t^{Depart}_{vm} \geq w_{p_{vm} p_{v(m+1)}} \tag{3.32}$$

This constraint guarantees that all essential travel costs are taken into account by the path planner when computing feasible paths.

3.5 **Model 2: Job Scheduling Model**

As presented above, the comprehensive model covers many actual aspects and integrates the path planning with collision avoidance and the job scheduling. However, compared to the path planning, the job scheduling is of more concern in some situations, due to the following reasons:

(1) Computational efficiency. When solving a large problem in the comprehensive model, the computation cost may be prohibitively increased, and it is also challenging to obtain an optimal solution in the limited time period afforded during the port operation.
(2) Operational interest. Production managers of the automated container terminal are usually more concerned with the job scheduling part, as resource utilisation and port costs are more dependent on the job scheduling.

As explained before, Eq(3.24)-Eq(3.32) are related to many practical challenges, including the presence of multiple levels of container stacking and sequencing, variable container orientations, and vehicular dynamics that require finite acceleration and deceleration times. By taking them out of the comprehensive model formulation, the job scheduling model for container transfers by ASCs can be thereby be presented as follows:

Minimise

$$\sum_{v\in V}\sum_{s\in S}(\lambda_1 C_s^{TravelTime}+\lambda_2 C_s^{SC\_waiting}+\lambda_3 C_s^{QC\_waiting}+\lambda_4 C_s^{TK\_waiting}+\lambda_5 C_s^{HP\_finishing})X_{vs} \tag{3.4}$$

subject to:

$$\sum_{v\in V}\sum_{s\in S} y_{js}X_{vs}=1,\forall j\in J \tag{3.10}$$

$$\sum_{s\in S} X_{vs}=1,\forall v\in V \tag{3.11}$$

$$X_{vs}\in\{0,1\},\forall v\in V,\forall s\in S \tag{3.12}$$

$$T_j^S<T_j^F,\forall j\in J \tag{3.13}$$

$$\alpha_{ab}=1,\forall(a,b)\in J\rightarrow T_a^S<T_b^S \tag{3.14}$$

$$\beta_{ab}=1,\forall(a,b)\in J\rightarrow T_a^F<T_b^F \tag{3.15}$$

$$y_{js}X_{vs}=1,\forall v\in V,\forall j\in J,\forall s\in S\rightarrow T_j^S\geq w_{p_v u_j}+t_0 \tag{3.16}$$

$$T_j^F-T_j^S\geq w_{u_j d_j},\forall j\in J \tag{3.17}$$

$$y_{as}y_{bs}=1,T_a^S<T_b^S,\forall(a,b)\in J,\forall s\in S\rightarrow T_b^S-T_a^F\geq w_{d_a u_b} \tag{3.18}$$

$$y_{as}y_{bs}=1,\forall(a,b)\in J,\forall s\in S\rightarrow T_a^F<T_b^S\vee T_b^F<T_a^S \tag{3.19}$$

$$\left|T_a^S-T_b^S\right|\geq\Delta t^{QC},T_a^S\geq t_q^D+\Delta t^{QC},\forall(a,b)\in J_q^D,\forall q\in Q \tag{3.20}$$

$$\left|T_a^F-T_b^F\right|\geq\Delta t^{QC},T_a^F\geq t_q^U+\Delta t^{QC},\forall(a,b)\in J_q^U,\forall q\in Q \tag{3.21}$$

$$\left|T_a^S-T_b^S\right|\geq\Delta t^{TK},T_a^S\geq t_g^E+\Delta t^{TK},\forall(a,b)\in J_g^E,\forall g\in G \tag{3.22}$$

$$\left|T_a^F-T_b^F\right|\geq\Delta t^{TK},T_a^F\geq t_g^I+\Delta t^{TK},\forall(a,b)\in J_g^I,\forall g\in G \tag{3.23}$$

The objective function is the same as the comprehensive model's objective, and all the constraints are from the job scheduling constraints in Section 3.4.3.1.

3.6 Matlab Seaport Simulation

The comprehensive model has been coded in Matlab with all variables and all constraints, while the collision-free path planner described in Lau *et al* (Lau et al., 2008), which is based on the Halpern's Algorithm (Halpern, 1977) was coded exclusively in C++ and interfaced with the Matlab model. The model was simulated using fleet sizes of (4, 8, 12, 16, 20) ASCs, a job-horizon of (2, 3, 4, 5) jobs and a total of 100 jobs. The main purpose of the Matlab simulation is to validate the mathematical model in case it misses any constraints or requirements. Regarding the proposed algorithms, they have been implemented in the Matlab simulator. However, the Matlab simulator is much simpler than the real system, so all the experiments are based on the real system rather than the simulator.

3.6.1 Matlab Seaport Simulator

To validate the efficacy of the formulated model and verify the feasibility of solutions, a simple algorithm was developed to plan and schedule jobs sequentially using a greedy heuristic based on nearest-vehicle-first strategy (Figure 3-9). However, this approach does not guarantee optimal job allocation, since the current path planner computes paths sequentially using the existing time-windows for each planned path.

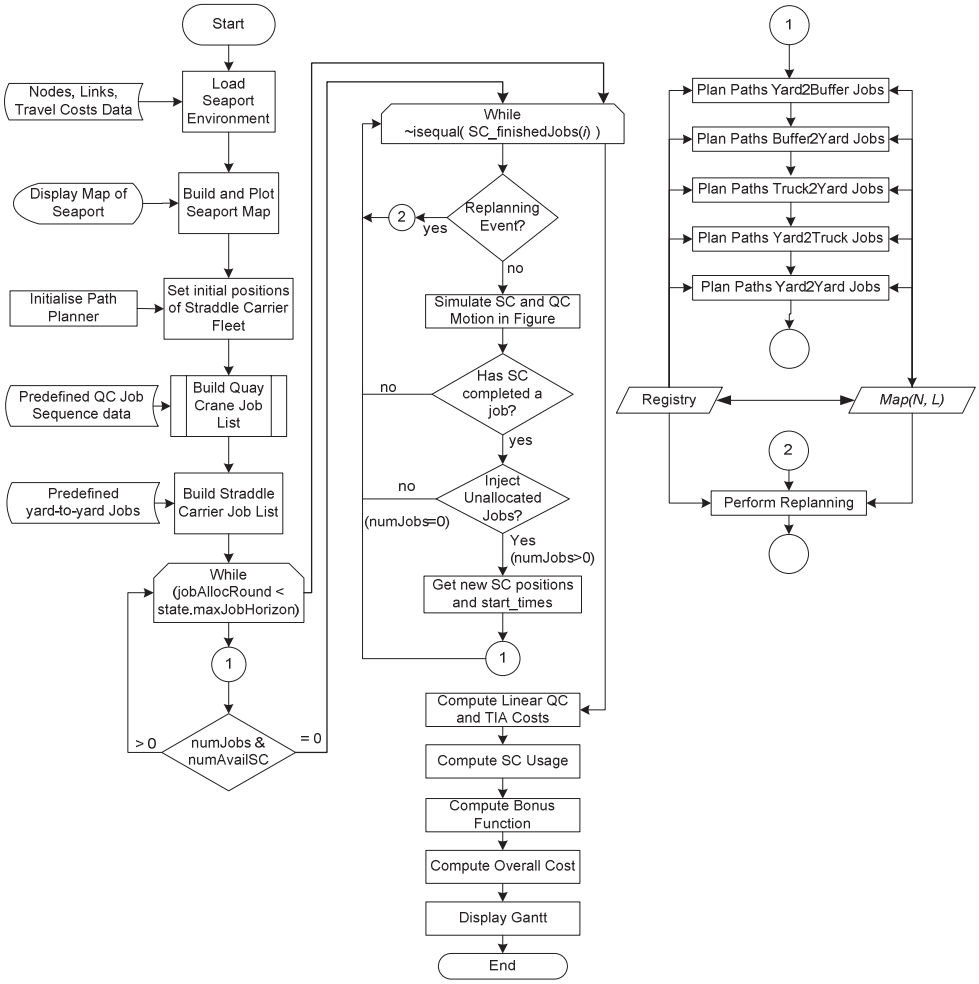


Figure 3-9: Flow diagramming for sequential job allocation and job injection approach based on a greedy nearest-vehicle-first heuristic.

3.6.2 Replanning in the Seaport Simulator

Replanning events are examined following the computation of the initial schedule, as shown in Figure 3-9. Replanning requires the state-tracking of all jobs, seaport resources and containers before initiating the path planning algorithm to provide planning for unfinished and replanned jobs as shown in Figure 3-10.

In this study, a single replanning event occurs at ($t = 1000$ time steps) and no replanning ($t = \infty$). This allows for verification of replanned schedules and validation of the replanning algorithm. Although our testing was restricted to a single replanning event, the replanning algorithm can be initiated at anytime in the seaport simulator, which more realistically models the dynamic nature of the seaport environment. Furthermore,

frequent replanning of schedules with short job horizons supports the notion of transforming an intractable scheduling problem which may be based on a day-long job horizon into a problem that is significantly more tractable.

Finally, a limitation of the replanning algorithm is the requirement that all seaport resources (ASCs and QCs) should only stop on nodes. If a resource stops on a link, then an additional control heuristic must move the resource onto a node in the real seaport environment.

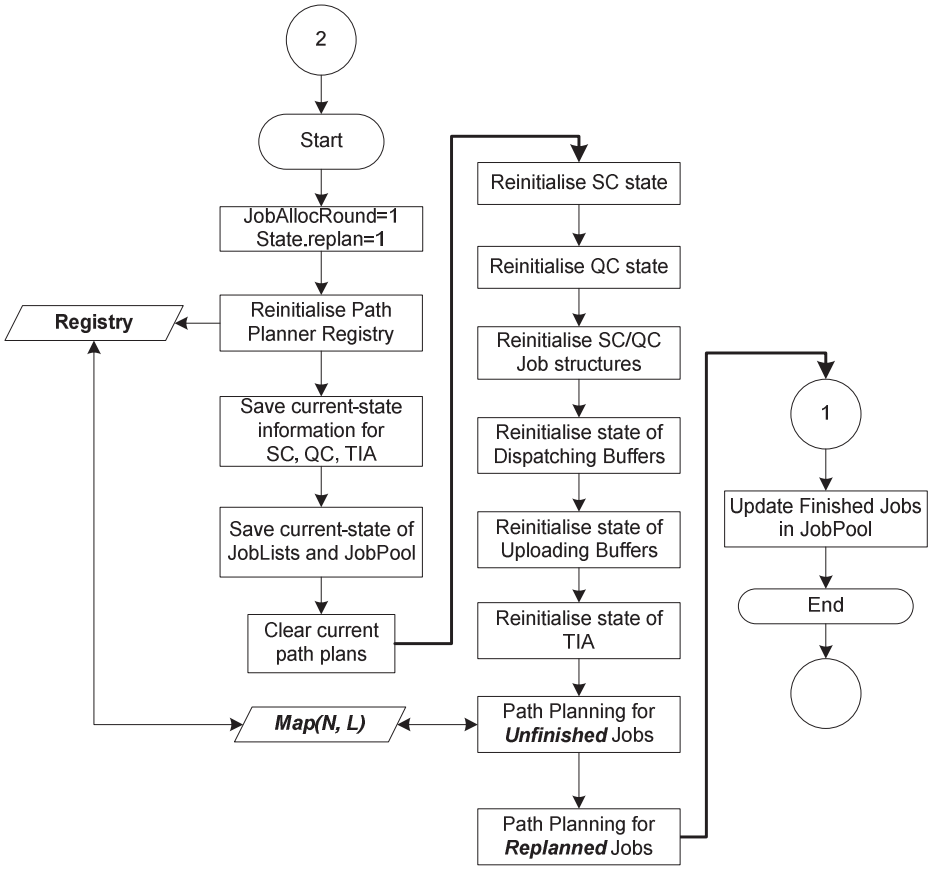


Figure 3-10: Flow diagramming which illustrates the replanning processes within the seaport simulator.

3.6.3 Model Validation within the Seaport Simulator

In order to validate the analytical model which was encoded in the Matlab seaport simulator, we employed a hierarchical assertion structure throughout the code for checking all the model constraints. This hierarchical approach ensures that any

constraint violations are flagged at the functional, subsystem and system level of the entire software system.

At the lowest level, each individual function contains a number of assertion checking routines for any associated model constraints that are encoded by that particular function. For example, the function for planning a single path (`PlanSinglePath`), before applying motion profiling must not violate the following constraints Eq(3.24) and Eq(3.28)-Eq(3.32). At the next level, additional constraint checking is performed on the combined or integrated set of functions. For example, the function (`ProcessY2Tstate`) takes the result of a planned single path to handle higher level job sequencing for yard-to-truck jobs. As such, the constraints Eq(3.10)-Eq(3.15), and Eq(3.22) must not be violated at this level.

Finally, a suite of assertions perform the final level of constraint checking of the combined subsystems of the seaport simulator as illustrated in Figure 3-11.

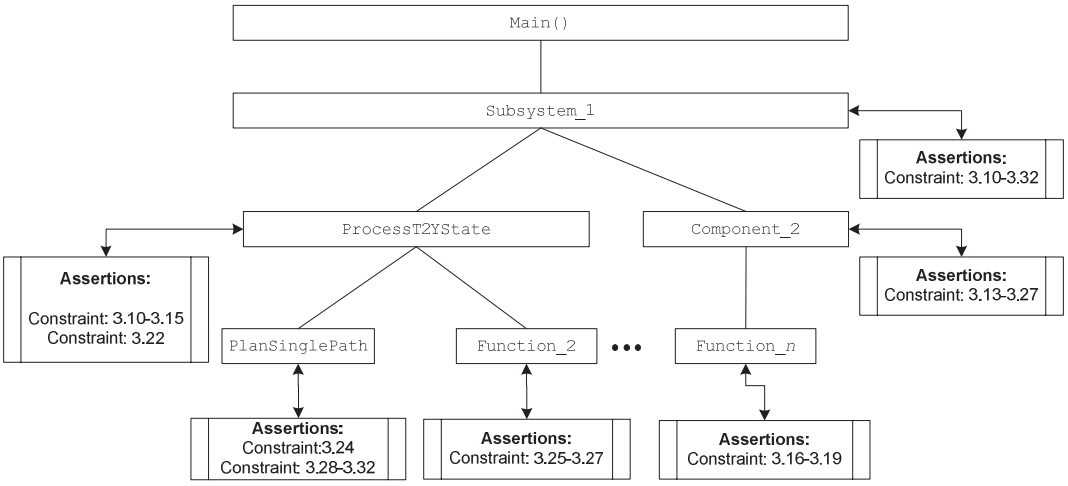


Figure 3-11: Model Validation Method – Hierarchical assertion checking within the Matlab Seaport Simulator ensures model constraints are not violated.

3.7 **Summary**

This chapter formulated two mathematical models of the automated container. The comprehensive model covers the path planning and the job scheduling, and many practical constraints, while the job scheduling model focuses on the optimisation of the job scheduling for container transfers. The comprehensive model is more complex, and

aims to integrate the path planning and job scheduling. In order to check and validate all constraints, a Matlab simulator was developed with a simple sequential solution. The job scheduling model is derived from the comprehensive model, and it aims to achieve a good solution for job scheduling that is purely based on operational considerations. The following chapters will introduce the related solution techniques for the comprehensive model and the job scheduling model respectively.

4. Job Grouping Approach for Planning Container Transfers

Having taken into account many practical constraints in the automated container terminal, Chapter 3 formulated the comprehensive model of container transfers by ASCs. The model covers path planning with collision avoidance, job scheduling, the presence of multiple levels of container stacking and sequencing, variable container orientations, and vehicular dynamics. The solution of the overall planning problem includes the allocation of jobs and a detailed trajectory of each ASC, which includes the position and status (such as picking up a container, setting down a container, travelling with a container, or empty travelling) of each ASC at each time step. With all the defined constraints, it becomes challenging to figure out an efficient or even feasible plan.

This chapter proposes a practical job grouping approach which aims to solve the integrated problem for better performance than the widely used sequential planning approach. It reduces ASC waiting time by grouping jobs using a guiding function. The performance of the current sequential job planning method and the proposed job grouping approach are evaluated and compared statistically using a pooled *t*-test for 30 randomly generated yard configurations.

4.1 Motivation

Generally, job grouping is viewed as an optimisation mechanism to enhance efficiency of machinery utilisation as mentioned in (Crama and Oerlemans, 1994, Logendran et al., 2005). In (Crama and Oerlemans, 1994), the job grouping problem was to partition the jobs into a minimum number of feasible groups. Since the number of variables is potentially huge, the authors used a column generation approach. The study in Longendran *et al* (Logendran et al., 2005) investigated the group scheduling problem which is comprised of two levels of scheduling, and aimed to minimise the makespan in a flexible flow shop. Another study examined the use of a typical job grouping strategy for the allocation of jobs in a grid computing application (Muthuvelu et al., 2005). The

job grouping approach resulted in improved performance in terms of low processing time and costs when applied to a large number of jobs where each user job holds small processing requirements.

This thesis is in favour of job grouping to adequately handle time-critical requirements for the scheduling of container transfers in the yard environment as these serve to reduce the ASC waiting time and enhance the productivity of the ASC fleet. It would appear that, to date, the job grouping concept has not been studied in the scheduling of container transfers at an automated seaport terminal.

In this chapter, collision avoidance trajectory planning is used together with the job grouping approach to solve the container transfer problem. The collision-free path planner is a prioritised multi-vehicle path planning algorithm (Lau et al., 2008). This path planning algorithm is extended from Halpern's algorithm (Halpern, 1977) to propagate feasible time and cost windows for each vehicle to arrive at and depart from each position (subject to the time-dependent position's availability). Such time windows are propagated iteratively from the known starting time of the vehicle at the starting position until a feasible arrival window is found at the destination. The key feature of this algorithm is that the paths generated will consider the motion of all other active ASCs and, as a result, will go around or give way (via waiting at a position or shunting aside and subsequently resuming) to ASCs with already planned paths. Importantly, this path planning approach is more realistic than simplified VRPs where path lengths only need to be calculated once, regardless of the changing occupancy of the various positions in the environment.

4.2 Job Grouping Approach

The key advantage of the job grouping strategy is that a plan can be more efficient, and it also can lead to cost savings (e.g. waiting time) for planning, since job groups may accomplish more to meet the objectives effectively. The basic principle of job grouping is to encourage ASCs to conduct some yard consolidation jobs instead of just waiting before or after performing a QC or TK related job. For each QC and TK related job, there are predefined pickup or setdown times which allow the path planner to avoid

introducing delays to QC and TK operations. In general, ASCs are encouraged to arrive at the pickup node or setdown node before the predefined time. However, unnecessary waiting is caused when ASCs arrive at the pickup/setdown nodes earlier than the predefined action time. As a result, the total schedule duration is stretched to some extent and can have negative impacts on the overall productivity. Nevertheless, it is possible to cluster jobs into groups that may have smaller total durations and also produce less ASC waiting time through the effective grouping of some *Y2Y* jobs to QC and TK related jobs. This strategy is achieved using a job grouping algorithm with an associated guiding function.

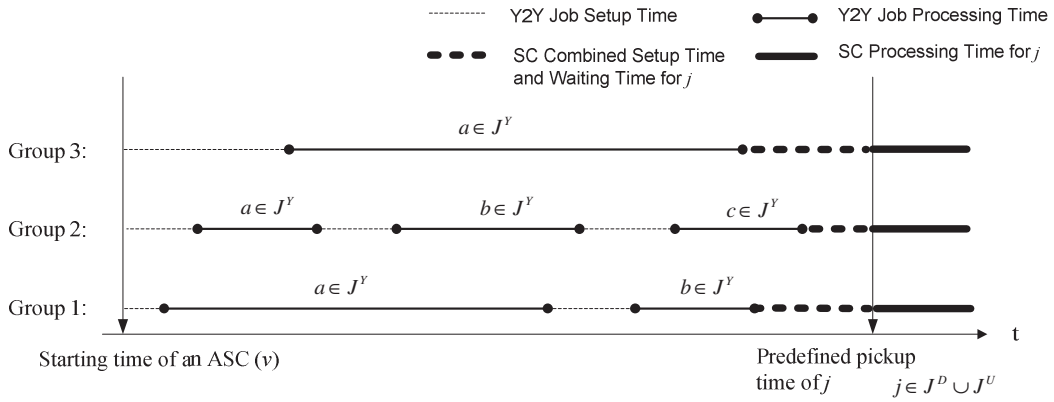


Figure 4-1: An example of job grouping.

As the example shown in Figure 4-1, an ASC (v) is trying to group *Y2Y* jobs (J^Y) before starting to perform the pickup of job (j), which can be a QC discharging or uploading job and has a predefined pickup time. The job setup time refers to the time required for an ASC to travel from its current node to the pickup node while the job processing time is the time required for an ASC to travel from the pickup node to the setdown node. In this example, there are three possible job groupings (*1, 2, 3*), which all meet the predefined pickup time of the job (j). Yet, it is necessary to have some criteria to tell which job group is the best.

Due to the different time/distance length of jobs, a simple grouping strategy is to select the group which can accumulate the greatest number of jobs. However, this approach can cause short-distance jobs to be grouped in preference to long-distance jobs. This may not effectively benefit the schedule, since long jobs would be held-over until the end of the schedule. That is, the finishing time may be delayed by scheduling long-

distance jobs at the end of the schedule. Furthermore, this may reduce the robustness of a schedule, since scheduling with bias towards short-distance jobs over long-distance jobs can make the schedule more vulnerable to uncertainties such as ASC breakdowns and new jobs arrivals. Therefore, we propose a guiding function which treats short-distance jobs and long-distance jobs equally and aims to effectively improve the schedule robustness.

4.2.1 Guiding Function of Job Grouping

Primarily, the guiding function aims to encourage early starting and finishing times for *both* short-jobs and long-jobs. The length of a job (short or long) refers to the processing duration required between pickup and setdown of a container. In general, a greedy-rule based strategy may complete all the jobs with short durations (short jobs) in preference to long jobs (jobs with long durations). However, such a biased strategy does not provide robust schedules, since the accumulation of jobs towards the end of a schedule can actually delay all future job allocations particularly if the long job is also delayed, as discussed before.

The guiding function for a group of jobs is given by:

$$f^G = \sum_{j \in J_v^G} f_j^G = \sum_{j \in J_v^G} \left(\int_{T_j^S}^{T_j^F} \frac{\varepsilon_j}{T_j^F - T_j^S} \times \frac{1}{\sqrt{t - t_0}} dt \right) \tag{4.1}$$

Where, $J_v^G \subset J^Y$ is a potential cluster of jobs grouped by (v). ε_j is the minimum theoretical processing time based on Dijkstra's algorithm (from pickup to setdown) for the job (j). $\frac{\varepsilon_j}{T_j^F - T_j^S}$ indicates the efficiency of the planned path for j and $\frac{1}{\sqrt{t - t_0}}$ encourages early starting and finishing of j, where t_0 is the starting time of the plan. For a job (j), equating the integral from T_j^S to T_j^F gives:

$$f_j^G = \int_{T_j^S}^{T_j^F} \frac{\varepsilon_j}{T_j^F - T_j^S} \times \frac{1}{\sqrt{t - t_0}} dt = \frac{\varepsilon_j \times 2(\sqrt{T_j^F - t_0} - \sqrt{T_j^S - t_0})}{T_j^F - T_j^S} \tag{4.2}$$

The following figures show the behaviour of the guiding function for different job starting times and job finishing times when $T_j^F - T_j^S$ = *constant* and T_j^S = *constant*, respectively.

Assume $T_j^F - T_j^S$ = *constant=10 sec* and ε_j = *8 sec*, then the guiding function is calculated with different starting times. As shown in the graph of Figure 4-2, as the job starting time T_j^S increases the nominal value of the grouping value decreases. Therefore, the guiding function (f_j^G) encourages the job to be started early.

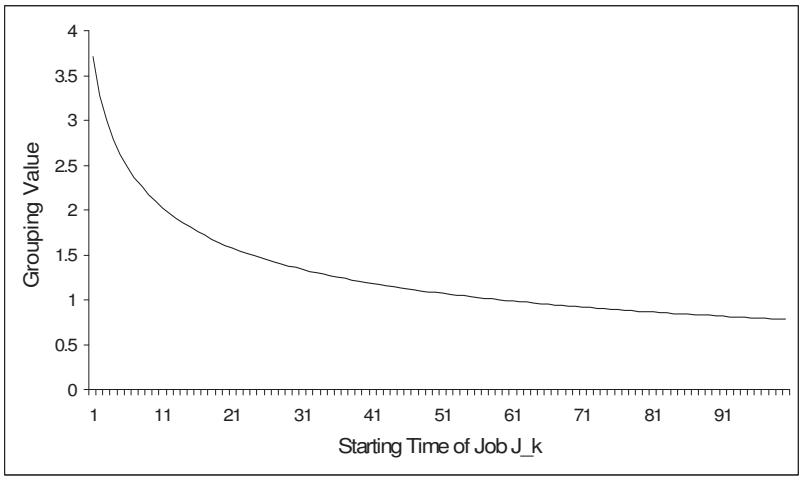


Figure 4-2: Plot of Eq(4.2) for job starting times (sec).

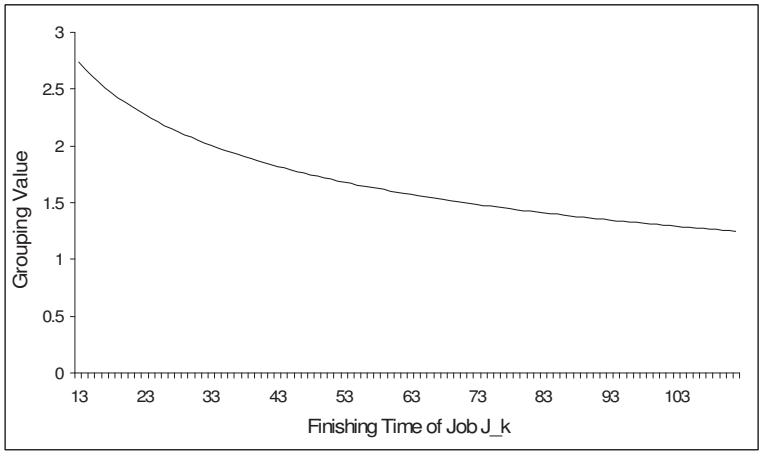


Figure 4-3: Plot of Eq(4.2) for job finishing times (sec).

Assume T_j^S =*constant* = *5 sec* and ε_j = *8 sec*, then the guiding function is calculated with different finishing times. From the plot in Figure 4-3, as the job finishing time T_j^F increases the nominal value of the grouping value decreases. Therefore, the guiding function f_j^G allows for the job to be finished early. Furthermore, the guiding function

can be altered to suit different yard configurations and job features. In this thesis, the form that the guiding function takes is based on a typical port configuration.

Regarding the intuition of the guiding functions, , it basically, as presented earlier, wants to promote the early starting and finishing times, so as to maximise the productivity, because the accumulation of jobs towards the end of a schedule could actually delay all future job allocations particularly if the long job is also delayed. The key consideration is that a proper guiding function should be based on the defined objective function.

4.2.2 Grouping Algorithm

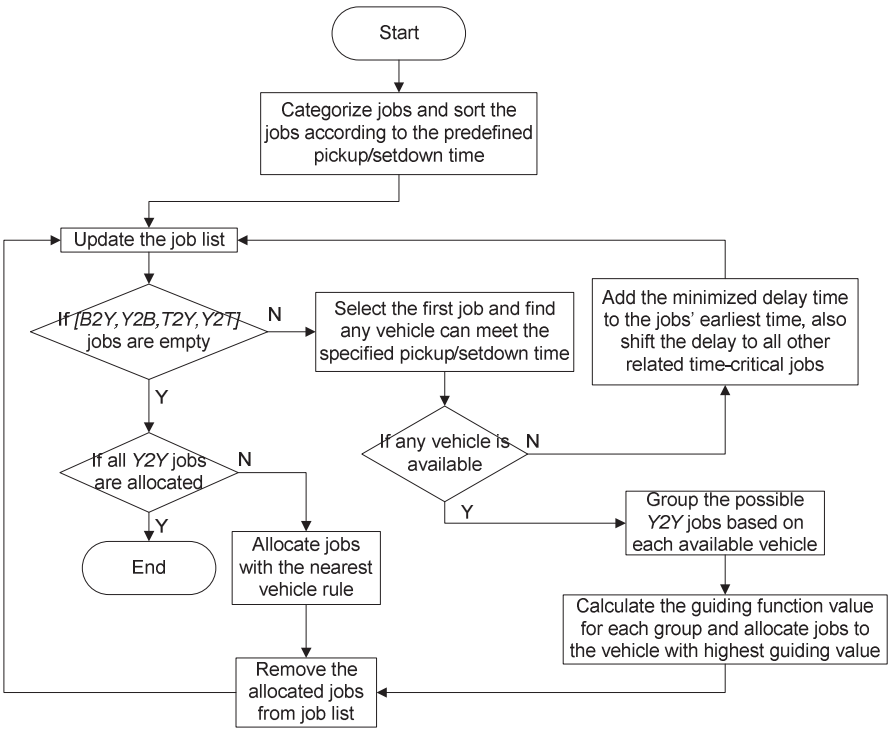


Figure 4-4: Flowchart of job grouping method

The proposed grouping method is to group yard jobs with corresponding QC or TK jobs according to the guiding function, so as to reduce the ASC waiting time. As shown in Figure 4-4, the job grouping algorithm starts by categorising jobs into two categories: *[B2Y,Y2B,T2Y,Y2T]* jobs and *Y2Y* jobs. The first section of the job list is then sorted according to the predefined pickup/setdown time. Next, if any QC and TK jobs exist, then select the first job and search for ASCs which are able to perform the job while satisfying the predefined pickup/setdown time of *the selected job*. All suitable ASCs are

added to a selection pool. If the selection pool is empty, then calculate the minimum delay based on the job's nearest ASC and put the nearest ASC to the selection pool. Next, for each ASC in the selection pool, search for *Y2Y* jobs that can be finished by the ASC while satisfying the predefined setdown/pickup time of *the selected job*. Then, calculate the guiding function using Eq(4.1) for each ASC in the pool and select the ASC with the highest guiding function value. This ASC is then assigned to the grouped job(s). Finally, remove all assigned jobs from the job list, plan and register the path(s) for the ASC and the allocated jobs. The algorithm continues to iterate through these steps until the job list is empty. If there are only *Y2Y* jobs remaining after corresponding QC and TK jobs have been allocated, then allocate those jobs based on the nearest vehicle rule.

The pseudo code of the job grouping algorithm is:

Job Grouping Algorithm for Scheduling Container Transfers:

Input: Job list (*J*) and the initial position (p_v^I) of each ASC.

Result: A plan for all jobs and all ASCs.

1: Initialise the job list by getting all job pickup nodes, setdown nodes and predefined pickup/setdown times if any; initialise all ASCs' starting positions;
2: Reformulate the job list by categorising jobs into (*B2Y, Y2B, T2Y, Y2T*), and (*Y2Y*). Sort the new job list according to the predefined pickup/setdown time if any;
3: **for** k=1:M // M is the total number of QC and TK jobs
4: **for** i=1:|V| // |V| is the total number of ASCs
Calculate readyTime(v, j) by calling path planner (Lau et al., 2008);
5: **if** readyTime(v, j) < predefined time of job (j)
Find the number of *Y2Y* jobs that can be grouped by calling the path planer;
Calculate the guiding function value of grouped jobs;
Put the ASC into selection pool;
end
end
6: **if** the selection pool is empty
add the minimum delay to the predefined time of job (j);
update all other job time intervals;
repeat steps 4 and 5;
7: **else**
select the ASC which has the highest guiding function value and allocate the grouped job and job (j) to the ASC;
Plan the associated paths and register them;
Update the ASC's available time for the next iteration;
Remove the assigned job from the job list;
end
end
8: **while** Y2Y is not empty
allocate the jobs to ASCs based on nearest vehicle rule;
end

The job grouping algorithm implementation has been coupled with our path planner (Lau et al., 2008), which aims to find the shortest paths for multiple vehicles with collision avoidance. One advantage of this grouping approach is that delay times can be effectively reduced for important jobs such as QC and TK related jobs, provided there are enough ASCs available. Moreover, the total ASC waiting time is reduced by grouping *Y2Y* jobs with the guiding function, and consequently the robustness and productivity of a schedule can effectively be improved.

4.3 **Sequential Job Planning Approach**

Sequential job allocation is often considered as a typical solution in a manufacturing system that deals with many different components, particularly for the job scheduling problem with priority constraints. In order to prevent QC waiting and TK waiting, sequential job allocation with action on the nearest vehicle is a practical method to handle the scheduling problem.

This approach aims to ensure QC related jobs and TK related jobs have the higher priority of scheduling than *Y2Y* jobs. Initially all jobs are sorted according to their predefined pickup/setdown times. At each iteration of the algorithm, the first job in the list is allocated to its nearest vehicle so as to reduce delay for QCs or TKs, and also minimise ASC travel time. The allocated job is removed from the job list and the algorithm iterates until the job list is empty as illustrated in Figure 4-5.

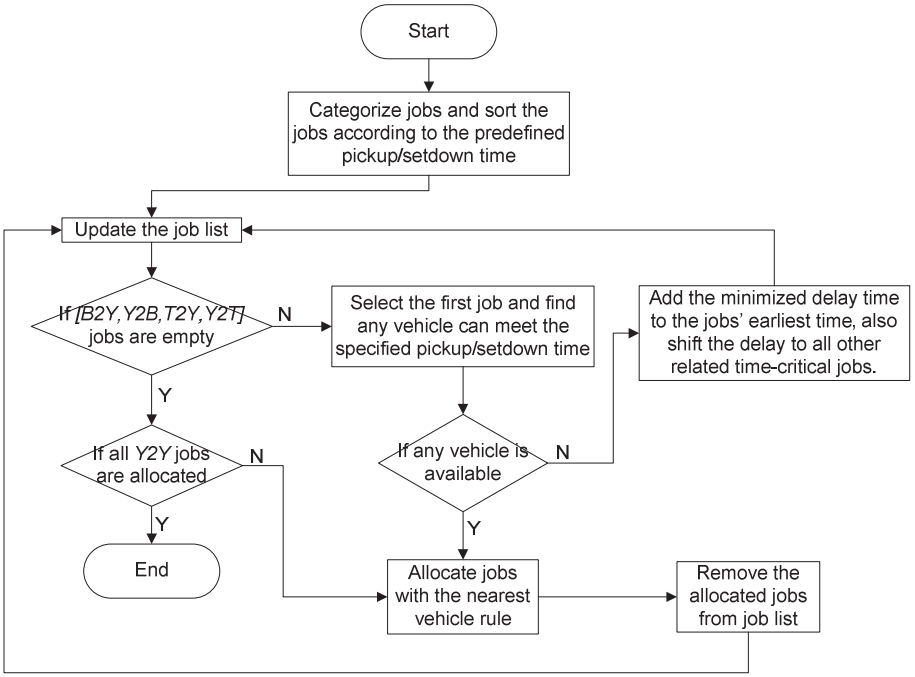


Figure 4-5: Flowchart of sequential job allocation method

The sequential job allocation algorithm, which was implemented as part of this study is also closely coupled with our path planner (Lau et al., 2008). The path for each vehicle is registered sequentially after allocating each job.

4.4 Experimental Results

This section presents a performance comparison between the job grouping approach and the sequential job allocation method. Both the two approaches have been implemented in Matlab 2009b, while our path planner with collision avoidance (Lau et al., 2008) is implemented in C++. All experiments were performed on a High Performance Computing Linux Cluster, configured using 2x3.33Ghz Quad Core Xeon with 24GB of DDR3-1333 ECC Memory.

In order to properly compare the performances of the two approaches, we employ a statistical hypothesis testing using the two-sampled pooled *t*-test (Walpole et al., 2006). The experiments with the two approaches in this thesis were conducted using 30 independent trials, which represent the different configurations of initial vehicle positions and jobs. These trails were generated uniformly at random. Statistical differences are indicated according to the two-sample pooled *t*-test (Refer to Appendix)

with a level of significance at p=0.05 (95% confidence) and (n_1 + n_2 – 2) degrees of freedom. If the absolute *t-value* is greater, then the performance between two methods is more significantly different.

Table 4-1: Parametric Configuration

Parameter	***Value***
Number of Total Jobs (J)	80
Number of Vehicles (ASCs)	20
Number of High Priority Jobs (J^H)	*2*
(λ_1, λ_2, λ_3, λ_4, λ_5)	*(1, 1, 20, 5, 1)*
Plan Starting Time (t_0)	$t_0 = 0$
QCs Discharging Time (t_q^D)	$t_q^D = t_0 + 300000$ms
QCs Uploading Time (t_q^U)	$t_q^U = t_0 + 420000$ms
QC Turnaround Time (Δt^{QC})	$\Delta t^{QC} = 120000$ms
Import TKs Starting Time (t_g^I)	$t_g^I = t_0 + 600000$ms
Export TKs Starting Time (t_g^E)	$t_g^E = t_0 + 720000$ms
TK Turnaround Time (Δt^{TK})	$\Delta t^{TK} = 120000$ms

Table 4-1 shows the parameter values used in the experiments. The parameter values have been based on those typically found in seaport terminal operations. Eighty mixed jobs have been experimented on. These include *B2Y*, *Y2B*, *T2Y*, *Y2T* and *Y2Y* jobs, and there are twenty available ASCs.

Table 4-2: Number of Jobs for Three Different Scenarios

	QC Associated		***TK Associated***		
	B2Y	***Y2B***	***T2Y***	***Y2T***	***Y2Y***
High QC Load	25	25	5	5	20
Medium QC Load	15	15	5	5	40
Low QC Load	5	5	5	5	60

In order to compare the job grouping approach with the sequential planning method, a set of performance metrics are derived from objective function Eq(3.4): ASC travel time($C_s^{TravelTime}$), ASC waiting time($C_s^{SC\_waiting}$), QC waiting time($C_s^{QC\_waiting}$), TK waiting time($C_s^{TK\_waiting}$) and HP finishing time ($C_s^{HP\_finishing}$) . Experimental results are based on three different performance metrics used to evaluate the effectiveness of the proposed algorithms. The performance metrics are considered using three different QC scenarios, including low-, medium- and high-QC loading as described in Table 4-2.

The experimental results for the performance metrics and loading scenarios are shown in Figure 4-6 to Figure 4-8. A lower processing time indicates better performance.

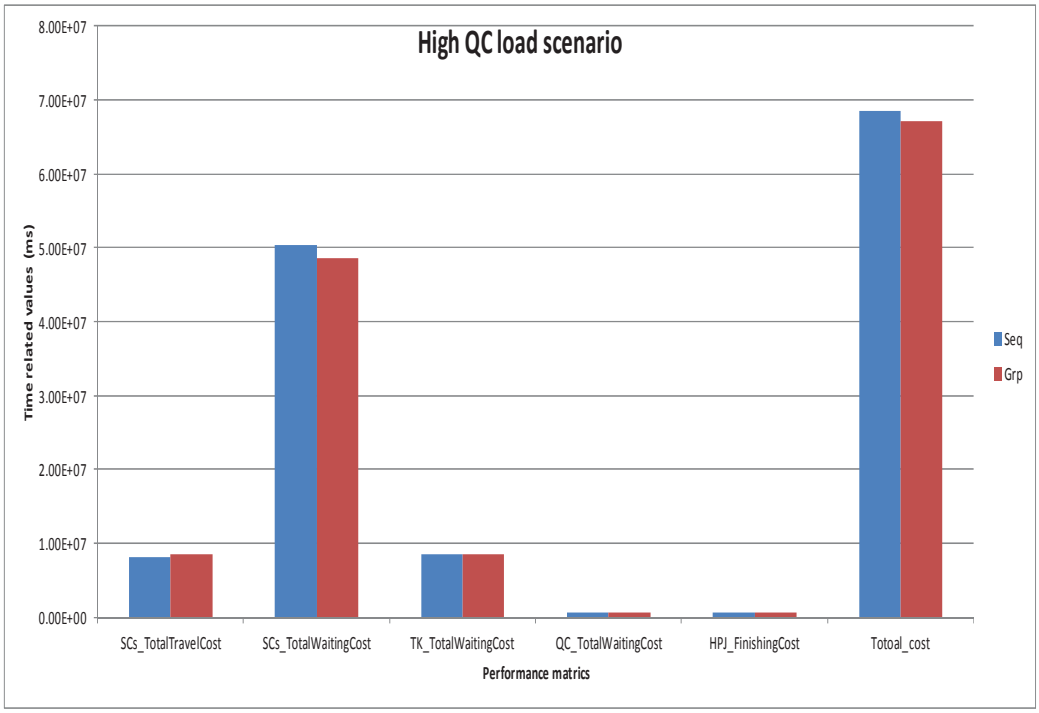


Figure 4-6: Performance comparison for the high QC load scenario.

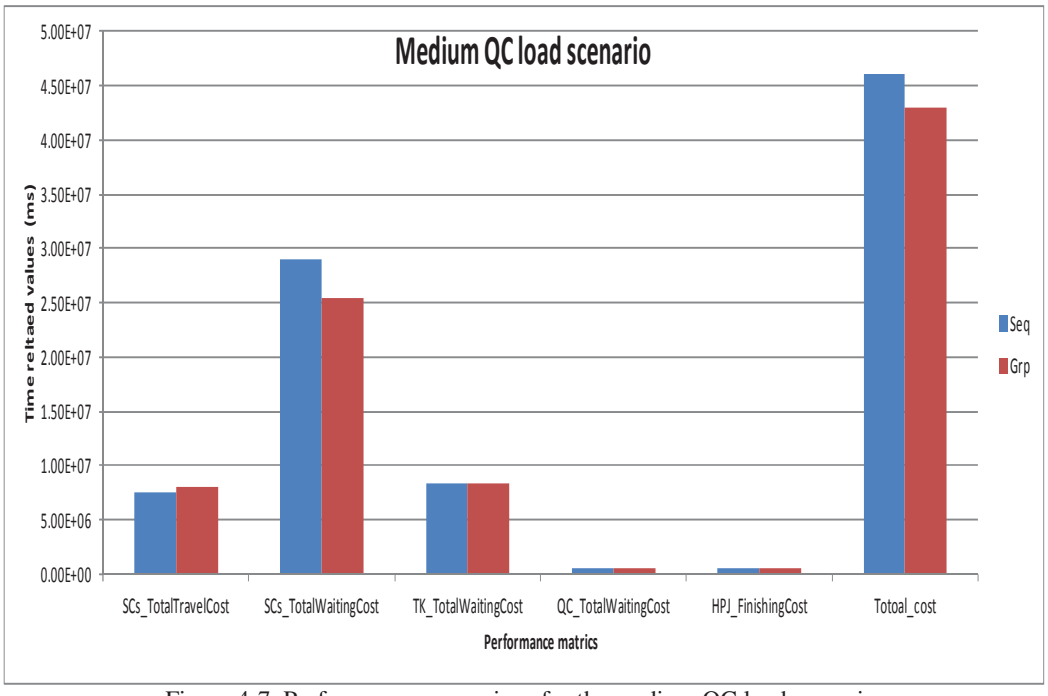


Figure 4-7: Performance comparison for the medium QC load scenario.

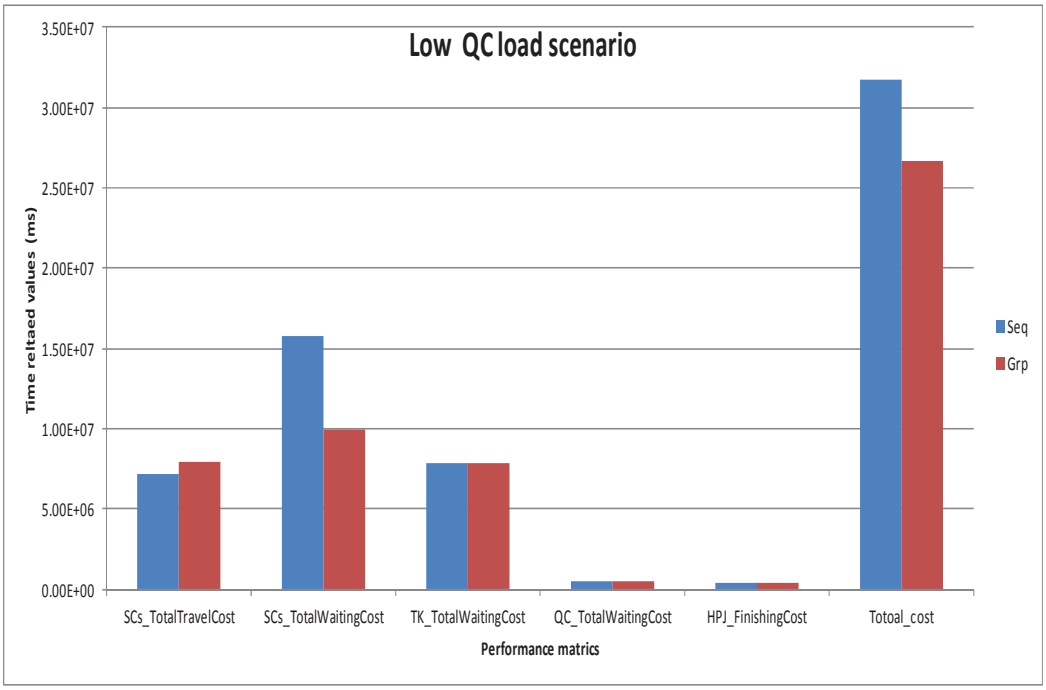


Figure 4-8: Performance comparison for the low QC load scenario.

Table 4-3 summarises the average performance and standard deviation of the job grouping (Grp) approach and the sequential job allocation (Seq) approach. A significant improvement in the performance of the grouping algorithm with respect to the sequential job allocation algorithm is indicated by the *t*-value.

Table 4-3: Comparison of the Job Grouping Algorithm VS Sequential Job Allocation Algorithm

		High QC load			Medium QC load			Low QC load		
		MEAN	STDEV	*t*-value	MEAN	STDEV	*t*-value	MEAN	STDEV	*t*-value
ASC Waiting	**Seq**	5.04E+07	2.78E+05	**-22.19**	2.91E+07	2.34E+05	**-49.56**	1.58E+07	1.40E+05	**-69.25**
	Grp	4.86E+07	3.37E+05		2.55E+07	3.20E+05		9.99E+06	4.40E+05	
Total ASC Travel	**Seq**	8.19E+06	3.27E+05	4.01	7.56E+06	3.02E+05	5.83	7.23E+06	2.88E+05	8.02
	Grp	8.54E+06	3.38E+05		8.05E+06	3.40E+05		7.95E+06	3.95E+05	

According to the critical value (*t* = *-2.00*), the results show the job grouping algorithm outperforms the sequential job allocation algorithm for ASC waiting time in the three different scenarios. When QC load is at the high-level (50 QC associated jobs amongst 80 jobs), the *t*-values of ASC waiting time are *-22.19*. When QC load is at the medium level (30 QC associated jobs amongst 80 jobs), the *t*-values of ASC waiting time are *-49.56*. This is because it is possible to group 20 more *Y2Y* jobs than can be grouped at the high-level QC load. For a low QC load, the t-values for ASC waiting time are significantly different at *-69.25*, since there are 60 *Y2Y* jobs, which can be grouped.

Particularly for the medium- and low-level of QC load scenarios, the job grouping approach performs significantly better than the sequential job allocation approach. This is because ASCs are encouraged to perform some yard area jobs before taking a QC or TK related job, instead of waiting for access to the QC area or TIA. However, the sequential job allocation algorithm has a significantly less total ASC travel time than the job grouping algorithm for the three different loading scenarios, regarding the *t*-values (*4.01*, *5.83*, and *8.02*). This is because ASCs have to travel more routes when trying to group *Y2Y* jobs together with the time-critical jobs. Therefore, the extra travel time of the job grouping approach is the trade-off when minimising ASC waiting time which is more important to the seaport operation. Table 4-4 summarises the overall performance of the job grouping algorithm relative to the sequential job allocation algorithm.

Table 4-4: Performance of the Job Grouping Algorithm Relative to the Sequential Job Allocation Algorithm

	High QC load	***Medium QC load***	***Low QC load***
ASC Waiting Time	4%	12%	37%
Total ASC Travel Time	-4%	-6%	-10%

Last but not least, it is very difficult to obtain the optimal solution for the large problem due to the large number of combinations in the solution space and associated computational costs in finding the optimum. However, in this thesis an exhaustive search was performed to find the optimal solution for a much smaller data set. This allowed for the optimal solution to be compared with those solutions found by the job grouping approach. In this experiment, a small test scenario was set up: 2 ASCs, 2 B2Y jobs and 3 Y2Y jobs, and this was executed for 1000 random configurations of initial vehicle positions and jobs. The experimental results show that the exhaustive search found the optimal solution for all configurations, while the job grouping approach managed to find optimal solutions 84% of the time.

Overall, the effectiveness of the job grouping approach is shown by numerical experiments on different sets of generated data. Although there is no benchmark available yet for the problem, the performances of the job grouping are statistically significantly better than performances of the sequential planning method at the port and close to optimum solutions for small data sets. The downside of the sequential job allocation approach is that some ASCs have an unnecessary waiting time, since they may arrive early at the work point and wait for a long time until other higher priority

QC and TK jobs are transferred to the yard area. Furthermore, it limits ASC utilisation so as to avoid introducing QC and TK delay.

4.5 **Summary**

This chapter proposed a simple job grouping approach as an enhancement of the sequential planning method for solving the comprehensive model. The proposed approach combines the planning with a guiding function, which encourages early starting and finishing for the sake of efficiency. The experimental results showed that the proposed approach can significantly reduce the ASC waiting time for container transfers in the Patrick AutoStrad container terminal, compared to the original sequential planning method.

However, the job grouping approach is based on the sequential local search algorithm, and it would be difficult to get the optimal or even close-optimal solution. In order to provide a good global solution for the job scheduling problem a modified GA will be proposed in this thesis for the job scheduling model.

5. Modified Genetic Algorithm for Scheduling Optimisation

Compared to the comprehensive model, the job scheduling model does not consider the collision avoidance of a fleet of ASCs, and does not take into account many other practical constraints, like variable container orientations and vehicular dynamics that require finite acceleration and deceleration times. However, the job scheduling model is mainly for the global optimisation of ASC job scheduling.

Solving the job scheduling problem using a global optimisation approach is expected to provide higher productivity in automated container terminals. GA is a method which is very easy to understand and is easily transferrable to existing simulations and models. Therefore, this chapter proposes a GA-based optimisation approach for solving the job scheduling problem of container transfers at the Patrick AutoStrad container. Additionally, a practical contribution of this chapter is that the modified GA-based approach has been fully implemented on a trial basis in the live scheduling system at the Patrick container terminal and it effectively improves the performance of the seaport container terminal.

5.1 The Modified GA Approach

In general, GAs are global optimisation techniques (Bo et al., 2006, Cus and Balic, 2003, Smith and Smith, 2002, Goldberg, 1989, Holland, 1992) that avoid many of the shortcomings that exist in classical local search techniques on difficult search spaces. GAs use operators such as reproduction, crossover and mutation as a means of preserving beneficial information with the overall goal of finding a better solution to the problem. In addition, the GAs work through using codification of the parameter space rather than the parameters themselves. The objective function can be easily defined as a measure of fitness for solution performance, which allows the GA to retain useful solutions and inhibit those which are less useful. Based on these features, a GA-based optimisation method is proposed here to solve the modelled problem. The two-part chromosome representation (Carter and Ragsdale, 2006) is adopted and a

chromosome validation and repair method is introduced. Further, a strategy is outlined to effectively handle the sequence and timing constraints during the fitness calculation.

It is nonetheless not practical to use a GA to solve the comprehensive model. Firstly, if we use a GA to search all possible collision-free paths and schedules for each ASC, it would take too long to get a feasible solution for even a small problem. Secondly, it is not easy or even realistic to integrate a path planning algorithm into a GA to solve the comprehensive problem. Lastly, if we solve the path planning problem (by other path planning algorithms) and the job scheduling problem (by a GA) separately, then there might not be a good reason to claim that a GA should be used to solve the comprehensive problem.

5.1.1 Two-part Chromosome Representation

The key to finding a good solution using a GA lies in developing a good chromosome representation of candidate solutions to the problem. A good GA chromosome should reduce or eliminate redundant chromosomes from the population. Redundancy in the chromosome representation refers to a solution being able to be represented in more than one way and appearing in the population multiple times. These multiple representations increase the search space and slow the search. So far, the two-part chromosome technique (Carter and Ragsdale, 2006) has been viewed as the best representation with minimum redundancy for MTSP. The MTSP maps very well to our scheduling problem as each ASC can be regarded as a salesman and each job (including pick-up and set-down operations) can be viewed as a city. However, the major difference is that we take into account complex constraints (e.g. job sequence and timings) and practical performance metrics.

The two-part chromosome technique, as the name implies, divides the chromosome into two parts. The first part of length $n = |J|$ represents a permutation of n jobs and the second part of length $m = |V|$ gives the number of jobs assigned to each ASC. The total length of the chromosome is $n+m$ in this representation. The m values present in the second part of the chromosome must sum to n in order to represent a valid solution. In the example of Figure 5-1, the shaded area shows the second part of the chromosome. Here the first ASC (ASC_1) will process jobs 6, 9, 1 and 7 in that order, the second ASC

(ASC_2) will process jobs 8 and 4 in that order and the third ASC (ASC_3) will conduct jobs 5, 2 and 3 in that order.

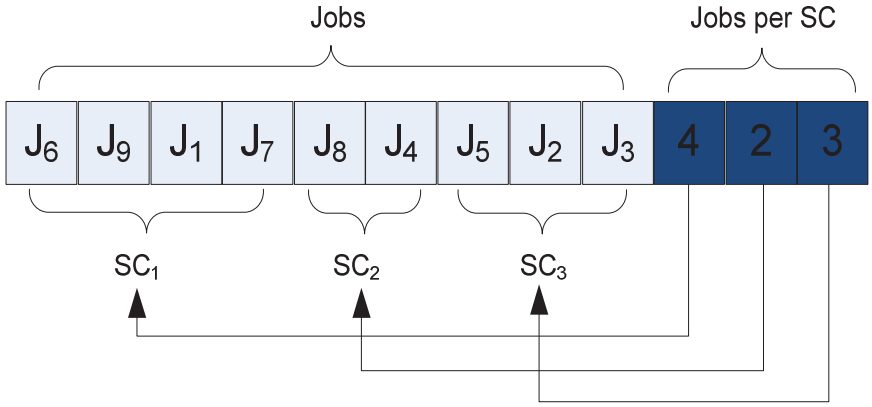


Figure 5-1: Example of two-part chromosome representation for a 9-job schedule with 3 ASCs

Using the two-part chromosome for solution representation, there are $n!$ possible permutations for the first part of the chromosome. The second part of the chromosome represents a positive vector of integers $(x_1, x_2, ..., x_m)$ satisfying $x_1 + x_2 + ... + x_m = n$ where $x_i \geq 0, i = 1, 2, ..., m$. There are $\binom{n+m-1}{n}$ distinct positive integer-valued *m vectors* that satisfy this requirement. Hence, the solution space for the two-part chromosome is of size $n!\binom{n+m-1}{n}$. Moreover, compared to MTSP with the two-part chromosome representation, the ASCs in this problem have different initial depots and they do not have to travel back to their starting depots after finishing all jobs. For the problem which was modelled in Section 5.1, each candidate schedule *s* for an ASC can be regarded as a two-part chromosome in the GA population. In Figure 5-1, for example, a candidate schedule *s* for ASC_1 is represented by four jobs $J_s = (J_6, J_9, J_1, J_7)$ and the related decision variable $X_{1s} = 1$.

5.1.2 Chromosome Validation and Repair Operation

Due to time related constraints (Eq(3.14) and Eq(3.15)) in scheduling, some containers must be picked up or set down following a fixed order by a single ASC, particularly for QC/TK associated jobs. However, within a randomly created two-part chromosome or a newly generated chromosome via crossover/mutation it is possible to have a few jobs which are not consistent with the predefined order of picking up and setting down.

Hence, based on each ASC, it is necessary to check whether the generated chromosome is feasible in terms of satisfying all timing constraints. If any part of the generated chromosome is inconsistent with any fixed order for container pickup or setdown, then we apply a chromosome repair operation so as to ensure the chromosome can represent a feasible schedule.

The above algorithm is designed to check the order of conducting jobs based on each ASC, and also repair any inconsistent part according to a predefined order. In the second part of the chromosome, for each ASC, the algorithm finds the assigned jobs in the first part chromosome based on the predefined conducting order if any, and then checks if the assigned jobs are consistent with the predefined sequence. If not, it then performs a repair operation by adjusting the jobs' positions in the first part of the chromosome. After all iterations, a valid chromosome is generated and this can represent a feasible schedule for container transfers.

The pseudo code of the chromosome validation and repair operation algorithm is:

Chromosome validation and repair operation :

Input: Any unchecked chromosome.

Output: A chromosome representing a feasible schedule.

```
for i = 1:NumASCs // NumASCs: total number of ASCs
   for k = 1:chromo(SecondPartHead + i) // SecondPartHead: head index of second part chromosome
        if the k^th job is associated with a predefined order
            Store the index and its position in the chromosome;
        end
   end
   if any two indexes are inconsistent with the predefined order
        Sort all indexes according to the predefined order;
        for k = 1:chromo(SecondPartHead + i)
              Adjust the job's position in the chromosome;
        end
   else
        Keep the part in the original chromosome;
   end
end
```

As to the frequency of calling the repair algorithm, it really depends on the actual problem's complexity and the initial generated random population. If there are more ASCs and jobs, then the repair frequency might be increased as well.

5.1.3 Fitness Calculation Strategy

Our fitness function is based on the objective function Eq(3.4), however, the fitness calculation cannot be directly applied on a chromosome. This is because some performance metrics are not independent and cannot be calculated simply based on the chromosome, particularly for the QC or TK associated jobs. During fitness evaluation, the total ASC waiting time is dependent on QC and TK operations. QC uploading time or discharging time is directly related to the total QC waiting time, while the TK exporting or importing time is associated with the total TK waiting time. Both QC waiting and TK waiting are also dependent on ASCs' pickup time and setdown time. That is, all types of timings are correlated closely.

To evaluate the overall fitness of a schedule we initially calculate the total ASC travel time by summing up all job setup times and job processing times, which all are based on Dijkstra's algorithm (Dijkstra, 1959). The job setup time refers to the duration of empty travelling time for an ASC moving to a particular job pick-up position while the processing time refers to the duration of time for an ASC transporting a container from a job pick-up position to the set-down position.

Following the above, the total ASC waiting time, the total QC waiting time and the total TK waiting time are calculated together. The two-part chromosome is then transformed into a set of groups based on each ASC and each job assigned to an ASC is accordingly mapped into different slots. A pair of raw (*estimated*) pick-up times and raw set-down times is created initially for each job by assuming there is no ASC waiting time. By assuming there is no QC waiting time a set of raw QC uploading times and QC discharging times are created for each QC based on the QC starting time (t_q^D and t_q^U) and turnaround time (Δt^{QC}). Likewise, TK raw timings are created in the same way based on the TK starting time (t_g^E and t_g^I) and turnaround time (Δt^{TK}). For each slot we find all QC/TK related jobs with raw timings and update the dependent jobs (via α_{ab} or β_{ab}). If its dependent job has been updated then we check the job raw timings and identify the ASC waiting, QC waiting or TK waiting times and then update all related timings. If its dependent job has not been updated yet then we ignore the job until its dependent job has been updated. However, since each QC has two buffer nodes, the evaluation on QC waiting time and related ASC waiting time should take into account the two buffer nodes. That is, for each uploading QC job ($j \in J_q^U, j \geq 2$) its QC

waiting time should be calculated based not only on the current raw QC timings but also the set-down time of the second previous job ($(j-2)\in J_q^U$). Likewise, for each discharging QC job ($j\in J_q^D, j\geq 2$) the QC waiting time should be calculated based not only on the current raw QC timings but also on the pick-up time of the second previous job ($(j-2)\in J_q^D$). After all iterations for the slots have been performed, the ASC waiting time, QC waiting time and TK waiting time can be calculated and all related timings updated. An example is provided in Figure 5-2 to further illustrate the process.

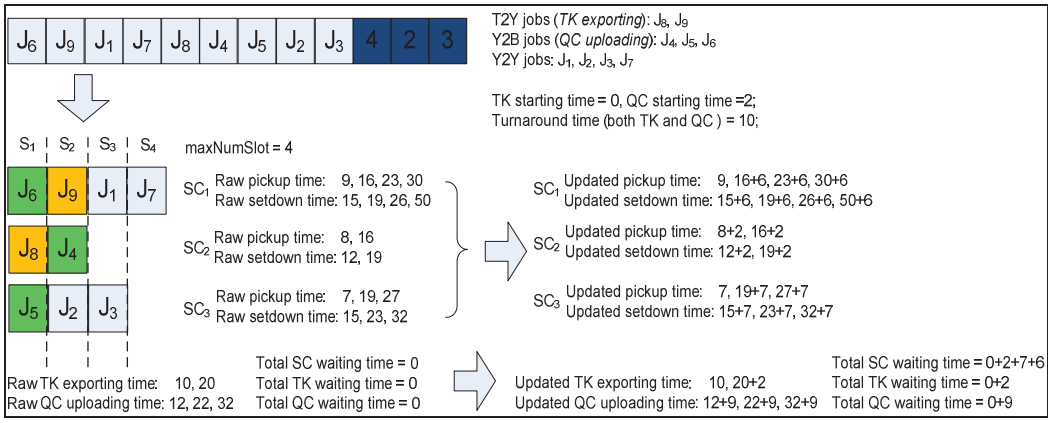


Figure 5-2: An example of calculating ASC and QC waiting times and updating related timings

In Figure 5-2, J_8 and J_9 are TK exporting jobs, J_4, J_5 and J_6 are QC uploading jobs and the other jobs are Y2Y jobs. As each QC has two buffer nodes, J_6 is dependent on J_4 and J_4 must be set-down before setting down J_6 at the same buffer. Here, J_5 can be set-down at another buffer as long as it satisfies the timing of uploading QC for J_5. J_8 is required to be picked up before picking up J_9. The raw timings of TK and QC are calculated using the starting times of QC and TK and the turnaround time.

In the first slot (S_1), J_5 and J_8 need to be checked for related raw timings and J_6 should be checked after checking J_4. The raw QC uploading time for J_5 is 22 and the raw set-down time of J_5 is 15, therefore ASC_3 needs to wait for 7 time units. Accordingly, all pick-up and set-down timings of J_2 and J_3 should be updated by adding the 7 time units. By checking the timings of J_8 ASC_2 needs to wait for 2 time units as the raw pick-up time is 8 and raw TK exporting time is 10. Owing to this, all pick-up and set-down timings of ASC_2 should be updated by adding 2 time units.

In the second slot (S_2), J_4 and J_9 need to be checked for related timings. The latest updated set-down time of J_4 is 21 and the raw QC uploading time is 12 for J_4, so the QC needs to wait for 9 time units. The QC discharging time for J_6 should be updated to 31 from 22. As J_6 is at the previous slot and dependent on J_4 and J_6 should be checked before J_9 in this slot. The latest updated QC uploading time of J_4 is 21 and the raw set-down time of J_6 is 15 and then ASC_1 needs to wait for 6 time units. Accordingly, the pick-up and set-down timings of J_9, J_1 and J_7 should be updated by adding the 6 time units. Regarding J_9, the raw TK exporting time is 20 and the pick-up time for ASC_1 is updated to 22 and TK needs to wait for 2 time units. For the third and fourth slots it is not necessary to check and update the timings because there is no QC/TK related job. Overall, the total ASC waiting time is 15, the total TK waiting time is 2 and the total QC waiting time is 9.

Fitness calculation:

Input: A valid two-part chromosome.

Output: Fitness value (f) for all performance metrics.

Variable and parameters initialisations for the chromosome and set f=0;
Generate the raw timings for all QCs and TKs based on their turnaround times;

for i = 1:numASCs
  for k = 1:chromo(SecondPartHead + i) // *for each assigned job of the ASC*
    Calculate and store the ideal pickup time ($t_{start}^{J_k}$) and setdown time ($t_{finish}^{J_k}$) for the assigned job (J_k);
    $f = f + \lambda_1$ * the travel time for the job setup and job processing;
  end
end

maxNumSlots **=** max(values of the second part chromosome);

for i = 1: maxNumSlots
  Get QC and TK related jobs which have not been checked yet in slot (i).
  for k=1:numUnchecked // search unchecked job, check if the job's dependent job has been updated;
    if the job's dependent job has been updated
      Compare the raw timings of pickup/setdown with the related QC and TK timings;
      Calculate the waiting time and update all raw timings and all QC and TK timings related jobs;
      if the waiting time is caused by ASC
        $f = f + \lambda_2$ * ASCWaiting;
      elseif QC needs to wait
        $f = f + \lambda_3$ * QCWaiting;
      elseif TK needs to wait
        $f = f + \lambda_4$ * TKWaiting;
      end
    end
  end
end

for i=1:numHighPriorityJobs
  $f = f + \lambda_5 * t_{finish}^{J_i}$;
end

The performance of high priority job finishing times can be evaluated based on all previously updated timings. The total value of high priority job finishing times is the summation of set-down times (T_j^F , $j \in J^H$) for all high priority jobs. The following pseudo code shows the chromosomal fitness calculation of a valid two-part chromosome using the objective function.

5.1.4 Selection and Crossover

We utilise the rank-based roulette-wheel (Goldberg, 1989) selection in our GA. With this approach, each chromosome is assigned a portion of an imaginary roulette wheel, based on a chromosome's rank. Chromosomes which have a higher fitness value are allocated a larger segment of the roulette wheel. Selection of a parent requires random generation of a number between 0 and 1. The chromosome occupying the section of the roulette wheel covered by the randomly generated percentage is chosen as a parent. The second parent is selected in the same manner.

A modified crossover operator is required to handle the two-part chromosome. For the first part of the two-part chromosome, our modified crossover operator is consistent with the ordered crossover (Carter and Ragsdale, 2006), which is commonly used for planning with sequence constraints. The parent chromosomes are selected using the rank-based roulette-wheel scheme described above. Given two parent chromosomes, two random crossover points are selected, partitioning them into a left, middle and right portion. The ordered two-point crossover behaves in the following way: child A inherits its middle section from parent A, and its left and right sections are determined, as shown in Figure 5-3.

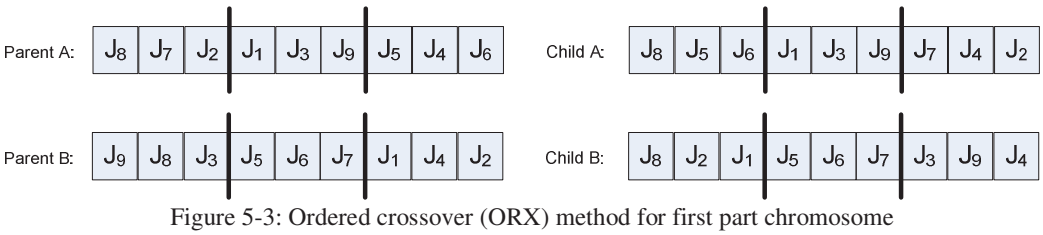


Figure 5-3: Ordered crossover (ORX) method for first part chromosome

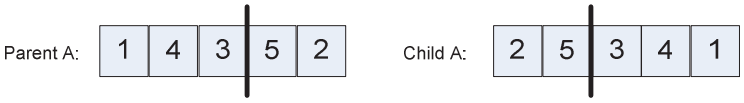


Figure 5-4: Single point asexual crossover for second part chromosome

For the second part of the two-part chromosome, we adopt a single point asexual crossover operator, as shown in Figure 5-4. This method simply cuts the second part of the chromosome into two sections and reverses the order in which the two pieces are arranged. This type of crossover ensures that the second part of the chromosome remains feasible (with the sum of the values in the chromosome equalling n). After the crossover operations for both the first and second parts of the two-part chromosome it is necessary to perform a chromosome validation and repair for all generated children.

5.1.5 Mutation

The mutation operator used in our GA is based on the swap mutation (Goldberg, 1989), which consists of randomly swapping two genes in the first part of the two-part chromosome. This swap mutation operator performs a very vital task because it provides the means by which jobs are exchanged to seek improvement. Crossover has the ability to drastically impact the groupings of jobs, but has little impact on the ordering of the jobs within each group. Mutation is needed to explore alternative solutions, as a low probability of mutation can maintain diversity of the solutions in the population. In addition, after each mutation operation, it is also necessary to perform a chromosome validation and repair for all newly generated chromosomes.

5.1.6 Replacement

We use the replacement policy of Steady-State GA (DeJong, 1975). Parents are selected to produce offspring and then a decision is made as to which individuals in the population to select for deletion to make room for the new offspring. Steady-State GA is an overlapping system, since parents and offspring compete for survival. A percentage of the replacement determines how much of the population of each generation is replaced by the newly generated children.

5.2 Experimental Results

This section presents performance comparisons between the GA-based approach and the sequential job scheduling method via simulation experiments and live testings in the Patrick AutoStrad scheduling system. The sequential job scheduling method is derived

from the sequential job planning approach introduced in Chapter 4, but it doesn't deal with path planning in this chapter. In our simulation experiments both the GA approach and sequential job scheduling algorithms have been implemented in C++. Live testings in the Patrick AutoStrad terminal were accomplished by integrating the GA-based algorithm with the scheduling system.

5.2.1 Simulation Testings

Firstly, we conducted the comparison on the performance of the GA-based method and the sequential job scheduling approach for scheduling a small amount of jobs (24 jobs).

There is a trade-off between the computation time and solution quality. As the port planning process is iterative, each planning round may have the similar problem as the previous round. Therefore, it is unnecessary to get the GA run too long at each round, and also the solution found by the GA would still be useful for the next round. In other words, the solution is still being improved as long as the planning process is on-going. So, using the GA does not mean that the computation time will be a big challenge.

Table 5-1: Parametric configuration for scheduling 24 mixed jobs

Parameter	***Value***
Number of Total Jobs (J)	*24*
Number of Vehicles (V)	*8*
Number of High Priority Jobs (J^H)	*2*
(λ_1, λ_2, λ_3, λ_4, λ_5)	*(1, 1, 20, 5, 1)*
Plan Starting Time (t_0)	$t_0 = 0ms$
QCs Discharging Time (t_q^D)	$t_q^D = t_0 + 10000ms$
QCs Uploading Time (t_q^U)	$t_q^U = t_0 + 10000ms$
QC Turnaround Time (Δt^{QC})	$\Delta t_{QC} = 80000ms$
Import TKs Starting Time (t_g^I)	$t_g^I = t_0 + 200000ms$
Export TKs Starting Time (t_g^E)	$t_g^E = t_0 + 200000ms$
TK Turnaround Time (Δt^{TK})	$\Delta t_{TK} = 100000ms$
GA population size	*100*
Number of generations in GA	*500 (Approx. less than 1 minute)*
Crossover probability rate in GA	*0.85*
Mutation probability rate in GA	*0.01*
Crossover method in GA	*Order crossover + Asexual crossover*
Selection method in GA	*Roulette-wheel selection*
Mutation method	*Swap*
Replacement in GA	*Steady-state GA*
Replacement percentage	*50%*

Table 5-1 shows the parameter values for the experiment on the simple scheduling scenario which includes QC and TK related jobs and *Y2Y* jobs. We have attempted to

base the parameter values on those typically found in seaport terminal operations. The 24 mixed jobs consist of *B2Y* (6 jobs are associated with one discharging QC), *Y2B* (6 jobs are associated with one uploading QC), *T2Y* (2 jobs are associated with one exporting TK), *Y2T* (2 jobs are associated with one importing TK) and *Y2Y* (8 jobs), and there are 8 available ASCs.

Experimental results include different performance metrics used to evaluate the effectiveness of the proposed algorithms. These metrics include total ASC travel cost, total ASC waiting cost, total TK waiting cost, total QC waiting cost and total high priority job finishing cost, all of which have been defined in Section 3.4.2 of Chapter 3. Since the GA is a stochastic search algorithm, it is necessary to perform at least 30 independent runs to obtain the average values (*GA_Average*) and the best result (*GA_Best*) found by the GA. Please note that *GA_Average* and *GA_Best* are based on the overall cost function, rather than individual cost functions, such as total ASC travel cost, total ASC waiting cost, total TK waiting cost, total QC waiting cost and or high priority job finishing cost.

As shown in Figure 5-5, the GA-based approach outperforms the sequential job scheduling algorithm (SQT) for the total cost of scheduling the 24 jobs for both *GA_Average* and *GA_Best*. The solution found by GA improved the overall performance in this experiment by 42.50% (*GA_Average*) and 45.10% (*GA_Best*). The GA achieves a better result than SQT for ASC total waiting cost, TK total waiting cost, while SQT has only slightly better values for ASC total travel cost and high priority job (HPJ) finishing cost. For the QC total waiting cost, the performance of SQT and GA are similar because the weight of scheduling QC related jobs is large (λ_3=20) and both algorithms try to reduce QC total waiting cost as much as possible. The GA sacrifices a little ASC total travel cost and HPJ finishing cost so as to gain much more reduction in ASC total waiting cost and TK total waiting cost.

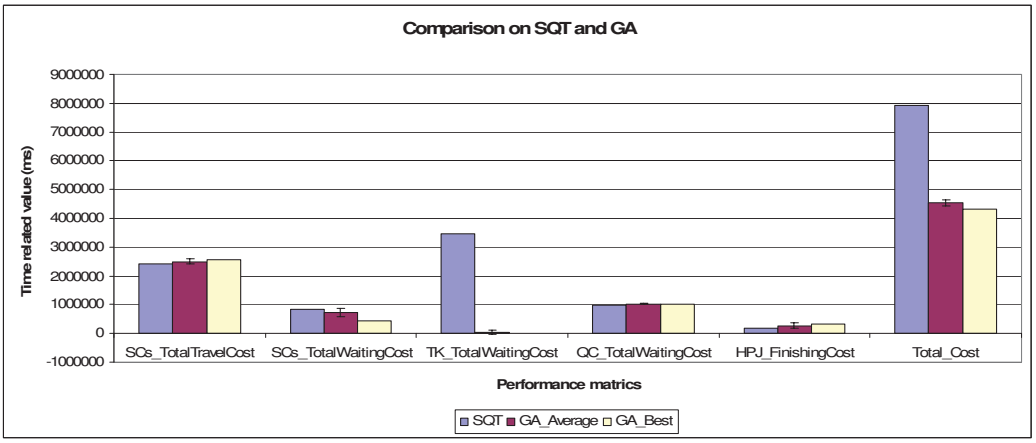


Figure 5-5: Comparison on scheduling for 24 mixed jobs with 8 ASCs

The performances of the GA-based method and the sequential job scheduling approach were also compared for scheduling a large number of jobs (80 jobs). Table 5-2 shows the parameter values for the experiments on the large number of jobs. The parameter values are based on those typically found in seaport terminal operations. We experiment on 80 mixed jobs which include *B2Y* (15 jobs are associated with one discharging QC), *Y2B* (15 jobs are associated with one uploading QC), *T2Y* (5 jobs are associated with one exporting TK), *Y2T* (5 jobs are associated with one importing TK) and *Y2Y* (40 jobs), and there are 20 available ASCs.

Table 5-2: Parametric configurations for scheduling 80 mixed jobs

Parameter	***Value***
Number of Total Jobs (J)	*80*
Number of Vehicles (V)	*20*
Number of High Priority Jobs (J^H)	*8*
(λ_1, λ_2, λ_3, λ_4, λ_5)	*(1, 1, 20, 5, 1)*
Plan Starting Time (t_0)	$t_0 = 0ms$
QCs Discharging Time (t_q^D)	$t_q^D = t_0 + 10000ms$
QCs Uploading Time (t_q^U)	$t_q^U = t_0 + 10000ms$
QC Turnaround Time (Δt^{QC})	$\Delta t_{QC} = 80000ms$
Import TKs Starting Time (t_g^I)	$t_g^I = t_0 + 200000ms$
Export TKs Starting Time (t_g^E)	$t_g^E = t_0 + 200000ms$
TK Turnaround Time (Δt^{TK})	$\Delta t_{TK} = 100000ms$
GA population size	*200*
Number of generations in GA	*1000 (Approx. 2 minutes)*
Crossover probability rate in GA	*0.85*
Mutation probability rate in GA	*0.01*
Crossover method in GA	*Order crossover + Asexual crossover*
Selection method in GA	*Roulette-wheel selection*
Mutation method	*Swap*
Replacement in GA	*Steady-state GA*
Replacement percentage	*50%*

As shown in Figure 5-6, the GA-based approach outperforms the sequential job scheduling algorithm (SQT) for the total cost of scheduling the 80 mixed jobs with both *GA_Average* and *GA_Best*. The solution found by GA reduced the overall cost in this experiment by 26.10% (*GA_Average*) and 34.20% (*GA_Best*). The GA achieves a better result than SQT for ASC total waiting cost, TK total waiting cost, while SQT has only slightly better values for ASC total travel cost, QC total waiting cost and HPJ finishing cost.

Based on the above experimental results, we employ a statistical hypothesis testing using the two-sampled pooled *t*-test (Walpole et al., 2006) to show the statistical significance between GA and SQT. This result indicates the performance difference between the two algorithms. If the absolute *t-value* is greater, then the performance between two methods is more significantly different.

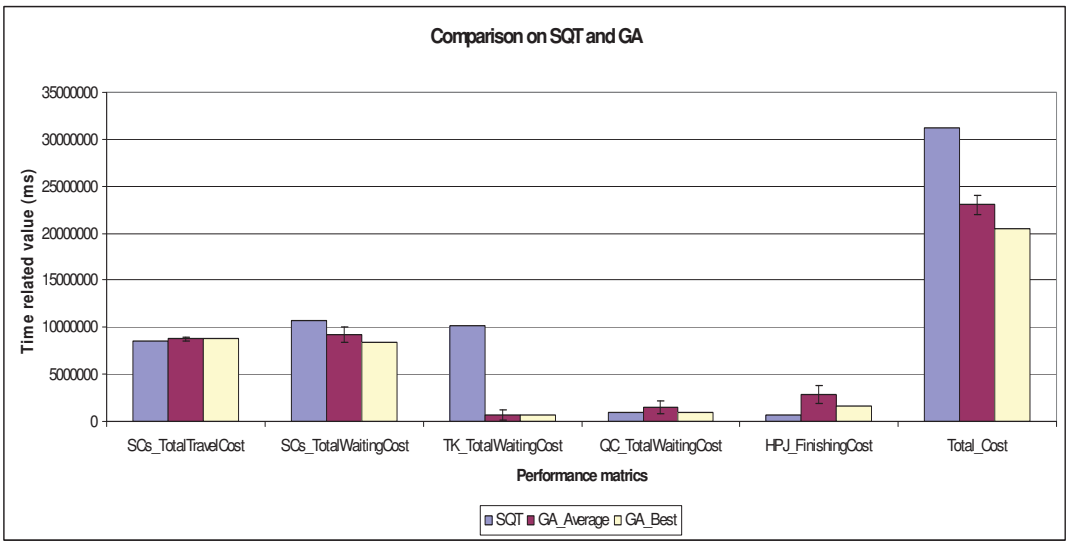


Figure 5-6: Comparison on scheduling for 80 mixed jobs with 20 ASCs

Table 5-3: Comparison of GA vs Sequential Job Scheduling Algorithm for Total Cost

		MEAN	STDEV	*t*-value
24 mixed jobs	**SQT**	7.90E+06	0.00E+00	**-166.18**
	GA	4.50E+06	1.11E+05	
80 mixed jobs	**SQT**	3.11E+07	0.00E+00	**-296.89**
	GA	2.30E+07	1.50E+05	

Table 5-3 summarises the average performance and standard deviation of the GA and SQT algorithms for the total cost. A significant improvement in performance of the GA with respect to the SQT algorithm is indicated by the *t*-value. According to the critical

value (t = *-2.00*), the results show the performance of the GA is from a statistical perspective, significantly better than the performance of the SQT algorithm for both 24 mixed jobs and 80 mixed jobs.

5.2.2 Live Testing Results at Patrick AutoStrad Terminal

Our GA-based approach has been implemented on a trial basis in the scheduling system of the Patrick AutoStrad Terminal in Brisbane, Australia. To show the effectiveness of the GA-based approach, we obtained a set of random live testing results from the Patrick AutoStrad Terminal and compared them to the actual scheduling optimiser (OP) of Patrick.

Table 5-4: GA settings

Parameter	***Value***
GA population size	*100*
Number of generations	*50 (Approx. 2 minutes)*
Crossover probability rate	*0.85*
Mutation probability rate	*0.01*
Crossover method	*Order crossover + Asexual crossover*
Selection method	*Roulette-wheel selection*
Mutation method	*Swap*
Replacement in GA	*Steady-state GA*
Replacement percentage	*50%*

Table 5-4 shows the general settings for the GA throughout the testing. All the problems listed in Table 5-5 represent a random snapshot based on typical operations at the Patrick AutoStrad terminal. The planning performance of the OP and GA-based approach are calculated by an evaluation function in Patrick's system. Both OP and GA are integrated with a local optimising process which conducts the local optimisation for each schedule based on a local hill climbing mechanism. Moreover, a number of additional practical requirements are taken into account in the actual operating system including *ASC contention* and the *MIN*/*MAX* number of ASCs. *ASC contention* is used to penalise the job scheduling when some of the ASCs are likely to travel on paths which are close together and might cause some traffic delay. The *MIN*/*MAX* number of ASCs needs to be ensured in particular sets of jobs. Fundamentally, the *MIN* number is to ensure that for the duration of the planning horizon, the throughput of container movements respects a certain level of expected throughput that roughly equates to the number of ASCs set as the *MIN* for any order of work. At the same time, the *MAX*

number applies a large cost to any period of time on the planning horizon in which more than *MAX* ASCs are simultaneously working on the same order of work. The precise details are not presented in this thesis due to commercial considerations. Nevertheless, the GA-based approach outperforms the OP method with positive improvements through the 12 testings as shown in Table 5-5.

For the 12 different port scenarios, the GA outperformed OP, except Case 8. Variations in the improvement are because each case has different conditions, such as the number of ASCs, number of jobs, pick-up and set-down locations of jobs and the ASC initial position. In case 8, as the problem is relatively simple (i.e. there are only 3 jobs and 16 available ASCs), the performance scores of GA and OP are the same, while the most significant difference is made by GA at case 4, where the performance score is improved by 8292. Overall, the effectiveness of the GA-based approach is shown by numerical experiments and live testing results on different sets of data.

Table 5-5: Various testing results from Patrick AutoStrad scheduling system

Snapshot problems	***Score of OP***	***Score of GA***	***Raw Improvement by GA***
Case 1 (ASCs=16, Jobs=21)	471781	471704	77
Case 2 (ASCs=20, Jobs=32)	104447	104109	338
Case 3 (ASCs=14, Jobs=32)	280304	277861	2443
Case 4 (ASCs=18, Jobs=14)	105647	97355	8292
Case 5 (ASCs=14, Jobs =21)	361782	360192	1590
Case 6 (ASCs=18, Jobs=9)	58896	58853	43
Case 7 (ASCs=18, Jobs=10)	69908	69282	626
Case 8 (ASCs=16, Jobs=3)	55675	55675	0
Case 9 (ASCs=19, Jobs=17)	53057	53040	17
Case 10 (ASCs=13, Jobs=20)	81647	81564	83
Case 11 (ASCs=11, Jobs=60)	119716	119664	52
Case 12 (ASCs=14, Jobs=47)	70870	70747	123

5.3 **Summary**

This chapter has presented a modified mathematical model derived from the problem of container transfers at the Patrick AutoStrad container terminal located in Brisbane, Australia. This model incorporates QC related operations, ASC scheduling and TK

related operations. Additionally, a practical contribution of this thesis is the GA-based approach which was presented to solve the job scheduling problem and was compared to a sequential job scheduling method for different scheduling scenarios. The proposed approach has been fully implemented on a trial basis in the live scheduling system at the Patrick container terminal and it effectively improves the performance of the seaport container terminal.

Having applied the two-part chromosome encoding technique with the GA, the thesis will now turn to examine the enhancement of the crossover approach. Accordingly, the next chapter will analyse some limitations of the existing crossover approach with the two-part chromosome encoding, and propose a new crossover approach for further improvement.

6. A New GA Crossover Approach for Further Improvement

Since the two-part chromosome GA was proposed (Carter and Ragsdale, 2006) for solving the MTSP with the minimum search space, this thesis adopted the encoding technique and modified the GA for solving the job scheduling problem in Chapter 5. Two limitations were found for the existing crossover strategy. Firstly, it severely limited the diversity in the second part of the chromosome, which in turn greatly restricted the search ability of the GA. Secondly, the existing crossover approach had a tendency to break useful building blocks in the first part of the chromosome, which of necessity reduced the GA's effectiveness and solution quality. In this chapter, we propose a new crossover approach named Two-part Chromosome Crossover (TCX) to overcome these limitations and improve the chances of solving the job scheduling problem.

6.1 Overview and Analysis of the Existing Strategy

6.1.1 The Two-part Chromosome Encoding Technique

As introduced in Section 5.2 of Chapter 5, the two-part chromosome technique divides the chromosome into two parts. The first part of length n represents a permutation of n jobs and the second part of length m gives the number of jobs assigned to each ASC. Therefore, the total length of the chromosome is $n+m$ in this representation. The m values in the second part of the chromosome must sum to n in order to represent a valid solution.

In addition, another two different chromosome representations are commonly used in GAs: one-chromosome representation (Tang et al., 2000) and two-chromosome representation (Park, 2001, Malmborg, 1996). However, both of these chromosome representations have more redundant solutions in the search space compared to the two-part chromosome representation (Carter and Ragsdale, 2006), as the one-chromosome representation's size of solution space is $(n+m-1)!$ (Carter and Ragsdale, 2006), and the

two-chromosome representation's size of solution space is $n!m^n$ (Carter and Ragsdale, 2006).

6.1.2 Existing Crossover Method for Two-part Chromosomes

In an important study (Carter and Ragsdale, 2006), Carter and Ragsdale employed a combined crossover technique (ORX+A) for two-part chromosomes. The first part of the chromosome uses the classic ordered crossover (ORX) operator (Goldberg, 1989), while the second part of the chromosome uses an asexual crossover operator (Chatterjee et al., 1996). That is, when performing crossover, two-part chromosomes are separated and two independent crossover methods are used for that purpose.

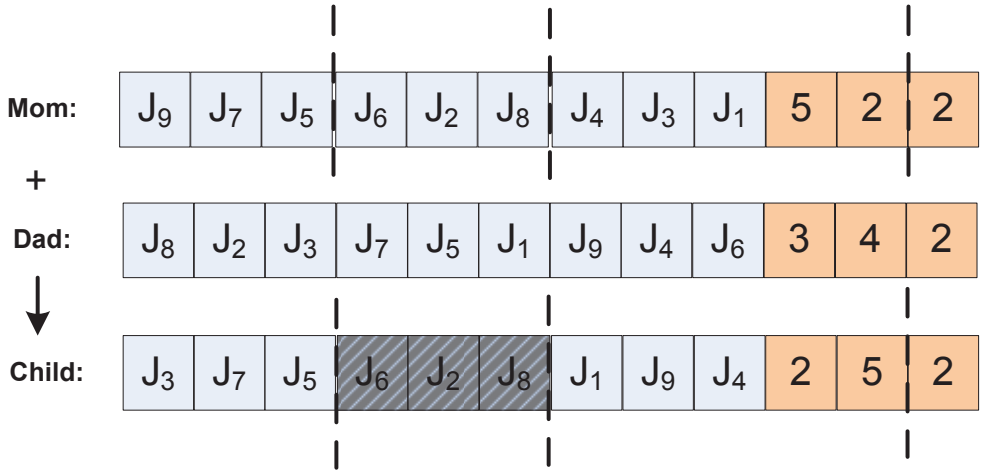


Figure 6-1: Example of using the ORX+A crossover method (Carter and Ragsdale, 2006) for two-part chromosomes

Specifically, Figure 6-1 is an example illustrating the overall process of the crossover operation for two-part chromosomes that were proposed in the study by Carter and Ragsdale (Carter and Ragsdale, 2006). This example shows that two parent chromosomes generate a child based on the mother's genes. For the first part chromosome, the ordered crossover behaves in the following way: the child inherits its middle section from its mother, and its left and right sections are determined according to its father's genes. The shadowed genes (J_6, J_2, J_8) in the child's first part chromosome are directly copied from its mother, and the other genes are filled according to the sequence of left genes in its father's chromosome. For the second part chromosome, a single point asexual crossover operator is utilised as shown in Figure 6-1. This method simply cuts the second part of the mother's chromosome into two

sections ((5, 2) and (2)) and reverses the order in which the two pieces were arranged. This type of crossover ensures that the second part of the chromosome remains feasible (with the sum of the values in the chromosome equalling *n*). Likewise, when performing crossover based on the father's genes, another child can be generated as well. Normally, each pair of parents can generate two children, i.e. a sister and a brother.

6.1.3 Limitations of the Existing Method

The crossover method (Carter and Ragsdale, 2006) has two limitations, which can reduce the search performance. First, in order to ensure that the second part of the chromosome is valid (i.e. a positive vector of integers $(x_1, x_2, ..., x_m)$ satisfying $x_1 + x_2 + ... + x_m = n$ where $x_i > 0, i = 1, 2, ..., m$), an asexual crossover has been adopted. The asexual crossover operator only changes the order of genes in the second part of the chromosome. This approach may limit the diversity of the whole population, because some other feasible numerical combinations would not appear in the second part of the chromosome, during the initialisation of the population. For example, in Figure 6-1, the second part of the mother and father's chromosomes are <5, 2, 2> and <3, 4, 2> respectively and their children would never have a chance to reach other feasible genes in the second part of their chromosomes, such as <1, 1, 7>, <1, 2, 6> and <1, 3, 5>, which are all still valid and feasible chromosomes. Additionally, for the MTSP problem (*n*=9, *m*=3), there are 7 fundamental gene combinations: <1, 1, 7>, <1, 2, 6> , <1, 3, 5>, <1, 4, 4>, <2, 2, 5>, <2, 3, 4> and <3, 3, 3>. Therefore, the GA population must contain the 7 fundamental gene combinations in the second part of the chromosome, so that the asexual crossover may have the chance to reach all possible solutions in the second part of the chromosome.

When the size of the problem gets bigger, the number of fundamental combinations increases dramatically and it is impossible to have all possible combinations in the limited population. To calculate the number of fundamental combinations in the second part of the chromosome, we define a function $f(n, m, k)$, where *n* is the number of jobs, *m* is the number of ASCs; *k* is the minimum number of jobs each ASC must visit. In our case, we only consider $f(n, m, 1)$, where *k*=1 and each ASC must conduct at least one

job. However, the general form $f(n,m,k)$ is necessary for a recursion process to make the calculation relatively convenient. The recursive function can be defined as:

$$f(n,m,k) = \sum_{i=k}^{\left\lfloor \frac{n}{m} \right\rfloor} f(n-i, m-1, i) \qquad (6.1)$$

Where $\left\lfloor \frac{n}{m} \right\rfloor$ is the floor value of (n/m). When m = 1, $f(n,1,k) = 1$. That is, if there is only one ASC, then the number of fundamental combinations is only one (i.e. <n>), regardless of the values of n and k.

Proof of Eq(6.1): Suppose the number of jobs conducted by each of the m ASC is represented by $(x_1, x_2, ..., x_m)$. To compute the number of fundamental combinations we just need to consider the non-decreasing combinations, that is x_1 <= x_2 <= … <= x_m. Since the total number of jobs is n and the total number of ASCs is m it is clear that the upper limit of x_1 is $\left\lfloor \frac{n}{m} \right\rfloor$, otherwise there must be some $x_i < x_1$ which is not valid. Now since we are computing $f(n,m,k)$ and each ASC needs to do at least k jobs, the minimal value of x_1 is k. Thus all the possible values for x_1 are from k to $\left\lfloor \frac{n}{m} \right\rfloor$ and why in Eq(6.1) the sum of i is from k to $\left\lfloor \frac{n}{m} \right\rfloor$. Now suppose x_1 is fixed as i (any integer from k to $\left\lfloor \frac{n}{m} \right\rfloor$), then the problem is for m-1 ASCs to conduct ($n - i$) jobs. Since we are only considering non-decreasing combinations, the minimal number of jobs that each ASC needs to do is $x_1 = i$, giving $f(n-i, m-1, i)$ in Eq(6.1).

For example, given n = 5, m = 2 and k=1 then the total number of fundamental combinations can be calculated using Eq(6.1):

$$f(5,2,1) = \sum_{i=1}^{2} f(5-i, 2-1, i) = f(5-1, 2-1, 1) + f(5-2, 2-1, 2) = f(4,1,1) + f(3,1,2) = 1+1 = 2$$

In fact, there are only two fundamental combinations <1, 4> and <2, 3> for the case n = 5 and m = 2. Based on Eq(6.1), we provide an analysis of the number of fundamental combinations in the second part of the chromosome for the job scheduling problem by increasing the problem size. Table 6-1 shows the exponential nature of the possible number of fundamental combinations for 12 job scheduling problems. Note that the cell

with "-" means that the number of combinations is greater than (7E+07). In addition, Table 6-2 shows the limited number of fundamental combinations (as a percentage) when using the existing crossover method, compared to the actual numbers of all fundamental combinations. Assume that the initial population size is 100, and all second part chromosomes contain different fundamental combinations. Then the percentages are calculated using the population size (100) divided by the number of calculated fundamental combinations in Table 6-1.

Table 6-1: The number of fundamental combinations in the second part of the chromosome for each job scheduling problem (number of jobs_number of ASCs) showing the exponentially increasing number of combinations with increasing *m*

51_3	51_5	51_10	100_3	100_5	100_10	100_20	150_3	150_5	150_10	150_20	150_30
2.17E+02	2.82E+03	1.95E+04	8.33E+02	3.82E+04	2.98E+06	1.05E+07	1.88E+03	1.88E+05	-	-	-

Table 6-2: Limited search space of the existing method for the second part of the chromosome

51_3	51_5	51_10	100_3	100_5	100_10	100_20	150_3	150_5	150_10	150_20	150_30
46.08%	3.55%	0.51%	12.00%	0.26%	0.00%	0.00%	5.33%	0.05%	0.00%	0.00%	0.00%

Another important characteristic of the GAs is that children are expected to stochastically inherit a subset of the parents' genes during each crossover process. For the job scheduling problem, each ASC has an independent tour, which represents the sequence of conducting jobs by the ASC itself and does not have any direct relationship with other ASC in the scheduling. However, when using the existing crossover method, the likelihood of children inheriting parents' sub-tours is relatively limited. This is mainly because the existing crossover method treats the first part of the chromosome as a whole and ignores the important mapping of gene segments to independent ASCs in the second part of the chromosome. In other words, the existing method would potentially break important building blocks or highly fit gene segments (tours) of the parent chromosomes. This situation may worsen when the number of ASCs (*m*) and the number of jobs (*n*) increase.

Overall, the search performance of GAs with the two-part chromosome representation can be seriously degraded as a result of the above two shortcomings of the crossover method.

6.2 **A New Crossover Approach**

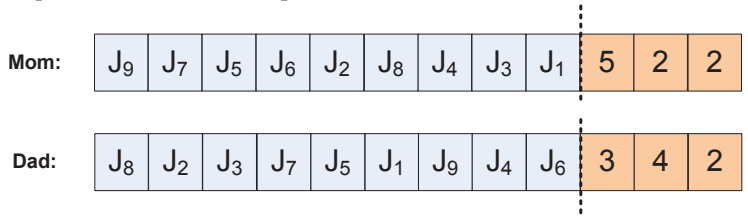
In this section, we propose a new crossover approach (TCX) for the two-part chromosome crossover to improve the GA search performance for solving the job scheduling problem.

TCX treats each ASC separately when performing crossover in the first part of the chromosome. This ensures that highly fit building blocks that may be present in the subtours of parental chromosomes are maintained during the reproduction process and inherited by the offspring chromosomes. In addition, TCX greatly enhances the diversity in the second part of the chromosome by increasing the number of feasible combinations, while always satisfying the constraint: $(x_1, x_2, ..., x_m) \quad x_1 + x_2 + ... + x_m = n$ where $x_i > 0, i = 1, 2, ..., m$. Instead of using the existing asexual crossover, TCX reproduces genes in the second part of the chromosome according to the results of the crossover operation that was performed in the first part of the chromosome. Therefore, new types of fundamental combinations may be generated and the diversity in the second part of the chromosome is increased dramatically.

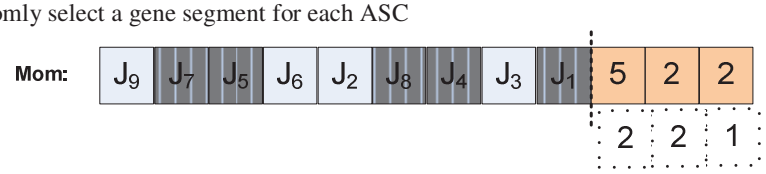
An example is used below to illustrate the process of using TCX to generate a child chromosome of (9 jobs and 3 ASCs). Five basic steps are involved. Firstly, a pair of parent two-part chromosomes (Mother and Father) are initialised, and the Mother chromosome is used as the base for generating a Child in this example. Secondly, we randomly select a gene segment (i.e. sub-tour) from the first part of the Mother's chromosome. In this case, the selected gene segments are <J_7, J_5> for ASC 1, <J_8, J_4> for ASC 2 and <J_1> for ASC 3. Hence, the numbers of randomly selected genes are (2, 2, 1) as shown in the dashed area of Step 2. Next, the order of the remaining genes is sorted according to the positions in the first part of the Father's chromosome. In this example, the remaining genes in the first part of the Mother's chromosome are <J_9, J_6, J_2, J_3>, and the order of these genes is shuffled to <J_2, J_3, J_9, J_6> according to the first part of the Father's chromosome. Next, based on the number of unselected genes, we compute a uniform random number, between 1 and the current value of unselected genes, to determine how many new genes will be added for each ASC in the Child chromosome. Here, the unselected genes are <J_2, J_3, J_9, J_6> and if, for example, we iteratively generate the integer sequence (2, 1, 1) of uniform random numbers between 1 and the number of currently unselected genes, then in the Child chromosome, ASC 1

gets genes <J_2, J_3>, ASC 2 gets <J_9>, and ASC 3 gets <J_6>. Moreover, during this step it is possible to generate a sequence containing one or more zeros, for example (4, 0, 0) or (3, 1, 0). This does not create infeasibility in the second part of the chromosome. Therefore, for the first part of the Child's chromosome, ASC 1 would have <J_7, J_5, J_2, J_3>, ASC 2 would have <J_8, J_4, J_9>, and ASC 3 would have <J_1, J_6>. Lastly, we construct the two-part chromosome for the Child by updating the information in the second part of its chromosome. In this example, <4, 3, 2> is generated by summing the each position of the intermediary second part vectors of <2,2,1> + <2,1,1> = <4,3,2> , which remains feasible and bounded.

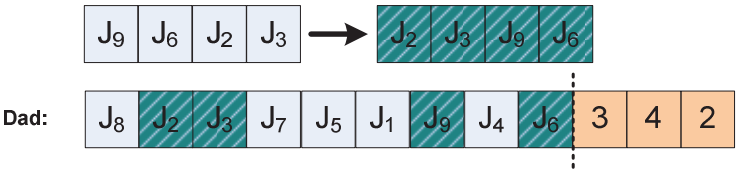
Step 1: Initialise a pair of chromosomes as parents



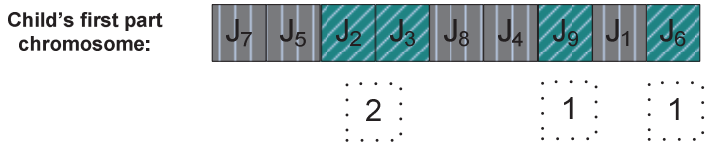
Step 2: Randomly select a gene segment for each ASC



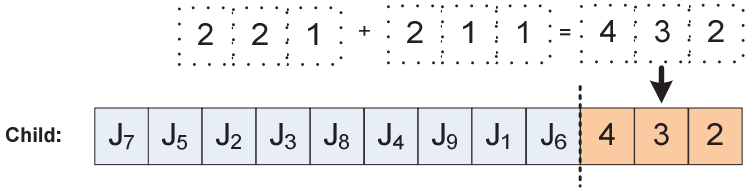
Step 3: Shuffle gene positions according to the first part of the Father's chromosome



Step 4: Add genes for each ASC



Step 5: Construct the Child's two-part chromosome.



In the second part of the Child's chromosome, the TCX operator will generate a new combination, which is probably different from the original fundamental combination in

the second part of the Mother's chromosome. As shown in the above example, the second part of the Child's chromosome is <4, 3, 2>, while the Mother's is <5, 2, 2>. However, if using the existing crossover approach, the second part of the Child's chromosome would have been the same as the Mother's chromosome <5, 2, 2>. Subsequently, taking the Father's chromosome as the base, another Child chromosome can be produced by going through the five steps described above. A pair of Child chromosomes is generated after each TCX crossover operation. The following pseudo code shows the proposed TCX crossover method.

Two-part chromosome crossover operation:

Input: A pair of two-part parent chromosomes.

Output: A pair of two-part child chromosomes.

```
//Process of generating a Child (Sister)

Select a parent's chromosome as the Mother base for the Child;
for m = 1:numASC //numASC: the total number of ASCs{
  Randomly generate an integer number between 1 and AssignedJobs[m] and store in Segment[m]; //
  assignedJobs[m]: the number of assigned jobs of ASC m
  if AssignedJobs [m] > Segment[m]
     Randomly generate an integer number between 1 and (AssignedJobs [m] - Segment[m]) and set it as the
     possible starting position for the gene segment;
     for k = 1:Segment[m]
        Copy each gene from the gene segment for a Child;
        Copy each gene into savedGenesPool;
     end
  else
     Copy the entire part of ASC m for Child;
     Copy the entire part of ASC into savedGenesPool;
  end
  totalSavedGenes = totalSavedGenes + Segment[m];
end
totalUnsavedGenes = numJobs – totalSavedGenes;
for m = 1: numASC
  if m != numASC
     Randomly generate an integer number between 1 and totalUnsavedGenes for ASC m to add genes;
  else
     The ASC will add all unsaved genes;
  end
  According to the order of the unsaved genes in the first part of the Father's chromosome, add the randomly
  generated number of genes to the Child;
end
Based on the construction of the first part of the Sister's chromosome, calculate the number of assigned jobs in the
second part of the chromosome.

//Process of generating a Child (Brother)

Select another parent's chromosome as the base for the Child;
Repeat the process of generating the Sister;
```

It can be seen that, TCX has two advantages when two-part chromosomes are used to solve the job scheduling problem. Firstly, the offspring have a better chance of inheriting highly-fit subtours from the parental chromosomes, because the mapping between each ASC and the corresponding job subtour are considered separately in TCX. The second advantage is the dramatic increase in diversity of the second part of the

chromosome because children have the opportunity to reach any feasible combinations within the entire search space defined in the second part.

When applying the new crossover TCX for solving our job scheduling problem, the crossover operator of the GA presented in Chapter 5 is replaced with the new crossover TCX. Figure 6-2 shows an integrated diagram describing the overall solution approach.

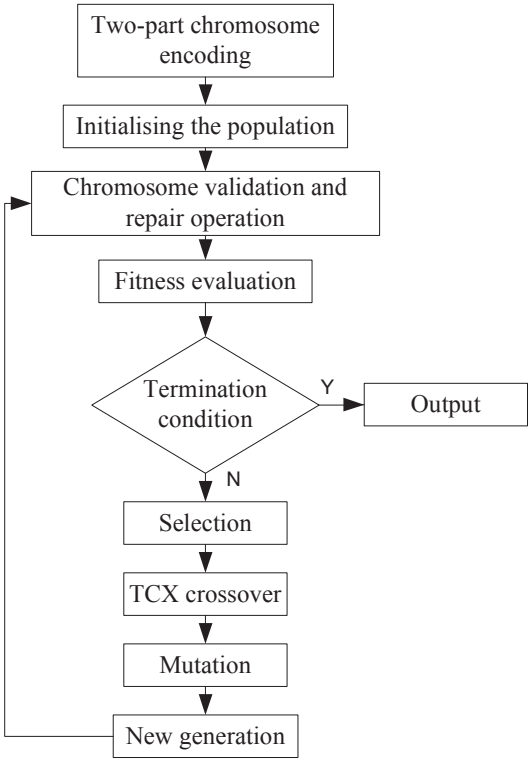


Figure 6-2: The modified GA flowchart for solving the job scheduling problem

Specifically, the two-part chromosome encoding, chromosome validation and repair operation, and fitness evaluation are presented in Section 5.2 of Chapter 5. The experimental performance of TCX for solving the job scheduling problem will be presented in following section.

6.3 **Experiments for Job Scheduling Optimisation**

A comparison was conducted on the performance of the existing ORX+A crossover (used for the modified GA in Chapter 5) and the proposed TCX crossover operator for scheduling a small amount of jobs (24 jobs). All parameter values for the experiment on

the simple scheduling scenario are from Table 5-1 in Chapter 5. The 24 mixed jobs consist of *B2Y* (6 jobs are associated with one discharging QC), *Y2B* (6 jobs are associated with one uploading QC), *T2Y* (2 jobs are associated with one exporting TK), *Y2T* (2 jobs are associated with one importing TK) and *Y2Y* (8 jobs), and there are 8 available ASCs.

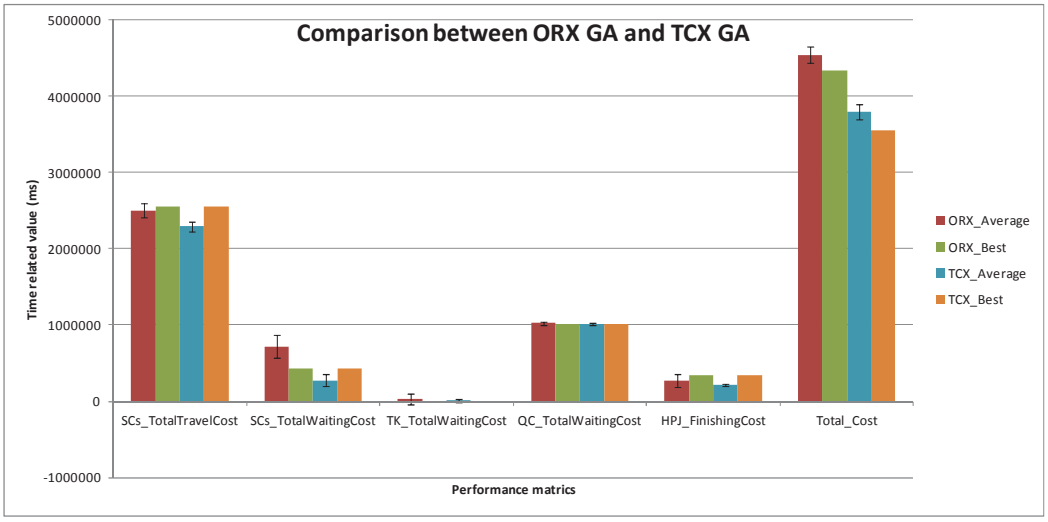


Figure 6-3: Comparison on scheduling for 24 mixed jobs with 8 ASCs

As shown in Figure 6-3, TCX approach outperforms the existing ORX+A for the total cost of scheduling the 24 jobs for both *Average* and *Best*. Please note that *Average* and *Best* are based on the overall cost function, rather than individual cost functions, such as total ASC travel cost, total ASC waiting cost, total TK waiting cost, total QC waiting cost and or high priority job finishing cost. The solution found by TCX improved the overall performance in this experiment by 16.31% (*Average*) and 17.97% (*Best*). TCX achieves a better result than ORX+A for ASC total travel cost, ASC total waiting cost, TK total waiting cost, high priority job (HPJ) finishing cost. For the QC total waiting cost, the performance of TCX and ORX+A are similar because the weight of scheduling QC related jobs is large (λ_3=20) and both algorithms try to reduce QC total waiting cost as much as possible.

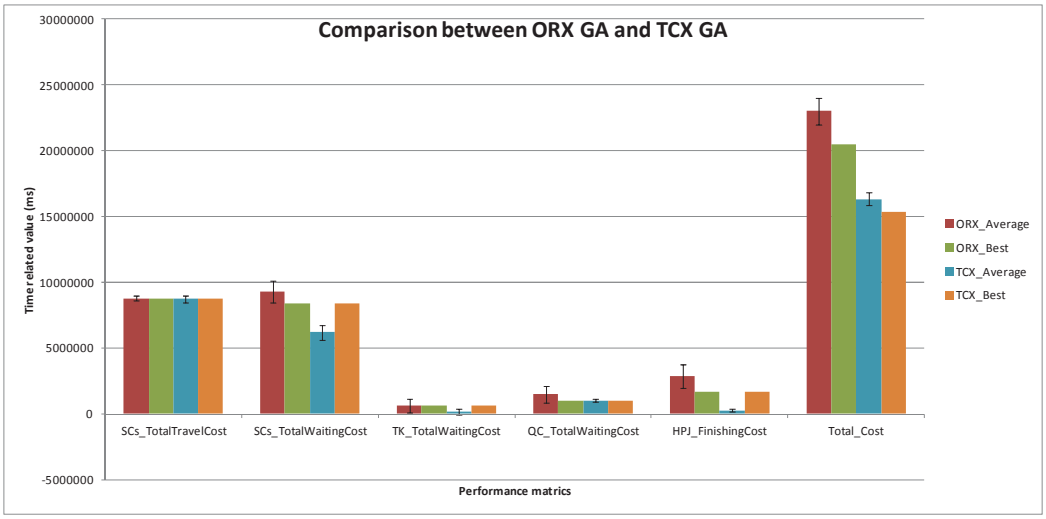


Figure 6-4: Comparison on scheduling for 80 mixed jobs with 20 ASCs

We also compared the performances of TCX and the existing ORX+A approach for scheduling a large number of jobs (80 jobs). Table 5-2 in Chapter 5 shows the parameter values for the experiments on the large number of jobs. The parameter values are based on those typically found in seaport terminal operations. Eighty mixed jobs are experimented on which include *B2Y* (15 jobs are associated with one discharging QC), *Y2B* (15 jobs are associated with one uploading QC), *T2Y* (5 jobs are associated with one exporting TK), *Y2T* (5 jobs are associated with one importing TK) and *Y2Y* (40 jobs), and there are 20 available ASCs.

As shown in Figure 6-4, TCX approach outperforms the existing ORX+A method for the total cost of scheduling the 80 mixed jobs with both *Average* and *Best*. The solution found by GA reduced the overall cost in this experiment by 29.02% (*Average*) and 25.07% (*Best*). TCX achieves a better result than ORX+A for ASC total waiting cost, TK total waiting cost, QC total waiting cost and HPJ finishing cost, while they have quite similar ASC total travel cost.

A *t*-test was performed to show the statistical significance between ORX and TCX. This result indicates the performance difference between the two crossovers. If the absolute *t-value* is greater, then the performance between two methods is significantly different.

Table 6-3: Comparison of ORX GA vs TCX GA for Total Cost

		MEAN	STDEV	*t*-value
24 mixed jobs	**ORX**	4.54E+06	1.11E+05	**-27.40**
	TCX	3.80E+06	9.85E+04	
80 mixed jobs	**ORX**	2.30E+07	1.00E+06	**-32.65**
	TCX	1.63E+07	4.97E+05	

Table 6-3 summarises the average performance and standard deviation of TCX and ORX+A for the total cost. A significant improvement in performance of TCX with respect to ORX+A is indicated by the *t*-value. According to the critical value (*t* = *-2.00*), the results show the performance of TCX is in a statistical sense significantly better than the performance of ORX+A for both 24 mixed jobs and 80 mixed jobs.

6.4 **Computational Testing Methodology for the MTSP**

As discussed in Section 2.2.4 of Chapter 2, there are many differences between the MSTP and our job scheduling problem. However, they may share the encoding technique and crossover method when applying GAs, particularly with the two-part chromosome technique, as mentioned in Section 5.2.1. Moreover, the MTSP is a well-known problem, and maps very well to our scheduling problem. Hence, the proposed TCX can also be directly applied to solve the MTSP using the two-part chromosome technique.

To evaluate the benefits of the proposed TCX crossover operator for solving the MTSP, computational experiments were conducted to compare the performance of four crossover methods on a set of problems created for the MTSP. The proposed TCX is compared to ORX+A, CYX+A and PMX+A, and these are combined with an asexual crossover for the two-part chromosome representation. The same strategy is presented in Section 5.2 for GA selection, crossover, mutation and replacement. The test problems were selected from a standard collection of TSPs from the Library of Travelling Salesman Problems (Reinelt, 2001) that were transformed into MTSPs by using more than one salesperson (m) to complete the tour. The test problems are Euclidean, two-dimensional symmetric problems with 51, 100, and 150 cities. Throughout this chapter, these test problems are referred to as MTSP-51, MTSP-100, and MTSP-150, respectively. These MTSP problems appear to be well-suited to the purpose of this chapter, which is to examine the limitations of the existing crossover

operators for two-part chromosomes and to compute the effectiveness of the proposed TCX operator. They permit direct comparison with the previous approaches and even though they are viewed as a simplified representation of practical problems, they remain representative as benchmarking problems.

Two different fitness (or objective) functions were considered. The first fitness function measured the total travel distance travelled by all of the salesmen. For the problems, each of the *m* salesmen were required to visit at least one city (other than the home city). This objective represents a situation in which there is a group of salesmen and there are no constraints associated with the maximum number of cities visited by any one salesman. The second fitness function measured the longest route among the *m* salesmen. Typically, this objective function is used to equalise the workload among the available salesmen when scheduling tasks.

In the simulation experiments, all GA programs were implemented in C++ with GAlib 2.4.7 (GAlib, 2007). Table 6-4 summarises the experimental conditions of 12 different problem size (*n*) and salesman (*m*) combinations along with the run times for each type of problem. The stopping criterion is the number of generations. All salesmen start and end their individual tours in the same city. Table 6-5 shows the parameter values for 'all' the experiments performed in this chapter.

Table 6-4: Computational test conditions

Number of cities (*n*)	**Number of salesmen (*m*)**	**GA generations**
51	3, 5 and 10	50000
100	3, 5, 10 and 20	100000
150	3, 5, 10, 20 and 30	200000
128	10, 15 and 30	200000

Table 6-5: Parametric configuration for GA

Parameter	***Value***
GA population size	*100*
Crossover probability rate in GA	*0.85*
Mutation probability rate in GA	*0.01*
Selection method in GA	*Roulette-wheel selection*
Mutation method	*Swap*
Replacement in GA	*Steady-state GA*
Replacement percentage	*20%*

6.5 Computational Results for the MTSP

This section presents performance comparisons between the proposed TCX approach and the other three crossover methods for two-part chromosomes. To correctly show the performance differences among the various crossover operators, two sets of experiments are conducted for two objective functions. For each objective function, the GA is first run without seeding (i.e. without any local optimisation) for the benchmark problems and all initial populations are randomly created. Subsequently, the same benchmark experiments are conducted using the GA with seeding. In this case, each of the starting populations is seeded with a candidate solution produced by a simple greedy heuristic to give the GA a good starting point in the search space.

Since GAs belong to the class of stochastic search algorithms, statistical hypothesis testing is employed using the two-sampled pooled *t*-test (Walpole et al., 2006). The experiments for the four crossover approaches in this chapter are conducted using 30 independent trials (each pair of $n_1 = n_2 = 30$). This occurs for both seeded and unseeded GA experiments. That is, the TCX is compared to each of the other three crossover methods. In this chapter, the null hypothesis can be stated such that: 'TCX does provide higher solution quality when applied to 30 trials'. Statistical differences are indicated according to the two-sample pooled *t*-test (Appendix) with a level of significance at $p = 0.05$ (95% confidence). The statistical hypothesis tests in this chapter use two-tailed critical regions to indicate whether there is a significant improvement by TCX ($t \leqslant -2.00$) or a significant degradation by TCX ($t \geqslant 2.00$). Statistical tests resulting in a *t*-value of ($-2.00 < t < 2.00$) do not provide enough statistical evidence to refute or confirm the null hypothesis, which indicates similar performance between the two algorithms. Moreover, it is assumed that the experimental results follow a standard normal distribution under the null hypothesis and the experimental data has been sampled independently from the two populations being compared. Furthermore, a variation of the *t*-test known as the Welch's t-test is employed. This is used when the two population variances are assumed to be different and must be estimated separately. In addition, experimental results and the MTSP problems used in this chapter including the cost matrices can be downloaded from http://ims.uts.edu.au/MTSP/MTSP_dataset_solutions.zip.

6.5.1 Experiments for Minimising Total Travel Distance

The first objective function is to minimise the total travel distance of all the salesmen. This objective reflects the goal of minimising the distance required to visit all *n* cities. The only constraint used with this objective is that each salesman must visit at least one city (other than their home city). Without this constraint, the GA could reduce the number of salesmen in the problem, hence possibly reducing the MTSP to a TSP. For this objective, as the number of salesmen increases the total travel distance of the combined trips also tends to increase, because each salesman must start and return to the home city.

6.5.1.1 GA without Seeding for Minimising Total Travel Distance

Table 6-6 summarises the mean value and standard deviation of the 30 trails of each approach for testing problems with the objective of minimising the total travel distance. The significant improvements in the performance of TCX with respect to each other method are indicated by the *t*-values. According to the critical value (t = -2.00), all calculated *t*-values as shown in Table 6-7 are less than -2.00 for the 12 problems. In other words, the solutions found by TCX are statistically better than the solutions found by the other three crossover methods (ORX+A, CYX+A and PMX+A) for all cases (MTSP-51, MTSP-100 and MTPS-150).

Table 6-6: Experimental results for minimising total travel distance.

		m = 3			*m* = 5			*m* = 10			*m* = 20			*m* = 30		
Problem	Crossover	Mean	Stdev	**Best**	Mean	Stdev	**Best**	Mean	Stdev	**Best**	Mean	Stdev	**Best**	Mean	Stdev	**Best**
MTSP-51	TCX	510	24	**466**	536	26	**499**	636	17	**602**	-	-	-	-	-	-
	ORX+A	584	29	**517**	621	39	**551**	709	33	**648**	-	-	-	-	-	-
	CYX+A	591	43	**511**	622	44	**530**	710	42	**633**	-	-	-	-	-	-
	PMX+A	601	38	**513**	606	40	**537**	705	34	**625**	-	-	-	-	-	-
MTSP-100	TCX	32708	2267	**28943**	34179	2006	**30941**	36921	1964	**32802**	46976	1773	**44112**	-	-	-
	ORX+A	41516	3356	**36713**	42416	2806	**36196**	44631	2997	**38717**	54265	3059	**47971**	-	-	-
	CYX+A	41911	3195	**35791**	43634	2804	**35421**	45150	3241	**40894**	52916	2884	**46466**	-	-	-
	PMX+A	41441	3423	**33802**	42063	3931	**33908**	44786	3467	**39785**	52142	2688	**46212**	-	-	-
MTSP-150	TCX	55851	2588	**51126**	61596	4759	**51627**	61360	3888	**54473**	69701	4340	**62456**	84008	5285	**76481**
	ORX+A	67037	3745	**60090**	68018	3377	**62539**	72113	3637	**63899**	81696	5372	**71933**	96122	4562	**88515**
	CYX+A	67463	4454	**55335**	69860	4342	**61521**	71584	4845	**63126**	83471	4197	**75146**	97106	3911	**89008**
	PMX+A	68152	5140	**58303**	69112	4011	**60761**	72620	4334	**64975**	81178	4920	**73281**	95752	4923	**87402**

Table 6-7: *t*-test results of TCX versus (ORX, CYX and PMX) crossover approaches for minimising total travel distance.

Problem	Comparison	$m = 3$	$m = 5$	$m = 10$	$m=20$	$m = 30$
MTSP-51	TCX VS ORX+A	-10.77	-10.03	-10.93	-	-
	TCX VS CYX+A	-8.96	-9.32	-9.13	-	-
	TCX VS PMX+A	-11.04	-8.16	-9.98	-	-
MTSP-100	TCX VS ORX+A	-11.91	-13.08	-11.78	-11.29	-
	TCX VS CYX+A	-12.87	-15.02	-11.89	-9.61	-
	TCX VS PMX+A	-11.65	-9.79	-10.81	-8.79	-
MTSP-150	TCX VS ORX+A	-13.46	-6.03	-11.06	-9.51	-9.50
	TCX VS CYX+A	-12.35	-7.03	-9.01	-12.49	-10.91
	TCX VS PMX+A	-11.71	-6.61	-10.59	-9.58	-8.91

6.5.1.2 GA with Seeding for Minimising Total Travel Distance

As mentioned previously, we also test the performance of TCX with the GA seeding enabled. We employ the same seeding technique as those of Carter and Ragsdale (Carter and Ragsdale, 2006) for minimising total travel distance and generate the greedy solutions by examining the present location of all the salesmen and calculating the closest unassigned city to each salesman. The unassigned city is then assigned to the closest salesman and the process is continued until all the cities are assigned to a salesman.

Table 6-8: Comparison of the mean results between two approaches using GA with seeding enabled for minimising total travel distance.

Problem	m	(Carter & Ragsdale, 2006)	TCX	Improvement (%)
MTSP-51	3	543	492	9.39%
	5	586	519	11.43%
	10	723	670	7.33%
MTSP-100	3	26653	26130	1.96%
	5	30408	28612	5.91%
	10	31227	30988	0.77%
	20	54700	44686	18.31%
MTSP-150	3	47418	44674	5.79%
	5	49947	47811	4.28%
	10	54958	51326	6.61%
	20	73934	62400	15.60%
	30	99547	78023	21.62%

In Table 6-8, we compare the average results in the literature (Carter and Ragsdale, 2006) with the mean results found by our TCX approach, using the simple GA seeding technique. For the 12 testing problems, TCX shows positive improvements compared to the results obtained in Carter and Ragsdale (Carter and Ragsdale, 2006). The greatest

improvement is achieved at MTSP-150 (m=30) with a 21.62% improvement in final solution quality. Therefore, by combining GA seeding and the proposed TCX operator a better solution quality can be achieved when compared to the approach in Carter and Ragsdale (Carter and Ragsdale, 2006) for the objective of minimum total travel distance.

However, seeding is not without risk, and may cause the GA to become stuck at a local minimum in some cases. For example, the average result 670 for the problem (MTSP-51: m=10) with seeding is worse than the mean result 636 without seeding in Table 6-8. Clearly, seeding may provide a good start point and reduce the search cost, but it might also reduce solution quality when GA is attracted by a local minimum.

6.5.2 Experiments for Minimising Longest Tour

Besides the objective of minimising the total travel distance, another common objective in real world MTSP applications is minimising the longest individual tour, which is also called makespan (Yuan et al., 2011). Minimising the longest tour has the goal of balancing the cities (or workload) among the salesmen and minimising the distance the salesmen travel. With the objective of minimising the longest tour, the fitness values decrease as the number of salesmen increase.

6.5.2.1 GA without Seeding for Minimising Longest Tour

Table 6-9: Experimental results for minimising longest tour.

		m = 3			m = 5			m = 10			m = 20			m = 30		
Problem	Crossover	Mean	Stdev	**Best**	Mean	Stdev	**Best**	Mean	Stdev	**Best**	Mean	Stdev	**Best**	Mean	Stdev	**Best**
MTSP-51	TCX	207	13	**182**	153	10	**135**	113	2	**112**	-	-	-	-	-	-
	ORX+A	216	12	**191**	165	11	**139**	128	13	**112**	-	-	-	-	-	-
	CYX+A	222	16	**188**	161	12	**138**	131	16	**112**	-	-	-	-	-	-
	PMX+A	218	11	**191**	161	10	**141**	130	12	**112**	-	-	-	-	-	-
MTSP-100	TCX	14365	1013	**12645**	10086	674	**8730**	7768	492	**6796**	6768	433	**6358**	-	-	-
	ORX+A	15137	1462	**12997**	11459	1053	**9415**	9286	1385	**7373**	8123	881	**6666**	-	-	-
	CYX+A	15759	1242	**13467**	11333	1278	**9507**	9151	1364	**7111**	8109	936	**6516**	-	-	-
	PMX+A	15238	1371	**12249**	11233	1177	**9267**	6890	1482	**7187**	8265	868	**6570**	-	-	-
MTSP-150	TCX	22523	1226	**20556**	16054	1227	**14096**	10722	927	**8475**	9640	789	**8423**	8759	806	**7169**
	ORX+A	24766	1689	**22015**	17646	1519	**15266**	15150	2006	**11788**	13669	1816	**10274**	12204	1376	**9182**
	CYX+A	23906	1941	**20915**	17608	1563	**14029**	14738	1750	**10779**	14111	1872	**8365**	13091	1295	**10694**
	PMX+A	24216	1802	**20347**	17741	1235	**15418**	14489	2139	**10738**	13810	1315	**11722**	12608	1667	**10080**

Table 6-9 summarises the mean value and standard deviation of the 30 trials for of each approach for the benchmarking problems for minimising the total travel distance. The significant improvements in performance of TCX with respect to each of the other crossover methods are indicated by the *t*-values in Table 6-10. TCX statistically outperforms the other three crossover methods (ORX+A, CYX+A and PMX+A) for the majority of the benchmarking problems ($t < -2.00$).

Table 6-10: *t*-test results of TCX versus (ORX, CYX and PMX) crossover approaches for minimising longest tour.

Problem	Comparison	$m = 3$	$m = 5$	$m = 10$	$m=20$	$m = 30$
MTSP-51	TCX VS ORX+A	-2.91	-4.41	-6.33	-	-
	TCX VS CYX+A	-4.05	-3.06	-6.27	-	-
	TCX VS PMX+A	-3.71	-3.26	-7.79	-	-
MTSP-100	TCX VS ORX+A	-2.38	-6.01	-5.66	-7.56	-
	TCX VS CYX+A	-4.76	-4.73	-5.22	-7.12	-
	TCX VS PMX+A	-2.80	-4.63	-6.39	-8.45	-
MTSP-150	TCX VS ORX+A	-5.88	-4.47	-10.98	-11.15	-11.83
	TCX VS CYX+A	-3.30	-4.28	-11.11	-12.06	-15.55
	TCX VS PMX+A	-4.25	-5.31	-8.85	-14.90	-11.39

6.5.2.2 GA with Seeding for Minimising Longest Tour

The performance of TCX with the GA seeding enabled was also tested. The same seeding technique as Carter and Ragsdale (Carter and Ragsdale, 2006) was employed for minimising longest tour and generating the greedy solutions by iterating through all the salesmen in a round-robin fashion and assigning the closest unassigned city to each salesman in turn. This continued until all the cities were assigned to a salesman.

Table 6-11: Comparison of crossovers with GA seeding for minimising longest tour.

Problem	*m*	(Carter & Ragsdale, 2006)	TCX	Improvement (%)
MTSP-51	3	203	203	0.00%
	5	164	154	6.10%
	10	123	113	8.13%
MTSP-100	3	13556	12726	6.12%
	5	10589	10086	4.75%
	10	9463	7064	25.35%
	20	8388	6402	23.68%
MTSP-150	3	19687	18019	8.47%
	5	14748	12619	14.44%
	10	11158	8054	27.82%
	20	10044	5673	43.52%
	30	8775	5270	39.94%

With the simple GA seeding technique, the average results in the literature (Carter and Ragsdale, 2006) are compared to the results found by our TCX approach. For the 12 problems, TCX has various positive improvements compared to the results obtained in Carter and Ragsdale's study (Carter and Ragsdale, 2006). The greatest improvement is achieved at MTSP-150 (m=20) with 43.52% Therefore, by combining the GA seeding and the proposed TCX operator, better solution quality can be achieved when compared to the approach in Carter and Ragsdale (Carter and Ragsdale, 2006) for the objective of minimum longest tour. Overall, the benchmarking experiments that were conducted for the two objective functions clearly suggest an increase in solution quality for the TCX technique, which is further increased through the introduction of initial seeding of the GA population. Taken together, the results presented in this chapter suggest that the proposed TCX operator enables the GA to find better solutions for solving the MTSP with the two-part chromosome representation.

6.5.3 The Robustness of the TCX

Since the solution quality and performance of the GA is somewhat dependent on the initial parameter settings it is useful to repeat the 'minimise the total travel distance' and 'minimise the longest tour' benchmarking experiments with a range of GA parameters, the results of which show the sensitivity of the GA and TCX operator to initial parameter configurations.

The number of salesman is fixed to m=10 for the three test problems (A) MTSP-51, (B) MTSP-100 and (C) MTSP-150, while varying several GA parameters which typically impact the solution quality and performance of the algorithm, namely GA population (*P = 50, 200, 1000*), crossover probability (*Pc = 0.7, 1.0*), mutation probability (*Pm = 0.02, 0.001*). The default GA parameter settings are *P = 100*, *Pc = 0.85* and *Pm = 0.01*.

Table 6-12: TCX Robustness to varying GA parameters for minimising total travel distance

	Default		*P* = 50		*P* = 200		*P* = 1000		*Pc* = 0.7		*Pc* = 1.0		*Pm* = 0.02		*Pm* = 0.001	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
A	636	17	655	27	623	19	616	18	642	24	645	18	644	20	630	21
B	36921	1964	37970	2950	36072	2119	35173	1511	38669	2389	38872	1844	37317	2412	36723	1991
C	61360	3888	63565	4221	60049	4803	56089	2983	64411	5079	65542	3034	88992	4287	57798	4259

Table 6-12 shows the results of minimising total travel distance, and the variation in GA population size shows an expected behaviour of the GA. In general, a smaller population size (P=50) results in lower solution quality and faster execution time while an increase of population size (P=200, 1000) results in higher or improved solution qualities and slower execution times for each of the three test problems. The variation of crossover probability also resulted in expected behaviour for the GA for all the three test problems. A lower crossover probability (Pc=0.7) reduces the exploration capability of the GA and results in a lower solution quality, while a high crossover probability (Pc=1.0) can destroy good candidate solutions with too much exploration and not enough exploitation of the GA. The variation of mutation probability is more subtle with the results not being in a statistical sense significantly different from the default GA settings for all the three test problems. The results suggest that a higher mutation probability (Pm=0.02), which increases the amount of disruption to each gene in the chromosome, provides a slightly lower solution quality. However, a lower mutation probability (Pm=0.001) produces a higher solution quality since the level of gene disruption is reduced by a factor of 10 on both parts of the chromosome.

Table 6-13 shows the results of minimising longest tour, and the variation in GA population size shows an expected behaviour of the GA. In general, a smaller population size (P=50) results in lower solution quality and faster execution time while an increase (P=200, 1000) results in higher or improved solution qualities and slower execution times for the more difficult test problems *B* and *C*. However, the solution quality for problem *A* does not vary greatly because the GA is able to find a near-optimal solution for each of the different population sizes. In general, a lower crossover probability (Pc=0.7) reduces the exploration capability of the GA, but results in improved solution quality by maintaining good solutions in the population, while a high crossover probability (Pc=1.0) can destroy good candidate solutions with too much exploration and not enough exploitation of the GA. In general, the results suggest that a slightly higher mutation probability (Pm=0.02, default Pm=0.01), produces a higher solution quality. However, a lower mutation probability (Pm=0.001) produces a lower solution quality since the level of gene disruption is reduced by a factor of 10 on both parts of the chromosome.

Table 6-13: TCX Robustness to varying GA parameters for minimising longest tour

	Default		*P* = 50		*P* = 200		*P* = 1000		*Pc* = 0.7		*Pc* = 1.0		*Pm* = 0.02		*Pm* = 0.001	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
A	113	2	115	4	115	4	112	0	113	3	173	7	113	3	121	6
B	7768	492	8073	562	7560	535	7289	467	7639	556	9224	1277	7610	515	8708	353
C	10722	927	11811	1006	10500	827	9631	850	10585	879	31386	885	14004	1064	12201	941

Depending on the type of problem being solved, whether the total travel distance or the longest tour is minimised, the level of exploration versus exploitation is controlled by key GA parameters which produce different solution qualities through their variation. The GA is able to robustly find near-optimal solutions that are typical of the GA parameter variations presented in this section.

6.5.4 Optimality

The TCX operator and GA presented in this chapter are derived from the class of meta-heuristic algorithms used to find *near-optimal* solutions to the symmetric MTSP problem. However, it is interesting to compare the proposed approach to optimal solutions for a subset of small-sized MTSP problems, which include (MTSP-11a):11 cities derived from distances51, (MTSP-11b): all 11 cities from sp11_dist, (MTSP-12a): 12 cities derived from distances51, (MTSP-12b): all 12 cities from uk12_dist and (MTSP-16): contains 16 cities derived from 'distances51'. Each of these small-sized MTSP data sets is described using a symmetric distance (cost) matrix. More specifically, the distances51 data set was obtained from Professor Arthur E. Carter (Carter and Ragsdale, 2006), while the other data sets of sp11_dist and uk12_dist were obtained from http://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html. In addition, GA parameters are default values that are specified in Table 6-5, the number of generations is 50,000 and 30 independent trials were used for the TCX/GA experiments conducted.

Computation of the optimal values for each problem was performed using the brute-force of an exhaustive search, which had CPU times of several orders of magnitude greater than the proposed TCX/GA algorithm. Furthermore, it must be acknowledged that other exact solution methods such as those presented in an early study (Gavish and

Srikanth, 1986) using a branch-and-bound algorithm, seemed to provide exact solutions to several medium-sized MTSP problems that are described using either symmetric/asymmetric and Euclidean/non-Euclidean datasets.

The results presented in Table 6-14 and Table 6-15, indicate that the GA with the proposed TCX operator is able to find optimal and near-optimal solutions to a set of different MTSP problems.

Table 6-14: Optimality Gap for small-sized symmetric MTSP problems for minimising total travel distance

Problem (m=3)	Optimal	TCX_MEAN	TCX_STDEV	TCX_BEST
MTSP-11a	198	198	0	198
MTSP-11b	135	135	1	135
MTSP-12a	199	199	0	199
MTSP-12b	2295	2295	0	2295
MTSP-16	242	247	8	242

Table 6-15: Optimality Gap for small-sized symmetric MTSP problems for minimising longest tour

Problem (m=3)	Optimal	TCX_MEAN	TCX_STDEV	TCX_BEST
MTSP-11a	77	77	0	77
MTSP-11b	73	73	0	73
MTSP-12a	983	987	4	983
MTSP-12b	77	77	0	77
MTSP-16	94	101	7	94

6.5.5 Examination of TCX Operator with an Additional Dataset

Since the proposed TCX operator and GA presented in this chapter utilise the two-part chromosome structure presented in the Carter and Ragsdale study (Carter and Ragsdale, 2006), we focused our benchmarking effort of the different crossover operators onto the Carter data sets (MTSP-51, MTSP-100 and MTSP-150). This allowed for a direct statistical comparison between Carter's results and those presented in this chapter. However, it was interesting to repeat the experiments of the four crossover operators using a completely different symmetric MTSP distance (cost) matrix. As such, we used the sgb128 data set, which was obtained from the following link: http://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html and considers 128 cities in the continental US and lower Canada. The distance matrix considers the actual road distances between the particular cities.

Table 6-16: Comparison of crossover methods for the sgb128 MTSP test problem for minimising the total travel distance

		m = 10				*m* = 15				*m* = 30			
Problem	Crossover	Mean	Stdev	Best	*t*-test	Mean	Stdev	Best	*t*-test	Mean	Stdev	Best	*t*-test
MTSP-128	TCX	42335	1697	**39478**	-	44294	1911	**40843**	-	54696	2189	**50809**	-
	ORX+A	43958	1214	**40922**	-4.33	45674	2133	**40524**	-2.64	56357	1973	**51439**	-3.09
	CYX+A	44133	1471	**40269**	-4.38	45282	1542	**42366**	-2.20	56376	1911	**52967**	-3.17
	PMX+A	44386	1825	**41671**	-4.51	46159	1926	**42024**	-3.76	56369	1623	**53416**	-3.36

Table 6-17: Comparison of crossover methods for the sgb128 MTSP test problem for minimising longest tour

		m = 10				*m* = 15				*m* = 30			
Problem	Crossover	Mean	Stdev	**Best**	*t*-test	Mean	Stdev	**Best**	*t*-test	Mean	Stdev	**Best**	*t*-test
MTSP-128	TCX	6716	428	**5912**	-	5979	404	**5295**	-	4814	435	**4003**	-
	ORX+A	8569	887	**6421**	-10.31	8166	947	**6051**	-11.36	6211	475	**5356**	-11.88
	CYX+A	8464	906	**6566**	-9.55	8007	739	**6324**	-13.19	6261	680	**4776**	-9.82
	PMX+A	8299	848	**6880**	-9.13	8016	1065	**6480**	-9.79	6354	632	**5540**	-11.00

The results in Table 6-16 for minimising the total travel distance suggest that the GA with TCX operator achieve statistically better solutions quality for all benchmarking tests when compared with the three alternative crossover techniques (ORX+A, CYX+A and PMX+A). Furthermore, in the benchmarking experiments minimising the longest tour, the GA and TCX operator achieve significantly better solution qualities for all experiments compared to the alternative methods as illustrated in Table 6-17. These results suggest that the proposed TCX operator robustly achieves good results in different benchmarking experiments.

6.6 **Summary**

It was found that the existing crossover operator for two-part chromosome encoding may limit the GAs' search ability. This chapter proposed a new crossover operator (TCX) for solving the job scheduling problem with further improvements. The proposed TCX can effectively enhance the search ability of the GA by generating more potentially useful solutions. The performance of ORX+A for the modified GA from Chapter 5 was compared to the proposed TCX. Experimental results show that the TCX operator enables the GA to produce higher solution quality than ORX+A when solving the job scheduling problem.

Furthermore, the proposed TCX can also be directly applied to solve the MTSP. The proposed TCX operator was evaluated and compared with three existing crossover operators for solving the MTSP, namely ORX+A, CYX+A and PMX+A. The benchmarking was performed using two different MTSP objective functions, namely the objective of minimising total travel distance and the objective of minimising the makespan. The experimental results show that the TCX operator enables the GA to produce higher solution quality than the existing crossover operators for solving the MTSP.

7. Conclusions and Future Research

This thesis has studied a challenging practical problem of optimising container transfers for a fleet of ASCs at the Patrick AutoStrad container terminal. The objectives of this thesis were to formulate an integrated mathematical model and develop efficient algorithms to coordinate a fleet of ASCs in a confined container terminal so as to improve the performance and productivity. This chapter summarises the principle contributions and provides suggestions for future research.

7.1 Summary

7.1.1 Two Optimisation Models of Container Transfers by ASCs

Firstly, an analytical optimisation model of container transfers using a fleet of ASCs, which integrates the quay-side, yard and land-side operational sub-problems, has been presented. This model incorporated many of the difficult practical constraints encountered at the terminal, which included the vehicle collision avoidance for path planning, multiple level container stacking and sequencing, variable container orientations and vehicular dynamics that require finite acceleration and deceleration times. Secondly, derived from the comprehensive model, a job scheduling model has been formulated to focus on the optimisation of job scheduling.

7.1.2 A Job Grouping Approach for the Comprehensive Model

A job grouping approach to scheduling ASCs for container transfers has been proposed. The job grouping approach can effectively improve the efficiency of the schedule for yard jobs, while reducing the QCs' and ASCs' waiting time by grouping jobs with a guiding function. The job grouping algorithm implementation has been coupled with the path planner, which aims to find the shortest paths for multiple vehicles with collision avoidance. One advantage of this grouping approach is that delay can be effectively reduced for important jobs such as QC and TK related jobs, provided there are enough ASCs available. Moreover, the total ASC waiting time is reduced by

grouping Y2Y jobs with the proposed guiding function, and consequently the overall efficiency of a schedule can be enhanced effectively. Overall, the effectiveness of the job grouping approach was demonstrated by numerical experiments on different sets of data, and it outperforms the sequential planning method. Although there is no benchmark testing problem available, the performance of the job grouping approach is significantly better in a statistical sense than the performance of the sequential planning method.

7.1.3 A Modified Genetic Algorithm for Optimising Job Scheduling

A modified GA-based approach was presented to solve the job scheduling problem and was compared to a sequential job scheduling method for different scheduling scenarios. The proposed approach has been fully implemented on a trial basis in the live scheduling system at the Patrick container terminal and it effectively improves the performance of the seaport container terminal. Overall, the effectiveness of the GA-based approach has been shown by numerical experiments and live testing results on different sets of data, and the modified GA outperforms the sequential planning approach.

7.1.4 A Novel Crossover Method

A new crossover operator for solving the job scheduling problem and the MTSP using GAs was developed. A two-part chromosome encoding technique was employed. This has previously been shown to minimise the size of the search space through the reduction of redundant candidate solutions. It was found that the existing crossover operator for two-part chromosome encoding does limit the GAs' search ability. The proposed TCX has proved to be effective as it enhances the search ability of the GA by generating more possible solutions. Experiments for job scheduling using TCX and ORX+A were conducted and the results showed that TCX can statistically demonstrate improvement. Furthermore, we also evaluated and compared the proposed TCX operator with three existing crossover operators, namely ORX+A, CYX+A and PMX+A. The benchmarking was performed using two different MTSP objective functions: minimising total travel distance and minimising the makespan. The

experimental results showed that the TCX operator can enable the GA to produce higher solution quality than the three existing crossover operators.

7.2 Future Work

This research focuses on developing generic algorithms for coordination of robot teams. The algorithms developed can be applied in other applications of robot teams such as in manufacturing factory, warehouse material handling, and mining. There are a number of topics which still require further work. These topics are beyond the scope of this current thesis, but it is hoped that these topics will inspire future work in this interesting area of planning container transfers at automated container terminals. Future research directions may include the following:

For the job grouping approach, the guiding function may be fine-tuned for extra improvement, and it would also be worthwhile to investigate the possibility of using a dynamical guiding function, so as to automatically fit into different situations.

Regarding the modified GA, using some local search or greedy algorithm to modify the sequence in each ASC will quickly further improve the performance of the algorithm. Accordingly, how to properly integrate a good local search strategy into the GA would be a good topic. Moreover, it would be useful to develop a parallel computing architecture to boost the convergence speed and solution quality of the GA. This would greatly reduce the computation time and increase the solution quality for the same criterion currently employed. It would also improve the productivity of the transhipment process. In addition, the new crossover approach has not been implemented and tested in the actual scheduling system at the Patrick AutoStrad terminal. It would therefore be good to verify its effectiveness when applying it to the purposes of solving the actual scheduling problem for container transfers.

In reality, the scheduling of quay-side, land-side and yard jobs needs to be closely coupled with the notion of frequent replanning. In general, if an ASC has not arrived at the pickup node of job then it may be reassigned to a different job after the application of replanning. The notion of replanning can be Markovian. In this sense, it is only the

current state of the entire seaport environment which governs future path planning, task allocation and scheduling of resources. Therefore, it will be important in the future to investigate the incorporation of replanning into the model and simulation environment .

On the other hand, as uncertainty is a critical concern in port operation, it is worthwhile to investigate and further improve the performance of the job grouping strategy and GA by considering a replanning scheme for both short term and long term planning. It would be interesting to investigate the feasibility of a simultaneous approach (combination of path planning and job scheduling) with POMDP (Partially Observable Markov Decision Process) under uncertainty circumstances which are usually considered more realistic. Also, with increasing automation of yard resources, the performance evaluation and analysis of long-term (e.g. multiple days, weeks or months) planning would be helpful to ensure a high degree of coordination and efficiency for all material handling equipment in the transhipment process including QCs, TKs and ASCs.

Appendix: Methodology of *t*-test

In order to interpret the experimental data within a systematic mathematical framework, it is necessary to adopt a formal decision making procedure. A hypothesis is merely an assertion that may be true or false. But to determine which of these two states is actually the case, the experimental data can be analysed using statistical methods. Therefore, it is imperative that the hypothesis testing is established within a statistical framework that affirms our conclusions. In this appendix we state the hypothesis testing technique described by Walpole *et al.* (Walpole et al., 2006). In general, this hypothesis testing technique is as follows:

State the null hypothesis H0. This is the hypothesis that is statistically tested. The null hypotheses can be stated such that, 'the proposed method ***does provide*** lower values of objective function(s).'

State the alternative hypothesis. The alternative hypotheses can be stated such that, 'the proposed method ***does not provide*** lower values of objective function(s).'

Define the level of significance (α). The level of significance is defined as the probability of committing a Type I error. A Type I error occurs when the null hypothesis (H0) is rejected when it is true. In this thesis, we employ a level of significance α=0.05 (95% confidence).

The experimental setting of our hypotheses requires testing of two independent populations. As such, we define the *test statistic* and *critical region* for testing two known means ($\bar{x}_1, \bar{x}_2$) and associated standard deviations (s_1, s_2) with n_1 and n_2 samples and independent observations. We can employ the Two-Sample Pooled t-Test with (n_1+n_2–2) degrees of freedom:

$$t = \frac{\overline{x}_1 - \overline{x}_2}{s_p \sqrt{1/n_1 + 1/n_2}}$$

where,

$$s_p = \sqrt{\frac{s_1^2(n_1 - 1) + s_2^2(n_2 - 1)}{n_1 + n_2 - 2}}$$

Since, $n_1=n_2=n=30$ for all experiments in this thesis:

$$t = \frac{(\overline{x}_1 - \overline{x}_2)}{\sqrt{\frac{s_1^2 + s_2^2}{n}}}$$

and require that $t > (t_{\alpha/2,\ (n1+n2-2)} = 2.00)$ or $t < (t_{\alpha/2,\ (n1+n2-2)} = -2.00)$ since we have selected $\alpha=0.05$ for $(n_1 + n_2 - 2)$ degrees of freedom. All statistical hypothesis tests in this thesis use two-tailed critical regions to indicate whether there is a significant improvement ($t \leq -2.00$) or a significant degradation ($t \geq 2.00$). Statistical tests resulting in a t-value of ($-2.0 < t < 2.00$) do not provide enough statistical evidence to refute or confirm the null hypothesis. This result indicates similar performance between the two algorithms.

Bibliography

BAKER, C. 1998. High Time for Straddles. *Cargo Systems*.

BALLIS, A. & ABACOUMKIN, C. 1996. A container terminal simulation model with animation capabilities. *Journal of advanced transportation,* 30**,** 37-57.

BEKTAS, T. 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega,* 34**,** 209-219.

BELLMORE, M. & HONG, S. 1974. Transformation of multisalesmen problem to the standard traveling salesman problem. *Journal of the Association Computer Machinery,* 21**,** 500-504.

BENT, R. & HENTENRYCK, P. V. 2006. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research,* 33**,** 875-893.

BIANCHESSI, N. & RIGHINI, G. 2007. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers and Operations Research,* 34**,** 578-594.

BIERWIRTH, C. & MEISEL, F. 2009. A survey of berth allocation and quay crane scheduling problems in container terminals *European Journal of Operational Research,* 202**,** 615-627.

BISH, E. K., LEONG, T., LI, C., NG, J. W. C. & SIMCHI-LEVI, D. 2001. Analysis of a new vehicle scheduling and location problem. *Naval Research Logistics,* 48**,** 363-385.

BO, Z. W., HUA, L. Z. & YU, Z. G. 2006. Optimization of process route by genetic algorithms. *Robotics and Computer-Integrated Manufacturing,* 22**,** 180-188.

BRÄYSY, O. & GENDREAU, M. 2005. Vehicle routing problem with time windows, part II: metaheuristics. *Transportation Science,* 39**,** 119-139.

CARTER, A. E. 2003. *Design and application of genetic algorithms for the multiple traveling salesperson assignment problem.* PhD Research, Virginia Polytechnic Institute and State University.

CARTER, A. E. & RAGSDALE, C. T. 2006. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research,* 175**,** 246-257.

CHABRIER, A. 2006. Vehicle Routing Problem with elementary shortest path based column generation. *Computers and Operations Research,* 33**,** 2972-2990.

CHATTERJEE, S., CARRERA, C. & LYNCH, L. A. 1996. Genetic algorithms and traveling salesman problems. *European Journal of Operational Research,* 93**,** 490-510.

CHOI, E. & TCHA, D.-W. 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research,* 34**,** 2080-2095.

CHRISTOFIDES, N., MINGOZZI, A. & TOTH, P. 1981. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming,* 20**,** 255-282.

CORDEAU, J. F. 2006. A branch and cut algorithm for the dial-a-ride problem. *Operations Research,* 54**,** 573-586.

CORDEAU, J. F., DESAULNIERS, G., DESROSIERS, J., SOLOMON, M. M. & SOUMIS, F. 2002. Vehicle routing problem with time windows. *In:* TOTH, P. & VIGO, D. (eds.) *the vehicle routing problem, SIAM monographs on discrete mathematics and applications.*

CORDEAU, J. F., GENDREAU, M., HERTZ, A., LAPORTE, G. & SORMANY, J. S. 2004. New heuristics for the vehicle routing problem. *Technical Report G-2004-33.* Montreal, Canada.

CORDEAU, J. F. & LAPORTE, G. 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological,* 37**,** 579-594.

CRAINIC, T. G. & KIM, K. H. 2006. *Handbook in Operations Research & Management Science: Transportation,* North-Holland, Elsevier.

CRAMA, Y. & OERLEMANS, A. G. 1994. A column generation approach to job grouping for flexible manufacturing systems. *European Journal of Operational Research,* 78**,** 58-80.

CUS, F. & BALIC, J. 2003. Optimization of cutting process by GA approach. *Robotics and Computer-Integrated Manufacturing,* 19**,** 113-121.

DAGANZO, C. F. 1989. The crane scheduling problem. *Transportation Research,* 23**,** 159-175.

DANIELS, S. C. 1988. *Real-time conflict resolution in automated guided vehicle scheduling.* PhD Research, Pennsylvania State University, USA.

DAVIS, L. Applying Adaptive Algorithms to Epistatic Domains. The International Joint Conference on Artificial Intelligence, 1985 Los Angeles. IEEE Computer Society Press, 162-164.

DEJONG, K. 1975. *An analysis of the behaviour of a class of genetic adaptive systems.* PhD Thesis, University of Michigan.

DESROSIERS, J., DUMAS, Y. & SOUMIS, F. 1986. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences,* 6**,** 301-325.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik,* 1**,** 269-271.

DUINKERKEN, M. B., DEKKER, R., KURSTJENS, S. T. G. L., OTTJES, J. A. & DELLAERT, N. P. 2006. Comparing transportation systems for inter-terminal transport at the maasvlakte container terminals. *OR Spectrum,* 28**,** 469-493.

DUMAS, Y., DESROSIERS, J. & SOUMIS, F. 1991. The pickup and delivery problem with time windows. *European Journal of Operational Research,* 54**,** 7-22.

DUMITRESCU, I. 2005. Polyhedral results for the pickup and delivery travelling salesman problem. *Technical Report CRT-2005-27.* Centre de recherche sur les transports, Universit´ de Montr´al.

FU, L. 2002. A simulation model for evaluating advanced dial-a-ride paratransit systems. *Transportation Research Part A: Policy and Practice,* 36**,** 291-307.

FUKASAWA, R., LYSGAARD, J., DE ARAGÃO, M. P., REIS, M., UCHOA, E. & WERNECK, R. F. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Proceedings of IPCO X, 2004 NewYork, Columbia University.

GALIB. 2007. *A C++ Library of Genetic Algorithm Components* [Online]. http://lancet.mit.edu/ga/. 2011].

GAMBARDELLA, L. M., MASTROLILLI, M., RIZZOLI, A. E. & ZAFFALON, M. 2001. An optimization methodology for intermodal terminal management. *Journal of Intelligent Manufacturing,* 12**,** 521-534.

GAVISH, B. & SRIKANTH, K. 1986. An optimal solution method for large-scale multiple traveling salesmen problems. *Operations Research,* 34**,** 698-717.

GEN, M. & CHENG, R. 1997. *Genetic Algorithms and Engineering Design,* New York, Wiley.

GENDREAU, M., LAPORTE, G. & POTVIN, J. Y. 2002. Metaheuristics for the capacitated vehicle routing problem. *In:* TOTH, P. & VIGO, D. (eds.) *The vehicle routing problem, SIAM monographs on discrete mathematics and applications.*

GOLDBERG, D. E. 1989. *Genetic algorithms in search, optimization, and machine learning,* Boston, MA, USA, Addison Wesley Longman.

GOLDBERG, D. E. & LINGLE, J. R. 1985. Alleles, Loci and the TSP. *The First International Conference on Genetic Algorithms and Their Applications.* New Jersey.

GOLDEN, B. L., WASIL, E. A., KELLY, J. P. & CHAO, I. M. 1998. Metaheuristics in vehicle routing. *In:* CRAINIC, T. & LAPORTE, G. (eds.) *Fleet management and logistics.* Boston.

GOLIAS, M. M., BOILE, M. & THEOFANIS, S. 2009. Berth scheduling by customer service differentiation: a multi-objective approach. *Transportation Research Part E: Logistics and Transportation Review,* 45**,** 878-892.

GORENSTEIN, S. 1970. Printing press scheduling for multi-edition periodicals. *Management Science,* 16**,** B373-383.

GRUNOW, M., HANS-OTTO, G. & LEHMANN, M. 2004. *Dispatching multi-load AGVs in highly automated seaport container terminals,* New York, Springer.

GÜNTHER, H. O. & KIM, K. H. 2006. Container terminals and terminal operations. *OR Spectrum,* 28**,** 437-445.

HAGHANI, A. & JUNG, S. 2005. A dynamic vehicle routing problem with time-dependent travel times. *Computers and Operations Research,* 32**,** 2959-2986.

HALPERN, J. 1977. Shortest route with time dependent length of edges and limited delay possibilities in nodes. *Mathematical Methods of Operations Research,* 21**,** 117-124.

HARTMANN, S. 2004. *A general framework for scheduling equipment and manpower at container terminals,* New York, Springer.

HOLLAND, J. 1975. *Adaptation in natural and artificial systems,* Michigan, University of Michigan Press, Ann Arbor.

HOLLAND, J. H. 1992. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence,* Cambridge, MIT Press.

IBM. *ILOG Optimisation* [Online]. Available: www.ilog.com/products/optimization/ [Accessed June 30 2010].

IMAI, A., NISHIMURA, E., HATTORI, M. & PAPADIMITRIOU, S. 2007. Berth allocation at indented berths for mega-containerships. *European Journal of Operational Research,* 179**,** 579-593.

IMAI, A., NISHIMURA, E. & PAPADIMITRIOU, S. 2003. Berth allocation with service priority. *Transportation Research – Part B,* 37**,** 437-457.

IMAI, A., NISHIMURA, E. & PAPADIMITRIOU, S. 2006. Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. *European Journal of Operational Research,* 171**,** 373-389.

IRNICH, S. & VILLENEUVE, D. 2003. The shortest path problem with resource constraints and k-cycle elimination for k>=3. *Technical Report G-2003-55.* Montreal, Canada.

KALLEHAUGE, B., LARSEN, J. & MADSEN, O. B. G. 2001. Lagrangean duality applied on vehicle routing with time windows—experimental results. *Technical Report IMM-REP-2000-8, Informatics and Mathematical Modelling.* Technical University of Denmark.

KELLEGÖZ, T., TOKLU, B. & WILSON, J. 2008. Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem. *Applied Mathematics and Computation,* 199**,** 590-598.

KIM, C. W. & TANCHOCO, J. M. A. 1991. Conflict-free shortest-time bi-directional AGV routing. *International Journal of Production Research,* 29**,** 2377-2391.

KIM, K. H. & BAE, J. W. 2004. A look-ahead dispatching method for automated guided vehicles in automated port container terminals. *Transportation Science,* 38**,** 224-234.

KIM, K. H., KANG, J. S. & RYU, K. R. 2004a. A beam search algorithm for the load sequencing of outbound containers in port container terminals. *OR Spectrum,* 26**,** 93-116.

KIM, K. H. & NGUYEN, V. D. 2009. A dispatching method for automated lifting vehicles in automated port container terminals. *Computers & Industrial Engineering,* 56**,** 1002-1020.

KIM, K. H. & PARK, K. T. 2003. A note on a dynamic space-allocation method for outbound containers. *European Journal of Operational Research,* 148**,** 92-101.

KIM, K. H. & PARK, Y. M. 2004. A crane scheduling method for port container terminals. *European Journal of Operational Research,* 156**,** 752-768.

KIM, K. H., WON, S. H., LIM, J. K. & TAKAHASHI, T. 2004b. An architectural design of control software for automated container terminals. *Computers & Industrial Engineering,* 46**,** 741-754.

KOO, P. H., LEE, W. S. & JANG, D. W. 2004. Fleet sizing and vehicle routing for container transportation in a static environment. *OR Spectrum,* 26**,** 193-209.

KOZAN, A. & PRESTON, P. 2007. *Mathematical modelling of container transfers and storage locations at seaport terminals,* Berlin, Springer.

KULKARNI, A. J. & TAI, K. 2010,. Probability Collectives: A multi-agent approach for solving combinatorial optimization problems. *Applied Soft Computing,* 10**,** 759-771.

LAPORTE, G. & SEMET, F. 2002. Classical heuristics for the capacitated vehicle routing problem. *In:* TOTH, P. & VIGO, D. (eds.) *The vehicle routing problem, SIAM monographs on discrete mathematics and applications.*

LAU, H., PRATLEY, T., LIU, D. K., HUANG, S. & PAGAC, D. 2008. An implementation of prioritized path planning for a large fleet of autonomous straddle carriers. *18th Triennial Conference of the International Federation of Operational Research Societies (IFORS).* Sandton, South Africa.

LAU, H. Y. K. & ZHAO, Y. 2007. Integrated scheduling of handling equipment at automated container terminals. *Annals of Operations Research,* 159**,** 373-394.

LEE, D. H., WANGA, H. Q. & MIAO, L. 2008. Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E: Logistics and Transportation Review,* 44**,** 124-135.

LI, C. L. & VAIRAKTARAKIS, G. L. 2004. Loading and unloading operations in container terminals. *IIE Transactions,* 36**,** 287-297.

LI, F., GOLDEN, B. & WASIL, E. 2005. Very large-scale vehicle routing: new test problems, algorithms and results. *Computers and Operations Research,* 32**,** 1165-1179.

LI, H. & LIM, A. A metaheuristic for the pickup and delivery problem with time windows. Tools with Artificial Intelligence, Proceedings of the 13th International Conference on, 2001. 160-167.

LIU, C.-I., JULA, H., VUKADINOVIC, K. & IOANNOU, P. 2004. Automated guided vehicle system for two container yard layouts. *Transportation Research Part C: Emerging Technologies,* 12**,** 349-368.

LIU, D. K. & KULATUNGA, A. K. 2007. Simultaneous planning and scheduling for multi-autonomous vehicles. *In:* DAHAL, K. P., TAN, K. C. & COWLING, P. I. (eds.) *Evolutionary Scheduling.* Berlin: Springer-Verlag.

LIU, W., LI, S., ZHAO, F. & ZHENG, A. An ant colony optimization algorithm for the Multiple Traveling Salesmen Problem. IEEE Conference on Industrial Electronics and Applications, 2009. 1533 - 1537.

LOGENDRAN, R., CARSON, S. & HANSON, E. 2005. Group scheduling in flexible flow shops. *International Journal of Production Economics,* 96**,** 143-155.

LU, Q. & DESSOUKY, M. 2004. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science,* 38**,** 503-514.

LYSGAARD, J., LETCHFORD, A. N. & EGLESE, R. W. 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming, Series A,* 100**,** 423-445.

MALMBORG, C. 1996. A genetic algorithm for service level based vehicle scheduling. *European Journal of Operational Research,* 93**,** 121-134.

MEERSMANS, P. J. M. & WAGELMANS, A. P. M. 2001. Dynamic Scheduling of Handling Equipment at Automated Container Terminals. *Erasmus Research Institute of Management,* ERS-2001-69-LIS**,** 1-20.

MESTER, D., BRÄYSY, O. & DULLAERT, W. 2007. A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications,* 32**,** 508-517.

MUTHUVELU, N., LIU, J., SOE, N. L., VENUGOPAL, S., SULISTIO, A. & BUYYA, R. 2005. A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids. *Proceedings of the 2005 Australasian workshop on Grid computing and e-research.* Newcastle, New South Wales, Australia: Australian Computer Society.

NANRY, W. & BARNES, J. 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological,* 34**,** 107-121.

NARASIMHAN, A. & PALEKAR, U. S. 2002. Analysis and algorithms for the transtainer routing problem in container port operations. *Transportation Science,* 36**,** 63-78.

NELMES, G. 2005. Container port automation. *In:* CORKE, P. & SUKKARIEH, S. (eds.) *The 5th International Conference on Field and Service Robotics.* Port Douglas, Australia: Springer.

OLIVER, I. M., SMITH, D. J. & HOLLAND, J. R. C. A study of permutation crossover operators on the TSP. The Second International Conference on Genetic Algorithms, 1987 New Jersey. 224-230.

OSMAN, M., ABO-SINNA, M. & MOUSA, A. 2005. An effective genetic algorithm approach to multiobjective routing problems (morps). *Applied Mathematics and Computation,* 163**,** 769-781.

PARK, Y. B. 2001. A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. *International Journal of Productions Economics,* 73**,** 175-188.

PARRAGH, S., DOERNER, K. & HARTL, R. 2008. A survey on pickup and delivery problems: Part I: Transportation between customers and depot. *Journal fürBetriebswirtschaft,* 52**,** 21-51.

PISINGER, D. & ROPKE, S. 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research,* 34**,** 2403-2435.

PSARAFIS, H. 1980. A dynamic programming solution to the single vehicle, many-tomany, immediate request dial-a-ride problem. *Transportation Science,* 14**,** 130-154.

PSARAFIS, H. 1983. An exact algorithm for the single-vehicle many-to-many dial-aride problem with time windows. *Transportation Science,* 17**,** 351-357.

REINELT, G. 2001. *TSPLIB* [Online]. Available: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/.

RULAND, K. S. & RODIN, E. Y. 1997. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & Mathematics with Applications,* 33**,** 1-13.

RYAN, J. L., BAILEY, T. G., MOORE, J. T. & CARLTON, W. B. Reactive tabu search in unmanned aerial reconnaissance simulations. The 30th conference on Winter simulation 1998 Los Alamitos, CA, USA. IEEE Computer Society Press, 873-879.

SAVELSBERGH, M. W. P. & SOL, M. 1995. The general pickup and delivery problem. *Transportation Science,* 29**,** 17-29.

SEXTON, T. R. & BODIN, L. D. 1985a. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. *Transportation Science,* 19**,** 378-410.

SEXTON, T. R. & BODIN, L. D. 1985b. Optimizing single vehicle many-to-many operations with desired delivery times: II. routing. *Transportation Science,* 19**,** 411-435.

SINGH, A. & BAGHEL, A. S. 2009. A new grouping genetic algorithm approach to the multiple traveling salesperson problem. *Soft Computing,* 13**,** 95-101.

SMITH, G. C. & SMITH, S. S. F. 2002. An enhanced genetic algorithm for automated assembly planning. *Robotics and Computer-Integrated Manufacturing,* 18**,** 355-364.

SPEARS, W. M. & DE JONG, K. A. An Analysis of Multi-Point Crossover. Foundations of Genetic Algorithms, 1991 Morgan Kaufmann. 301-315.

STEENKEN, D. 1992. Fahrwegoptimierung am Containerterminal unter Echtzeitbedingungen *OR Spectrum,* 14**,** 161-168.

STEENKEN, D., VOB, S. & STAHLBOCK, R. 2005. *Container terminal operation and operations research – a classification and literature review,* New York, Springer.

STEENKEN, D., VOß, S. & STAHLBOCK, R. 2004. Container terminal operation and operations research - a classification and literature review. *OR Spectrum,* 26**,** 3-49.

SVESTKA, J. A. & HUCKFELDT, V. E. 1973. Computational experience with an m-salesman traveling salesman algorithm. *Management Science,* 17**,** 790-799.

TAILLARD, E. 1993. Parallel iterative search methods for vehicle routing problems. *Networks,* 23**,** 661-673.

TANG, L., LIU, J., RONG, A. & YANG, Z. 2000. A multiple traveling salesman problem model for hot rolling scheduling in Shangai Baoshan Iron & Steel Complex. *European Journal of Operational Research,* 124**,** 276-282.

TARANTILIS, C., IOANNOU, G. & PRASTACOS, G. 2005. Advanced vehicle routing algorithms for complex operations management problems. *Journal of Food Engineering,* 70**,** 455-471.

VIS, I. & HARIKA, I. 2005. *Comparison of vehicle types at an automated container terminal,* New York, Springer.

VIS, I. & KOSTER, R. 2006. Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research,* 147**,** 1-16.

VIS, I. F. A. & HARIKA, I. 2004. Comparison of vehicle types at an automated container terminal *OR Spectrum,* 26**,** 117-143.

VIS, I. F. A., KOSTER, R., DE ROODBERGEN, K. J. & PEETERS, L. W. P. 2001. Determination of the number of automated guided vehicles required at a semi-automated container terminal. *Journal of the Operational Research Society,* 52**,** 409-417.

WALPOLE, R. E., MYERS, R. H., MYERS, S. L. & YE, K. 2006. *Probability & statistics for engineers & scientists*, Prentice Hall.

WANG, X. B. & REGAN, A. C. 2002. Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological,* 36**,** 97-112.

YANG, C. H., CHOI, Y. S. & HA, T. Y. 2004. Performance evaluation of transport vehicle at automated container terminal using simulation. *OR Spectrum,* 26**,** 149-170.

YUAN, S., LAU, H., LIU, D. K., HUANG, S. D., DISSANAYAKE, G., PAGAC, D. & PRATLEY, T. 2009. Simultaneous dynamic scheduling and collision-free path planning for multiple autonomous vehicles. *Proceedings of the 2009 IEEE International Conference on Information and Automation.* Macau, China.

YUAN, S., SKINNER, B. T., HUANG, S. D., LIU, D. K., DISSANAYAKE, G., LAU, H. & PAGAC, D. 2011. A job grouping approach for planning container transfers at automated seaport container terminals. *Advanced Engineering Informatics,* 25**,** 413-426.

ZHAO, W. & GOODCHILD, A. V. 2010. The impact of truck arrival information on container terminal rehandling. *Transportation Research Part E: Logistics and Transportation Review,* 46**,** 327-343.