Tabu Search for Ship Routing and Scheduling

A thesis submitted for the degree of Doctor of Philosophy

By: Khaled Al-Hamad

School of Information Systems, Computing and Mathematics
Brunel University

December 2006

Abstract

This thesis examines exact and heuristic approaches to solve the Ship Routing and Scheduling Problem (SRSP). The method was developed to address the problem of loading cargos for many customers using heterogeneous vessels. Constraints relate to delivery time windows imposed by customers, the time horizon by which all deliveries must be made and vessel capacities. The objective is to minimise the overall operation cost, where all customers are satisfied. Two types of routing and scheduling are considered, one called single-cargo problem, where only one cargo can be loaded into a ship, and the second type called multi-cargo problem, where multiple products can be carried on a ship to be delivered to different customers.

The exact approach comprises two stages. In the first stage, a number of candidate feasible schedules is generated for each ship in the fleet. The second stage is to model the problem as a set partitioning problem (SPP) where the columns are the candidate feasible schedules obtained in the first stage. The heuristic approach uses Tabu Search (TS). Most of the TS operations, such as insert and swap moves, tenure, tabu list, intensification, and diversification are used. The results of a computational investigation are presented. Solution quality and execution time are explored with respect to problem size and parameters controlling the tabu search such as tenure and neighbourhood size.

The results showed that the average of the solution gap between TS solution and SPP solution is up to 28% (for small problems) and up to 18% for large problems. However, obtaining an optimal solution requires a large amount of computer time to produce the solution compared to obtaining approximate solutions using the TS approach. The use of Tabu Search for SRSP is novel and the results indicate that it is viable approach for large problems.

Acknowledgements

First I thank God for helping me to succeed in the accomplishment of this thesis.

I wish to express my deepest gratitude and sincere appreciation to Professor Ken Darby-Dowman for his continuous guidance, encouragement, careful reading, correction, and constructive criticism of my thesis.

I am grateful to Dr Cormac Lucas and Mathematical department staff for any help they provided me during this study.

Special thank to my parents who always prayed to God to support and help me during the four years I spent doing my research.

To my wife Wedad, who supported and encouraged me to complete my thesis. I also dedicate this work to my children Mohammad, Omar, Ibrahim, Munira and Ola.

I am also grateful to my brothers and sisters for their support and who prayed to God to help me.

Finally, I would like to thank all my friends for their good wishes that I succeed.

Table of Contents

Abstract

Acknowledgements

Chapter 1: Introduction

1.1 Transportation Management 1

1.1.1 The Economic Importance of Transportation 1

1.1.2 Modes of Transportation 1

1.2 Sea Transportation 2

1.2.1 Increase in the number of ships world wide 2

1.2.2 Type of ships 3

1.2.3 Cost of operation and other expenses 4

1.2.4 Terms in Maritime Transportation Systems 5

1.2.5 Classification Scheme for The Ship and Scheduling Problem 6

1.3 Methods Of Scheduling 9

1.4 The aims of this research project 10

Chapter 2: Solution Techniques for Routing and Scheduling 12

2.1 Integer Programming (IP) 12

2.1.1 Branch and Bound B&B method 14

2.1.2 Illustration of (0-1) integer programming 19

2.1.3 Applications of (0-1) integer programming 20

2.2 Approximate Technique (Heuristic methods) 22

2.2.1 Tabu Search (TS) 22

2.2.1.1 Tabu Search Process 23

A. Neighbourhood Search 23

B. Tabu List 27

C. Tabu Tenure 30

D. Aspiration Criteria 30

E. Longer term memory 31

F. Intensification and Diversification strategies 31

2.2.1.2 Tabu Search Applications 32

2.2.2 Simulated Annealing (SA) 34

2.2.2.1 Applications on SA 38
2.2.3 Genetic Algorithms (GAs) 40
2.2.3.1 Background of Genetic Algorithms (GAs) 40
2.2.3.2 Applications using GAs 46

Chapter 3: A Review of the Literature on Ship Routing and Scheduling 51
3.1 An Overview of Research on Transportation Routing and Scheduling 51
3.2 Land Routing and Scheduling Models 51
3.2.1 Vehicle Routing and Scheduling 51
3.2.2 Railroad Scheduling Models 52
3.3 Airline Routing and Scheduling 53
3.4 Ship Routing and Scheduling Modes 54
3.4.1 Liner Operations 55
3.4.2 Tramp Operations 58
3.4.3 Industrial Operations 58

Chapter 4: Ship Scheduling: Specification and algorithms 63
4.1 Problem specification 63
4.1.1 Definition of the Ship Routing and Scheduling Problem (SRSP) 63
4.1.2 Classes of SRSP 63
4.1.3 Notation 64
4.1.4 Exact algorithm 65
4.1.4.1 Generation of the candidate schedules 67
A. Single-cargo 68
B. Multi-cargoes 73
4.1.5 Heuristic Approach 81
4.1.5.1 Tabu Search 83
A. Single-cargo 84
B. Multi-cargo 89
C. Intensification and Diversification 91

Chapter 5: Design and implementation of solution Approaches 94
5.1 Introduction 94
5.2 Exact Approach Design 94
5.3 Approximate Approach 100
5.3.1 Conditions 104
5.4 Model Validation 106
Chapter 6: Computational Experiment 107
6.1 Design 107
6.1.1 Input data 107
6.2 Tabu Search Parameter evaluation 109
6.2.1 TS Parameters Evaluation 110
6.2.1.1 Selecting Nbr_i by Systematic or Random method .. 110
6.2.1.2 Size of Neighbourhood (*NZ*) 111
6.2.1.3 Diversification ($Divert_i$) 114
6.2.1.4 Intensification 115
6.2.1.5 Minimum Tenure (*MinTn*) 117
6.2.1.6 Problem size 119
6.2.2 Single-cargo and Multi-cargo Comparison 120
6.2.3 Comparison between SPP approach and TS approach 122
Chapter 7: Conclusions and Further Researches 126
Bibliography 130
Appendices 140

List of Figures

Figure 2.1: Branch and bound first step 16

Figure 2.2: Branch and bound second step 17

Figure 2.3: Branch and bound tree 18

Figure 2.4: Depth-First Search (DFS) and Breadth-First Search BFS) 19

Figure 2.5: Searching within neighbourhood distances less than 80 miles 26

Figure 2.6: Searching within neighbourhood distances less than 95 miles 27

Figure 2.7: Module insulation problem 28

Figure 2.8: Initial solution for module insulation problem 28

Figure 2.9: New solution for module insulation problem 28

Figure 2.10: Current solution for module insulation problem 31

Figure 2.11: TSP with four customers 35

Figure 2.12: Genetic process 41

Figure 2.13: Chromosome and fitness 42

Figure 2.14: Second parent 42

Figure 2.15: Five arrows represent the possible position
for crossover point 43

Figure 2.16: One-point crossover 43

Figure 2.17: PMX crossover operation 44

Figure 2.18: Several common mutation operations 44

Figure 2.19: Swap nodes 45

Figure 2.20: Two parents with 2-point crossover 47

Figure 2.21: Crossover operation 47

Figure 2.22: Duplicated genes 47

Figure 2.23: Duplicated genes 48

Figure 2.24: Two cutting 49

Figure 2.25: OX method 49

Figure 4.1: Generating all candidate feasible schedule flowchart for
Single-cargo 70

Figure 4.1: Generating all candidate feasible schedule flowchart for
Multi-cargo 75

Figure 4.3: If the answer is YES for question (1) 78

Figure 4.4: If the answer is YES for question (4) 79

Figure 4.5: If the answer is NO for question (4) 79
Figure 4.6: If the answer is NO for question (3) 80
Figure 4.7: Presentation of trip in Multi-cargo case 83
Figure 4.8: Two ships delivered five cargo-ports 85
Figure 4.9: Deleting cargo-port 2 from ship 2 and inserting in ship 1 85
Figure 4.10: Potential insert trial places for cargo-port *i* 86
Figure 4.11: Swap move 86
Figure 4.12: Tabu list and the Tabu tenure 87
Figure 5.1: Structure of Exact Method 94
Figure 5.2: User Interface 95
Figure 5.3: Generation of candidate feasible schedule 97
Figure 5.4: Model formulation using MPL 99
Figure 5.5: Structure of Approximate Method 100
Figure 5.6: Initial solution 101
Figure 5.7: The neighbourhood of cargo-port 11 102
Figure 5.8: Results obtained by applying tabu search 103
Figure 6.1: Presents different size problems 120

List of Tables

Table1.1: Lloyd's Register of Shipping "World Fleet Statistics" 3

Table 1.2: capital costs for various classes of bulk carriers 4

Table 1-3: Three different ship types with their fuel consumption by tonnes for each day at normal speed 5

Table 2.1: The distances (miles) between each pair of customers 25

Table 2.2: Neighbours that are with less than 80 miles apart 25

Table 2.3: Neighbours that are with less than 95 miles apart 26

Table 2.4: Distance (miles) between customers, where 0 denotes to the depot ... 37

Table 2.5: Bracketed values in the Solution columns represent cost function values 38

Table 2.6: Distance between each customer (miles) 41

Table 4.1: Possible set of candidate feasible schedules 67

Table 4.2: All candidate feasible schedules for Single-cargo problem 73

Table 4.3: All candidate feasible schedules for Multi-cargo problem 81

Table 4.4: Example of schedule of *n* cargo-ports 83

Table 4.5: Nbr_i for cargo-port *i* 88

Table 6.1: All generated random data involved in SRSP 107

Table 6.2: Problem features to Evaluate Selecting Nbr_i 110

Table 6.3: Average cost obtaining in the evaluation of the procedure to select Nbr_i 111

Table 6.4: Problem features to Evaluate Selecting *NZ* 111

Table 6.5: Average number of iterations for each *NZ* for each case 112

Table 6.6: the difference in quality of the objective function for all cases 113

Table 6.7: The percentage of times each value of *NZ* produced the best solution among forty problem instances 113

Table 6.8: Three different problem size with number of iterations 114

Table 6.9: Results of applying diversification on three different problem sizes 115

Table 6.10: Results of applying first method of intensification on three different problem sizes 116

Table 6.11: Results of applying second method of intensification on three different problem sizes 117
Table 6.12: The evaluation of *MinTn* 118
Table 6.13: problems of different *MinTn* 119
Table 6.14: Problem features to evaluate problem size 119
Table 6.15: The objective cost and solution time for four different problem sizes 121
Table 6.16: SPP and TS computational results 123

Notation and Abbreviation

The following notation is used in this thesis:

Data:

$V = \{1,..,m\}$, denotes the set of m ships to be scheduled, indexed by v, where $v \in V$

$N = \{0,1,..,n\}$, denotes the set of n cargo-ports to be visited, indexed by i, where $i \in N$ and 0 denotes to origin and $N \setminus \{0\}$ denotes cargo-ports

AV_v denotes to the availability time of ship v at the origin $\forall v \in V$

e_i earliest arrival time for cargo-port i $\forall i \in \{N \setminus 0\}$

l_i latest arrival time for cargo-port i $\forall i \in \{N \setminus 0\}$

Q_i cargo quantity of cargo-port i (tons) $\forall i \in \{N \setminus 0\}$

d_{ik} distance (in days) between cargo port i and cargo port k, where $i,k \in N$

CT_v capacity of ship v (tons) $\forall v \in V$

PC_{iv} port entrance due (fee) at cargo port i for ship v $\forall i \in \{N \setminus 0\}, \forall v \in V$

SP_v sailing cost using ship v (per day) $\forall v \in V$

LD_{iv} time required for loading cargo-port i onto ship v (days) $\forall i \in \{N \setminus 0\}\ \forall v \in V$

UL_{iv} time required for unloading cargo-port i from ship v (days) $\forall i \in \{N \setminus 0\}\ \forall v \in V$

OC_{iv} operating cost for ship v to handle cargo-port i, including loading and unloading costs $\forall i \in \{N \setminus 0\}\ \forall v \in V$

WP_v waiting cost (idle in the ocean) for ship v (per day) $\forall v \in V$

S_v denotes to a set of candidate schedules are available for ship v, and j indexed to specific schedule $\forall v \in V$

C_{vj} denotes to the cost for using schedule j for ship v $\forall v \in V$, $j \in S_v$

$$SC_{ivj} = \begin{cases} 1 & \text{if schedule } j \text{ for ship } v \text{ servicing cargo } i \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i \in \{N \setminus 0\}\ \forall v \in V \quad j \in S_v$$

Decision Variables

f_{iv} actual arrival time for ship v to cargo-port i $\quad \forall i \in \{N \setminus 0\}, \forall v \in V$

wv_{iv} duration of waiting (idle) time for ship v until time-window of cargo-port i opens (days) $\forall i \in \{N \setminus 0\}, \forall v \in V$

$$x_{vj} = \begin{cases} 1 & \text{if schedule } j \text{ for ship } v \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \forall v \in V \quad j \in S_v$$

The following notations are used in the Tabu Search (TS) approach.

Notations:

TN	Tenure (Tabu list size)
MinTn	Minimum tenure size
MaxTn	Maximum tenure size
$InsTn_i$	cargo-port i entered insert tabu list, where $i \in N$
$SwTn_i$	cargo-port i entered swap tabu list, where $i \in N$
IncTn	number of iterations to increase tenure by one
Nbr_i	neighbourhood set of cargo-port i, where $i \in N$
NZ	neighbourhood size
$NbrSel_i$	cargo-port i is selected to form it's neighbourhood, where $i \in N$
$NbItr_i$	number of iterations within the neighbourhood of cargo-port i, where $i \in N$
AllItr	number of iterations for the entire problem
Rpt_i	number of times cargo-port i entered tabu list, where $i \in N$
$Divert_i$	cargo-port i will be hold for a number of time of forming new neighbourhood, not to be chosen, where $i \in N$
S	the current solution
S^*	the best-known solution

f^*	value of S^*
$N(S)$	the neighbourhood of S
$N'(S)$	non-tabu subset (admissible subset)

The following of abbreviations are used,

B&B	Branch and Bound
BFS	Breadth-First Search
DFS	Depth-First Search
GA	Genetic Algorithm
IP	Integer Programming
LNS	Local Neighbourhood Search
LP	Linear Programming
LR	Linear Relaxation
MIP	Mixed Integer Programming
PIP	Pure Integer programming
SA	Simulated Annealing
SPP	Set Partitioning Problems
SRSP	Ship Routing and Scheduling Problem
TS	Tabu Search
TSP	Travelling Salesmen Problem
VRP	Vehicle Routing Problem
ZIP	Zero-one Integer Programming

Chapter 1: Introduction

1 Introduction

1.1 Transportation Management

1.1.1 The Economic Importance of Transportation

The continuous growth of the world population and of its standard of living, combined with reduction of local resources, increases the dependence of the world economy on international trade. Initial data available for 2004 indicate that growth in world output was 3.8%, which is 1.3% above the 2.5% recorded for 2003[80]. This reflects the fact that most regions of the world experienced simultaneously positive economic growth.

Freight transportation is a vital component of the economy. It supports production, trade, and consumption activities by ensuring the efficient movement and timely availability of raw materials and finished goods. Freight transportation represents a significant part of the cost of products, as well as of the national expenditure of any country (Crainic and Laporte [35]). This explains the highly competitive environment for freight transportation firms. Carriers have to quickly adjust to changing economic and regulatory conditions; offer reliable, high quality, low cost services to their customer, and clearly make profit. Both the planning and operational units of a company have to work together towards the achievement of these goals.

1.1.2 Modes of Transportation

Transportation can be accomplished in different ways: overland by road or rail, by air, or by sea. Each mode has its own particular characteristics. Ronen [92] and Christiansen et al.[29] mention several significant differences between sea, air, and land transportation modes:

1- The ability of ports to accommodate a particular ship differes (e.g. a deep water harbour may be required for a large ship).

2- A voyage by a ship (also aircraft) usually has one or very few unloading locations, whereas land based vehicles frequently have many unloading locations.

3- Ships come in a large variety of sizes with different compartment sizes. When a shipment consists of multiple products that have to be loaded on the same vessel, the product quantities will have to be adjusted to fit the available compartment.

4- Higher uncertainty is involved in scheduling ships due to much longer voyages.

5- Destination change can occur whilst the ship is underway. This is seldom the case, for aircraft or and based vehicles.

On the other hand, in some aspects, aircraft are more similar to ships:

1- Both ships and aircraft have uncertainty in their operations due to their higher dependence on technology and weather conditions.

2- Both ships and aircraft pay port fees and both operate on international routes.

3- Aircraft and ships involve large capital and operating costs, compared to road vehicles.

Since this research is focused on ship routing and scheduling, it is important to give background and information about ships.

1.2 Sea Transportation

1.2.1 Increase in the number of ships world wide

The latest available sea transportation statistics show that the world shipping fleet has increased to a record level of 633,321 million gross tons in 2004, an increase of 28 million gross tons from a year earlier. Over the last ten years there has been a growth of 33% in the number of ships, resulting in 89,960 ships operating in 2004. Table 1.1 shows the total number of ships and gross tonnage in the period 1990-2004. In 2004 world shipments of tanker cargos reached 2.32 billion tons, a growth of 4.2% over the previous year. About 76.4% of this tanker trade is in crude oil, with the remainder in petroleum products. In 2004, 1,397 new building contracts were placed for various ship types, an impressive increase of 20.5% in comparison with 2003 (1159 contracts)[80]. Approximately, 70% of the value and 90% of the volume of all the goods transported worldwide are made by sea. [6]. These facts emphasize the reliance of the world economy on seaborne trade and hence highlight the need for

well-organized and reliable maritime transportation systems. Routing and scheduling of ships requires a significant level of fleet management planning.

Year	Number of ships	Gross tons 000 GT
1990	78,336	423,627
1991	80,030	436,027
1992	79,845	444,305
1993	80,655	457,915
1994	80,676	475,859
1995	82,890	490,662
1996	84,264	507,873
1997	85,494	522,197
1998	85,258	531,893
1999	86,817	543,610
2000	87,546	553,054
2001	87,939	574,551
2002	89,010	585,583
2003	89,899	605,218
2004	89,960	633,321

Table1.1: Lloyd's Register of Shipping "World Fleet Statistics"

1.2.2 Type of ships

There are four type of ships employed around the world:

1- *Passenger ships*, where people are carried on vacation trips of various durations.

2- *Bulk carriers*, which are designed to carry specific commodities (liquid bulk and dry bulk ships). They vary in size. The Ultra large Crude Carriers (ULCC) are up to 500,000 deadweight tons (dwt), where as the typical is size around 100,000 to 150,000 dwt, such as Suezmax [80].

3- *General cargo*, where the ship is designed to carry non-bulk cargoes. The traditional ships were less than 10,000 dwt. Recently, these ships have been replaced by container ships, where loading and unloading can be implemented efficiently.

4- *Roll on-Roll off* are ships designed to carry vehicles and trains to be loaded directly on board.

1.2.3 Cost of operation and other expenses

Because of the ship size, the major feature of the economics of shipping is capital cost. Cruise ships represent the most expensive class of ships. For example, the Queen Mary2, which launched the first trip in 2004 from Southampton to Fort Lauderdale in United States, cost 800 million dollars [60]. Table 1.2 illustrates capital costs for various classes of bulk carriers [80].

Classes of bulk carriers	Capital cost (Millions of dollars)
2,500 TEU full containership	42
30-50,000 dwt bulk carrier	30
80-105,000 dwt tanker	56
75,000 m^3 LPG	77
250-280,000 dwt tanker	105

TEU: 20-foot equivalent unit
LPG: liquefied petroleum gas

Table 1.2: capital costs for various classes of bulk carriers

Port entrance dues vary between ports, and differences also occur due to ship size, (loaded or empty), and time remaining in port. For example, in the port of Duisburg in Germany, a ship of 310,513 dwt, loaded with 158,503 tonnes, will be charged 27,013 Euro for up to three days, while the charge will increase to 32,447 Euro for more than three days, up to a maximum of ten days. A bulk carrier of 47,471 dwt, loaded with 42,904 tonnes will be charged 4132 Euro for up to three days, while charge will increase to 4963 Euro for more than three days, up to a maximum of ten days[61].

The average speed of ships is about 15 Knots (about 28 km per hour). A ship can travel about 575 km per day, under such circumstances. Newer ships can travel at speeds of between 25 to 30 knots (45 to 55 km per hour). The cost of fuel consumption (bunker) depends on ship size and the speed. Table 1.3 presents three

different ship types with their fuel consumption by tonnes for each day at normal speed, where the price of fuel per tonne equal to 71 dollars according to Kuwait Oil Tanker Company (KOTC) officials (price applicable in May, 2006) [5].

Ship Type	Bunker Consumption **Sailing**		Bunker Consumption **Idle**	
	Volume Tonne/Day	Cost (Dollars)	Volume Tonne/Day	Cost (Dollars)
ULCC	96	6816	10	710
VLCC	96	6816	3	213
LPG	86	6106	7	497

ULCC: Ultra Large Crude Carrier
VLCC: Very Large Crude Carrier
LPG: liquefied petroleum gas

Table 1-3: Three different ship types with their fuel consumption by tonnes for each day at normal speed

Most companies resort to the market to charter ships to deliver their shipments. The price for chartering is based on ship size and the time of the year. For example, the chartering rate for a 55,000 dwt tanker is $27,100 per day in August 2004, increasing to $55,000 per day in December 2004. In November 2004, the charter rate for a five-year-old VLCC was 90,000 dollars per day. In August 2004, a 48,263 dwt ship was chartered for a trip to China at 14,200 dollars per day.

1.2.4 Terms in Maritime Transportation Systems

To understand the maritime transportation operation, some terms needs to be clarified before proceeding. Ronen [92] mentioned these terms as follows: *Shipping* refers to moving of cargoes by ship. *Routing* refers to defining a sequence of ports of call to ships. *Scheduling* refers to routing with specific times allocated to ships for their stay in the ports. There are different meanings of time in maritime transportation, *Short term* is usually up to several weeks, *medium term* refers to up to several months, while *long term* is over six months. Operations in shipping have three general modes. The first one is *liner operation* which is like a bus line with a timetable. Such ships compete for cargos. The schedule of a liner ship is influenced by the demand for its service (cargo available), where the route is defined before the ship leaves the port. A

liner may be scheduled to call at a particular port more than once on a single voyage (subject to cargo being available). The second mode, *tramp operation* is like a taxi-cab operation, where ships can be sent depending on the availability of cargos. Usually, the cargo is a whole shipload with a single origin and one or two destinations. The objective of both liner and tramp operations are usually to maximise profits per time unit. Moreover, liner and tramp operations is common in shipping companies. The last mode, *industrial operation*, is similar to private truck fleet operation, where the owner operates the fleet of ships. The objective of this mode is to minimise the sum of the overall costs for all ships in the fleet, at the same time, ensuring that all cargoes have been delivered. Such a fleet often lacks the required capacity (fleet capacity is less than cargos volume), in which case the owner has to resort to chartering from others. There are two types of chartering: spot chartering which is for one or two specified tasks, and time chartering for a period of time (months or years).

1.2.5 Classification Scheme for The Ship and Scheduling Problem

In the section, a little modified classification scheme to the one proposed by Ronen [92] is presented. Different problems may involve different difficulties.

A. Mode of Operation

1- Liner

2- Tramp

3- Industrial

4- Other Modes (Naval, Barge, Coast Guard and Fishing)

B. Loading and Discharging Times

1- Specified (ship scheduling problem)

2- Time windows

3- Open (routing problem)

C. Number of Origins

1- One

2- Multiple

D. Number of Discharging Ports

1- One

2- Multiple

E. Number of Loading Ports per Vessel Voyage

1- 1- One

2- Multiple

F. Number of Discharging Ports per Vessel Voyage

1- 1- One

2- Multiple

G. Number of Products to be Shipped

1- 1- One

2- Multiple

H. Fleet Size

1- 1- One

2- Multiple

I. Compartments Capacities of Vessels

1- 1- One

2- Multiple

J. Type of Vessels

1- 1- One

2- Multiple

K. Status of vessels

1- Owned

2- Time Chartered

3- Spot Chartered

L. Demands (Shipment Sizes)

1- Deterministic (continuous, discrete)

2- Stochastic (continuous, discrete)

M. Cruising speed as a Decision Variable

1- Yes

2- No

N. Fleet Size and Composition

1- Specified and cannot be changed (short term problem)

2- Can be changed (medium term problem)

3- Constant over a scheduling period

4- Changes permitted over a scheduling period

O. Port Entry Constraints on Vessels

1- Exist

2- None

P. Sea Route Constraints on Vessels

1- Exist

2- None

Q. Cost

1- Fixed costs (operating cost and capital cost)

   a. In operation

   b. In lay-up

   c. Change of status

2- Variable costs

   a. Fuel consumption

   b. Port entry charges

   c. Time in ports

   d. Unit shipping cost

   e. Demurrage (cost of vessel's waiting time)

3- Penalties incurred by late shipments

R. Objectives

1- Minimise costs

2- Maximise profits

3- Maximise utility

S. Cargo Transshipment

1- Allowed

2- Excluded

T. Time between Events

1- Deterministic

2- Stochastic

U. Other Problem-Specific Characteristics

1- Some customers impose condition that shipments of products are to be made based on specified minimum and maximum aspiration storage levels.

2- Some customers state in advance the quantities of shipments of products and the times.

3- Some customers do not accept shipments of certain products if the compartments carrying such products on the previous trip, have carried different product. For example, if the compartments carrying crude oil have carried Naphtha on the previous trip [8].

1.3 Methods Of Scheduling

The traditional way of scheduling is called "rule of thumb". This way is implemented by sending the larger ships to the farther destinations. For example, if there are three ships available of 10,000, 8,000, and 5,000 tonnes of capacity respectively. The scheduler will try to load the first ship with the cargo to the farthest destination, and if that cargo does not fill the ship, then the scheduler may add more cargo(es) to other destinations in the same direction [93]. The rule of thumb approach may be specified as follows (Ronen [93]):

1- Read data and set up the problem.

2- If fleet capacity is not enough go to 10.

3- Select the largest ship remaining (if none left go to 8).

4- Select the cargo at the farthest port remaining (if none left go to 8).

5- Add the port to the ship's route, calculate left over capacity on the ship and quantity of the cargo left for other ships).
6- If ship is full go to 3.
7- If the ship is capable of entering an additional port, add the cargo of the port closest to the ship's discharging ports; go to 6.
8- For each ship, calculate the shortest route.
9- Calculate the cost of the schedule.
10- Stop

This way of scheduling is not efficient in comparison with other scientific methods, such as exact or approximate methods. Much research has been published using these scientific methods, and excellent results have been reported. Some of these methods will be described in Chapter 3.

1.4 The aims of this research project

The reason for the interest in this class of problems is for two reasons. First, since maritime transportation is an expensive mode to transport cargoes and services, any reduction in cost resulting from appropriate routing and scheduling of a fleet of ships will contribute a large monetary saving. This can occur by using mathematical programming models. Secondly, contributions made for this class of problems are based on a numerous solution algorithms. These solution algorithms are created from either optimisation or heuristic approaches. Much research are carried out to address this class of problems. However, researchers faced the challenge of complexity when making use of mathematical programming based models to generate good solutions. It seems that the underlying nature of the problem is simple, but mathematically it is challenging and complex.

In this thesis, both optimisation and heuristic methods are used. The aim of this thesis is to present and evaluate a new algorithm for a class of ship routing and scheduling problem. Golden et al. [51], presented a good survey on Vehicle Routing Problems (VRP). They showed that one of the best metaheuristics for solving VRP is Tabu Search (TS). Baker and Ayechew [12] reported that TS is better than a Genetic Algorithm (GA) approach for solving VRP. The contribution offered by the research reported in this thesis is based on a heuristic approach obtained by designing a new

model based on Tabu Search (TS). To measure the quality of the output, an optimisation algorithm based on Set Partitioning Problem (SPP) has been applied.

Therefore, the aim of this research is to design a reliable model, which is capable of generating a good schedule for industrial sectors, which can be used in proactive, simple and efficient way.

This thesis is organised as follows. Chapter 2 presents a background of solution techniques for routing and scheduling in general. Some applications for each technique will be provided. Chapter 3 gives a literature review of different classes of routing and scheduling problems and methods for addressing each one. A full description of ship routing and scheduling problem (SRSP) addressed in this thesis, is presented in Chapter 4. First, an exact approach using integer programming will be described, followed by a full explanation of a heuristic method to solve the problem. Chapter 5 will specify the design and implementation of solution approaches. Computational results and analysis for each approach are presented chapter 6. Finally, chapter 7 offers some conclusions and suggestion for future work.

Chapter 2: Solution Techniques for Routing and Scheduling

Introduction

Operations Research has long recognised the need for systematic mathematical techniques for solving complicated problems. There are two types of techniques, exact and approximate techniques. This chapter presents an explanation for each type with application studies. The first section is devoted to explaining the technique of Integer Programming, while the following section will present a description of approximate approach with some application studies.

2.1 Integer Programming (IP)

A **mathematical program** is a constrained optimization problem in which a function of n nonnegative variables is to be maximized or minimized subject to m constraints. Thus, a mathematical program identifies an *extreme* (i.e., minimum or maximum) point of an objective function $f(x_1, x_2, .., x_n)$, subject to a set of constraints $g_i(x_1, x_2, .., x_n) \leq b_i, i = 1, .., n$. **Linear programming (LP)** is a special case of mathematical programming where both the function f and the constraints are *linear*. This can be represented as follows:

Objective Function:

$$Min\ (or\ \ Max)\ \ f(x_1, x_2, .., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n$$

Subject to

Set of *m* constraints:

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \begin{pmatrix} \leq \\ \geq \\ = \end{pmatrix} b_i \qquad i = 1,2,..,m$$

Sign restrictions:

$$x_j \geq o \qquad \text{for } j = 1, .., n$$

Pure Integer programming (PIP) is a special case of linear programming in which all the variables are required to take integer values. It can be represented as follows:

Objective Function:

$$Min\ (or\ \ Max)\ \ f(x_1, x_2, .., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n$$

Subject to

Set of *m* constraints:

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \begin{pmatrix} \leq \\ \geq \\ = \end{pmatrix} b_i \qquad i = 1,2,..,m$$

Sign restrictions and integrality conditions:

$$x_j \geq \text{o} \ \ \text{and integer} \quad \text{for } j = 1,..,n$$

A *mixed integer programming* problem (MIP) is a relaxation of PIP where some but not all of the variables are required to be integer values. It can be represented as follows:

Objective Function:

Min(orMax)

$$f(x_1, x_2, .., x_p, y_1, y_2, .., y_q) = c_1x_1 + c_2x_2 + ... + c_px_p + d_1y_1 + d_2y_2 + ... + d_qy_q$$

Subject to

Set of (*m*) constraints:

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{ip}x_p + h_{i1}y_1 + h_{i2}y_2 + ... + h_{iq}y_q \begin{pmatrix} \leq \\ \geq \\ = \end{pmatrix} b_i \qquad i = 1,2,..,m$$

Sign restrictions:

$(x_j \geq 0 \text{ and integer}), \text{for } j = 1,.., p.$

$(y_k \geq 0), \text{for } k = 1,.., q.$

A *zero-one integer programming* problem (ZIP) is a special case of PIP in which each variable is required to take a value of zero or one, which represents a binary choice. It can be represented as follows:

Objective Function:

$$Min\ (or\ \ Max)\ \ f(x_1, x_2, .., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n$$

Subject to

Set of *m* constraints:

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \begin{pmatrix} \leq \\ \geq \\ = \end{pmatrix} b_i \qquad i = 1,2,..,m$$

Sign restrictions:

$$x_j \in \{0,1\}$$

LP and IP models are increasingly used within decision-making (e.g. network design in the telecommunication sector [55], aircraft scheduling [53, 62], production scheduling in the industry sector [45, 113]). The simplex method, developed by George Dantzig in 1947 (see [121]), is very successful in solving LP problems and efficient software is capable of solving large-scale LP problems. Solution methods for IP have progressed quickly since the work of Land and Doig [69] who proposed using Branch and Bound (B&B), and since the research reported in this thesis requires IP problems to be solved, the following section will present more details about B&B.

2.1.1 Branch and Bound B&B method

The most widely used method for solving IP problems is Branch and Bound B&B. Large scale integer problems generally have a very large number of possible solutions. For example, the Travelling Salesmen Problem (TSP) with 11 nodes (which is quite small) has (11-1)! = 3,628,800 possible solutions. To date the most effective way off tackling these hard combinatorial problems (any optimization problem that has a finite number of feasible solutions) is the B&B method, which provides a systematic partial enumeration of the solution space [121]. It involves searching a tree which is developed using a binary branching process. B&B starts by considering the problem with the entire feasible solution space. For a maximization problem, for example, if the objective function after relaxation has an optimal solution with all variables integer values, then an optimal solution has been found and the procedure stops. Otherwise, the feasible region will be divided into two or more regions. This process is called branching. The branching operation will be repeated on identified sub-regions and will produce sub-regions (nodes) to form a tree structure. The method

of finding the upper bounds for the optimal solution within each feasible sub-region is called bounding (upper bounds in maximization problems and lower bounds in minimization problems). The major tool in branch and bound is searching within a sub-region to establish better objective function values. Searching starts by selecting a variable x with fractional value ($>N$ but $<N+1$). The solution space between (N,$N+1$) will be eliminated. Variable x ($x \leq N$) will form a sub-region, while the other sub-region will be formed by ($x \geq N+1$). Upper bounds will be provided by a linear programming relaxation of the problem. If the upper bound for sub-region A in the tree is less than the value of the best integer feasible solution found to date, then sub-region A will be terminated and this operation is called *pruning (or fathoming)*. On the other hand, if the value of the upper bound is greater than the value of the best integer feasible solution found to date, then this objective function value for sub-region A is set as a new upper bound and is called *incumbent*. Hence, the best objective function value is obtained in the search tree operation together with the corresponding feasible solution. This operation continues until all sub-regions of the search tree are either pruned or solved, and the best integer feasible solution is proven to be optimal [38, 121] . For minimization problems, the same procedure is applied expect that decisions are made on the basis of lower bounds instead of upper bounds. To illustrate the B&B process, consider the following example [121];

Example: consider the following integer programming problem:

$$\max z = 8x_1 + 5x_2$$

$$\text{s.t.} \quad x_1 + x_2 \leq 6$$

$$9x_1 + 5x_2 \leq 45$$

$$x_1, x_2 \geq 0; \quad x_1, x_2 \text{ integer}$$

B&B method starts by finding the solution to the Linear Relaxation (LR). The LR solution is $z = 41.25$, $x_1 = 3.75$, and $x_2 = 2.25$. Since not all the variables are integers, the solution is not integer feasible. The integer solution will not have a value greater than 41.25. The first step is to select a fractional variable. For example, select $x_1 = 3.75$

This operation forces x_1 to integer and implemented by branching on x_1 to create two sub-problems (sub-regions). One with $x_1 \leq 3$ and the other with $x_1 \geq 4$ as shown Figure 2.1:

The solutions to the two sub-problems are:

$x_1 = 3$, $x_2 = 3$, $z = 39$

and $x_1 = 4$, $x_2 = 1.8$, $z = 41$

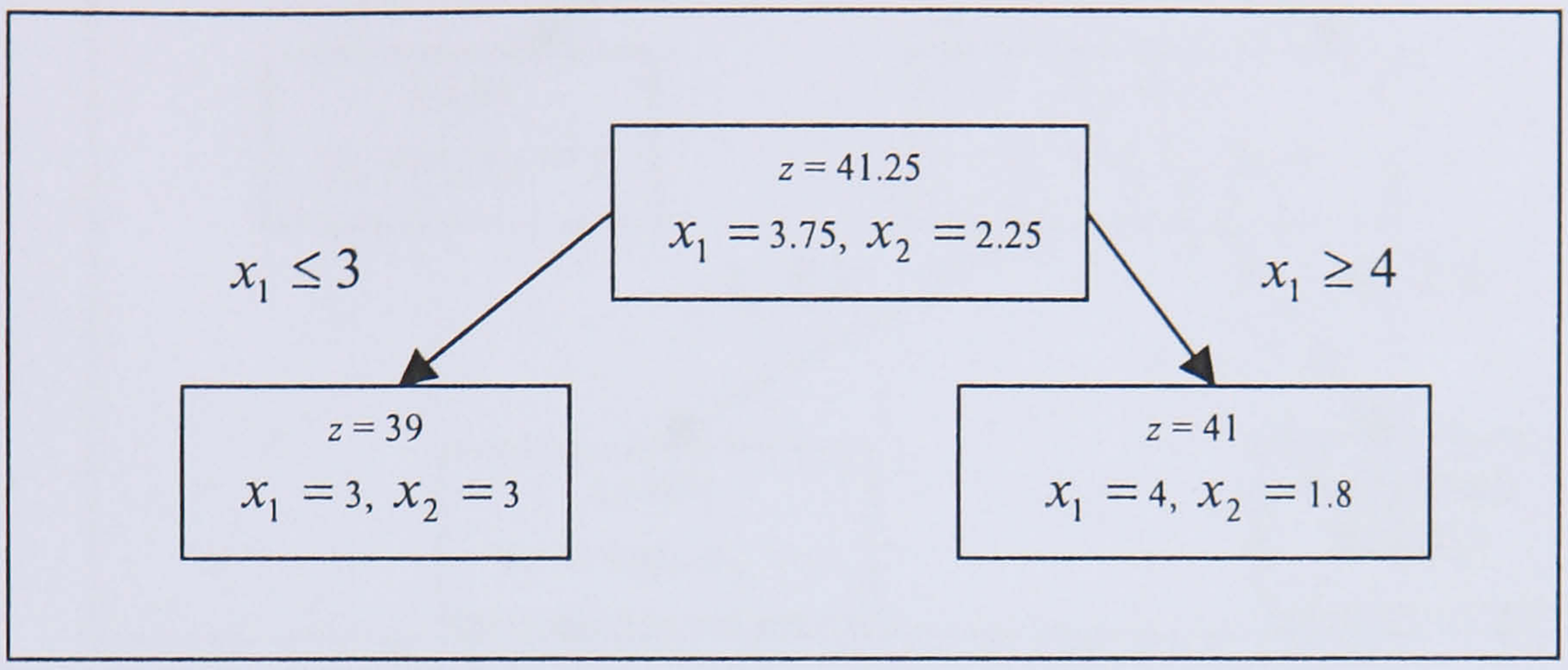


Figure 2.1: Branch and bound first step

In general, selecting the sub-problem with the highest solution value is the next step. The sub-problem with $x_1 = 3$ has no fractional variables, and the left path is therefore pruned with an integer feasible solution and the right branch will be developed by branching on x_2.

The solutions to the two sub-problems are:

$x_2 = 1$, $x_1 = 4.46$, $z = 40.6$

and, No feasible solution (does not satisfy constraint 2 in original IP)

Figure 2.2 illustrates the branch and bound diagram for the next operation. The sub-problem with $x_2 = 2$ has no feasible solution, and the right path is therefore pruned and the left branch will be developed by branching on x_1.

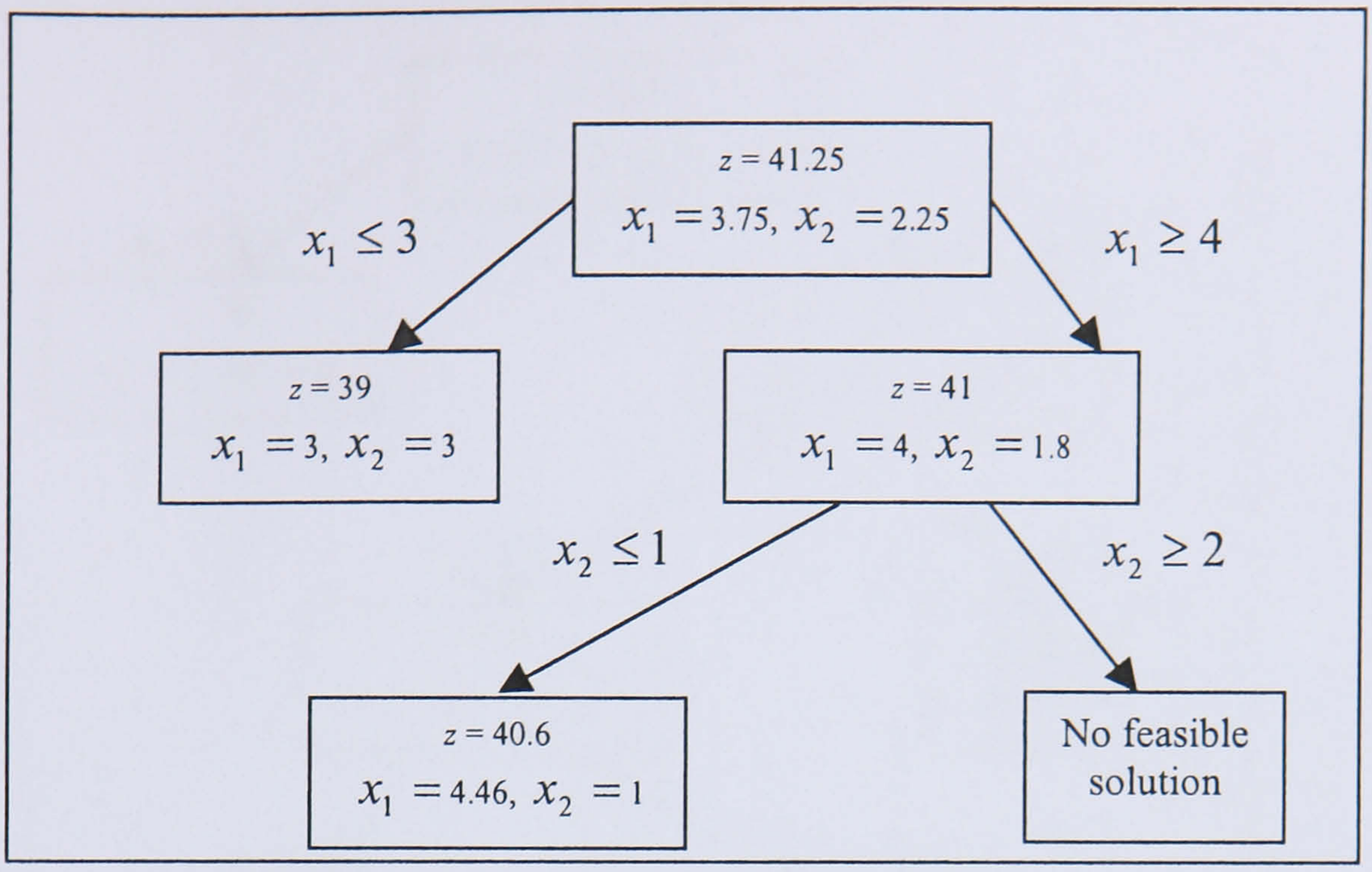


Figure 2.2: Branch and bound second step

The solutions to the two sub-problems are:

$x_1 = 4\ ,\ x_2 = 1\ ,\ z = 37$

$x_1 = 5\ ,\ x_2 = 0\ ,\ z = 40$

The best integer feasible solution found to date (the incumbent solution) has value z = 40. Since there are no unfathomed nodes remains in the tree, this solution is the optimal solution to the problem. In general, the incumbent solution is optimal if no unfathomed node has an upper bound greater than the incumbent value. Figure 2.3 illustrates the B&B for the entire problem, where the last rectangle (shaded) represents the optimal integer feasible solution.

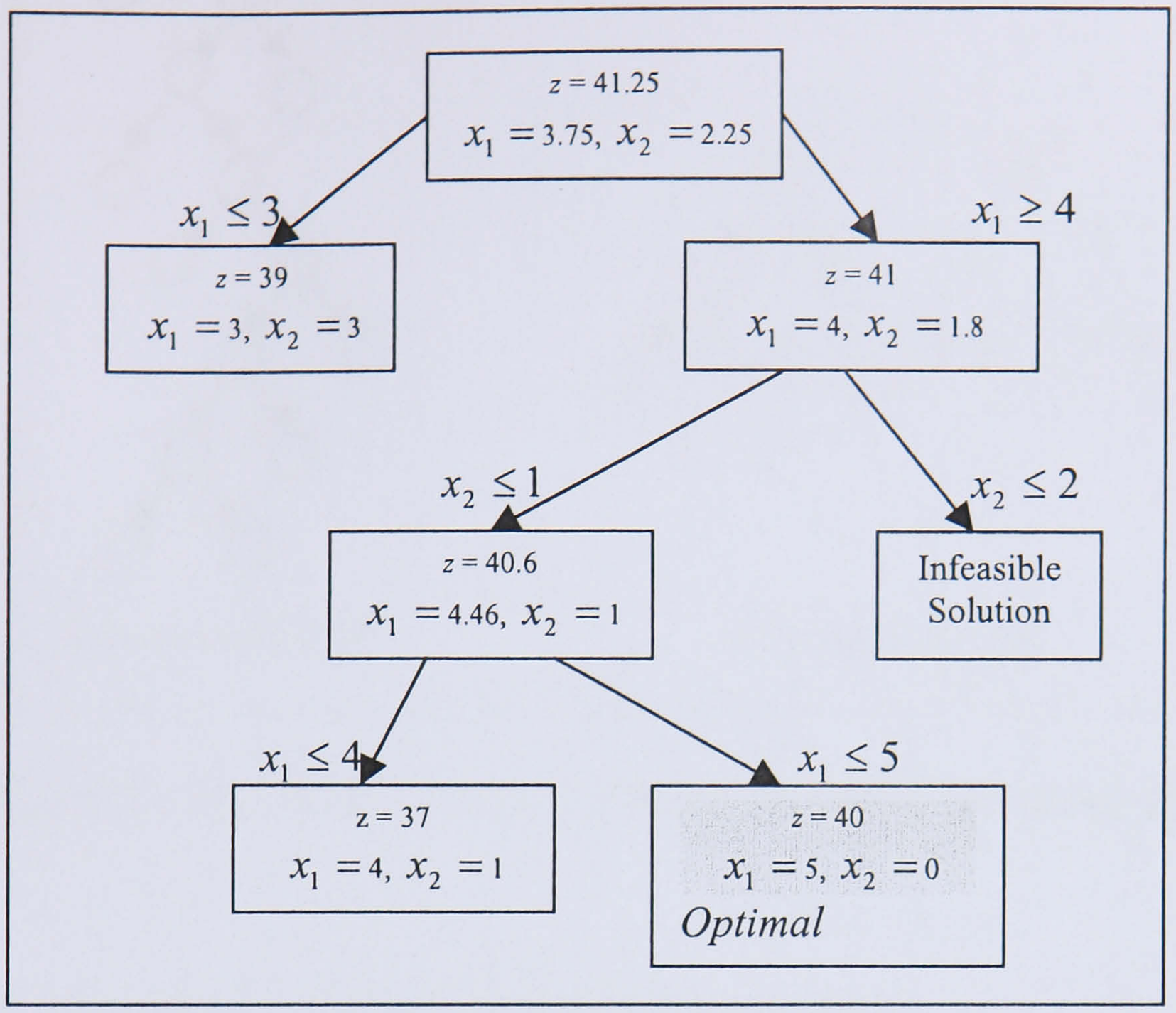


Figure 2.3: Branch and bound tree

In general, there are many different methods for selecting the next sub-region. One of these methods and widely used is called *Depth-First Search* (*DFS*) also known as Last-In-First-Out (LIFO). The way of applying this method is by selecting one of the new created sub-regions, where the search is implemented down one side of the branch and bound tree, with the intention of quickly finding a candidate solution. When a node is fathomed (pruned), the process backtracks from this node toward the root until it finds the first node that has a sub-region that has not yet been explored [121]. An alterative method called a *Breadth-First Search* (*BFS*), which starts at level zero. In the first stage, all nodes will be considered at level one. In the second stage all nodes will be considered at second level. The BFS traversal stops when all nodes have been considered. However, it needs a large amount of memory, which makes the approach impractical for solving large-scale problems. On the other hand, *DFS* does not need a large amount of memory, but needs extensive duplicate searching [124]. Figure 2.4 illustrates these two rules.

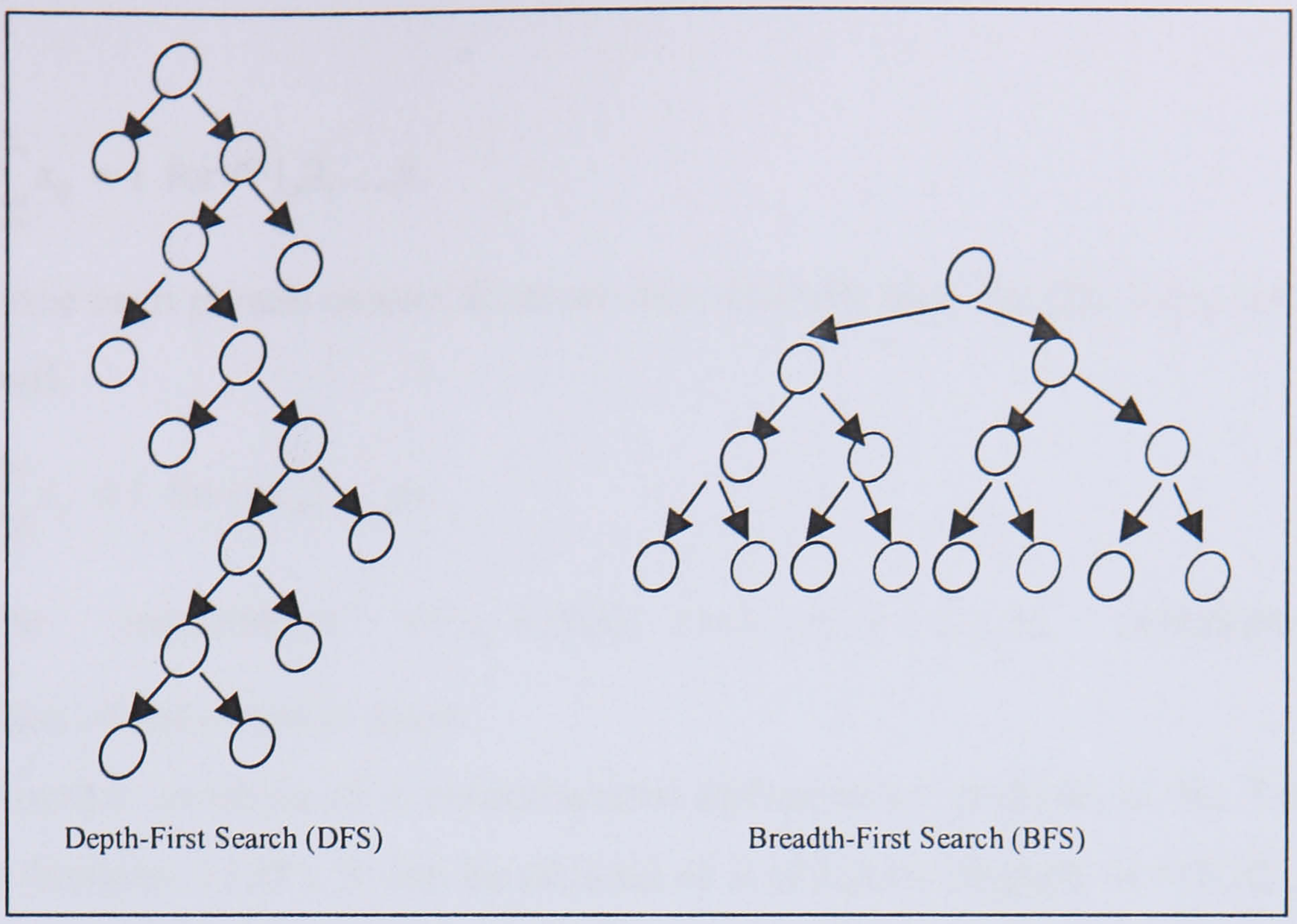


Figure 2.4: Depth-First Search (DFS) and Breadth-First Search (BFS)

2.1.2 Illustration of (0-1) integer programming

Since the Ship Routing and Scheduling Problem (SRSP) -which is addressed in this thesis- is of type (0-1) integer programming, it is useful to present some applications of this type. Any optimization problem that has a finite number of solutions is called a combinatorial optimization problem. There are many examples of combinatorial optimization problems such as assignment problems, one of which solves the problem of selecting m objects (vehicle, machine, manpower, etc.) to implement n events (commodity, product, job, etc.), where $m \geq n$. The requirement is to select l objects, where $m \geq l$, to implement all events at minimum total cost. For example, suppose there are n jobs and m people, where $m \geq n$ and each job must be implemented by exactly one person, the problem is to make a decision between these two possibilities. By using binary variables, it can be modelled as follows:

$$x_{ij} = \begin{cases} 1 \text{ if } j \text{ is selected to do job } i \\ 0 \text{ otherwise} \end{cases}$$

Variable x_{ij} represents the action of selecting person j to job i, where i=1,2,…,n. and j=1,2,…,m. To ensure that each job is implemented by only one person, the following constraints are required.

$$\sum_{j=1}^{m} x_{ij} = 1 \text{ for } i=1,2,\ldots,n.$$

Since each person cannot do more than one job, then the following constraints are required.

$$\sum_{i=1}^{n} x_{ij} \leq 1 \text{ for } j=1,2,\ldots,m.$$

The requirement of $x_{ij} \in (0,1),\ i = 1,..,n,\ j = 1,...,m$ completes the specification of the solution space.

Another example of a combinatorial optimization problem is the Travelling Salesman Problem (TSP). It can be defined as a complete digraph $G=(V,A)$, where $V = \{1,...,n\}$ is the vertex set and $A = \{(i,j) : i,j \in V\}$ is the arc set. The objective is to visit each vertex once and only once, passing through all vertices in minimum overall distance (time). One more example of a combinatorial optimization problem is called Vehicle Routing and Scheduling Problem (VRSP). This problem is concerned with determining the composition and routing of a fleet of vehicles in order to visit a pre-specified set of customers with known delivery demands from a central depot. Each customer is visited exactly once. The total demand of a route does not exceed the capacity of the vehicle specified to it, and the total cost is minimized [71].

In general, the best method of finding an optimal solution for a combinatorial optimization problem is by using B&B method [125].

2.1.3 Applications of (0-1) integer programming

There are many applications of (0-1) integer programming, usually solved by the B&B. Tozkapan et al. [119] addressed the two stage assembly scheduling problem. This problem can be described as if there are n jobs to be processed, where each job requires m-1 components to be processed. In the first stage, each job to be processed needs m-1 independent machines. After finishing stage one, an assembly machine at stage two assembles the components. Each machine can accomplish one job at a time. The objective is to minimize the sum of weighted finishing times of all jobs. The authors tackled this problem using the B&B method. The results showed that the method could handle test problems with $n \leq 20$. Chung et al. [31] addressed the m-machine permutation flow shop problem, where each of n jobs have to be implemented on machines $1,..,m$ in order. The operation time to finish each job on

each machine is known. Each machine can process only one job and each job can be processed by using only one machine, at any time. The objective is to minimize the total flow time. The algorithm they proposed was based on B&B, where a depth-first plus backtracking search technique was used. They presented computational results, which showed that this algorithm is capable of solving problems of up to 15 jobs.

Facility location is a problem that can be solved by using IP. Facility location can be defined as follows. Given set of demand points, a distance function, and a parameter p it is required to find a set of p supply objects (points, lines, warehouse, etc.); where the objective function is to minimize distance. This type of problem is encountered in many application areas, such as telecommunications networks, where one may wish to disperse radio transceivers to serve cellular phones in order to minimize interference problems. Other applications are material distribution, supply chain management, and transportation. Pisinger [84] addressed a type of facility location problem, called the p-dispersion-sum problem (PDSP). This problem can be defined as establishing p facilities at some of m predefined locations. The distance between two facilities i and j is obtained by a square matrix (d_{ij}), where i , j =1,..,m. The objective of this problem is to minimise the maximum distance between any demand point and its supply point. The author applied the B&B method to solve this problem, where experiments showed that the algorithm is capable of handling problems with a size of up to 90 locations.

(0-1) integer programming can be used to solve planning problems, such as, aircraft fleet routing and scheduling. This problem occurs in the long range planning operations of many airlines. Ioachin et al. [62] considered this problem where a given set of flights that present schedule flexibility on departure time and the flights had to be covered at a minimum cost. Each individual flight has its identifier such as origin and destination stations, a duration, etc. The authors tackled this problem of weekly fleet routing and scheduling by proposing an approach which solved problem samples with up to 80000 arcs (connection between origin and destination) in less than 21 seconds.

Zamani [124] considered a project scheduling problem called constrained project scheduling (RCPS). This problem can be defined as a set of n activities (jobs), which are given, where each activity has a fixed integer duration and a fixed amount of one or more different resources. This is subject to a set of precedence relations that

specify permissible activity orders. Once activities start they may not be interrupted. There are specified fixed limits on the availability of each resource type. The objective of this problem is to minimize the processing time. The author solved this problem using a B&B method based on a depth-first method, where the computational results showed that the algorithm is capable of solving problems of up to 100 activities and six different resource types. The author presented a numerical example to explain the effectiveness and the application of the algorithm. Maniezzo and Mingozzi [75] also considered project scheduling using B&B, where each node of the tree is associated with job *i*. The computational results reflect the ability of the algorithm to solve problems with up to 100 activities.

2.2 Approximate Technique (Heuristic methods)

It is very important before introducing heuristic methods to explain the meaning of heuristics. Heuristics are approximate solution techniques and have been used since the beginning of Operations Research to obtain solutions of complex combinatorial problems. With the growth in the number of difficult problems, where the level of difficulty in solving mathematically problems is measured by the time, memory space required, or number of steps or arithmetic operations required to solve them. The interesting aspect is usually complexity with respect to the size of the input, *N*. Since many of these problems are NP-hard, there was little hope of finding an efficient solution procedure to solve all instances of such problems.

This issue emphasized the role of heuristics for solving the combinatorial problems. Many heuristics have been proposed to solve these difficult problems such as, Genetic Algorithm (GA) [57], Simulated Annealing (SA) [66], or Tabu Search (TS).

2.2.1 Tabu Search (TS)

Tabu search is a memory-based search method, established by Glover in 1986 [47]. The method is capable of providing optimal or near optimal solutions. This capability for solving large combinatorial problems encountered in various practical settings attracted many researchers interested in finding optimal or near optimal solutions to this type of problem.

Tabu search is a meta-heuristic which leads a local search procedure to discover the solution space away from local optima. A meta-heuristics refers to a master strategy that leads and modifies other heuristics to generate solutions beyond those solutions generated by a procedure that may converage to a local optimum.

The main idea of Tabu Search (TS) is based on the Local Neighbourhood Search (LNS) improvement method. LNS can be described as operations of iterative search called *move*, which start from an initial feasible solution (obtained by using any heuristic method such as greedy algorithm or randomly). The search continues by applying a series of local moves. Examples of moves are adding or deleting an element from a set, changing the value assigned to a variable, or interchanging the position of two jobs on a machine, and so on [48]. At each iteration, the search moves to an improved feasible solution. This procedure carries on until the search reaches a local optimum with respect to the transformations that it considers. One important component of TS is the use of adaptive memory, that allows LNS methods to overcome local optima. This idea is implemented in LNS by using information to guide the search from one solution to the next, avoiding *cycling*. Cycling is defined as revisiting past sequences of solutions. Tabu Search uses memories called *tabu lists* (forbidden lists), which record recently made moves or visited solutions (history of moves). Therefore, when a solution is visited, the movement from which it was obtained will be considered tabu. A move made in iteration t is called tabu until a certain number e of iterations are executed at which point, after iteration $(t+e)$, a move will be non-tabu.

A background to some of the effective elements of tabu search is presented in the following sections.

2.2.1.1 Tabu Search Process

Tabu search can be applied directly to many types of decision problems. On the other hand, introducing mathematical notation to explain many classes of these problems is helpful to communicate the principal ideas of tabu search.

A. Neighbourhood Search

Tabu search starts in the same way as neighbourhood search. In neighbourhood search, each value $x \in X$ has an associated set of

neighbours, $N(x) \subset X$, where $N(x)$ refers to all elements that neighbour x. Each value $x' \in N(x)$ is reached from x by an operation called *move*, and x is said to move to x' when such an operation is implemented. There are two types of trail moves, *insert* and *swap*. An *insert* move consists, for example in VRP problems, of taking a customer from one route and inserting it into a different route. A *swap* move consists of exchanging two customers belonging to two different routes. For large problems, where a neighbourhood may include many elements, or for some problems where these elements are costly to examine, the best choice of TS makes it highly important to isolate a candidate subset of the neighbourhood and examine it instead of examining the entire neighbourhood. A candidate subset size depends on the problem type and the move definition and the best candidate subset size is a critical step. In comparison to TS, Local Neighbourhood Search (LNS) permits moves to neighbouring solutions that improve the current objective function, value $f(x)$, and ends with a local optimum solution when no improving solutions can be obtained. On the other hand, the Tabu Search method is capable of avoiding being trapped at a local optimum by moving to a different search area. The following steps illustrate Local Neighbourhood Search using descent method:

1. Choose $x \in X$ to start the process.
2. Find $x' \in N(x)$ such that $f(x') < f(x)$.
3. If no such x' can be found, x is the local optimum and the method stops.
4. Otherwise, set $x = x'$ and go to step 2.

At the end of the process, x is a local optimal solution with respect to its neighbourhood. However, a local optimum in most cases will not be a global optimum. In some cases, the solution of a specific problem may get trapped at a local optimum.

To provide a background for understanding one of the elements of neighbourhood search (the relationship between neighbours), the following example illustrates the basic operation.

An illustrative example

As a basis for illustration, consider the travelling salesman problem (TSP), and suppose there is a depot and five customers (A, B, C, D, E). The distances (miles) between each pair of customers are shown in Table 2.1.

	Depot	**A**	**B**	**C**	**D**	**E**
Depot	0	100	119	78	67	134
A	100	0	70	129	90	50
B	119	70	0	30	92	117
C	78	129	30	0	79	145
D	67	90	92	79	0	61
E	134	50	117	145	61	0

Table 2.1: The distances (miles) between each pair of customers

It is desirable that a search method is capable of finding an optimal or near optimal solution by examining a small subset of the total number of possible solutions (in this case 5! =120). To achieve this, a neighbourhood search can be defined as the relationship between customers, for example, distances. Consider two cases of constructing neighbourhoods: case 1, customer *i* is a neighbour of customer *j* if the distance between them is less than 80 miles, while in case 2 the distance limit is 95 miles. This procedure defines the size of the neighbourhoods and reduces searching time. Table 2.2 illustrates case 1, while Table 2.3 illustrates case 2.

	Depot	**A**	**B**	**C**	**D**	**E**
Depot	0	-	-	78	67	-
A	-	0	70	-	-	50
B	-	70	0	30	-	-
C	78	-	30	0	79	-
D	67	-	-	79	0	61
E	-	50	-	-	61	0

Table 2.2: Neighbours that are with less than 80 miles apart

	Depot	A	B	C	D	E
Depot	0	-	-	78	67	-
A	-	0	70	-	90	50
B	-	70	0	30	92	-
C	78	-	30	0	79	-
D	67	90	92	79	0	61
E	-	50	-	-	61	0

Table 2.3: Neighbours that are with less than 95 miles apart

The neighbourhood search in case 1 will be restricted within a small space, where the search will produce four search paths. Figure 2.5 illustrates the four possible search as for case 1, where last column is the overall distance (miles) for each specific neighbourhood search.

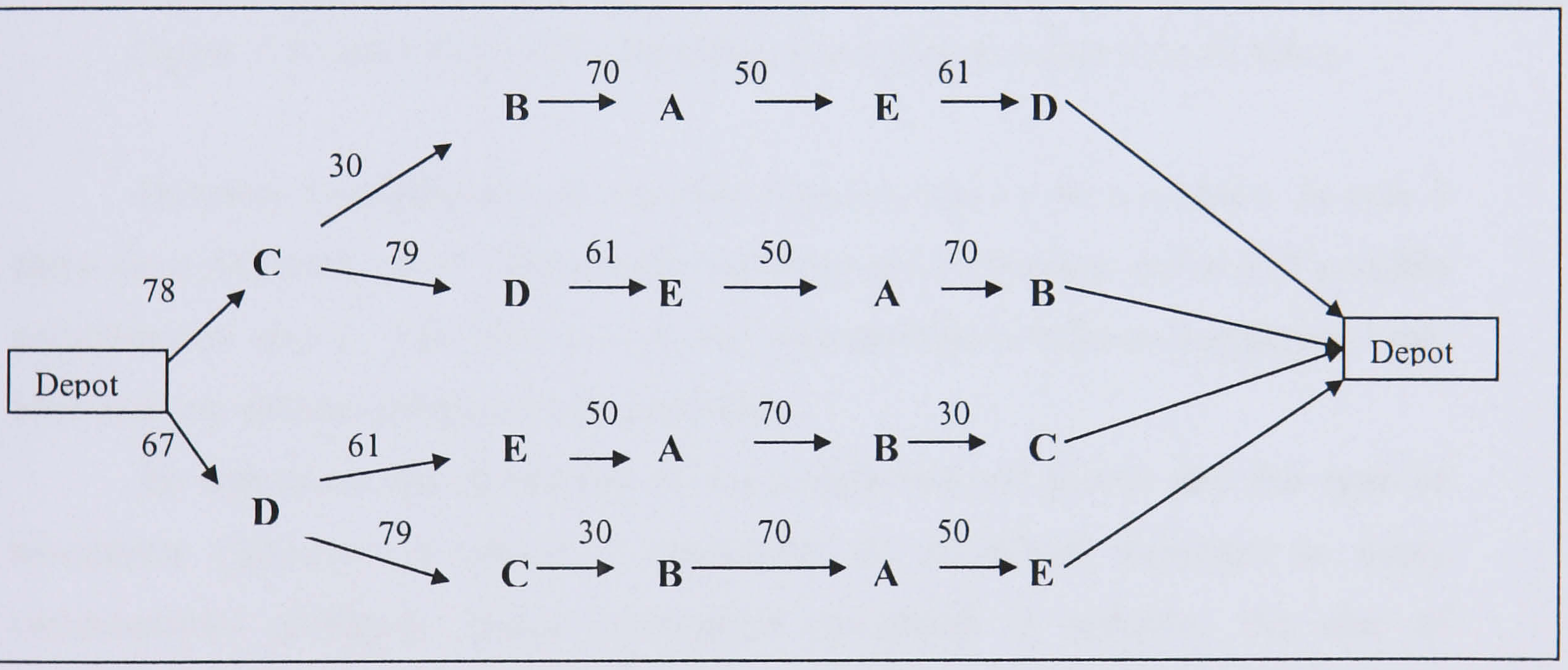


Figure 2.5: Searching within neighbourhood distances less than 80 miles

On the other hand, for case 2, the neighbourhood search will be greater than in case 1. Figure 2.6 illustrates all possible neighbourhood searches when the distance is less than 95 miles.

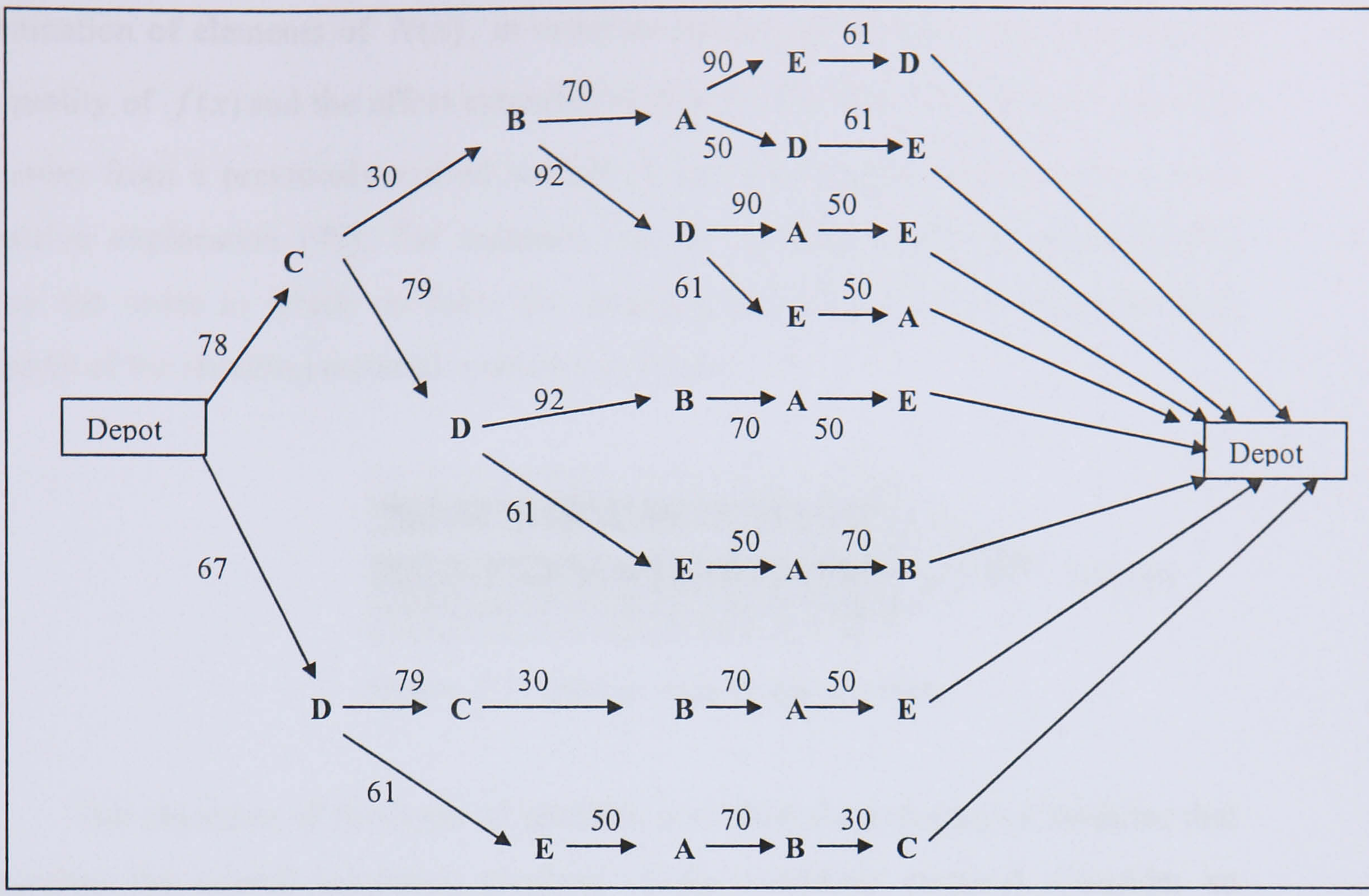


Figure 2.6: *Searching within neighbourhood distances less than 95 miles*

Defining a neighbourhood can reduce search time to get a solution. In case 1 there are 4 solutions out of 120 possible solutions and 8 solutions out of 120 possible solutions for case 2. This way of solving large problems reduces computing time. However, an optimal solution is not guaranteed.

To determine the dimension of the neighbourhood search and the type of movement (dynamic or static) is considered an important condition in many combinatorial problems. Some researchers succeeded in reducing the size of neighbourhood whilst, at the same time, finding good solutions in a short computing time. For example, Toth and Vigo [118] addressed the Vehicle Routing Problem (VRP) and presented an algorithm to find solutions in a short computing time, by reducing the size of the neighbourhood to explore only the potentially most promising moves.

B. Tabu List

The guidance mechanisms of TS are introduced to go beyond the local optimal termination value of a descent method. Thus an important step to consider in TS is to determine an appropriate candidate list strategy (tabu list) for narrowing the

examination of elements of $N(x)$, in order to achieve an effective tradeoff between the quality of $f(x)$ and the effort extended to find it. The tabu list helps the search to get away from a previously visited section of the search space and execute a more extensive exploration [49]. For instance, on the module insulation problem [91], where the order in which modules are arranged determines the overall insulating property of the resulting material as shown in Figure 2.7.

Figure 2.7: Module insulation problem

The objective of this type of problem is to find the ordering of modules that maximizes the overall insulating property of the combined material. Consider an initial solution for this problem constructed as the one shown in Figure 2.8 and consider the resulting material to have an insulating property of 10 units.

								Value
2	5	7	3	4	6	1	=	10

Figure 2.8: Initial solution for module insulation problem

As mentioned previously, swap pairs of elements exchange their positions. If a swap for example, between materials 4 and 5 is applied, a new construction as shown in Figure 2.9 is created with an insulating property of 16 units.

								Value
2	4	7	3	5	6	1	=	16

Figure 2.9: New solution for module insulation problem

In this case, (4, 5) will be considered classified as tabu and enters the tabu list. Exchange positions are not allowed for a number of iterations (tenure) of n iterations – this issue will be discussed in section 2.3-.

The length of the tabu list L plays a critical part in most tabu search algorithms. Thesen [117] presented some examples to show that the wrong choice of

L may lead to a very inefficient algorithm. Anderson et al.[9] reported that list lengths between 7 and 15 worked well for a path assignment problem. Where Allahverdi and Al-Anzi [7] showed that when L exceeds 5, results remain the same with no significant improvement. Taillard [110] reported that the successful use of lists when it is implemented randomly change in length. It is clear that the most efficient length of the list depends on the problem being solved and the algorithm being used.

We can illustrate tabu search steps as follow:

Notation:

S	the current solution
S^*	the best-known solution
f^*	value of S^*
$N(S)$	the neighbourhood of S
$N'(S)$	non-tabu subset (admissible subset)

Initialization

Construct an initial solution S_0 using any heuristic method such as greedy method or randomly.

Set $S = S_0$, $f^* = f(S_0)$, $S^* = S_0$, T (Tabu list) = ϕ.

Search

Select S in *argmin* $[f(S')]$; $S' \in N'(S)$

If $f(S) < f^*$, then set $f^* = f(S)$, $S^* = S$;

Record tabu for the current move in T and delete oldest entry;

endwhile.

The most commonly used termination criteria in TS are after a fixed number of iterations (or a fixed amount of CPU time) or after some number of iterations without any improvement in the solution.

Record and delete tabu elements in and from the tabu list during the exploration process are obtained by using two types of tabu list. The first one is called *short term memory* to maintain the most recently visited tabu transformations. In some applications, the short term memory is sufficient to provide very high quality solutions. However, in general, TS becomes significantly stronger by including the second type, *long term memory*, where the history through all the exploration process as a whole is kept.

C. Tabu Tenure

A tabu list consists of a list of moves the search has recently encountered. The moves on the tabu list cannot be revisited for a particular number of iterations called the *tabu tenure*. Once a move is on the tabu list for a number of iterations equal to the tabu tenure, it will be removed from the list and considered as a non-tabu element, and is again a valid move choice. The size of the list can be either variable (dynamic) or fixed [17]. In the case of using variable list size, called Dynamic tabu tenure, tabu tenure starts out small and grows rapidly if cycling occurs and decreases gradually as cycling is diminished. Varying the tabu tenure during the search provides one way to induce a balance between closely examining one region and moving to different spaces of the solution area. There are several ways to implement dynamic tabu tenure, we will mention two of them. The first implementation is called *Random Dynamic Tenure*, often given one of two forms. Both forms use a tenure range defined by parameters $t_{\min}$ and $t_{\max}$. The tabu tenure t is randomly selected within this range and usually uses a uniform distribution. In the first form, the selected tenure remains constant for $t_{\max}$ iterations, and then a new tenure is chosen by the same operation. The second form draws a new tabu tenure t for every attribute that becomes tabu at a given iteration. The second implementation is called *Systematic Dynamic tenure*, and this consists of creating a sequence of tabu search tenure values in the range defined by $t_{\min}$ and $t_{\max}$. Then this sequence is used rather than using a uniform distribution to allocate the current tabu tenure value [49]. Thesen [117] reported, by extensive experiments, that the best value for $t_{\min}$ is one, and the best value for $t_{\max}$ is L (the length of the list).

D. Aspiration Criteria

Aspiration Criteria techniques are introduced in tabu search to determine when tabu activation rules can be ignored. An improving move is not accepted if it is tabu list unless it leads to an overall the best solution of the search process. In this case, *aspiration criteria* can be used, which allows one to ignore the tabu status of a move if this move leads to a value better than the best-known value found by the search so far. Consider the module insulation problem, if the current solution as is shown in

Figure 2.10 and the resulting material has an insulating property of 18 units, and the best solution so far is 22 units.

2	3	5	3	4	6	1	=	Value 18

Figure 2.10: Current solution for module insulation problem

If exchanging (3,7) classified tabu (tabu list), and exchanging positions will result in a solution of 24 units, which is better than any visited so far (22 units), then its tabu classification may be overridden.

A basis for one of these criteria arises by introducing the concept of *influence*, which measures the degree of changes induced in solution structure [91].

E. Longer term memory

Short term memory components in some applications are adequate to generate very high quality solutions. On the other hand, TS becomes significantly stronger by introducing long term memory. Long term memory operates principally as a basis for diversifying the search [72]. If a search begins cycling within, and cannot escape a region, it needs some starting event forcing the search to move to another region.

F. Intensification and Diversification strategies

Intensification strategies are based on initiating a return to attractive regions to search them more carefully. Intensification strategies can be applied to stress the search to a more promising region of the solution space, so that the moves to the local optimum are intensified. On the other hand, d*iversification* strategies can be used to enlarge the search into less explored regions by forcing the moves out of the local optimum. Diversification strategies are usually based on some form of long term memory of the search, such as a *frequency* memory, in which one records the total number of iterations that various solution components have been involved in the selected moves or have been presented in the current solution. For instance, in Vehicle Routeing Problem (VRP) application, one could calculate how many times each customer is moved from its current route. These techniques could be considered the most critical issue of designing TS heuristic [13].

2.2.1.2 Tabu Search Applications

Tabu search has been found to be very effective for a variety of combinatorial problems [74]. Tabu search methods has been applied intensively for various types of optimisations problems and enjoyed success in many of these problems. TS has been compared against simulated annealing and found to find better solutions more efficiently for many combinatorial optimisation problems [4]. Scheduling provides one of the most successful areas for modern heuristics techniques in general and tabu search in particular.

Some of these studies, which have used tabu search as a method to solve their problems, will be mentioned. There are of course many of applications, not mentioned here because of limitation of the space. Jaszkiewicz [64] presented a short survey on TS for combinatorial optimisation problems. The following are some of the applications reported [49].

1- Planning and scheduling

1.1 Scheduling in Manufacturing Systems [90].

1.2 Audit Scheduling [41].

1.3 Scheduling a Flow-Line Manufacturing Cell [103].

2- Telecommunications

2.1 Hub-and-Spoke Communication Networks [101].

2.2 Design of Optical Telecommunication Networks [102].

3- Parallel Computing

3.1 Multiprocessor Task Scheduling in Parallel Programs [117].

3.2 Quadratic assignment Problem on a connection Machine [26].

4- Transportation, Routing and Network Design

4.1 The Fixed Charged Transportation Problem [107].

4.2 The Vehicle Routing Problem [13].

4.3 Vehicle Routing Problem with Time Windows [56].

4.4 Routing and Distribution [36].

5- Optimization on Graphs

4.1 The P-Median Problem [98].

4.2 Graph Partitioning [40].

4.3 Graph Drawing [68].

It is useful to mention some of the applications concerned with transportation and routeing in detail because of the similarity with our study. The need for more efficient logistical planning became the main subject for most transport and distribution companies due to global competition. Taillard [109] introduced a novel approach where the set of vertices are decomposed into sub problems that may be solved separately in order to speed up the iterative search method. Results indicated that it performed better than all other tabu search approaches. Two different decomposition schemes are proposed for the uniform and nonuniform problems. Vertices and tours are exchanged between the subproblems after a summary resolution of the subproblems. Brandao and Mercer [20] presented a tabu search for the multi-trip vehicle routeing and scheduling problem (MTVRSP). The algorithm consists of three stages and with the initial solution of each stage being the best solution found in the previous stage. The aspiration criteria are considered which allows a move to be performed that is tabu in the current iteration. Two types of moves are also considered, insert move which consists of moving one variable from one position to another and swap move which consists of interchanging two variables of the candidate set. They reported an improvement of 17% and 22% over the manual solution for the overall cost and time, respectively. Ho and Haugland [56] studied a variant of the general VRP with time windows and split deliveries (VRPTWSD). A customer whose demand exceeds the vehicle capacity can be served by the option of splitting a demand. They solved the problem using three steps. The first step, involved computing an initial feasible solution on the basis of a simple analysis of travel time and waiting time. In the second step, they improved the solution by using tabu search. The neighbourhoods were based on some common operators; relocate, exchange, 2-opt, and relocate split. Comparison with the best published VRPTW solutions in the literature had been made. Barbarosoglu and Ozgur [13] developed an algorithm for VRP namely DETABA, using most of the tabu search principles, but they proposed a new neighbourhood search procedure without any diversification and a new intensification scheme. Comparison was made to measure the performance against many tabu search algorithms using the well-known benchmark problems. The authors reported that the performance was better than all of them except that of Taillard [109].

2.2.2 Simulated Annealing (SA)

Simulated Annealing (SA) is considered as a method for finding a good approximate solution for optimization problems. The idea behind this method emerged when Metropolis et al. [78] in 1953 presented an approach to estimate the rate of cooling for liquid material to reach a solid state, a process known as annealing. Physical annealing is a process of heating solid material to a temperature past the material's melting point and then cooled back into a solid state. The atomic structure of the material will depend on the cooling rate. The preparation starts by, first melting the solid material followed by reducing the temperature gradually. If the reduction in the cooling temperature is performed very quickly, the shape of atoms will be nearly the same shape as in the previous structure. On the other hand, if the cooling process is implemented very slowly, the chance of obtaining a good shape will be high. The last shape will be obtained when the temperature reaches a steady frozen state. Kirkpatrick et al. [66] suggested that this kind of operation could be exploited and transferred to a method for solving optimisation problems by searching the feasible solutions, where feasible solutions can be analogous to construction of the materials. In Kirkpatrick's proposal, the term control parameter T which represents temperature is not constant, but it is decreased gradually after a number of iterations. A newly generated feasible solution which is different from the current solution is similar to a small displacement of an atom in the solid material to obtain a new shape. There are two important categories of decisions to be mentioned. The first one is a set of generic decisions related to simulated annealing itself and consists of: (i) initial temperature t_0, (ii) cooling schedule, (as mentioned previously, the temperature is not constant in SA), which is responsible for α, the rate of change the temperature, and (iii) lower limit or stopping function. The rate of temperature reduction is fundamental to the success of any annealing operation, and this can be achieved by specifying a number of iterations at each temperature and the rate at which the temperature is reduced. This can be performed either by using a small number of iterations at a high temperature, or by a large number of iterations at a low temperature. The determination of α is critical, either as a constant or a variable. There are different ways to determine α. The most popular is a geometric reduction function using the formula,

$$T_k = T_{k-1}.\alpha \qquad \text{Where } \alpha < 1$$

Research has shown that a high value of α performs better and in most published research a value between 0.8 and 0.99 is used [91]. Lundy and Mees [73] executed their problem using only one iteration at each temperature. On the other hand, the temperature rate of change was very slow and determined by the formula

$$T_k = \frac{T_{k-1}}{1 + \beta T_{k-1}} \quad \text{Where } \beta \text{ is close to zero}$$

The second category is about problem-specific decisions and consists of three decisions (i) space of feasible solutions, (ii) neighbourhood structure, and (iii) the cost function. In a problem such as TSP, where the problem consists of n customers, any route can be represented by a string of the numbers 1 to n. for example if n =10 (including the depot) the string could be formed as follows:

0—2—6—8—5—3—9—7—1—0

These strings shape the solution space. The size of solution space can be calculated, although in some types of problems it may difficult to calculate. For TSP with n customers, the size of solution space can be calculated as $n!$, whereas in a problem which is constrained by strict feasibility conditions it may be hard to calculate. Several results showed that the number of iterations depends on the size of the solution space. It is desirable to keep the solution space small [91]. The neighbourhood feasible solutions can be generated randomly, and the size of the neighbourhood is recommended to be small for large and complex problems. Consider a TSP problem with 4 customers to be visited and an initial solution as shown in Figure 2.11.

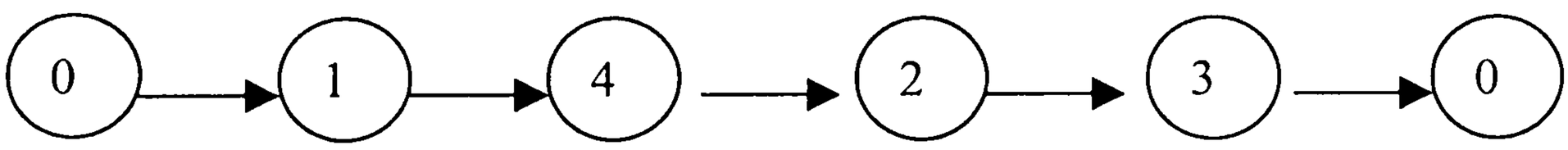


Figure 2.11:TSP with four customers

The selection of customers for deleting and inserting, or swapping considers the size of neighbourhood structure. By using the 2-Opt method and selecting customers 2 and 4 randomly, there is only one way to reconnect the two customers, 2 exchanged to position 4 and 4 exchanged to position 2. Whereas in 3-opt for the same problem there are 6 choices. In general TSP, with m customers, the size of neighbourhood is $m(m\text{-}1)$. The changes between the objective function value (the cost

function) for both the current solution and the newly generated feasible solution can be defined by δ.

$$\delta = FC_k - FC_{k-1}$$

Where FC denotes to function cost. If $\delta < 0$, then the newly generated feasible solution becomes the current solution, but if $\delta > 0$ the newly generated feasible solution has a probability of

$\exp(-\delta / T)$ of assumption as the current solution

Initially, the temperature is at its highest and therefore the acceptance probability is also at its highest value. At the end of the procedure, the temperature is close to zero and the correspondent acceptance probability is very small. This procedure will prevent simulated annealing from being trapped in local solutions. The simulated annealing algorithm can be stated as follows [91]:

(Simulated annealing for minimisation problem with solution space *S*, objective function f and neighbourhood structure *N*)

Select an initial solution s_0;

Select an initial temperature $t_0 > 0$;

Select a temperature reduction function α;

Repeat

Repeat

Randomly select $s \in N(s_0)$

$\delta = f(s) - f(s_0)$;

if $\delta < 0$

then $s_0 = s$

else

generate random x uniformly in the range [0,1];

if $x < \exp(-\delta / t)$ *then* $s_0 = s$;

until iteration_count=nrep

Set $t = \alpha(t)$;

Until stopping condition = true

s_0 is the approximation to the optimal solution.

---

1. Example of TSP

To simplify SA methodology, consider a TSP with six customers including the depot. The distance between each customer is presented in Table 2.4.

Customer	0	1	2	3	4	5
0	0	6	7	2	9	8
1		0	4	6	8	12
2			0	6	11	13
3				0	10	7
4					0	5
5						0

Table2.4: Distance (miles) between customers, where 0 denotes to the depot

Let the initial temperature be high, $T_0 = 100$, and the reduction parameter α after each iteration will be $T_k = T_{k-1} * 0.8$. If the cost of the current solution is lower than that of the new generated neighbouring solution, it will be accepted automatically. But if the difference giving rise to cost function, may be accepted subject to condition that depends on the control parameter (temperature) and the magnitude of the increase, which is formulated as

$$x < \exp(-\delta / t) \qquad \text{where } x \text{ is}$$

In this example let x be constant and equal to 0.9. The initial solution is generated randomly as follow (0-1-4-2-5-3-0) with cost function equal to 47 miles.

The neighbours are generated randomly by using 2-Opt method. Table 2.5 illustrates SA applied on TSP problem.

	Current Solution	Swap	New solution	T	δ	$\exp(-\delta/T)$ $\delta>0$	Acceptance Decision $x=0.9$ $x<\exp(-\delta/T)$?
1	0-1-4-2-5-3-0 (47)	(1)&(5)	0-5-4-2-1-3-0 (36)	100	< 0		YES
2	0-5-4-2-1-3-0 (36)	(4)&(3)	0-5-3-2-1-4-0 (42)	80	6	0.93 > 0.9	YES
3	0-5-3-2-1-4-0 (42)	(2)&(4)	0-5-3-4-1-2-0 (44)	64	2	0.97 > 0.9	YES
4	0-5-3-4-1-2-0 (44)	(1)&(3)	0-5-1-4-3-2-0 (51)	51	7	0.87 > 0.9	NO
5	0-5-3-4-1-2-0 (44)	(5)&(3)	0-3-5-4-1-2-0 (33)	51	< 0		YES
6	0-3-5-4-1-2-0 (33)	(3)&(2)	0-2-5-4-1-3-0 (41)	41	8	0.82 > 0.9	NO
7	0-3-5-4-1-2-0 (33)	(1)&(2)	0-3-5-4-2-1-0 (35)	41	2	0.95 > 0.9	YES

Table 2.5: Bracketed values in the Solution columns represent cost function values

In this example, the best solution is in row six, where the cost function has reduced to 33 miles.

There have been a number of independent implementations of SA in the solution of a variety of classical combinatorial optimization problems. Most of these studies have used different cooling rates and enhancement in some of SA parameters such as Marin and Salmeron [76], Wu et al. [122], Breedam [21, 22], Tan et al. [112], and Bent and Hentenyck [18]. The following section highlights on some of these studies.

2.2.2.1 Applications on SA:

A good description of SA is provided by Breedam [21], where VRP was solved using SA. The author used three methods of selecting customers for using solution update action. The first method, called the relocation method, tries to delete a customer or a string of customers from one route and insert into another route. The second method, called string exchange method, exchanges customers or a string of customers between two routes. The last method is a mixture of the previous methods, called string mix method, which tries to exchange or relocate a customer or a string of customers. The maximum number of customers to be relocated or exchanged was 3 customers. The initial temperature $T_0 = 1000$, and the total number of iterations was equal to 1000000. An iteration occurs every time a move was generated, whether it was feasible or not. The acceptance ratio was 0.01, while the stop temperature

$T_{final} = 1$. A comparison with 14 classical VRP problems [30] was made. Bent and Hentenryck [18] used a two-stage algorithm for pickup and delivery vehicle routing problems with time window (PDPTW), where the customers in this problem were divided into pickup and delivery pairs. A pair (a,b) in a route must visit customer a and b using the same vehicle. Meantime, the schedule must pickup customer a before the delivery to customer b. The objective function was to minimize the number of used vehicles and the total travel cost. The first stage was to reduce the number of vehicles using SA, while the second stage of minimizing the total travel cost was solved using a heuristic method. Because of the success in reducing routes on the VRPTW and overall simplicity of its implementation the authors selected SA to solve part of this problem. The initial solution started by serving each customer using its own vehicle. Each temperature started after finding the best solution so far, where the search was based on a number of local searches. Each local search has a number of iterations and decreases of temperature, where these two steps were repeated until a time limit was reached or the temperature has reached its lower bound. The cooling rate was specified as $\exp(-\delta / T)$. The initial temperature was $T_0 = 2000$ with cooling factor $\alpha = 0.9$, 2500 iterations per temperature and minimum temperature $T_{final} = 0.01$. The time allowed for running SA was 5 minutes. The acceptance condition number was selected randomly in the range [0,1]. The results they achieved were compared with standard PDPWT benchmarks [1]. Tan et al. [112] addressed VRPWT using heuristic methods, one of which was SA. The authors generated an initial solution, with initial temperature $T_0 = 100$. They decreased the temperature by applying the formula

$$T_k = \frac{T_{k-1}}{1 + \tau\sqrt{T_{k-1}}}$$

where T_k was the current temperature at iteration k and τ was the time constant in the range of (0,1). To speed up the cooling process, they introduced the square root of T_k. The neighbourhood solutions selected randomly or systematically with the acceptance ratio for those solutions according to

$$\exp(-\Delta / T)$$

Where $\Delta = C(s') - C(s)$, $C(s')$ was the cost of the new solution and $C(s)$ was the cost of the current solution. When the temperature reaches the final temperature

$T_f = 0.001$, or no more feasible moves in the neighbourhood, the temperature was reset using the formula

$$T_r = \max(\frac{T_0}{2}, T_b)$$

where T_r was the reset temperature and T_b was the temperature at which the best current solution was found. The authors adapted the 2-exchange approach for local neighbourhood search. The operation terminates after a given number of resets. Results showed that SA was very fast and offered reasonably good solutions.

2.2.3 Genetic Algorithms (GAs)

2.2.3.1 Background of Genetic Algorithms (GAs):

Genetic Algorithms (GAs) were introduced and developed by Holland [57], his colleagues, and his students at the University of Michigan. The aim of their study was to abstract and explain the adaptive technique of natural systems and to design artificial systems software that preserves the important mechanisms of natural systems [50]. GAs are considered as an adaptive technique that simulates the optimisation operation with the natural system of genes in a population of organisms as illustrated in Figure 2.12. The GA keeps a population of candidate members over many generations. The population consists of a number of strings of artificial chromosomes. A chromosome consists of a number of entities known as genes. The number of genes in a chromosome is usually the same for all chromosomes in the population. Each gene has a value associated with it. In most applications those numbers are either binary or integer. Variables are often called genes, and the values of variables are called alleles. The process starts by selecting parents according to the fitness value (quality). Operations such as crossover and mutation are applied to parents to produce offspring (children). The new generation is considered as the new population and the procedure reapplied.

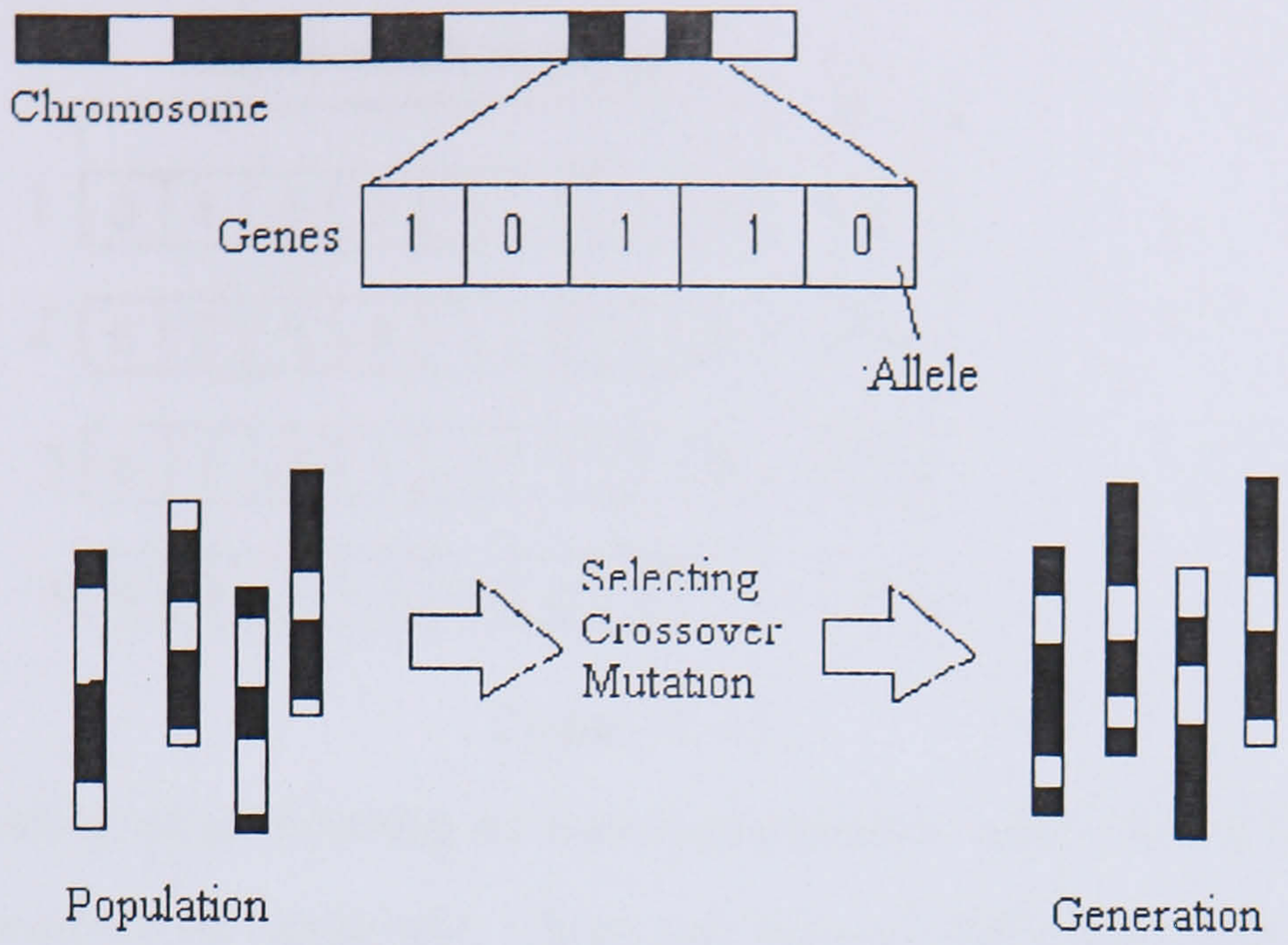


Figure 2.12: Genetic process (derived from Tan et al. [111])

The methodology and the basic operators of GAs can be illustrated as follows. Consider, for example the Travelling Salesman Problem (TSP), to visit six customers, starting and finishing at a depot. The objective of this problem is to serve all customers at minimum total distance. The distance between each customer is shown in Table2.6 where 0 denotes to the depot.

Customers	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0	0	15	12	20	25	16	9
1		0	10	8	7	8	12
2			0	17	8	9	13
3				0	21	14	9
4					0	18	15
5						0	11
6							0

Table2.6: Distance between each customer (miles)

A genetic algorithm starts by first generating initial solutions which form a population. Consider for this example that there is an initial population of size = 4 with the total distance (fitness value) for each string as shown in Figure 2.13.

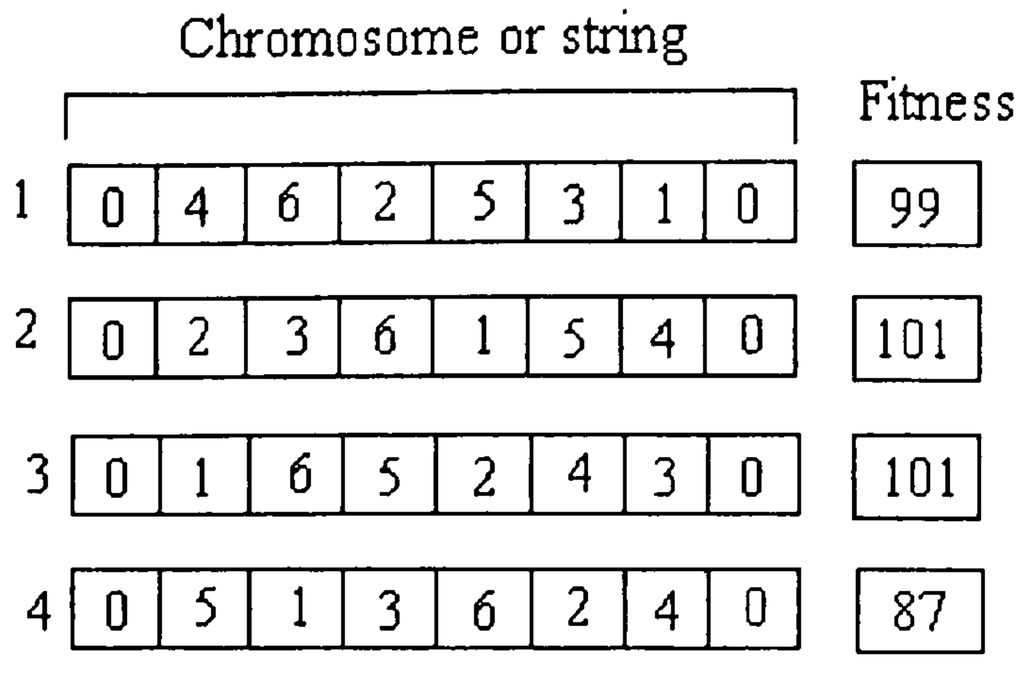


Figure 2.13

The operation of generating an initial population could be performed by using a systematic procedure or randomly. There are sets of operations that take this initial population and generate successive populations, which lead to improving the objective function over time. There are three basic operations in GA: (i) Reproduction, (ii) Crossover, and (iii) Mutation, usually applied in sequence. First, *reproduction* is an operation of selecting individual strings according to their fitness values. The selected strings are known as parents and can be produced by several methods. Since the TSP is a minimization problem the way of selecting a parent will be according to lower fitness values. This means, the lower the fitness value, the greater the probability that the parent will contribute one or more offspring (new generation) for the next generation. Referring to the example shown in Figure 2.11, string 4 has the lowest fitness value (87) and will be the first parent, while string 1 (99) will be the second parent as shown in Figure 2.14

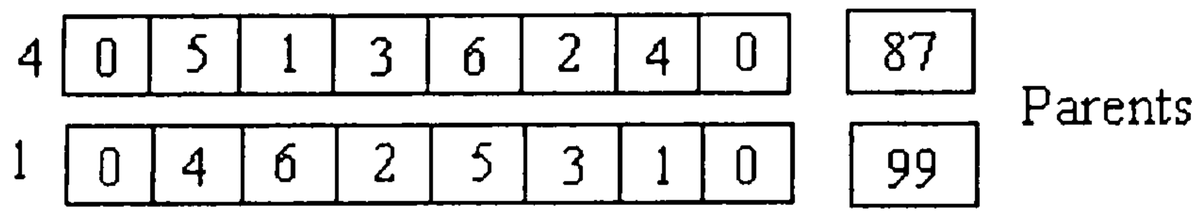


Figure 2.14: Second parent

These two strings (parents) will enter a mating pool for further genetic operator procedures.

Crossover operator is the next operation after reproduction, which is applied to the selected strings (parents). A crossover point is selected at random from the five possible positions as shown in Figure 2.15, and a new solution is created by combining partial solutions from the original parents.

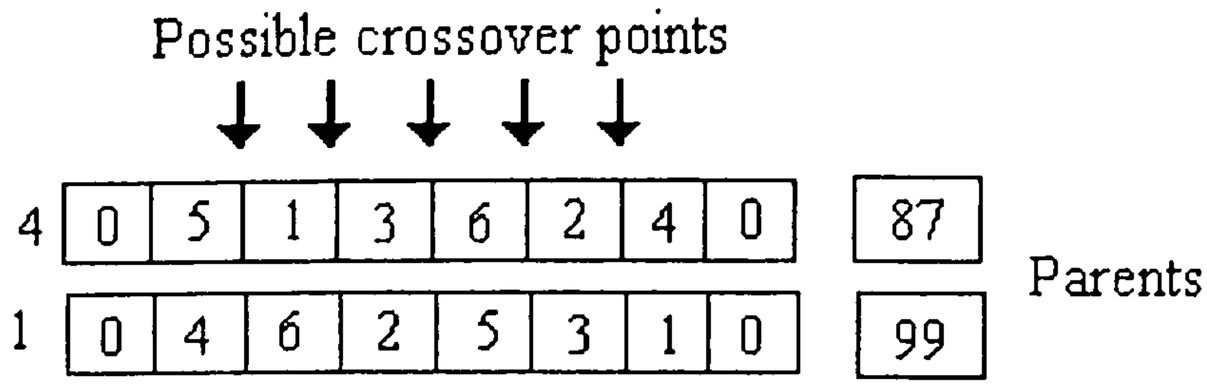


Figure 2.15: Five arrows represent the possible position for crossover point

Figure 2.16 illustrates an example for 1-point crossover operator, where all genes before the crossover point will remain and the two sets of genes after crossover point will be exchanged. There are other ways to apply crossover. For example, more crossover points can be chosen (2-point, 3-point, and multi-point crossover procedures have been developed from the simple 1-point). For this example, the crossover is located between the third position and the fourth position in the strings.

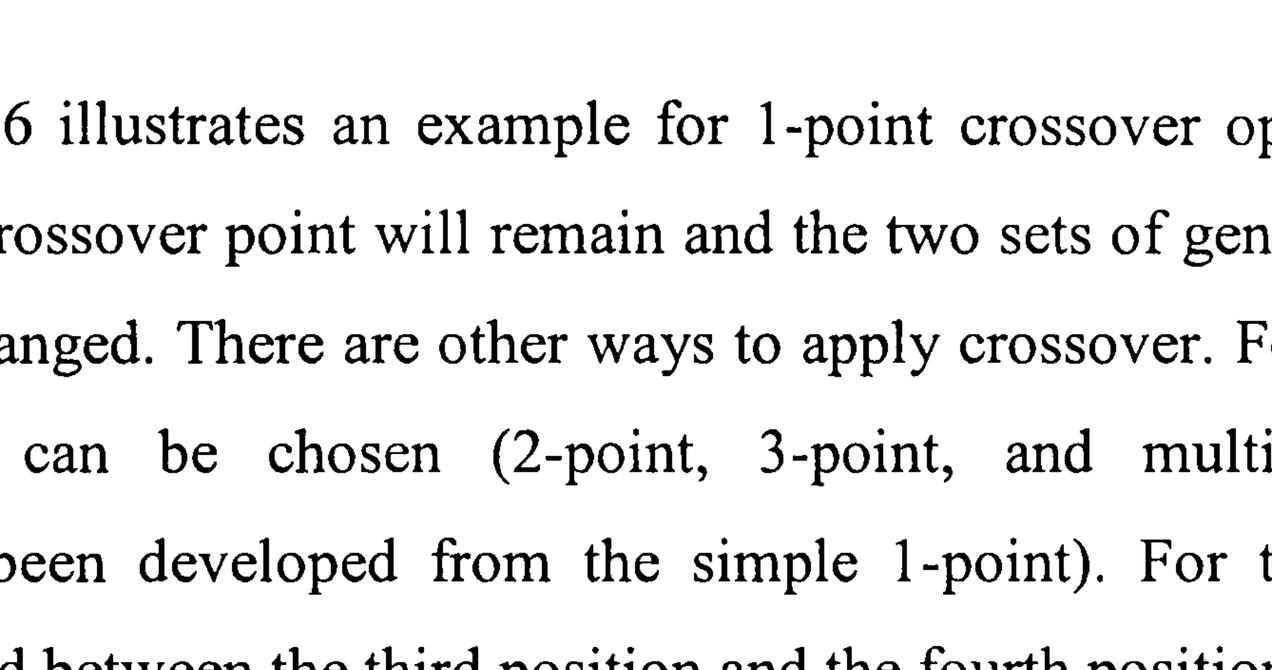


Figure 2.16: One-point crossover

All elements from the third position to the last position are swapped and two offspring are generated. As mentioned previously, variables are often called genes, and the values of variables are called alleles. It is obvious from the new generation (offspring) that there is duplication in some alleles (customer), such as 1 and 5 in first offspring and 4 and 6 in the second offspring. To avoid this duplication, partially matched crossover (PMX) is applied in this example. PMX was first proposed by Goldberg et al. in [3] where PMX proceeds by positionwise exchanges as shown in Figure 2.17. The same procedure as the one implemented previously is applied, where all elements from the third position to the last position are swapped. Since 1 has exchanged with 4 in position 6, 5 has exchanged with 6 in position 4, then to solve duplication, for positions 1 and 2, occupy position with the element has been exchanged with in the previous step. Where 1 occupies first position in Offspring 2, where 4 has been occupied previously, and 4 occupies second position in Offspring 1,

where 1 has been occupied previously. The same procedure applies on elements 5 and 6.

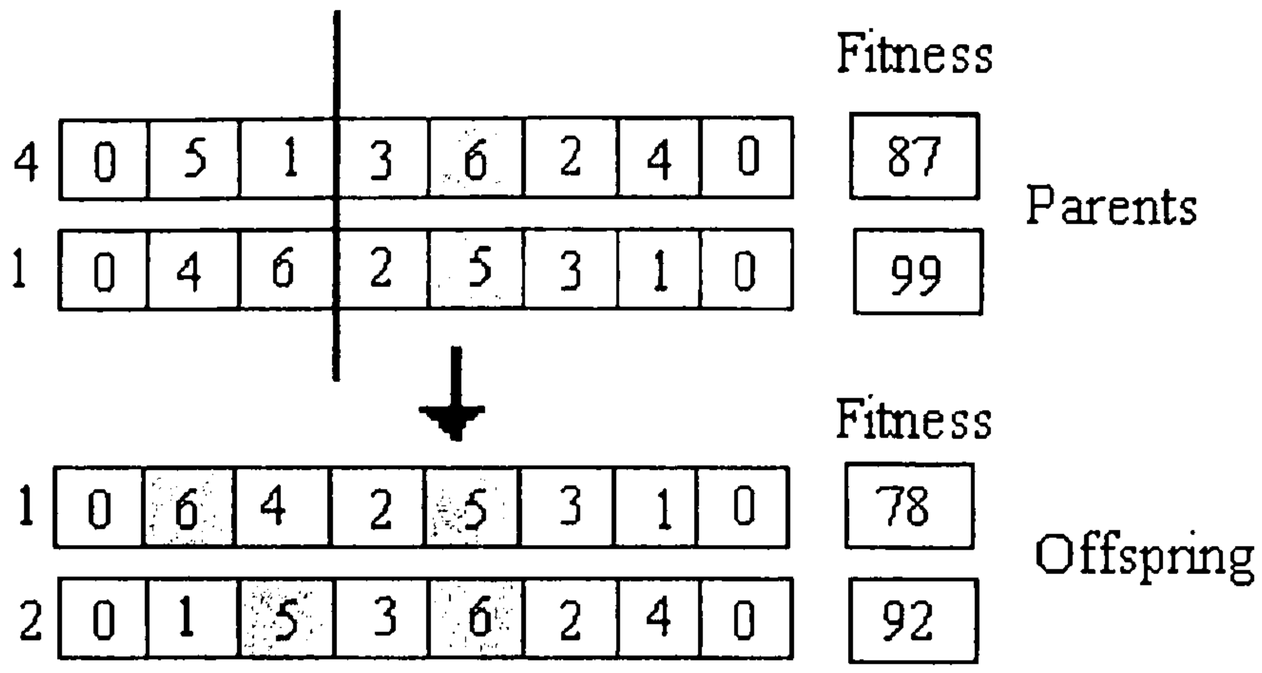


Figure 2.17: PMX crossover operation

It is obvious from the result obtained from the crossover operation that the fitness value in offspring 1 equals 78 which the best solution so far. *Elitism* should be used for saving the best-found solution. Elitism is an operation for searching for the best solution to date at various stages of the GA procedure. This guarantees the preservation of the best chromosomes at each generation.

The *Mutation* operator plays the next role in the simple GA. Mutation is achieved by swapping genes. There are several ways of implementing the mutation operator, such as Swap Nodes or Swap Sequence as shown in Figure 2.18.

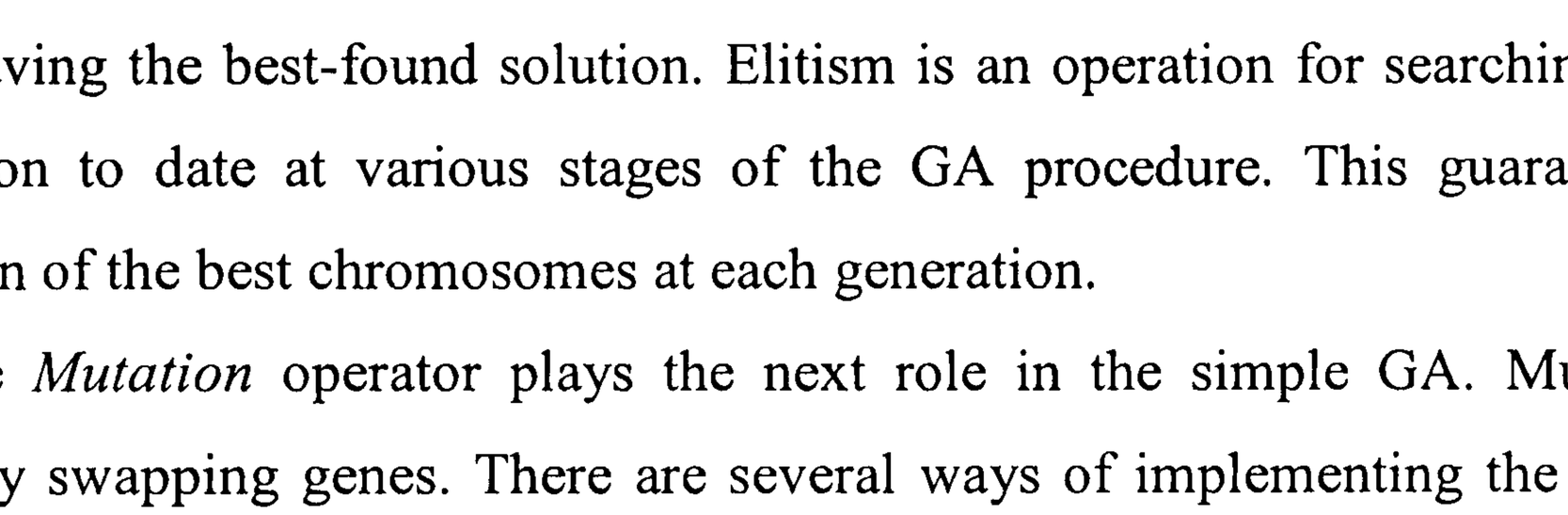


Figure 2.18: Several common mutation operations

In this example swap node will be implemented on both offspring as shown in Figure 2.19.

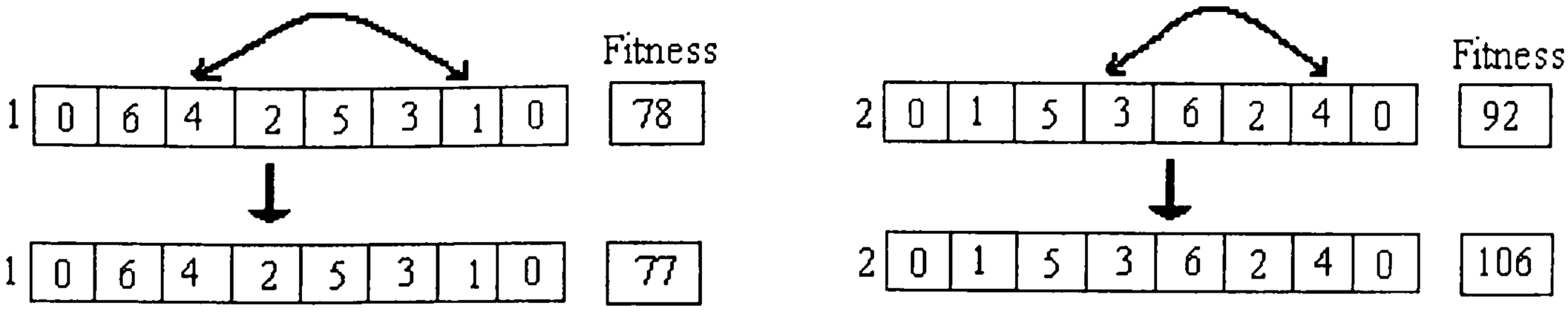


Figure 2.19: Swap nodes

Mutation involves a random element in order to help the search to obtain solutions that crossover alone may not encounter. The fitness value now is the best one so far which is clear from offspring 1 (fitness value = 77).

Once this operation is finished, the procedure is repeated by selecting a new parent from the initial population to create new offspring. The same operation is carried out and the process stops after a defined number of generations or a number of generations with no improvement in the best solution obtained.

The steps of simple GA can be summarised as follows:

Generate an initial population (randomly or structured method) with size = M

Repeat

Repeat

Repeat

Evaluate fitness (objective function) of each string in population

Select two parents from the population

Apply crossover to produce two offspring

Mutate offspring

Until number of generated offspring = M

Until stopping criterion is satisfied

It is important to mention some recommendations regarding other parameters associated with crossover and mutation operators. These parameters are the probability (rate) of crossover Pc and the probability of mutation Pm. The probability of crossover means the proportion of iterations at which crossover occurs, while the probability of mutation means the proportion of iterations at which a chromosome will be mutated during GA searching procedure. The crossover is necessary to exploit the population (to combine good genes) while mutation must be used only from time to time to diversify the search (to bring new genes). The value of Pc and Pm depend

on the type of the problem. In combinatorial optimization, for example, the usual practice is to perform one crossover at each iteration, i.e. Pc = 100% and Pm is between 2% to 5% [86]. Taniguchi and Shimamoto [114] found that Pc = 90% and $Pm = 2\%$.

In general, the value of Pc should be large, where Pm is recommended to be small. If the Pm is too large, the population is badly exploited.

This introduction to GAs covers only the basic version. There are many enhancements that have been proposed and the reader is refered to [50, 57, 91] for an in depth treatment of the subject. The following section presents a brief literature review.

2.2.3.2 Applications using GAs:

Baker and Ayechew [12] considered a vehicle routing problem (VRP) and tackled it using GA. The VRP consists of a number of customers with known demand, where each customer will be visited once. The problem is to find a set of delivery routes to visit all customers at minimal total cost. Two methods were used to generate the initial population. In the first method, solutions are generated using a structured approach and in the second method by a mixed approach which is part structured and part random. A population size of 30 was used for the smallest problems (50 customers), while a population size of 50 was used for large problems (from 75 customers up to 199 customers). Two parents are chosen from the population using the binary tournament method, where a group of T parents is selected from the population and compared. The one with the best fitness value is considered as the first parent, and the best value after the first parent is considered as the second parent. A standard crossover procedure is used to produce offspring. By experiments, the authors found that the best procedure was a 2-point crossover, where two crossover points are selected randomly. Offspring that duplicate an existing solution are eliminated. The authors applied a mutation operator by selecting two vehicles randomly and switched their positions. Computational results were presented, and a comparison with Simulated annealing (SA) and Tabu Search (TS) was made using through 14 vehicle routing problems which can be downloaded from OR-library (see Beasley [15]). Results showed that solutions were up 0.5% above best known results on average, with solution times that were not excessive. Choi et al. [28] solved a

symmetric travelling salesman problem using GA. In their study, infeasible solutions were generated and included in the population through genetic operators so that the solution can be utilized. The sum of arc costs in the tours represents the fitness value of a chromosome in the algorithm. Two crossover operations were presented, partially matched crossover (PMX) and tie break crossover (TBX). The methodology of TBX can be presented as follows: consider that two parents are given and two-crossover points (2-point) are selected randomly as shown in Figure 2.20.

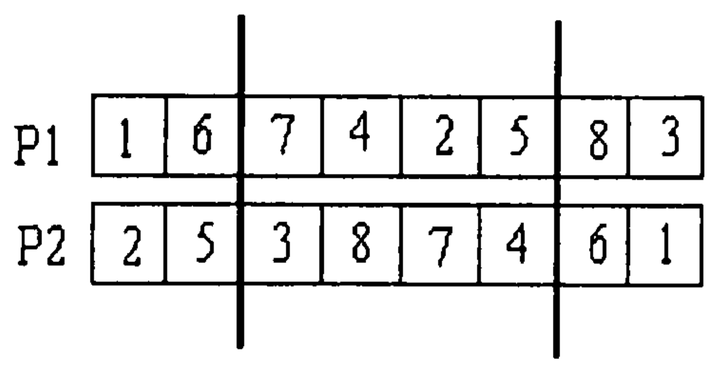


Figure2.20: Two parents with 2-point crossover

After applying the crossover operation, two new offspring are produced as shown in Figure 2.21.

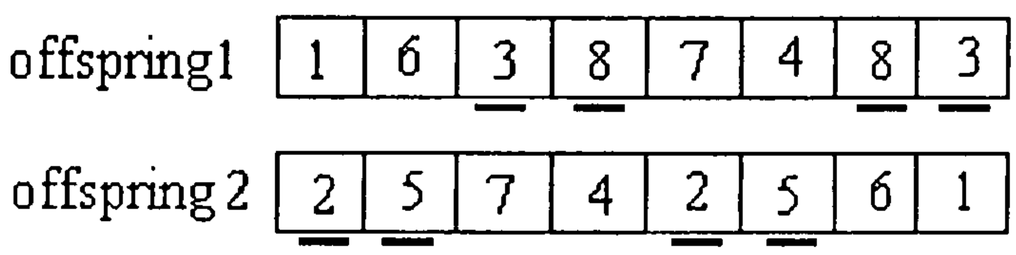


Figure2.21: Crossover operation

In this example some genes are duplicated as shown by an underscore. Solving the duplication can be performed by generating numbers between 0 and 1 randomly. For example (0.3 0.5 0.6 0.1) and (0.3 0.6 0.8 0.2). These numbers are added to duplicated genes as shown in figure 2.22.

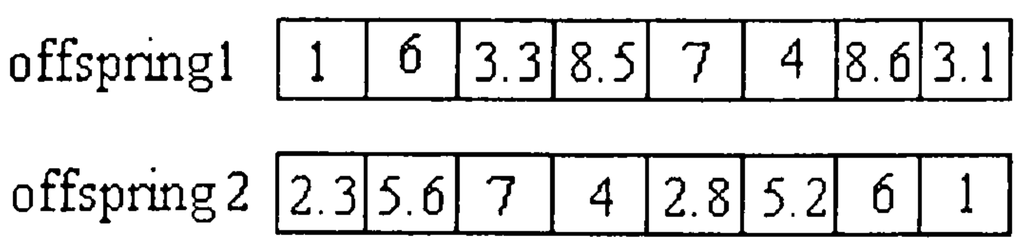


Figure 2.22: Duplicated genes

Finally, the smallest among the four underscored values in offspring1 change to allele value 2, the next smallest to 3, then 5, and the last to 8. The same operation is

applied to offspring2. The new two offspring, after removing the duplication, will be as displayed in Figure 2.23.

offspring1	1	6	3	5	7	4	8	2
offspring2	2	8	7	4	3	5	6	1

Figure 2.23: Duplicated genes

Through computational experiments the authors found that PMX and TMX performed better when they were used together rather than separately in the algorithm. Since 2-opt operation shifts the orientation of arcs, they used three-way swapping as a mutation operator to preserve the orientation of genes in parents. Because the algorithm they used generated infeasible solutions, they applied a repair algorithm called Karp's patching algorithm (details about this algorithm can be found [65]). The authors presented results that showed GA is efficient in producing high quality solutions for this type of problem. Tan et al. [111] considered vehicle routing problems (VRP). This problem involved routing a fleet of vehicles with limited capacities from the depot to serve a number of customers with known demands and predefined time window constraints. The objective is to minimise the total cost with a minimum number of vehicles without violating any constraints. The authors used heuristic methods to solve this problem, one of which was GA. The sequence for the customers in a chromosome represented the order of visiting these customers. Initial solutions were generated partially by using the push forward insertion heuristic PFIH (see Solomon [104]) and the remaining solutions were produced randomly. The population size was kept to 1000, and the total generation number was set at 500-1000. The fitness value associated with each chromosome is the total cost and the select on procedure is implemented by selecting the best chromosome fitness value mating pool. The PMX crossover operator was implemented and applied to each pair, where the crossover probability was more than 60%. The mutation operator was applied by selecting and swapping two customers in the same chromosome randomly. Infeasible solutions generated by crossover or mutation operations are deleted. GA was applied to problems with 100 customers served by a fleet of vehicles with between 2 and 21 vehicles. Compared to the best-published results the presented GA achieved an overall distance gap of not more than 4.3%. Prins [86] addressed VRP where chromosome repeated a sequence of n customer nodes. The population was

implemented as a number of chromosomes, and sorted in increasing order of fitness value. Three good solutions were generated using heuristics denoted as CW [32], MJ [79], and GM [46], then sorted as the first three chromosomes in the population. The other chromosomes were generated randomly and any identical chromosome deleted. The author adapted order crossover OX in this problem. This method does not generate any duplicated nodes. The methodology of this OX works as follows: suppose P1 and P2 are selected, then 2-point crossover pointed is implemented, where two cutting, before i= 4 and after j= 6, are selected randomly as shown in Figure 2.24.

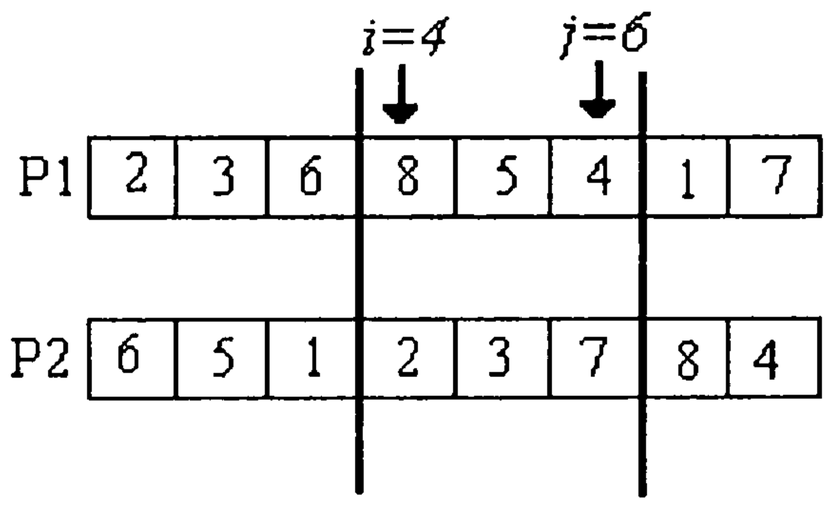


Figure 2.24: Two cutting

By applying the OX method, the offspring will be as shown in Figure 2.25

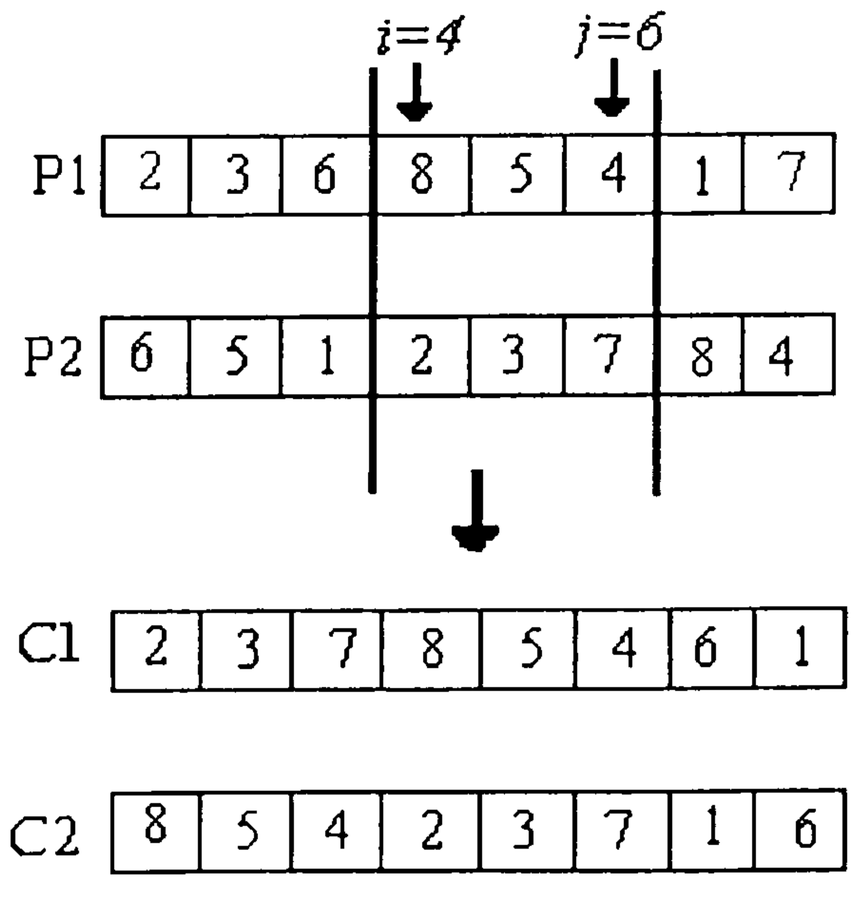


Figure 2.25: OX method

The substring between position 4 and 6 will be copied and placed in the same position for C1 (offspring1) and C2 (offspring2). Finally, P1 is transferred circularly

from j+1 forwards to complete C1 with the missing nodes. C2 is also filled using the same procedure with P1. The author applied local search procedure LS instead of simple mutation operators which achieved much better results. Offspring C generated by OX is improved by LS with fixed probability *Pm*. The GA was applied on 20 problems with a number of customers between 200 to 483 customers. The best standard setting of population size was equal 30, *Pm* started at 5% and increase up to 10%. GA gave the best solutions for 12 of the 20 problems.

Chapter 3: A Review of the Literature on Ship Routing and Scheduling

3.1 An Overview of Research on Transportation Routing and Scheduling

The transportation revolution poses challenges to transportation companies in providing a good service at an economic cost. Efforts continue to find creative solutions to the challenging environment for transportation. A survey of published work on routing and scheduling problems are presented in this chapter. The survey consists of three categories: (1) land transportation problems, (2) air transportation problems, and (3) ship routing and scheduling problems. Categories 1 and 2 are briefly summarised and the main focus is on ship routing and scheduling.

3.2 Land Routing and Scheduling Models:

This category consists of two subcategories: vehicle routing and scheduling models and railroad routing and scheduling models.

3.2.1 Vehicle Routing and Scheduling:

Most studies have been devoted to the general Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). Some of these studies were presented in Chapter 2.

The Vehicle Routing with Time Windows Problem (VRWTP) is the extension of VRP. This type of problem has many real world applications, such as gas and petroleum deliveries, bank deliveries, postal deliveries, and school bus routing. Many research studies have been conducted on VRWTP because of the challenging difficulty and strategic cost impact [97]. Russel and Chiang [97] considered VRWTP. The objective was to minimise the number of vehicles required and the total travel distance incurred by the fleet of vehicles. All feasible routes were generated and a set covering formulation was applied to obtain the optimal solution. Haghani and Jung [54] solved the problem of a pick-up or delivery vehicle routing problem with soft time window using heterogeneous vehicles with different capacities. The authors solved the problem using an exact approach based on mixed integer programming (MIP). The authors also proposed a Genetic Algorithm (GA) heuristic method for large-scale problems. For small size problems (10 demands), the GA produced

solutions close to the exact solutions. For large problems (30 demands), the GA generated solutions that were within 8% of optimality.

Yau [123] addressed a problem of truck scheduling. This problem is to schedule a fleet of trucks from many warehouses to many delivery points subject to some constraints such as truck capacity, loading and unloading time, and travelling time. The problem owner was STARLINK, a warehousing and a distribution company which based in Hong Kong. The author solved the problem using a heuristic method to build a complete schedule. Each step of the method involves two possible actions: (i) a delivery point may be inserted into the set of partial routes, or (ii) a delivery point in the partial solution may be moved to another position in the routes where the latter has the priority over the former. This procedure continues until all delivery points have been included into the routes. An average of 8.8% improvement in cost compared to the manual method used by STARLINK Company was reported.

3.2.2 Railroad Scheduling Models

A railroad system is considered an important part of the transportation network in many countries, both for passengers and cargoes. In the United States, for example, about 30% of total revenues of all carriers are delivered from railroad, and these account for 35.6% of total express freight transport (Assad [11]). Train routing and scheduling decisions represent a large problem that is made more difficult by a number of constraints on track, engine and crew availability (Assad [11]).

Train scheduling activities in planning applications are conducted in two phases. The first phase is line planning (i.e. sketch planning), which is to specify the routes, frequencies, and stop schedules of trains. The second phase is schedule generation, which constructs the arrival and departure times for each train at passing stations.

The train scheduling problem is to find an optimal train timetable, subject to a number of operational and safety requirements, such as crossing heading and overtaking constraints. The major objective of planning applications is to minimise the overall operational costs, while freight traffic demand and passenger satisfaction are considered.

Zhon and Zhong [126] considered a problem of double-track train scheduling for planning applications with multiple objectives. Double-track train means that there can be two trains in any direction at the same time. Two types of trains are designed

to operate on the same track system, namely high-speed trains (250-300 km/h) and medium-speed trains (160 km/h). The first objective is to minimise the expected waiting time for high-speed trains, while the second objective is to minimise the total travel times for both high-speed and medium-speed trains. The traditional planning approach usually schedules high-speed trains followed by medium-speed trains. To prevent extremely long travel times for medium-speed train passengers, the authors applied a branch and bound algorithm and heuristic method. Results were obtained by applying their approach on a case study based on a railroad in China.

To reduce the chance of a missed connection, buffer times are inserted into the schedule. Vansteenwegen and Oudheusden [120] presented a technique for creating robust timetables. Linear Programming was used to construct an improved timetable. The authors applied their approach on the Belgium Railway Company, where waiting cost was 40% lower compared to the existing timetable.

Delays can occur when a train on a single track switches to the sidetrack to allow another train travelling in the same direction to pass. Delays also occur if the length of the train's standstill interval at the arrival platform exceeds a certain lower bound, then the train may be shunted towards a parking area in order to release the arrival platform. Zwaneveld et al. [127] addressed a problem of routing trains through a railway station. Several aspects were taken in account such as capacity, safety and customer service. The two major objectives were, minimising the number of shunting movements, while the second objective was to maximise the preferences of the trains for platforms or routes. The authors solved the problem to optimality by using integer programming.

A comprehensive survey for train scheduling problems and other train problems is given by Cordeau et al. [33].

3.3 Airline Routing and Scheduling:

Commercial airlines face a series of operational decision problems regarding the deployment of their fleets. First, they have to assign aircraft types to flights (fleet assignments). Secondly, a specific aircraft has to be assigned to each flight (fleet routing), and finally, crews have to be assigned to the aircraft (crew scheduling). These problems have to be solved whilst meeting the aircraft maintenance requirements (maintenance planning) (Ronen [95]). Achieving a match between capacity and demand for optimal revenue is the objective of fleet assignment.

Many researchers presented models to solve these types of problems. Subramanian et al. [106] considered a fleet assignment problem at Delta Airlines. They solved the problem by using interior point methods [59]. They reported savings of up to $220,000 per day for the airline company by using this method. Ronen [95] considered a combined fleet assignment and fleet routing problem which involved maintenance activities and crew availability considerations. The problem was solved using an elastic set partitioning model that was embedded within a decision support system. Holstand and Sorenson [58] considered a maintenance problem at the Federal Aviation Administration (FAA). They solved the problem by generating several thousands feasible routes. Then, a minimum cost subset of these routes is selected by solving a set covering formulation. Subramaniam and Marsten [105] considered this problem by presenting an integer programming formulation and applied Lagrangian Relaxation.

Operational changes due to technical difficulties, bad weather, head winds on route, etc, may lead to delaying or cancelling flights, swapping aircraft among flights or using spare aircraft. These problems affect future deployment of aircraft and crews. This kind of problems called a Day of Operations Scheduling (DAYOPS) problem. Actually, many researchers presented models to find a good balance between the optimality of a proposed solution and the speed with which it is achieved (dispatchers usually adjust the planned schedules under stress and have little time to analyse cost-effective scheduling alternatives). Rakshit et al. [87] collected 251 cases of aircraft delays during one day. They succeeded in saving 8495 minutes by using simple swaps between planed aircraft. A conservative estimate of the value per minute of delay is $20. Thus a $169,900 saving in delay cost was achieved in the given period.

Teodornic and Guberinic [115] proposed a simplified model to minimise the overall passengers waiting time. Later, Teodornic and Stojkovic [116] developed a heuristic procedure which first minimises the number of cancelled flights and then minimises the overall passenger waiting time.

For more applications in this area, the reader can refer to Gopalan and Talluri [52] who presented a survey of mathematical models in airline schedule planning.

3.4 Ship Routing and Scheduling Modes

This category is further divided into three ship routing and scheduling sub-sections:

3.4.1 Liner Operations:

The literature on modelling techniques and approaches for liner shipping is quite limited. However, in recent years an increased activity in this area has become evident, due to China's economic boom (Lane et al.[70, 100] and Shantani et al [100]). In 1995 container liner ships made up 5.9% of the world fleet's total dead weight capacity. In 2001 there was an increase of 9% [29]. Despite these facts, liner shipping has drawn little attention from researchers, at least in its quantitative aspects. This might be due to the nature of some of variables and factors that influence the operations of a liner company, such as some minimum required services frequencies, subsidies, and government regulations. These factors have discouraged attempts towards a systematic approach to the analysis and optimization of liner transportation systems.

Since there exists a high degree of uncertainty with liner operations because of factors such as weather conditions, strikes, or mechanical problems, which made the routing or scheduling problems for ships less structured than in the case with other transportation modes [27]. The main modelling techniques relied mainly on simulation and heuristic decision rules to solve liner problems. Exact methods, such as linear, integer, or non-linear programming can be used to tackle liner fleet deployment and scheduling problems, given that the cargo forecasts are dependable. It is known that simulation techniques can provide an evaluation for the system or help users to choose the best solution among limited group of alternatives submitted to it. On the other hand, exact methods present an optimal solutions (Perakis and Jaramillo[83]).

Datz et al. [39] presented a simulation model for liner operations which produces a schedule and gives an estimation of the financial consequences for the schedule. The model took into account the chance that a "promised" cargo could be cancelled. Simulation was also used by Kydland [67] and Olson et al. [82] to solve liner problems. Kydland [67] utilized linear programming to developed a stochastic simulation model to determine the optimal number of ship required to offer a specified service frequency. Olson et al. [82] presented a deterministic simulation model to produce medium term regular schedules for a liner company, operating between the US west coast and Hawaii. Additionally, the model was used to investigate the impact of factors such as waiting in port for additional cargo, and to evaluate scheduling decisions.

Cho and Perakis [27] addressed the problem of determining a fleet size and design in optimal liner routes for a liner shipping company. They solved the problem by first, generating a number of candidate feasible routes for different ships, and then applied a linear programming formulation to solve it. The generated feasible routes are represented by the variables. The authors also presented a mixed integer programming model to extend the model to consider more investment by expanding fleet capacity such as building or purchasing new ships, or resorting to the market for chartering.

Fagerholt [42] considered a problem of designing an optimal fleet in a real liner shipping problem. The methodology used to solve the problem consisted of two steps; first generating ship routes based on a dynamic programming algorithm. In the second step, a Set Partitioning model (SPP) was formulated. The solution considered only ships with fixed speed. Later on, Fagerholt and Lindstad proposed an approach that could cater for different ship speeds. The model was applied on real problems belonging to offshore supply operations based in the Norwegian Sea. The model presented saving of $7 million compared to the manual method.

Boffey et al. [19] solved a problem of scheduling container ships over the North Atlantic route. The authors developed an optimisation model which was presented as an interactive computer program based on a greedy heuristic to generate schedules. This interactive computer program provided the user with information such as profitability, timing, and transit times.

Nemhauser and Yu [81] studied a model for rail service which can be used for a liner problem. They used dynamic programming to locate the optimal frequency of services to maximise the profit over the planning horizon.

Rana and Vickson [88] presented a deterministic mathematical programming model for optimally routing a simple container ship. The formulation involved nonlinearity, which was converted into a number of mixed integer programs. Bender's decomposition was applied to the mixed integer programs, where a specialized algorithm was used to solve the integer network subprograms. Later, Rana and Vickson [89] provided an extension to the work of their previous paper , by allowing multiple ships. The authors formulated the problem as a non-linear problemand solved it using Langrangean Relaxation, which decomposed it into several sub-problems, one for each vessel, and each sub-problem was decomposed into a number of mixed integer programs.

Perakis and Jaramillo [83] presented a linear programming model to minimize the annual operating costs of a fleet of liners. This minimization is equivalent to maximizing the profit. The operating costs are; fuel costs, daily running costs (such as salaries and benefits of the crew), port charges, and canal fees. The authors also presented two separated approaches to find the optimal speed and frequency of service on routes by using non-linear constrained optimisation. In a subsequent paper, Jaramillo and Perakis [63] continued the work. An optimum deployment of a liner fleet that contains both controlled and chartered ships subject to time, frequency and other constraints was developed. To avoid nonlinearity and to formulate the problem using linear programming, the authors resorted to fixing the speed of the ships and the frequency of the service on each route. They provided an example where the solution was derived by rounding the number of ships on a route obtained by using Linear Programming (LP). The authors used sensitivity analysis to present insights in to the impact of the various cost components and constraints on the profitability of the liner company. This analysis indicated that the operating costs are very sensitive to the composition of the fleet with more owned ships resulting in higher operating expenses. The example they presented was based on the data provided by the liner fleet of Flota Mercante Grancolombiana (FMG), a large liner company operating routes between Colombia and many countries such as US, and Japan. The authors compared their solution with the present fleet deployment of the company. The result presented a reduction of 3% in operating costs without any changes in the service frequencies. A reduction of 13% could be a achieved with a little modification on the service frequencies. Six years later, Powell and Perakis [85] provided an extension and improvement of the previous work. They solved the problem by an integer programming formulation. The model provides information such as the optimal deployment of a fleet of ships, service, route, and chartering from the market. The model was applied to a real world problem and resulting a significant saving compared to the actual deployment.

The sea container industry is confronted with the problem of allocating empty containers. If the inventory control fails in locating empty containers at demand points at the requested time, there are two possible decisions: load rejection or container leasing. As a result, a significant decision at the operational level is how to transfer empty containers in an efficient and timely way and / or lease containers. Shintani et al. [100] proposed a design method for containership liner shipping service network

that incorporates empty container repositioning among calling ports. The authors solved the problem by linear programming and a heuristic method based on Genetic Algorithm (GA). The objective was to maximise the profit for a liner shipping company, by finding a set of calling ports, the number of ships of each ship size, the resulting cruising speed to be deployed in the service network, and an associated port calling sequence. Experiments and analysis were presented based on a case study.

3.4.2 Tramp Operations

Very little work has been done on the allocation, routing, and scheduling of tramp shipping. Christiansen et al [29] refer to the shortage of research related to tramp shipping compared to the large number of comparatively small operators in the tramp market. Large shipping companies usually watch the tramp market as a secondary one, and this occurs because of uncertainty over ship availability and the reality that ships of a tramp operation when such an operation is very profitable (Ronen [92]).

The first author to address this type of problem was Appelgren [10], who used the Dantzig-Wolfe decomposition approach for routing and scheduling. Bronmo et al [24] considered a tramp ship scheduling problem. The problem involved pickups and deliveries of bulk cargoes between specified ports within a specific time frame. The objective was to minimise the profit of this operation. The authors formulated the problem as a Set Partitioning Problem (SPP), where all possible schedules are generated. The authors also proposed a heuristic method to tackle large size problems, in order to measure heuristic efficiency compared to the SPP optimal approach. Computational results were presented with excellent solutions.

Fagerholt [43] developed a decision support system (DSS) for ship fleet scheduling for both tramp and industrial shipping. The author designed an interactive computer program consisting of two different heuristic methods. The first heuristic method generates initial schedules for each ship, where as the second, a heuristic hybrid search method, is used to improve the solution to the ship scheduling problem.

3.4.3 Industrial Operations

Industrial operators have control over both the ships and the cargos. This type of operation is well known in carrying bulk and semi-bulk commodities, such as oil,

ore, coal, pulp, lumber, and sugar. Industrial operations have attracted more research than liner or tramp shipping (Perakis and Jaramillo [83]).

Most of existing ship routing and related scheduling studies are covered by the following three major review papers: Ronen [92, 95] and Christian et al. [29].

Seaborne routing and scheduling problems for bulk products can be presented into two separate types: *inventory routing* and *cargo routing* problems. First, inventory routing problems are constrained by inventory requirements, where the level of products at ports should be maintained. The second type, routing problems are usually constrained by the cargo, which is specified by loading/ unloading ports, and by time windows for loading and unloading.

Al-Khayyal and Hwang [6] considered an inventory routing problem. A heterogeneous fleet of ships with multiple compartments capable of carrying different products simultaneously were employed. The fleet is used to distribute multiple liquid products to many ports, where each port is either a producer or a consumer. The average production and consumption rate for each product is known. The problem is to determine which product is to be loaded (or unloaded) onto which ship, the quantity to be loaded/unloaded, and to schedule the arrivals and departures of the ships to maintain the inventory levels. The objective is to minimise the total daily cost of the ships. The authors formulated the problem using mixed integer linear programming and applied CPLEX 7.5 to solve the problem. More than 100 test problems were randomly generated and solved. Results and analysis were presented.

Ronen [96] considered an inventory routing problem. There is a single loading port at the origin and a single unloading port at destination, for each vessel's voyage, but a vessel may load multiple products (in separate compartments). This problem tried to determine when and how to ship each product from which origin to which destination and by which vessel. The objective function of this problem was to minimise the overall shipping cost and not violate the safety stock and storage volume. A mixed integer programming was presented as a first approach, and solved with a 1% optimality tolerance, while the heuristic approach was within 15% of the LP bound.

Much research has been conducted with seaborne routing and scheduling problems for bulk products for the second type. Dantzig and Fulkerson [37] considered a tanker scheduling problem for a homogeneous fleet, i.e., speeds, carrying capabilities, and operating expenses were similar for all ships. Meantime, loading and

unloading dates were predetermined. The authors considered that just one loading and one unloading port for each ship. The objective was to generate a specified schedule, where the number of ships is minimised. Later, Briskon [23] extended the problem the previous work by allowing several unloading ports. The author exploited dynamic programming to find out the schedule of unloading ports for each ship.

Bellmore et al. [16] extended the problem of Dantzig and Fulkerson [37] by allowing different types of ships and permitting partially loaded ships. Predefined delivery dates within a certain time interval were provided. The objective was to maximize the total benefit of deliveries. They solved the problem using a mixed integer linear programming model, where branch and bound was used. No application or results were presented.

Ronen [93] considered a single voyage ship-scheduling problem of an organization that ships quantities of bulk or semi bulk commodity from one origin to several destinations. A ship schedule specifies the cargos to be carried on a ship, the locations between which each of these cargos are carried and the timings of each activity (loading, unloading, transits). A feasible ship schedule is a schedule that satisfies all specified practical requirements. It is assume that the total capacity of the available fleet is greater than the total cargo demand, which means that there is no need to resort to chartering, was presented. Each ship is allowed to load cargoes to multiple destinations, and the demand at each destination can be met by more than one ship. Three ways to address the problem were presented: two different heuristic algorithms to minimize the cost per ton-mile of cargoes and an exact optimising algorithm (applied to a small sized problem due to excessive computing time). Ronen tried to minimize the cost of operating a fleet to deliver available shipments. He calculated the operating cost (consisting of port costs and bunker fuel consumption during the voyage), canal dues, the cost of the unit loaded to the vessel, demurrage (payment for holding a vessel on anchor, without employment) and unloading port charges for each route. Ronen presented 20 examples with between 3 to 13 vessels serving 5 to 13 ports (the exact algorithm was applied to 5 vessels and 7 ports). These examples illustrated the reduction of cost compared with the ad hoc methods currently used by fleet managers.

Brown et.al. [25] considered a crude oil tanker scheduling problem faced by a major company, and solved it using an elastic set-partitioning model. The oil company controlled several crude oil vessels of similar sizes to ship crude oil (full

shipload) from Middle East to North America and Europe. The problem was solved by generating all feasible schedules, and selecting the optimal subset of schedules. The model took into account all fleet cost components, and determined the optimal speed for the vessels, and the best routing of empty vessels. The model also determined which shipments to load on controlled vessels, and which to spot charter.

Bausch et al. [14] expanded the Brown et al. [25] model for scheduling shipments of refined oil products from several refineries to multiple destinations using either tankers or barges. A microcomputer system was designed with an EXCEL user interface. A detailed cost model was integrated in the system. The model can be used to find out the optimal speed of the ships and the need for spot chartering ships.

Mehrez et al. [77] considered the problem of an ocean transportation system for bulk products from an overseas port to transshipment ports on the Atlantic Coast and then over land to final destinations. This model took into account the number and the size of vessels to charter in each time period during the planning horizon, the number and location of the transshipment ports to use, and the delivery from port to final destination. The resulting formulation of the problem was a mixed-integer program.

Fagerholt and Christiansen [44] considered a problem of a bulk ship scheduling which was a multi-ship pickup and delivery with time window problems. Time windows were presented for each customer, both for the pickup and delivery ports. Different ship capacities and cruising speeds were available. Management may resort to the market to charter one or more ships. There were a number of types of ships with fixed compartments which enable different products to be loaded on the same route. The authors solved the problem using a set partitioning approach. The approach consisted of two stages: firstly, a number of candidate schedules generated for each ship with allocation of cargoes to the ships compartments. In the second stage, SPP was applied to minimise transportation cost. For large problems, a subset of candidate schedules were inclinded in order to reduce computational time and memory requirements. The model was applied on a real case from Hydro Agri's waterborne transportation of fertilizer in Northern Europe.

Sherali et al. [99] considered a problem for routing and scheduling ships. The main thrust of the research was focused on the Kuwait Petroleum Corporation (KPC) Problem. A fleet of ships was involved in delivering crude oil and refined oil-related products from Kuwait to ports around the world. The model considered a fleet of

ships consisting of controlled and chartered ships. There are two route options to deliver shipments to US or European countries: around the Cape of Good Hope or through Suez Canal. The problem was solved and formulated using an integer programming model. The results presented a good cost reduction compared with the ad-hoc schedule procedure used by Kuwait Petroleum Corporation (KPC).

Most cargo routing problems adopt a set partitioning problem (SPP) approach to solve the problems. However, heuristic methods such as Tabu Search, Genetic Algorithm, Simulated Annealing, or other heuristic methods are proposed for large-scale problems.

Chapter 4: Ship Scheduling: Specification and algorithms

4.1 Problem specification

4.1.1 Definition of the Ship Routing and Scheduling Problem (SRSP)

The Ship Routing and Scheduling Problem (SRSP) considered here involves efficient scheduling of a set V of a heterogeneous fleet of m ships (controlled and chartered ships, where the operations manager can resort to the market for spot chartered ships at a given cost), where $V = \{1,..,m\}$. $N = \{0,1,..,n\}$ is a set containing an origin port and n cargo-ports, where 0 represents the origin and $\{1,..,n\}$represents the set of cargo-ports to be transported by the ships. Each cargo-port consists of a given quantity to be loaded at the origin and delivered to a cargo port. For any cargo-port there are time-window constraints on the earliest and latest time for arrival at each cargo port. No ship is allowed to arrive outside the time-window. It is assumed that all ports can accommodate any ship. Each ship has a set of known attributes such as capacity and availability at the origin. All ships can carry any type of cargo. There are operations costs for any ship engaged in servicing one or more cargo-ports. These expenses relate to loading, unloading, crewing, and bunker (fuel) consumption in sailing. The cost of using a spot chartered ship is more than the cost of using a controlled ship of the same type. In addition to the above-mentioned costs, the overall cost for each ship includes port charges. The decisions required are to assign a ship to each cargo-port either by using a controlled or a spot ship at minimum overall cost. These decisions are usually made for a planning time horizon of several months with revisions to the schedule being made during the planning horizon as emergencies occur or new data become available.

Two classes of routing and scheduling problems are considered in this thesis and are discussed in the following section.

4.1.2 Classes of SRSP

There are two classes of scheduling problem considered in this thesis. The first one is called single-cargo schedule, while the second is called multi-cargo schedule. The single-cargo schedule problem consists of routes containing trips that service only one cargo-port on each trip, where each specific ship returns to the origin

after each trip for loading for the next trip. For example, consider there are 3 cargo-ports A, B, and C. A route can be presented as follows:

0—A—0—B—0—C—0

The multi-cargo schedule considers routes in which each trip may service more than one cargo-port. Consider the previous example, the route could be presented as follows:

0—A— B—0—C—0

Kuwait Petroleum Company (KPC) uses single-cargo schedules, whereas most other companies use multi-cargo schedules.

In this thesis, two computational approaches to handle SRSP are considered. The first is an optimisation approach based on the set partitioning problem (SPP) and the second approach is an approximate method based on Tabu search (TS). The following section will give details about notation, while the other sections will explain the use of each approach.

4.1.3 Notation

The following notation is used:

Data:

$V = \{1,..,m\}$, denotes the set of m ships to be scheduled, indexed by v, where $v \in V$

$N = \{0,1,..,n\}$, denotes the set of n cargo-ports to be visited, indexed by i, where $i \in N$ and 0 denotes to origin and $N \setminus \{0\}$ denotes cargo-ports

AV_v denotes to the availability time of ship v at the origin $\forall v \in V$

e_i earliest arrival time for cargo-port i $\forall i \in \{N \setminus 0\}$

l_i latest arrival time for cargo-port i $\forall i \in \{N \setminus 0\}$

Q_i cargo-port quantity of cargo-port i (tons) $\forall i \in \{N \setminus 0\}$

d_{ik} distance (in days) between cargo port i and cargo port k, where $i,k \in N$

CT_v capacity of ship v (tons) $\forall v \in V$

PC_{iv} port entrance due (fee) at cargo port i for ship v $\forall i \in \{N \setminus 0\}, \forall v \in V$

SP_v sailing cost using ship v (per day) $\forall v \in V$

LD_{iv} time required for loading cargo-port i onto ship v (days) $\forall i \in \{N \setminus 0\} \forall v \in V$

UL_{iv} time required for unloading cargo-port i from ship v (days) $\forall i \in \{N \setminus 0\} \forall v \in V$

OC_{iv} operating cost for ship v to handle cargo-port i, including loading and unloading costs $\forall i \in \{N \setminus 0\} \forall v \in V$

WP_v waiting cost (idle in the ocean) for ship v (per day) $\forall v \in V$

Decision Variables

f_{iv} actual arrival time for ship v to cargo-port i $\forall i \in \{N \setminus 0\}, \forall v \in V$

wv_{iv} duration of waiting (idle) time for ship v until time-window of cargo-port i opens (days) $\forall i \in \{N \setminus 0\}, \forall v \in V$

The following section will present details about the exact approach used to solve the problem.

4.1.4 Exact algorithm

The approach adopted to solve this type of problem is based on the Set Partitioning Problem (SSP). The major advantages of SPP models are that cost can be easily incorporated when generating all feasible schedules. The SPP model involves generating candidate feasible schedules and can be solved by use of a heuristic method or by optimisation depending on the desired solution quality and the time available for solution. SPP is a widely used model for solving routing and scheduling problems, Christiansen et al. [29] reported that 40% of the reviewed papers use the SPP models or a variant. This thesis adapted SPP for exact solution, where candidate feasible schedules are generated. The SPP formulation is the same for the two classes of problem considered here.

In this section, notations and formulation for solving SRSP are presented, while the full description of generation of the candidate schedules will be presented in the following section.

S_v denotes to a set of candidate schedules are available for ship v, and j indexed to specific schedule $\forall v \in V$

C_{vj} denotes to the cost for using schedule j for ship v $\forall v \in V$, $j \in S_v$

$$SC_{ivj} = \begin{cases} 1 & \text{if schedule } j \text{ for ship } v \text{ servicing cargo } i \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i \in \{N \setminus 0\} \; \forall v \in V \;\; j \in S_v$$

Decision variable

$$x_{vj} = \begin{cases} 1 & \text{if schedule } j \text{ for ship } v \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \Bigg\} \forall v \in V \;\; j \in S_v$$

The following is SPP based formulation for solving SRSP

$$Min \qquad \sum_{v \in V} \sum_{j \in S_v} C_{vj} x_{vj} \tag{4.1}$$

Subject to

$$\sum_{v \in V} \sum_{j \in S_v} SC_{ivj} x_{vj} = 1 \quad , \forall i \in N \tag{4.2}$$

$$\sum_{j \in S_v} x_{vj} \leq 1 \quad , \forall v \in V \tag{4.3}$$

$$x_{vj} \in \{0,1\} \quad , \forall v \in V, j \in S_v \tag{4.4}$$

The objective function (4.1) represents overall cost. Constraints (4.2) ensure that all cargo-ports have been delivered either by controlled ship or spot chartered ship. Constraints (4.3) ensure that each ship has used at most once.

The following section will explain the method of generating all possible candidate schedules SC_{ivj} .

4.1.4.1 Generation of the candidate schedules

Candidate schedule generation uses a procedure that generates a set of feasible candidate schedules, and limitation of generating of feasible schedules for large problems, each of which corresponds to a variable in the SPP model. Each individual candidate schedule has a route for a specific ship containing one or more cargo-ports. A number of candidate schedules for a specific ship are represented by a subset. The union for all subsets (for the set of all ships in the fleet) forms a set of candidate schedules. For example, suppose there are four cargo-ports A, B, C, and D to be delivered and two ships available to implement this task. Table 4.1 illustrates a possible set of candidate feasible schedules for each ship.

Ship	**Schedule**	**Cargo-port** / **Schedule** *j*	**A**	**B**	**C**	**D**	**Cost**
v	**Set**						
1	S_1	*1*	1	0	0	0	£ 2000
		2	1	0	1	0	£ 3500
		3	0	0	1	0	£ 1800
		4	0	0	1	1	£ 3450
		5	0	1	0	0	£ 1900
		6	0	1	0	1	£ 3550
		7	0	0	0	1	£ 2200
2	S_2	*1*	0	1	0	0	£ 1500
		2	0	1	0	1	£ 2980
		3	0	0	0	1	£ 1900
		4	0	0	1	0	£ 1700

Table 4.1: Possible set of candidate feasible schedules

There is a set of 11 candidate feasible schedules in Table 4.1, ship 1 has a subset of 7 candidate feasible schedules and ship 2 has a subset of 4 candidate feasible schedules. A value of one in the body of the table indicates that the corresponding cargo-port is delivered. For example, schedule 4 represents a candidate feasible schedule for ship 1, where the route in this schedule delivers cargo-ports C and D at a cost £ 3450. A cargo-port's quantity relative to ship capacity or the availability time of the ship at origin may restrict the ship from delivering a particular cargo-port. For

example, ship 2 cannot deliver cargo-port A since some or all of these restrictions cannot be met. These feasibility tests will be discussed later.

The candidate feasible schedules in this example are represented by SC_{ivj} in SRSP, for example, $SC_{113} = 0$ and SC_{313}=1, which means that cargo-port A in the third schedule for ship one will not be delivered, while cargo-port C will be delivered. The subset of candidate feasible schedules for ship v is denoted as S_v. In the previous example, $S_1 = \{1,..,7\}$ and $S_2 = \{1,..,4\}$. For each schedule there is an overall cost denoted by C_{vj}.

Since column generation for this type of problem may produce an enormous number of columns, an assumption has to be imposed to restrict the number of columns generated in order to reduce the computational time. However, heuristic methods (TS) is capable of solving large-scale problems of this type. Therefore, all cargo-ports are arranged in order according to earliest delivery time for each specific cargo-port before starting to generate candidate schedules. This generation of possible schedules will be accomplished in cargo-port order.

Since there are some differences between single-cargo and multi-cargo, the explanation for each class will be presented in separate sections.

A. Single-cargo

The generation of schedules is implemented in a systematic way by expanding an existing schedule by inserting a new cargo-port. Since the route in the schedule consists of origin and cargo-port nodes, there are at least 3 nodes in each route, where the first and the last nodes represent the origin. Before continuing to explain the method of generating candidate feasible schedules, it is necessary to illustrate the test of feasibility for each generated schedule.

Feasibility tests for each generated schedule will be implemented by considering each ship capacity or availability time at origin, as follows:

1. To ensure that each ship arrives within the required time-window, the following test is necessary

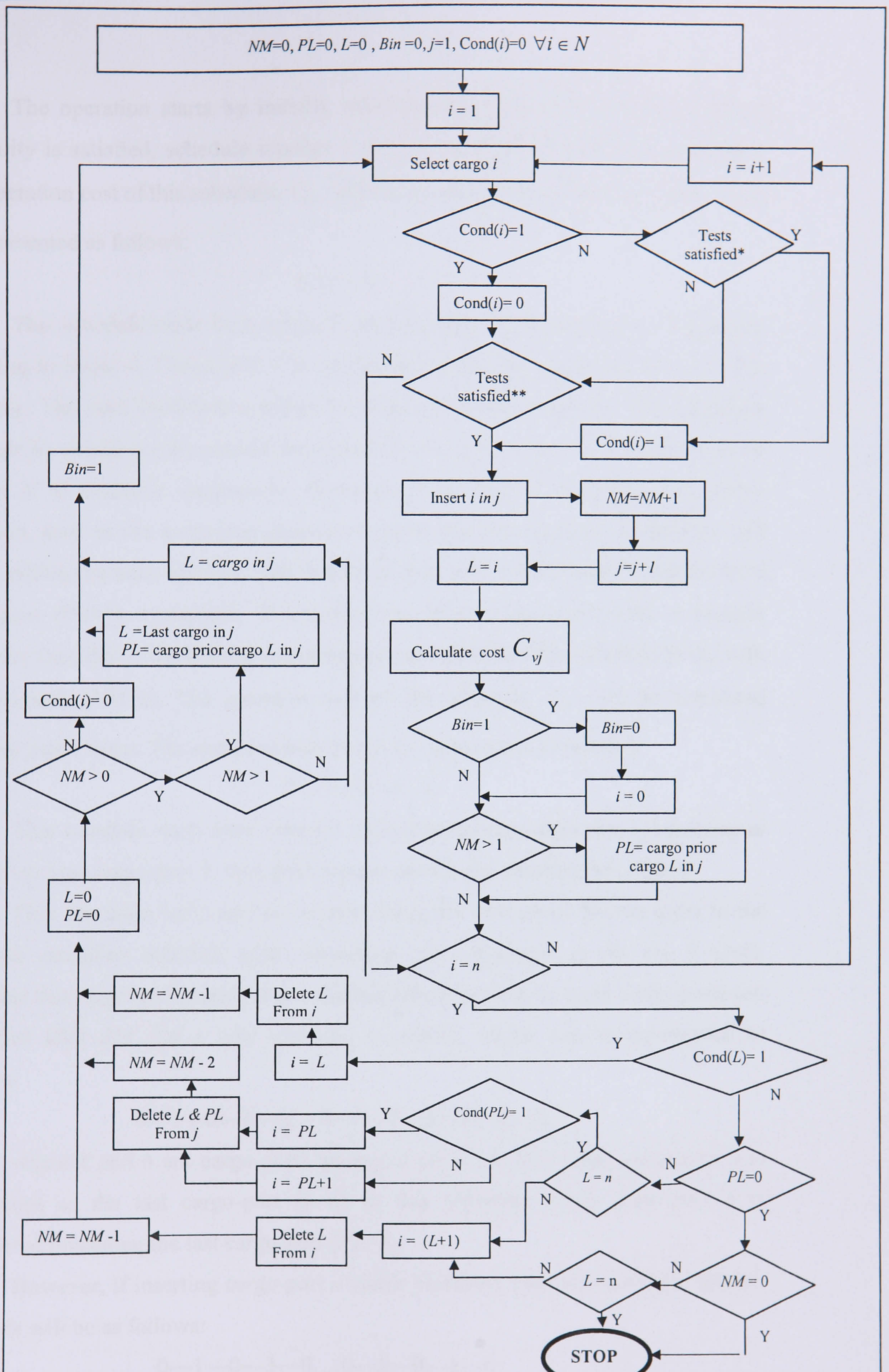
$$e_i \le f_{iv} \le l_i \qquad i \in N, v \in V \tag{4.5}$$

2. To ensure a ship has sufficient capacity to handle a specific cargo-port, it is necessary to add the following test

$$Q_i \leq CT_v \qquad \forall v \in V,\ where\ i \in N \qquad (4.6)$$

These two tests will apply for each generated schedule to ensure that SPP will consider only feasible schedules.

To understand the mechanism of generating feasible schedules, the following flowchart in Figure 4.1 illustrates this operation for only one ship, while the second ship will follow the same procedure.



Tests satisfied*: test of feasibility for conditions (4) & (5)
Tests satisfied**: test of feasibility for conditions (4) & (7)

Figure 4.2: Generating candidate feasible schedules Flowchart for Multi-cargo

The operation starts by initially selecting cargo-port 1 (i=1). If the test of feasibility is satisfied, schedule number 1 (j=1) is established, with one cargo-port. The operation cost of this schedule, C_{vj}, will be calculated after the insert event. It can be represented as follows:

0—1—0

This schedule starts from origin 0 and sailing to deliver cargo-port 1 and then returning to origin 0. Cargo-port 1 is considered as the last cargo-port (L=1) in this schedule. The next candidate schedule is obtained by extending this first candidate schedule by examining the second cargo-port (i=i+1, which means i=2) in the order of the set N of available cargo-ports. However, if the test of feasibility has shown violation, such as the cargo-port quantity exceeds the ship capacity or delivery will not be within the time-window, then this cargo-port will not be inserted and the third cargo-port (i=i+1) examined). If inserting the third cargo-port yields a feasible schedule, then this is considered as the second candidate feasible schedule (j=2), with 2 cargo-ports (NM=2). The operation cost of this schedule, C_{vj}, will be calculated after the insert event. The second schedule can be represented as follows:

0—1—0—3—0

This schedule starts from origin 0, sailing to deliver cargo-port 1, returning to origin 0 to load cargo-port 3, then delivering it and finally returning to origin 0.

This operation will carry on by examining the next cargo-port in order to the previous candidate schedule until examining last cargo-port in the set N (i=n). Consider that cargo-port n generates a feasible schedule, then no more cargo-ports can be added after that and a new schedule is created, which can be represented as follows:

0—1—0—3—0….0—k—0—h—0—n—0,

where k and h are cargo-ports belong to set N. At this stage, cargo-port n is considered as the last cargo-port (L=n) in this schedule, while cargo-port h is considered preceding the last cargo-port (PL=h).

However, if inserting cargo-port n yields violation, then the candidate feasible schedule will be as follows:

0—1—0—3—0….0—k—0—h—0

Cargo-port h is considered as the last cargo-port (L=h) in this schedule and cargo-port k is considered as preceding the last cargo-port (PL=k).

At this stage, the procedure will be performed according to the following question: was cargo-port *n* included in the last schedule? If the answer is YES, then the following procedure will be implemented:

1- $i=h+1$ ($i=PL+1$).

2- Delete cargo-ports *h* and *n* from the last schedule.

3- $NM=NM-2$.

If the answer is NO, then the following procedure will be implemented:

1- $i=h+1$ ($i=L+1$).

2- Delete cargo-port *h* from the last schedule.

3- $NM=NM-1$.

The previous schedule after deletion can be represented as follows:

0—1—0—3—0….0—*k*—0

After this stage, *L* and *PL* will be stated according to their position in this schedule.

The order in which cargo-ports are included in a route is not necessarily the same as they appear in the set *N*. For example, cargo-port 5 can be delivered before cargo-port 3 in the same schedule, if the test of feasibility is satisfied. This issue will be considered in the generation process. In the flowchart of Figure 4.2, the binary indicator *Bin* is set to one after the first generation phase in which candidate schedules contain cargo-ports in cargo-port list order. When the procedure recognises that *Bin* is equal to one, candidate schedules are generated with cargo-ports that are not in list order schedules. Consider the following schedule:

0—1—0—3—0….0—*k*—0—(*h*+1)—0

Where $Bin=1$ the generation process will generate new candidate schedules based on the above schedule. The search starts from cargo-port 1 ($i=i+1$, where $i=0$). The example of a schedule that could be generated is shown below.

0—1—0—3—0….0—*k*—0—(*h*+1)—0— (*h*)—0— (*n*)—0

The operations of select and insert will continue to be applied to generate a set of candidate feasible schedules S_v for specific ship *v* until no more feasible schedules

can be generated. Each ship in the fleet will have a set of candidate feasible schedules generated by using the same procedure.

To simplify this procedure, consider an example with three cargo-ports and one ship. If all constraints are relaxed, then Table 4.2 illustrates all the candidate schedules that can be generated:

No.	Candidate schedules	No.	Candidate schedules	No.	Candidate schedules
1	0—1—0	6	0—2—0	11	0—3—0
2	0—1—0—2—0	7	0—2—0—1—0	12	0—3—0—1—0
3	0—1—0—2—0—3—0	8	0—2—0—1—0—3—0	13	0—3—0—1—0—2—0
4	0—1—0—3—0	9	0—2—0—3—0	14	0—3—0—2—0
5	0—1—0—3—0—2—0	10	0—2—0—3—0—1—0	15	0—3—0—2—0—1—0

Table 4.2: All candidate feasible schedules for Single-cargo problem

After finishing each candidate schedule, the overall cost for each specific schedule is calculated by computing the number of sailing days multiplied by sailing cost SP_v and adding port entrance and operating cost (PC_{kv} and OC_{kv}).

B. Multi-cargoes

The generation of schedules for the multi-cargo problem is implemented in the same way as in the case of the single-cargo problem, by expanding an existing schedule by inserting one cargo-port at a time. Feasibility tests are applied by considering the two parameters, ship capacity and availability time at origin, as defined below:

1. To ensure that each ship arrives within the required time-window, the following test is applied

$$e_i \le f_{iv} \le l_i \qquad i \in N, v \in V \tag{4.7}$$

2. A ship must have sufficient capacity in order to handle the assigned cargo-ports, so it is necessary to add the following for each trip

$$\sum_{\substack{i,k \in N \\ i \neq k}} Q_k \le CT_v \qquad \forall v \in V \tag{4.8}$$

Two tests are applied for each cargo-port under consideration for insertion during the generation of new feasible schedules. If the first test is satisfied for a

specific cargo-port, a new candidate feasible schedule is generated. The first test is defined as follows:

$$f_{iv} + UL_{iv} + d_{ki} \leq l_k \qquad i,k \in N, v \in V \tag{4.9}$$

This test examines the feasibility of inserting cargo-port k on the same trip as cargo-port i, where the route after inserting cargo-port k can be represented as follows:

$$\ldots\!-\!i\!-\!k\!-\!\ldots\ldots$$

If (4.10) is satisfied in addition to (4.9) then the ship will have arrived before the delivery time-window

$$f_{iv} + UL_{iv} + d_{ki} < e_k \qquad i,k \in N, v \in V \tag{4.10}$$

In this case, waiting costs are imposed until the delivery time-window is satisfied. Calculation of waiting time can be computed as follows:

$$wv_{kv} = \max\{0, e_k - (f_{iv} + UL_{iv} + d_{ik})\} \qquad i,k \in N, v \in V$$

The second test is defined by (7).

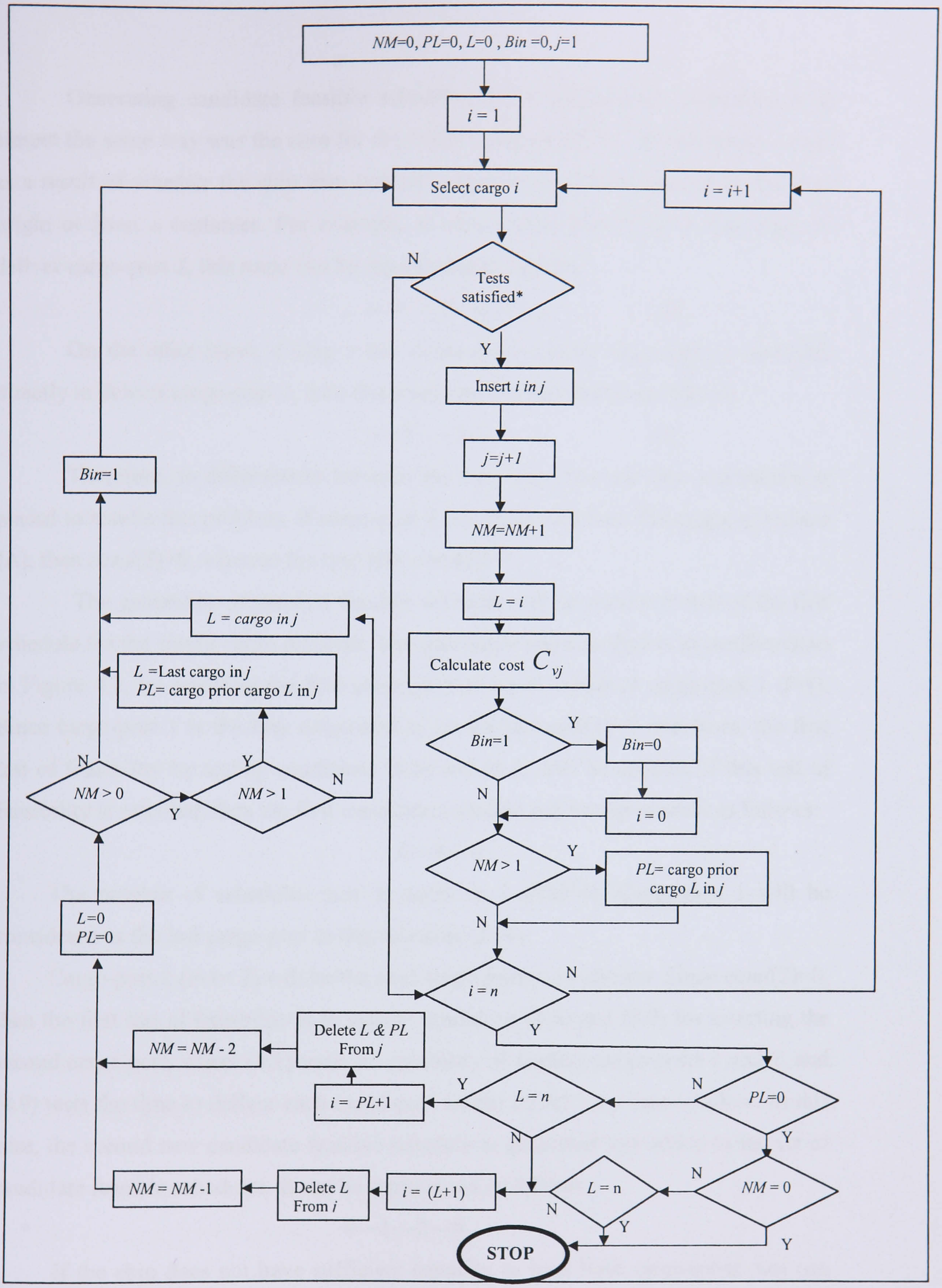
$$f_{iv} + UL_{iv} + d_{i0} + LD_{kv} + d_{0k} <= l_k \qquad i,k \in N, v \in V \tag{4.11}$$

If (4.11) is satisfied, ship v has sufficient time to deliver cargo-port i, return to the origin, load and then deliver cargo-port k before the close of the delivery time-window. The route after inserting cargo-port k can be represented as follows:

$$\ldots\!-\!i\!-\!0\!-\!k\!-\!\ldots\ldots$$

On the other hand, if neither (4.9) nor (4.11) are satisfied, then cargo-port (k+1) will be examined.

The mechanism of generating feasible schedules for a single ship is defined in the flowchart shown in Figure 4.2. This procedure is applied to each ship.



L= last cargo of schedule j

PL= Prior cargo L of schedule j

n: Last cargo of set N

NM: number of cargoes in schedule j

* Test of feasibility.

Comment: j for first ship will be equal 1, while j will continue for the next ships.

Figure 4.1: Generating candidate feasible schedules Flowchart for Single-cargo

Generating candidate feasible schedules for multi-cargo is implemented in almost the same way was the case for the single-cargo problem. The difference arises as a result of whether the ship that delivers a specific cargo-port has come from the origin or from a customer. For example, if ship v sails directly from the origin to deliver cargo-port 2, this route can be represented as follows:

…— 3 — 0 — 2— …. (A)

On the other hand, if ship v has finished delivering cargo-port 3, and sails directly to deliver cargo-port 2, then this route can be represented as follows:

…— 3 — 2— …. (B)

Therefore, to differentiate between the two types (A) and (B), a condition is placed to handle this problem. If cargo-port 2 was delivered from the origin as in case (A), then $cond(2)=0$, whereas for type (B), $cond(2)=1$.

The generation of the first feasible schedule will be similar to that of the first schedule for the single-cargo problem. The procedure starts as shown in the flowchart in Figure 4.2, by selecting the first cargo-port of set N, which is cargo-port 1 ($i=1$). Since cargo-port 1 is the first cargo-port to be tested, $cond(1)=0$. therefore, the first test of feasibility by testing conditions (4.8) and (4.9) will be applied. If this test of feasibility is satisfied, then the first candidate schedule can be represented as follows:

0—1—0

The number of schedules now is equal to 1 ($NM=1$). Cargo-port 1 will be considered as the last cargo-port in this schedule ($L=1$).

Cargo-port 2 ($i+1= 2$) will be the next cargo-port to be chosen. Since $cond(2)=0$, then the first test of feasibility is to satisfy conditions (4.8) and (4.9) for inserting the second cargo-port, where (4.8) tests the capability of loading cargo-ports 1 and 2, and (4.9) tests the time to deliver each cargo-port within its delivery time-window. In this case, the second new candidate feasible schedule is generated and added to the set of candidate feasible schedules. It can be represented as follows:

0—1—2—0

If the ship does not have sufficient capacity to load both cargo-ports, but can load cargo-port 1, if the ship deliver it, returns to the origin and loads cargo-port 2, which is delivered it within the time-window (i.e. it satisfies condition (4.11)), then the second new candidate feasible schedule can be represented as follows:

0—1—0—2—0

For the last two possible schedules, cargo-port 2 will be considered as the last cargo-port in the current schedule (L=2), while cargo-port 1 is considered to be the preceding cargo-port of the last cargo-port (PL=1). Meanwhile, $NM = 2 =$ (NM=NM+1).

The next cargo-port is cargo-port 3 (i+1 = 2) to be chosen, and the same procedure is applied by examining the first test of feasibility by applying conditions (4.8) and (4.9). In this case, the new candidate feasible schedule is generated and added to the set of candidate feasible schedules. It can be represented as follows:

0—1—2—3—0

If the first test of feasibility is not satisfied, then the second test of feasibility by applying conditions (4.8) and (4.11) will take place. In this case, the new candidate feasible schedule is generated and added to the set of candidate feasible schedules. It can be represented as follows:

0—1—2—0—3—0

Now, for the previous two possible schedules above, cargo-port 3 will be considered as the last cargo-port in the current schedule (L=3), while cargo-port 2 is considered to be the preceding cargo-port of the last cargo-port (PL=2). Meanwhile, $NM = 3 =$ (NM=NM+1).

Meanwhile, if condition (4.10) as well as condition (4.7) are satisfied, then waiting cost (wv_{iv}) will be added to the operating cost for this specific schedule.

This operation will continue by examining the next cargo-port in the set N and testing to see whether the cargo-port is to be included in a new candidate schedule. This process is repeated until the last cargo-port in the set N (i=n) is considered. Consider that cargo-port n satisfied the first two conditions (4.8) and (4.9), then the new candidate schedule can be represented as follows:

0—1—2—0—3—….—k—0—h—n—0,

On the other hand, if the first test of feasibility is not satisfied, then the second test of feasibility will be examined by applying conditions (4.8) and (4.11). The new candidate schedule can be represented as follows:

0—1—2—0—3—….—k—0—h—0—n—0,

Where k and h are cargo-ports belong to the set N. At this stage, for the last two possible schedules above, cargo-port n is considered as the last cargo-port (L=n),

while cargo-port h is considered to be the preceding cargo-port of the last cargo-port ($PL=h$).

However, if inserting cargo-port n yields violation, then no new candidate feasible schedule will be added, where the current feasible schedule will be as follows:

0—1—2—0—3—….—k—0—h—0

Cargo-port h is considered as the last cargo-port ($L=h$) in this schedule and cargo-port k is considered to be the preceding cargo-port of the last cargo-port ($PL=k$).

At this stage, the procedure will be performed according to the following question:

Is *cond*(L)= 1 ? —— (1)

If the answer is YES, then the following procedure will be implemented:

1- $i=L$

2- Delete cargo-port L from the last schedule.

3- $NM=NM$-1

Figure 4.3 illustrates an example of this case.

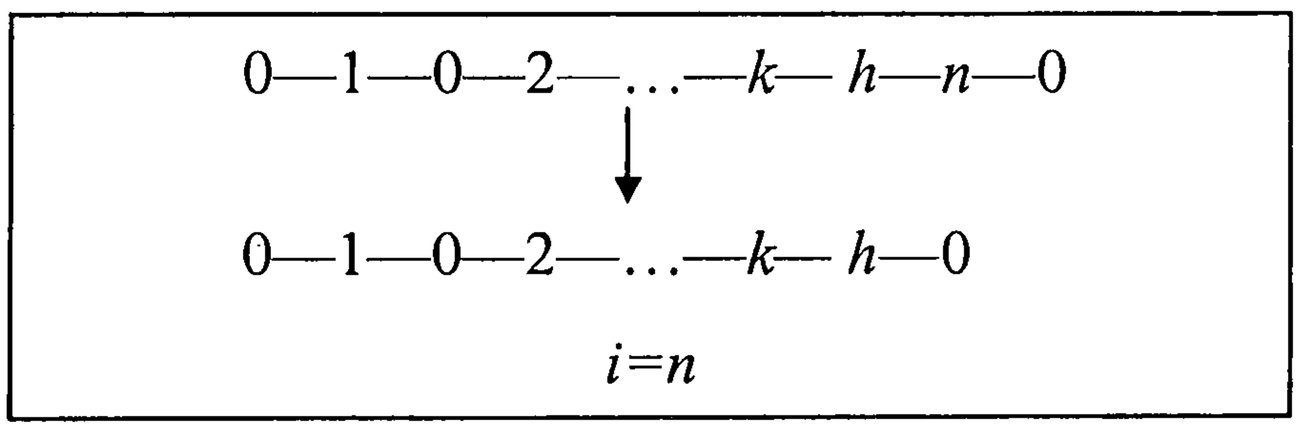


Figure 4.3: If the answer is YES for question (1)

In Figure 4.3, cargo-port n is deleted from the current schedule, at the same time, cargo-port n will be chosen ($i = n$) to examine whether it is to be included in a new candidate schedule.

Meanwhile, if the answer is NO for question (1), then the following question will be asked:

Is PL= 0 ? — (2)

(is there at most one cargo-port in the current schedule). If the answer is NO, then the following question will be asked:

Is L= n ? —— (3)

If the answer is YES, then the following question will be asked:

Is *cond*(PL)= 1? —— (4)

If the answer is YES for question (4), then the following procedure will be implemented:

1- $i=PL$

2- Delete cargo-ports PL and L from the last schedule.

3- $NM=NM-2$.

Figure 4.4 illustrates an example of this case.

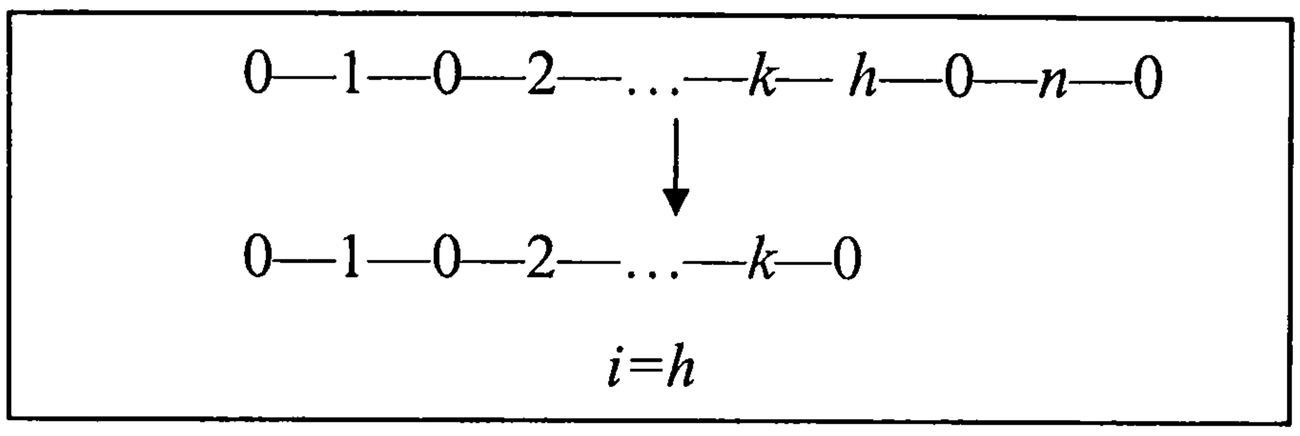


Figure 4.4: If the answer is YES for question (4)

In Figure 4.4, cargo-ports h and n will be deleted from the current schedule and cargo-port h will be chosen ($i = h$) to examine whether it is to be included in a new candidate schedule.

Meanwhile, if the answer is NO for question (4), then the following procedure will be implemented:

1- $i=PL+1$

2- Delete cargo-ports PL and L from the last schedule.

3- $NM=NM-2$.

Figure 4.5 illustrates an example of this case.

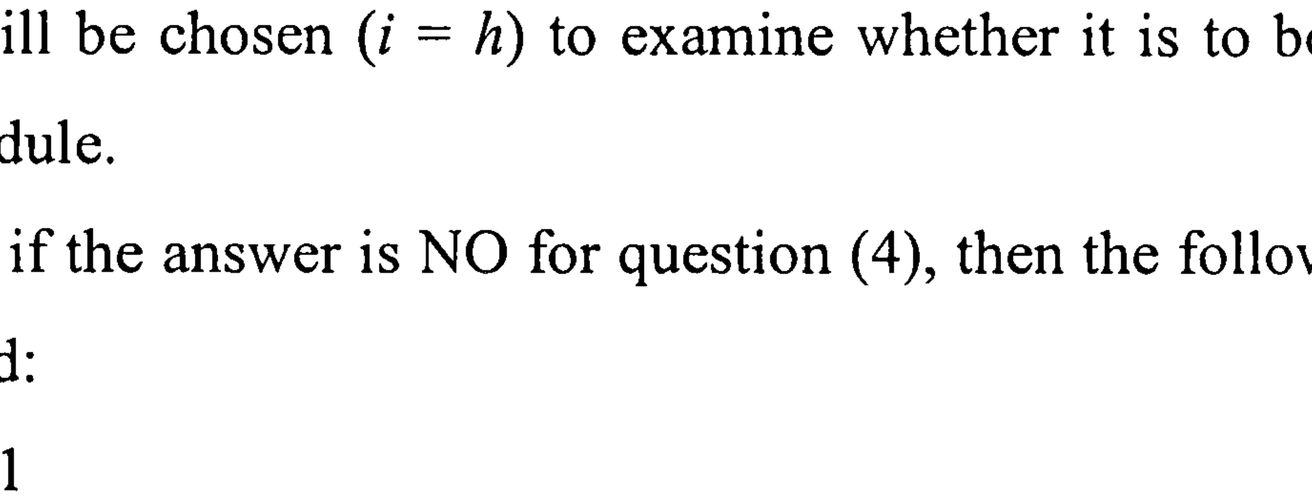


Figure 4.5: If the answer is NO for question (4)

In Figure 4.5, cargo-ports h and n are deleted from the current schedule and cargo-port (h+1) will be chosen ($i = h$+1) to examine whether it is to be included in a new candidate schedule.

On the other hand, if the answer is No for question (3), then the following procedure will be implemented:

1- $i=L+1$

2- Delete cargo-port L from the last schedule.

3- $NM=NM-1$.

Figure 4.6 illustrates an example of this case.

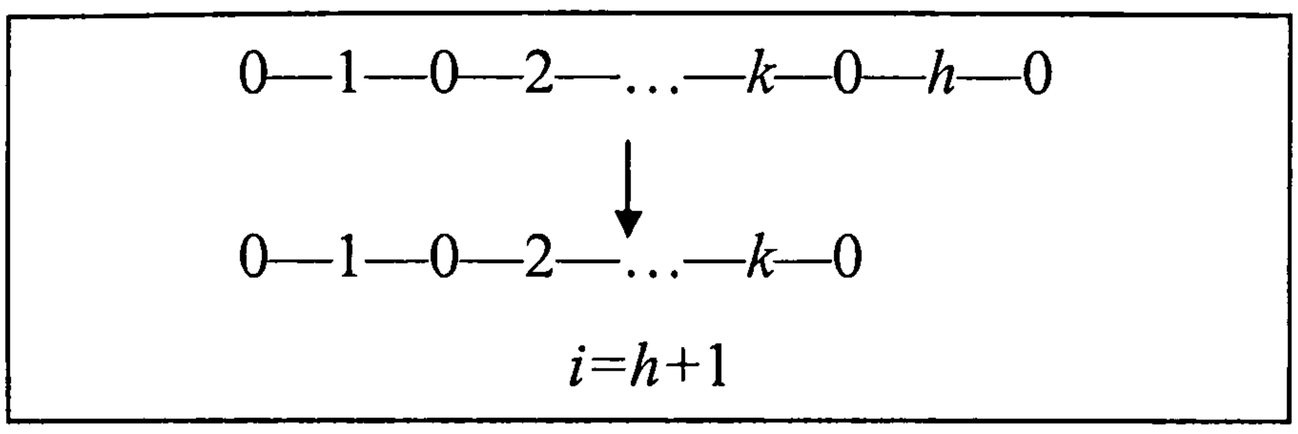


Figure 4.6: If the answer is NO for question (3)

In Figure 4.6, cargo-port h is deleted from the current schedule and cargo-port (h+1) will be chosen ($i = h+1$) to examine whether it is to be included in a new candidate schedule.

If the answer is YES for question (2), then the following question will be asked:

Is $NM= 0$? —— (5)

If the answer is YES, then *STOP*. Otherwise, the following question will be asked:

Is $L= n$? —— (6)

If the answer is NO, then following procedure will be implemented:

1- $i=L+1$

2- Delete cargo-port L from the last schedule.

3- $NM=NM-1$.

Otherwise, the generating operation will *STOP*.

To simplify this procedure, consider an example with three cargo-ports and one ship. If all constraints are relaxed, then Table 4.3 illustrates all the candidate schedules that can be generated:

No.	Candidate schedules	No.	Candidate schedules	No.	Candidate schedules
1	0—1—0	14	0—2—0	27	0—3—0
2	0—1—2—0	15	0—2—1—0	28	0—3—1—0
3	0—1—2—3—0	16	0—2—1—3—0	29	0—3—1—2—0
4	0—1—2—0—3—0	17	0—2—1—0—3—0	30	0—3—1—0—2—0
5	0—1—0—2—0	18	0—2—0—1—0	31	0—3—0—1—0
6	0—1—0—2—3—0	19	0—2—0—1—3—0	32	0—3—0—1—2—0
7	0—1—0—2—0—3—0	20	0—2—0—1—0—3—0	33	0—3—0—1—0—2—0
8	0—1—3—0	21	0—2—3—0	34	0—3—2—0
9	0—1—3—2—0	22	0—2—3—1—0	35	0—3—2—1—0
10	0—1—3—0—2—0	23	0—2—3—0—1—0	36	0—3—2—0—1—0
11	0—1—0—3—0	24	0—2—0—3—0	37	0—3—0—2—0
12	0—1—0—3—2—0	25	0—2—0—3—1—0	38	0—3—0—2—1—0
13	0—1—0—3—0—2—0	26	0—2—0—3—0—1—0	39	0—3—0—2—0—1—0

Table 4.3: All candidate feasible schedules for Multi-cargo problem

For each candidate schedule, there is overall cost C_{vj}. The overall cost C_{vj} is calculated by computing the number of sailing days multiplied by sailing cost SP_v adding port entrance and operating cost (PC_{kv} and OC_{kv}) and considering the waiting time cost ($wv_{iv} \times WP_v$) for each cargo-port.

4.1.5 Heuristic Approach

In the previous section, the exact approach using SPP was presented. Since the exact approach is capable of solving small size problems, but may have difficulty with large problems, the need for an approach to solve large-scale problems is essential. The following section presents a full description of the Tabu Search (TS) method, which is capable of solving large scale instances of SRSP. However, since this is a heuristic method, optimality is not guaranteed in all cases

To solve this problem, a list of all cargo-ports is arranged in sequence according to their earliest delivery time.

An initial solution can be obtained by using a Greedy Algorithm. The Greedy Algorithm is implemented by first, selecting the ship with the least overall cost ship. This method selects controlled ships first. A route is then created for this ship starting

from cargo-port number one in the list, provided all constraints such as capacity and delivery time window are satisfied. The process can be presented as follows:

1. The insertion of cargo-port i in a route for ship v, does not violate the delivery time window if $e_i \leq f_{iv} \leq l_i \quad , i \in N, v \in V$
2. The insertion of cargo-port i in a route for ship v, does not violate the capacity of ship v if $Q_i \leq CT_v \quad \forall v \in V$.

If cargo-port one has satisfied these constraints, then the cargo-port will be added to this ship. The same operation applies on the second cargo-port such that, if the constraints are satisfied, the cargo-port will be added to the route; otherwise, cargo-port two will be left for another ship, while the procedure continues to evaluate cargo-port three for this specific ship. This operation of testing all cargo-portes in the list continues for this specific ship until cargo-port n. At that stage, a schedule for this ship is obtained. The next step is to select the ship with the next lowest cost and considering the unallocated cargo-ports in the list. This procedure will carry on until no cargo-port remains unallocated.

Greedy Algorithm.

(1) **Until** number of $N = \phi$

(2) set $N = \{0,1,..,n\}$

(3) choose v, where $v \in V$, and $v \notin Hold$

(4) Select cargo-port i , where $i \notin served$,

(5) **If** v satisfies all constraints **then**

Cargo-port i served by ship v, $i \in served$

$n = n - 1$

If $i \neq n$ **then**

$i = i + 1$

go to (5)

Else, $v \in$ hold

go to (3)

(6) **End Until**

The result can be illustrated in Table 4.4:

Cargo-port	1	2	3	4	.....	*n*-3	*n*-2	*n*-1	*n*
Ship	3	1	5	*k*	.....	1	4	3	*k*

Table 4.4: Example of schedule of n cargo-ports

In Table 4.4, cargo-port 1 is served by ship 3, cargo-port 2 is served by ship 1, and so on until last cargo-port *n* is served by ship *k*.

The initial schedule consists of a set of routes. A route is represented, for example, as [0,3,…,*k*,0], which defines the sequence in which the cargo-ports are visited. A route for a ship *v* is denoted by R_v. A schedule consists of all routes together with the servicing time of each cargo-port in the route, and denoted as *S*.

In the multi-cargo case, each route is made up of some trips. A trip is a partition of the route, starting and finishing at the origin. Figure 4.7 illustrates the concept of a trip. Since in single-cargo case, the ship delivers only one cargo-port and returns to the origin, there is no need to use trips.

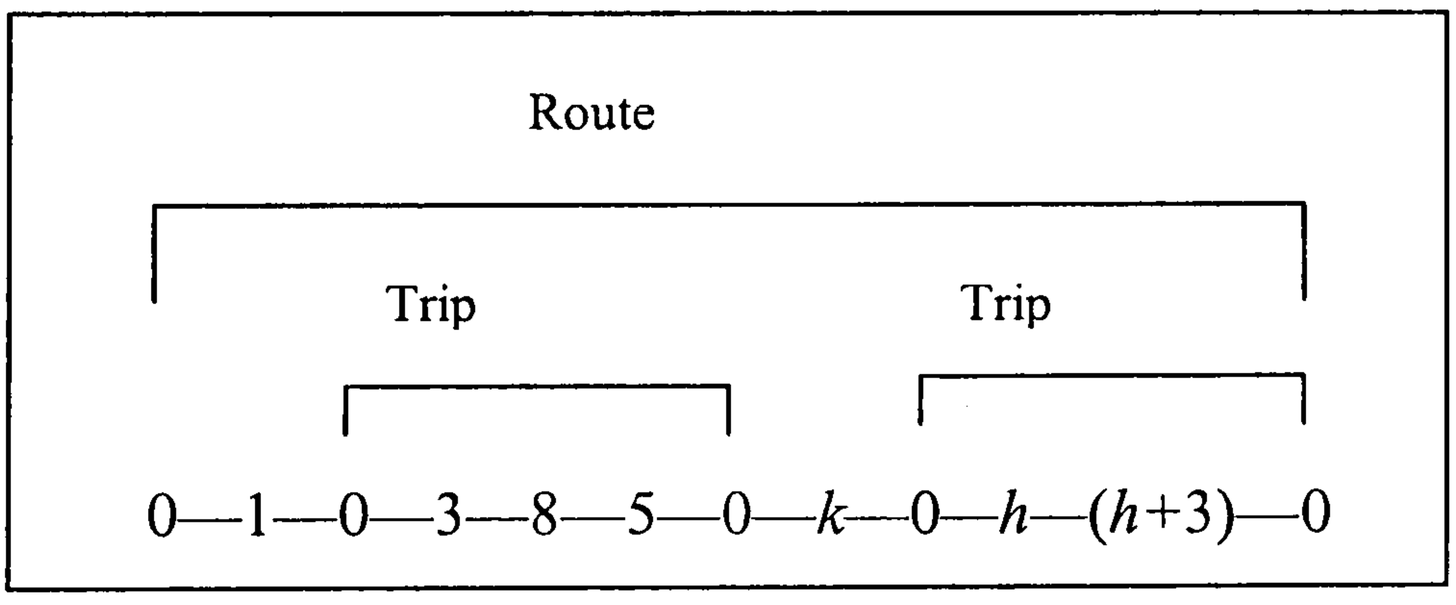


Figure 4.7: Presentation of trip in Multi-cargo case.

After forming the initial schedule, it is possible that one or more ships are not used in this schedule. These ships are referred as idle ships. These idle ships will be available for use during TS operation.

The following section explains the methodologies for solving this type of problem by using the tabu search method, and describes separately the characteristics of each component of the method.

4.1.5.1 Tabu Search

The Ship Routing and Scheduling Problem (SRSP) is solved using the Tabu Search method, proposed and developed by Glover [47]. The most important

characteristic of tabu search is the use of memory for the solution to solve difficult problems. There are many techniques used in Tabu Search such as attributes, tabu list, tabu list size (tenure), intensification, diversification, neighbourhood, and neighbourhood size, move and evaluation of the move, all which are adapted in this approach. Parameters defining the neighbourhood size and the tabu list size are considered critical in terms of solution quality and computing time. The most important point in Tabu Search is the need for experimentation to choose the best parameters and their values for each specific type of problem.

The following notations are used in the tabu search approach.

Notations:

TN	Tenure (Tabu list size)
MinTn	Minimum tenure size
MaxTn	Maximum tenure size
$InsTn_i$	cargo-port i entered insert tabu list, where $i \in N$
$SwTn_i$	cargo-port i entered swap tabu list, where $i \in N$
IncTn	number of iterations to increase tenure by one
Nbr_i	neighbourhood set of cargo-port i, where $i \in N$
NZ	neighbourhood size
$NbrSel_i$	cargo-port i is selected to form it's neighbourhood, where $i \in N$
$NbItr_i$	number of iterations within the neighbourhood of cargo-port i, where $i \in N$
AllItr	number of iterations for the entire problem
Rpt_i	number of times cargo-port i entered tabu list, where $i \in N$
$Divert_i$	cargo-port i will be hold for a number of time of forming new neighbourhood, not to be chosen, where $i \in N$

Since there are two classes of SRSP, the following section is devoted to the single-cargo problem. The multi-cargo problem will be discussed in section 4.2.3.

A. Single-cargo

It is important to explain some characteristics of tabu search. Insert and swap moves are the only moves used in tabu search. First, the insert move can be described

as deleting cargo-port i from route j and inserting it in route j', where the routes belong to different ships. For example, suppose there are five cargo-ports to be visited by two ships.

After applying the Greedy algorithm, the routes for each ship may be illustrated as shown in Figure 4.8.

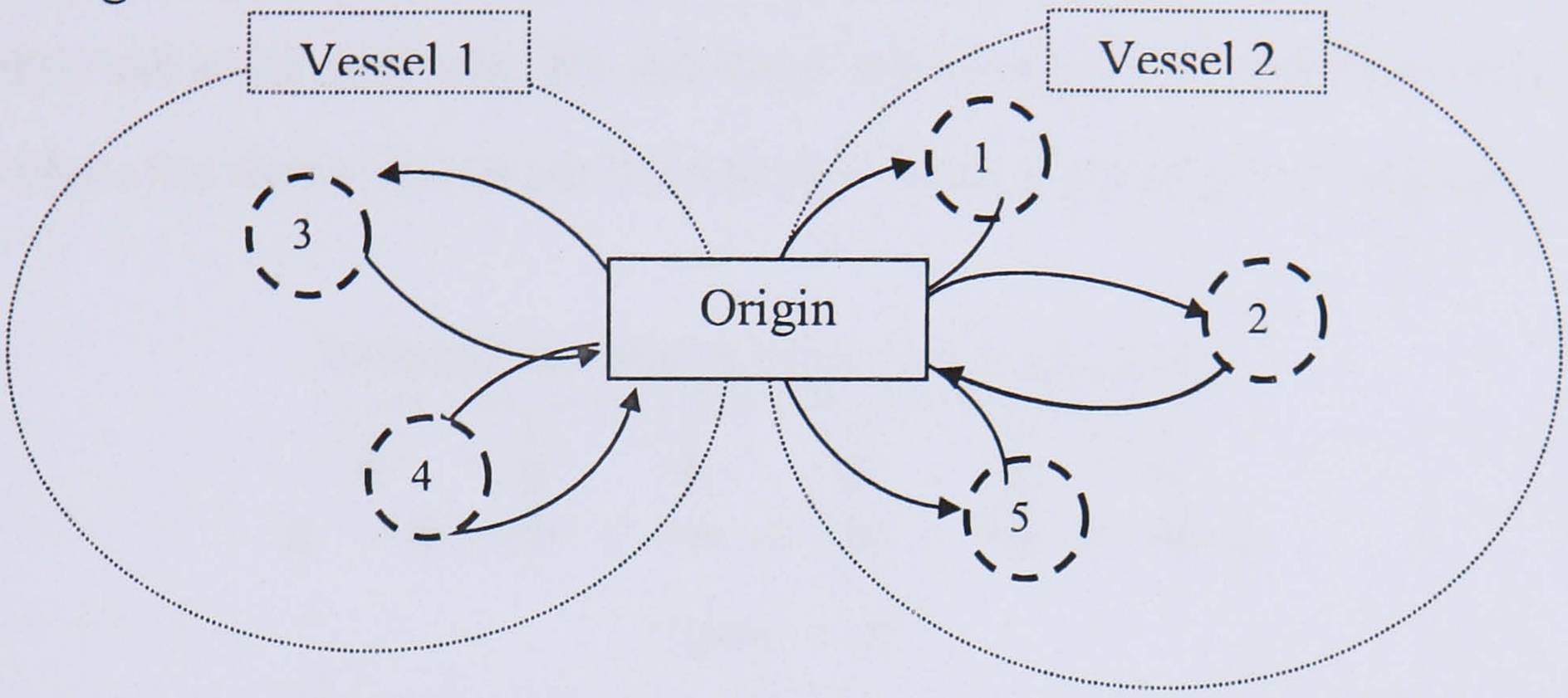


Figure 4.8: Two ships delivered five cargo-ports

The route of ship 1 is: 0 → 3 → 0 → 4 → 0

The route of ship 2 is: 0 → 1 → 0 → 2 → 0 → 5 → 0

The route for ship 1 states, that the ship is loaded at the origin to serve cargo-port 3 then returns to the origin to load cargo-port 4, which is delivered, and then returns to the origin.

Applying the insert move such that, deleting cargo-port 2 from ship 2 and inserting in ship 1 is shown in Figure 4.9.

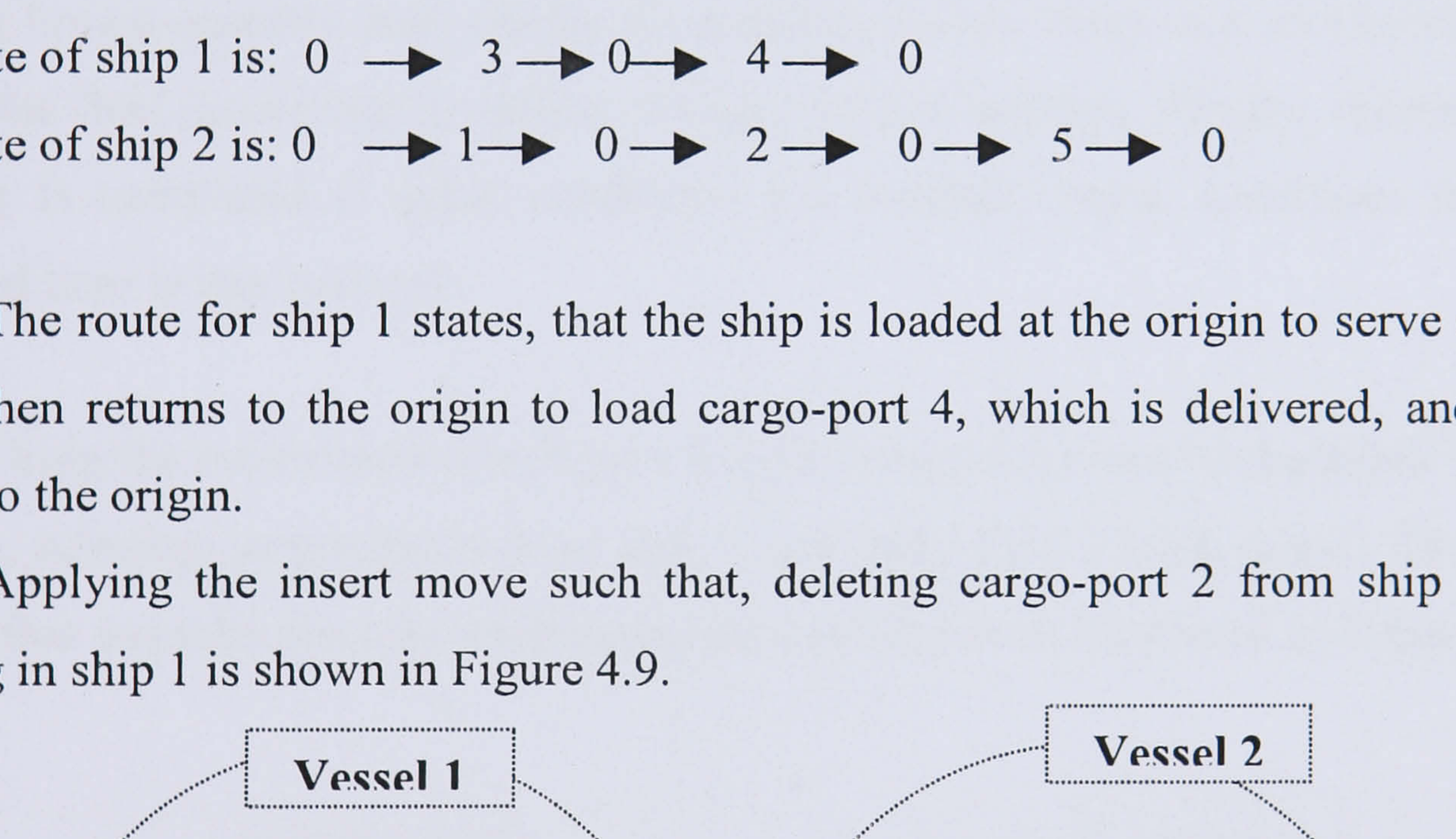


Figure 4.9: Deleting cargo-port 2 from ship 2 and inserting in ship 1

Now, after insertion, the routes for each ship will be as follows:

The route of ship 1 is: 0 → 3 → 0 → 2 → 0 → 4 → 0

The route of ship 2 is: 0 → 1 → 0 → 5 → 0

The insertion process considers many potential places for insertion in the candidate route. The first potential insertion trial place is immediately before the first cargo-port in the specified route, the second place is immediately before the second cargo-port, and so on until after the last cargo-port. Figure 4.10 illustrates route R_v of ship v, where the above arrows are the potential insert trial places for cargo-port i.

Figure 4.10

In the approach adopted here, the insert move starts by selecting the cargo-port with the lowest quantity size among all neighbourhoods. Next step, the lowest cost ship in the fleet (according to sailing cost per day) is selected. Finally, deleting and inserting is completed if some conditions are satisfied (these conditions will be explained later in this section).

Using the routes shown in Figure 4.8, the swap move can be described by, for example, selecting cargo-port 4 from ship 1 and cargo-port 1 from ship 2. The swap move action can take place by exchanging their positions as illustrated in Figure 4.11.

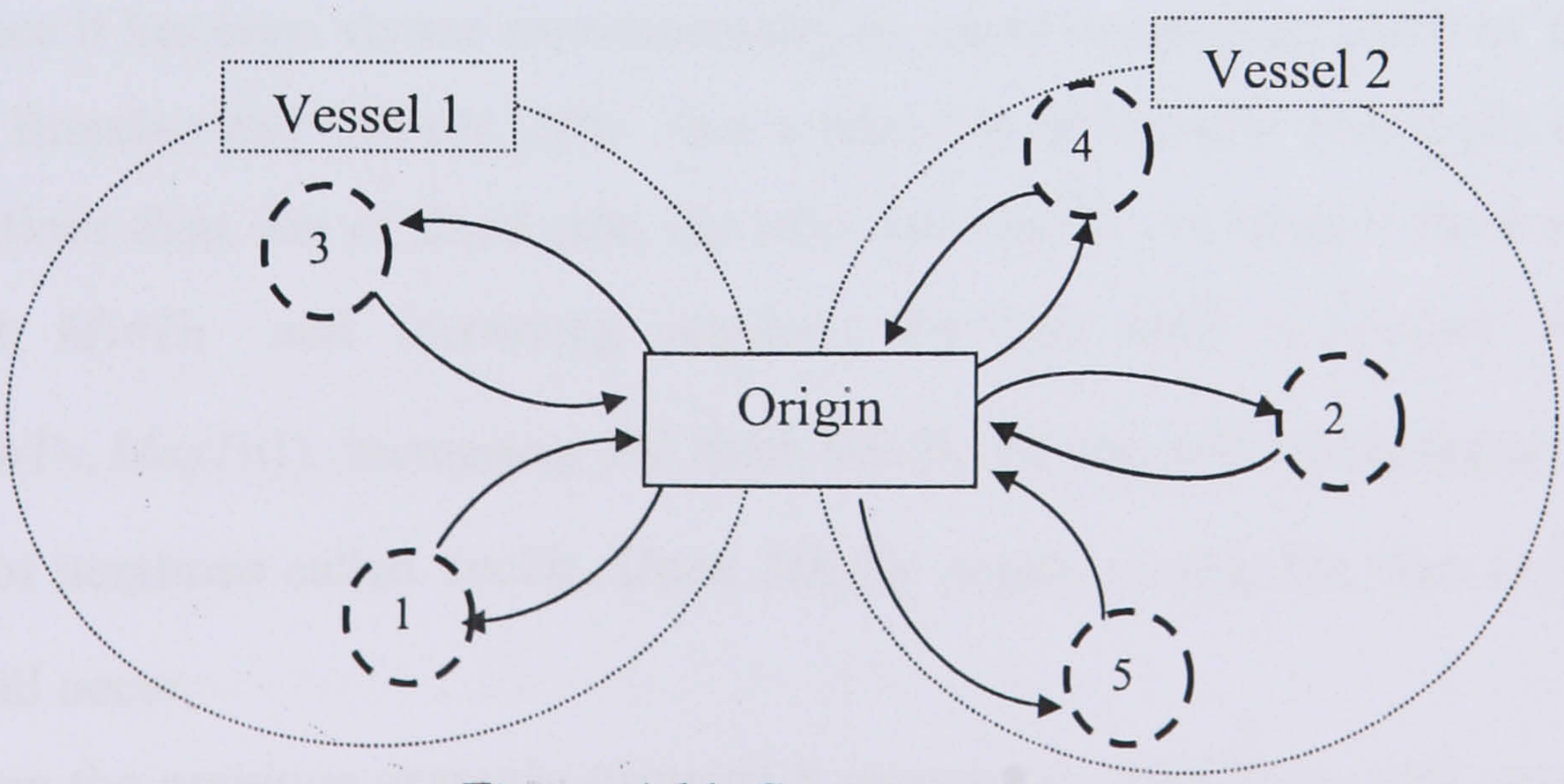
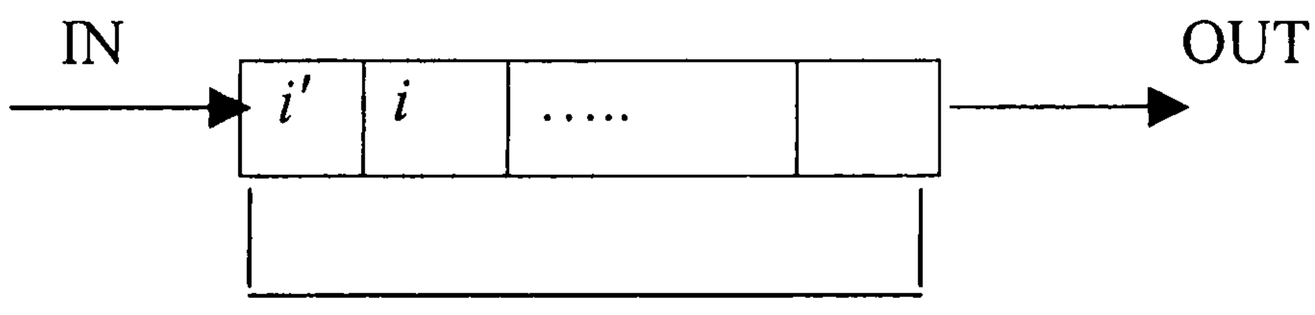


Figure 4.11: Swap move

A swap move starts by selecting the two cargo-ports of the lowest quantity sizes among all neighbourhoods. The next step is to exchange their positions, where each cargo-port will transfer to the other cargo-port's route (ship). Finally, a swap move is completed if some conditions are satisfied (these conditions will be explained in this section).

Each cargo-port that has recently encountered moves (either insert move or swap move) will enter the tabu list. The tabu list helps the search to move from a previously visited section of the search space and to execute a more extensive exploration. A tabu list usually consists of a list of moves the search has recently encountered. The cargo-ports on the tabu list cannot be revisited for a particular number of iterations called tabu tenure TN (in this model there are two tabu lists, one for insert move events $InsTn_i$ and the other for swap move events $SwTn_i$). Figure 4.12 illustrates the Tabu list and the Tabu tenure. After each iteration, the longest serving cargo-port in the tabu list will leave the list and the remaining cargo-ports in the list will move one place down the list, where for each iteration tabu elements will move one step toward dropping out of the tabu list.



TN iterations

Figure 4.12

Since it has been shown experimentally by many researchers (such as Taillard [108] and Brandao and Mercer [20]) that a tabu list of variable size tends to give better solutions than one of fixed size, the tabu list size TN, is taken to be a number starting at $MinTn$ and increasing regularly by one until it reaches $MaxTn$ ($TN \in [MinTn, MaxTn]$). Increasing TN from $MinTn_i$ by one will be implemented for a number of iterations called $IncTn$. Once $MinTn$ equals to $MaxTn$, then no further increase will occur.

Since the previous example contains 5 cargo-ports (which is quite small), all cargo-ports are neighbours for each other. For large-scale problems, the search among all cargo-ports in the schedule would require excessive computer time. Avoiding this difficulty can be achieved by selecting a neighbourhood and searching will be within

it. The neighbourhood structure or neighbourhood list Nbr_i for variable i (cargo-port i in the schedule) can be defined by different approaches. In this thesis two approaches are adopted, one by a systematic approach and the other random. The systematic approach is implemented by the following steps:

1. Select cargo-port i, which has the lowest quantity size among n cargo-ports and call it $NbrSel_i$.

2. Form the neighbourhood cargo-ports of cargo-port i (Nbr_i), by selecting the cargo-port, whose time-window starting time is closest to the cargo-port i time-window starting time, then select the next closest and so on until the neighbourhood has been determined. Represent these neighbours by Nbr_i.

3. The magnitude of Nbr_i is denoted by NZ, the number of cargo-port neighbours for cargo-port i, and must be defined.

Table 4.5 illustrates Nbr_i for cargo-port i.

Cargo-port	1	2	…	$i-b$	….	i	…	$i+b$	….	N
Ship	3	1	…		…..	k	…		…	j

Range Nbr_i for cargo-port i

Table 4.5

The second approach is to select cargo-ports to form a neighbourhood in a random way with the neighbourhood size fixed at NZ. Maximum tenure $MaxTn$ must not exceed NZ ($MaxTn \leq NZ$).

The neighbourhood size NZ is considered as a critical point in the moves operation. If NZ is too small it will restrict the search and a good solution is less likely to be found. On the other hand, if NZ is too large, it loses its purpose in diminishing the neighbourhood size. A good trade-off can be obtained by experimentation.

Execute moves of both types within Nbr_i will be carried out with some conditions. A trial insert by deleting the current ship and inserting another ship happens only if:

1. Selecting a cargo-port is within the neighbourhood (Nbr_i) of cargo-port i.
2. The insertion of cargo-port i in a route of ship v', does not cause the violation of other cargo-port's delivery time window. $e_{i'} \le f_{i'v'} \le l_{i'} \quad , i' \in N, v' \in V$
3. The insertion of cargo-port i in route (or idle) of ship v', does not violate the capacity of ship v' ($Q_i \le CT_{v'} \quad \forall v' \in V$).

A trial swap of two cargo-ports takes place if and only if:

1. Both selected cargo-ports exist within Nbr_i.
2. Both selected cargo-ports have different routes (ships).
3. The capacity for each of the two ships is sufficient to load the other cargo-port.
4. Each ship does not violate the other cargo-ports delivery time-window.

Execute moves of both types within Nbr_i will be carried out for a specified number, $NbItr_i$ of iterations, where insert is the first move followed by swap move. The search within Nbr_i stops after a given number of iterations, $NbItr_i$, has been executed without improvement. If searching in this specific area has been completed, the next step is to select another Nbr_i' and execute moves of both types for a number of iterations. Searching is based on the best solution found to date. The iteration limit for the entire problem is denoted as $AllItr$.

B. Multi-cargo

The elements used in the single-cargo problem are also used in the multi-cargo problem, while some differences on applying trail moves are encountered. First, by deleting cargo-port i from route R_v and inserting it in route $R_{v'}$, a new trip on route $R_{v'}$ is created. To simplify, consider cargo-port i, where $i \in R_v$. To insert cargo-port i

between cargo-port t and k, where $t,k \in R_{v'}$ and cargo-port t will be serviced before cargo-port k, one of the following four sets of conditions must be satisfied, otherwise, insert trail will not carry on:

1- State 1: with the following conditions

$$f_{tv} + UL_t + d_{t0} + d_{i0} + LD_i \leq l_i \tag{4.12}$$

$$f_{iv} + UL_i + d_{i0} + d_{k0} \leq l_k \tag{4.13}$$

Then the new route $R_{v'}$ can be represented as follows:

$$\cdots \longrightarrow t \longrightarrow 0 \longrightarrow i \longrightarrow 0 \longrightarrow k \longrightarrow \cdots.$$

2- State 2: if (4.13) is satisfied and the following test is also verified

$$LD_t + LD_i + d_{ti} \leq l_t \tag{4.14}$$

$$f_{tv} + UL_t + d_{ti} \leq l_i \tag{4.15}$$

Then the new route $R_{v'}$ can be represented as follows:

$$\cdots \longrightarrow t \longrightarrow i \longrightarrow 0 \longrightarrow k \longrightarrow \cdots.$$

3- State 3: if (4.12) is satisfied and the following test is also satisfied

$$LD_i + LD_k + d_{0i} \leq l_i \tag{4.16}$$

$$f_{iv} + UL_i + d_{ik} \leq l_k \tag{4.17}$$

Then the new route $R_{v'}$ can be represented as follows:

$$\cdots \longrightarrow t \longrightarrow 0 \longrightarrow i \longrightarrow k \longrightarrow \cdots.$$

4- State 4: if (4.15) and (4.17) are satisfied, additional to the following test:

$$LD_t + LD_i + LD_k + d_{ti} \leq l_t \tag{4.18}$$

Then the new route $R_{v'}$ can be represented as follows:

$$\cdots \longrightarrow t \longrightarrow i \longrightarrow k \longrightarrow \cdots.$$

These four states are encountered for both insert and swap trials. Meanwhile, the same tests of feasibility for the single-cargo problem, mentioned in the previous section must be satisfied.

After each move, the overall cost for both new routes must be calculated and added to the total cost of the schedule, replacing the costs of the previous two routes.

C. Intensification and Diversification

To explore the current neighbourhood Nbr_i further, *intensification* search is adopted. Two methods of intensification search are used in this approach. With the first method, if the search yields an improved solution, the iteration count will be set to zero. This means, that this area seems to be attractive. With the second method, if cargo-port i has repeatedly entered the tabu list a given number of times, the cargo-port will remain in tabu list for a number of iterations equal to $MaxTn$ (cargo-port i has entered the tabu list for a number of times denoted by Rpt_i). To explore a large search region and to prevent the method from being trapped at a local optimum, a *diversification* technique is used within this stage. Diversification search is a technique that tries to alleviate this problem by forcing the search to unexplored areas of the search space. This technique can be used only for the problem solved by using the systematic method of forming neighbourhood Nbr_i. Having searched the neighbourhood of cargo-port i, the process moves to another neighbourhood. This is achieved by ensuring that cargo-port i is not selected again for a number of time (called $Divert_i$) of forming new neighbourhood $Nbr_{i'}$. This method gives the chance to explore different neighbourhoods $Nbr_{i'}$, by selecting cargo-port i', and so on for the whole $AllItr$.

Tabu Search Algorithm

1) Initial S and the value f of S obtained from Greedy algorithm
2) $S^* = S, \quad f^* = f$
3) $All = 0$
4) Until $AllItr = NbItr_i * All$
5) $Divert_i = 0 \quad \forall i \in N$
6) $Dvt = D$
7) $NZ = M$
8) $IncTn = IncreaseByOne$
9) $InsTn_i = 0 \quad \forall i \in N$
10) $SwTn_i = 0 \quad \forall i \in N$
11) count = 0

12) $TN = MinTn$

13) **(Neighbourhood procedure)**
- $All = All + 1$
- If $Divert_i > 0 \quad \forall i \in N \quad then$
  - $Divert_i = Divert_i - 1$
- Select cargo-port i, where $i \in N$ and $Divert_i = 0$
- Form Nbr_i by size = NZ
- $Divert_i = Dvt$
- $NbItr_i = LocalIter$

14) If count = $IncTn$ *then*
- $TN = TN + 1$

15) (**Insert procedure**) select cargo-port k, where $k \in Nbr_i$, and $InsTn_k = 0$

16) Select route $R_{v'}$, where $R_{v'} \notin ChosenVessel$
- If inserting cargo-port k into ship $R_{v'}$ satisfies constraints then
  - Insert continues
  - $InsTn_k = TN$
  - If $f < f^*$ then
    - $f^* = f$
    - count = 0
- Else
  - If all $i \in Nbr_i$ are examined then
    - Keep the same route R_v
    - $InsTn_k = TN$
  - Else
    - $R_{v'} \in ChosenVessel$
    - Go to 16

17) **(Tenure procedure)**
- $\forall i \in Nbr_i$ If $InsTn_i > 0$ then
  - $InsTn_i = InsTn_i - 1$

18) (**Swap procedure**) select cargo-port h, $h \in R_v$ and $h \in Nbr_i$, and $SwTn_h = 0$

19) Select cargo-port g, where $g \in Nbr_i$ and $h \in R_{v'}$, where $R_{v'} \notin ChosenSwap$
- If exchange positions satisfy constraints then
  - Exchange continues
  - $SwTn_h = TN$
  - count = count +1
  - if $f < f^*$ then
    - $f^* = f$
    - count = 0

- Else
  - If all $v \in V$ are examined then
    - Keep the same route R_v
    - $SwTn_h = TN$
    - count = count +1
  - Else
    - $R_{v'} \in ChosenSwap$
    - Go to 19

20) **(Tenure procedure)**

- $\forall i \in Nbr_i$ If $SwTn_i > 0$ then
  - $SwTn_i = SwTn_i - 1$

21) **(Form a new Neighbourhood)**

- If count = $NbItr_i$ then
  - Go to 9
- Else
  - Go to 14

22) End until

The steps of tabu search are as described above, where the first schedule is obtained by applying the Greedy Algorithm. Steps 2 to 12 specify all the data involved in this operation. Step 13 forms a new neighbourhood every time the search finishes the search for any particular neighbourhood after $NbItr_i$ iterations. Step 14 is designed to increase *MinTn* when reaching a number of iterations equal to *IncTn*, during the search in every particular neighbourhood. Steps 15 to 17 represent the insert move, while steps 18 to 20 represent the swap move. Step 21 represents the two options of tabu search operation, where the search will be implemented on a new neighbourhood, if the iteration has reached the limit, otherwise, the search will continue for the same neighbourhood. Finally, step 22 stops the search if the number of iterations is equal to *AllItr* .

Chapter 5: Design and implementation of solution approaches

5.1 Introduction

This chapter provides details of the design and implementation of the two approaches for solving SRSP. The first approach is an exact method based on the Set Partitioning Problem SPP. The implementation language Visual Basic 6 [3] was used to generate all candidate feasible schedules. The MPL [2] software was used to create the model which was solved using the commercial optimisation solver CPLEX-10 [34]. The second approach is based on the heuristic method Tabu Search. The implementation language was Visual Basic 6 and all results are displayed using an Excel spreadsheet.

5.2 Exact Approach Design

5.2.1 Overall Structure

The structure for this approach consists of several stages, as shown in Figure 5.1. These stages are, in most respects, the same for both single-cargo and multi-cargo problems.

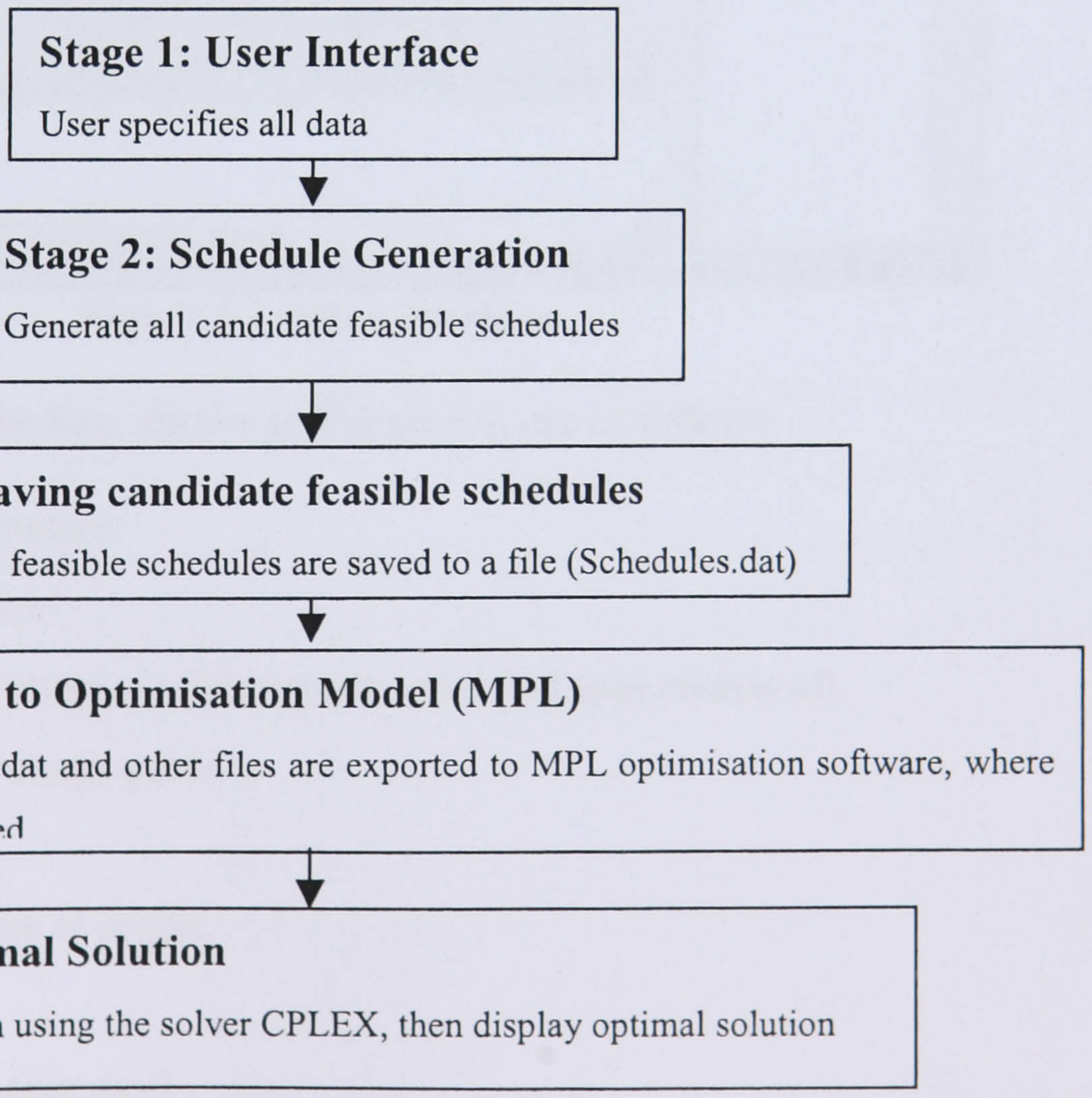


Figure 5.1: Structure of Exact Method

Stage 1: The user specifies the data that defines a particular problem instance of SRSP. This information can be entered using an interface called *feasible schedules generation*. This interface is designed using Visual Basic 6 (VB6 and run on a 2.00 GHZ Intel ® Pentium under Windows 2000. The interface design consists of several components as illustrated in Figure 5.2.

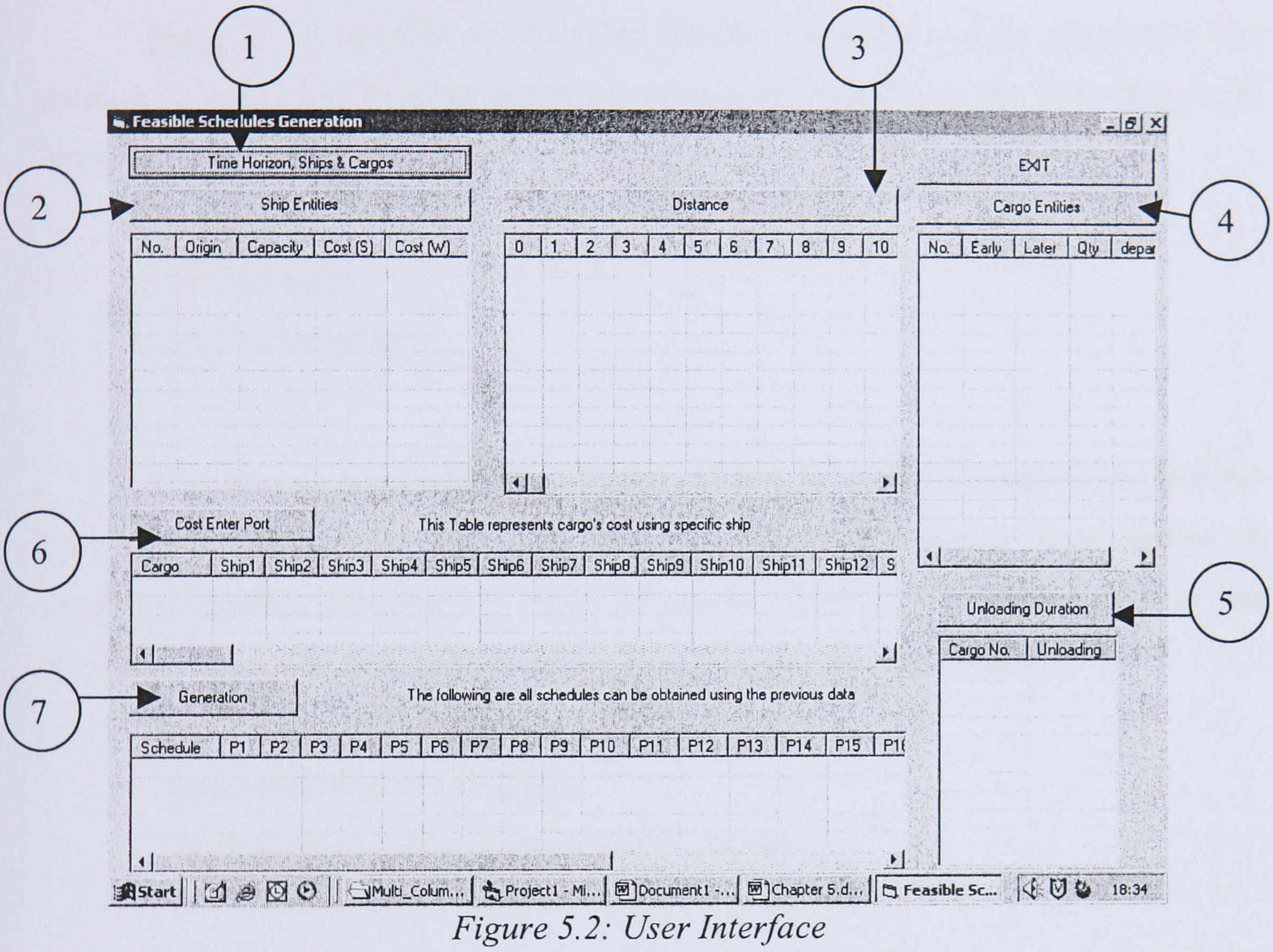


Figure 5.2: User Interface

The components of the interface, shown in Figure 5.2, are as follows:

1. Problem size parameters:
   - Time horizon
   - Number of ships available (controlled and spot chartered).
   - Number of cargo-ports.
2. Ship features:
   - Availability time at origin.
   - Ship capacity
   - Cost of sailing (per day)
   - Cost of waiting –idle- (per day)
3. Cargo ports distances

4. Cargo-port features
    - Time-window (start and end of delivery time-window)
    - Quantity (ton)
5. Loading and unloading time required for each cargo-port
6. Port entrance fee for each ship.

Stage 2: A number of candidate feasible schedules will be generated. The number of candidate feasible schedules generated depends on the following three factors:

1. Time horizon.
2. Number of ships.
3. Number of cargo-ports.

It is obvious that if there are a large number of ships and cargo-ports, and the time horizon duration is large, then the possibility of obtaining a large number of candidate feasible schedules is high. Moreover, three further factors control the number of candidate feasible schedules. These three factors are as follows:

1. The width of delivery time-window
2. Distance between cargo ports.
3. Cargo-port quantity in relation to ship capacity.

For the first factor, if the width of delivery time window is large for all or most cargo-ports, then this will enable ships to visit more customers. Moreover, with the second factor, if the distance between cargo ports is small, then a ship can deliver many cargo-ports. Finally, if ship capacities are large and the cargo-port quantities are small, then a ship can deliver many cargo-ports in the same trip before returning to the origin.

A number of candidate feasible schedules will be displayed as a list view on the interface once the user clicks on the "generation" icon (item 7 in Figure 5.3). The first column in the list view represents schedule number and ship number, the other columns except for the last one represent ship route, where numbers represent cargo-ports. A route could contain "0" between two cargo-ports, which means the ship will return to the origin port after unloading a cargo-port before delivering then its next

cargo-port. Since, in the single-cargo case, the ship automatically returns to the origin port after every delivery, there is no need to mention the origin in the list view. The last column represents the route's cost. Some of generated schedules are illustrated in Figure 5.3. The user can browse through the list view to review the generated schedules. For example, schedule number 40 in Figure 5.3 is delivered by ship1. The ship starts from the origin (represented by 0) to load cargo-port 1, then sails to deliver cargo-port 1, after that the ship will return to the origin to load cargo-ports number 4 and 5. The operation cost of this route is £165870.

Generation — The following are all schedules can be obtained using the previous data

Schedule	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
36 (1)	0	1	0	3	0	4		252880				
37 (1)	0	1	0	3	5		186915					
38 (1)	0	1	0	3	0	5		207045				
39 (1)	0	1	4	0	5		220770					
40 (1)	0	1	0	4	5		165870					
41 (1)	0	1	0	4	0	5		219940				
42 (1)	0	1	0	5	4		165870					

Figure 5.3: Generation of candidate feasible schedule

Stage 3: These candidate feasible schedules will be saved on an external file called *Schedule.dat*, and the operation cost for each schedule will be saved on another external file called *ScheduleCost.dat*. The representation for the schedule at this stage will not be the same as presented at stage 2. Each schedule will consist of schedule number and ship route. Each route contains an identifier for each cargo-port, where a value of "1" indicate that the cargo-port is included in this schedule, and the value of "0" indicates that it is not carried by this schedule. For example, schedule 40 in Figure 5.3 will be presented as (1 0 0 1 1), where cargo-port 1 delivered then cargo-ports 4 and 5, and cargo-ports 2 and 3 will not. Additional to the previous files, there are three more external files representing number of ships, number of cargo-ports, and the set of schedules for each ship. These three files called *ShipNumber.dat*, *CargoNumber.dat*, and *ScheduleNumber.dat* respectively. There is a total of five files, representing all data required for the optimisation. In chapter 4, the SPP formulation was defined using several parameters, where m represents the total number of ships, n represents the total number of cargo-ports, S_v denotes the set of candidate schedules for ship v, C_{vj} denotes the operation cost for schedule j using ship v, and finally, SC_{ivj}

equals one if schedule j for ship v services cargo-port i, and equals zero otherwise. This information will be presented as follows:

Total number of ships, m, will be saved at file *ShipNumber.dat.*

Total number of cargo-ports, n, will be saved at file *CargoNumber.dat*

Total number of candidate feasible schedules generated S_v, for all ship v, will be saved at file *ScheduleNumber.dat*

All costs, for each schedule, C_{vj}, will be saved at file *ScheduleCost.dat*

SC_{ivj} Set of candidate feasible schedules, for ship v, will be saved at file *Schedule.dat*

Stage 4: The five files mentioned previously will be input to the optimisation model. The SPP formulation will be created using software called MPL (Mathematical Programming Language). MPL is an advanced modelling system, which allows the user to formulate optimisation models in a clear and efficient way. Models developed in MPL can then be solved using an efficient commercial solver such as CPLEX-10. The MPL model file is divided into two major parts: the definition (model dimensions, data, variables) and the model (objective function and constraints). The definition part consists of index (dimension of the problem), data, and decision variables. The model part contains the algebraic model formulation. The model part consists of the objective function, the constraints, bounds (upper and lower bounds), integer variables, and binary variables. The definition part for solving SRSP using MPL is illustrated in Figure 5.4.

```
DATA
   MaxShip  := DATAFILE("ShipNumber.dat");
   MaxSched := DATAFILE("ScheduleNumber.dat");
   MaxCargo := DATAFILE("CargoNumber.dat");

INDEX
   ship=1..MaxShip ;
   schedule = 1..MaxSched ;
   cargo = 1..MaxCargo ;

DATA
   Sched[ship,schedule,cargo]:=        SPARSEFILE("Schedule.dat");
   cost[ship ,schedule]:=              SPARSEFILE("ScheduleCost.dat");
DECISION
  x[ship ,schedule]
```

Figure 5.4: Model formulation using MPL

MPL will import the five files directly to the model, as illustrated in Figure 5.4. The first data represent the dimension of the problem, number of ships, number of candidate schedules, and number of cargo-ports. The second data consists of a set of candidate schedules and the cost for each candidate schedule. The decision section of Figure 5.4, identifies the decision variables $x(v, j)$. $x(v, j)$ will equal one if schedule j for ship v is selected, and zero otherwise.

One of the most important features of MPL is its ability to handle very large sparse index and data sets. This feature allows the model formulator to identify only those cargo-ports that will be delivered by each specific ship. It can be presented as follows:

[Ship ID, Schedule number, Cargo number, Delivered index]

To clarify this part, consider the previous example (Figure 5.3), schedule number 36 delivering cargo-ports 1, 3, and 4. They will be listed as follows:

[1,36,1,1,]

[1,36,3,1]

[1,36,4,1]

The first number represents ship number, the second represents schedule number, the third represent cargo-port number, where the delivered index equal 1, which means this cargo-port is included in this schedule. This feature allows the model formulator to export only delivered cargo-ports for each specific ship, instead of having to examine all cargo-ports, which would be considerably slower.

Stage 5: The model produced using MPL is passed to the solver CPLEX-10. The solution results will be automatically retrieved from the solver and displayed. The result contains the objective function, followed by the selected schedules.

5.3 Approximate Approach

The approximate approach for solving SRSP was coded using Visual Basic 6 (VB6) and run using 2.00 GHZ Intel ® Pentium under window 2000. The structure of the design is shown in Figure 5.5.

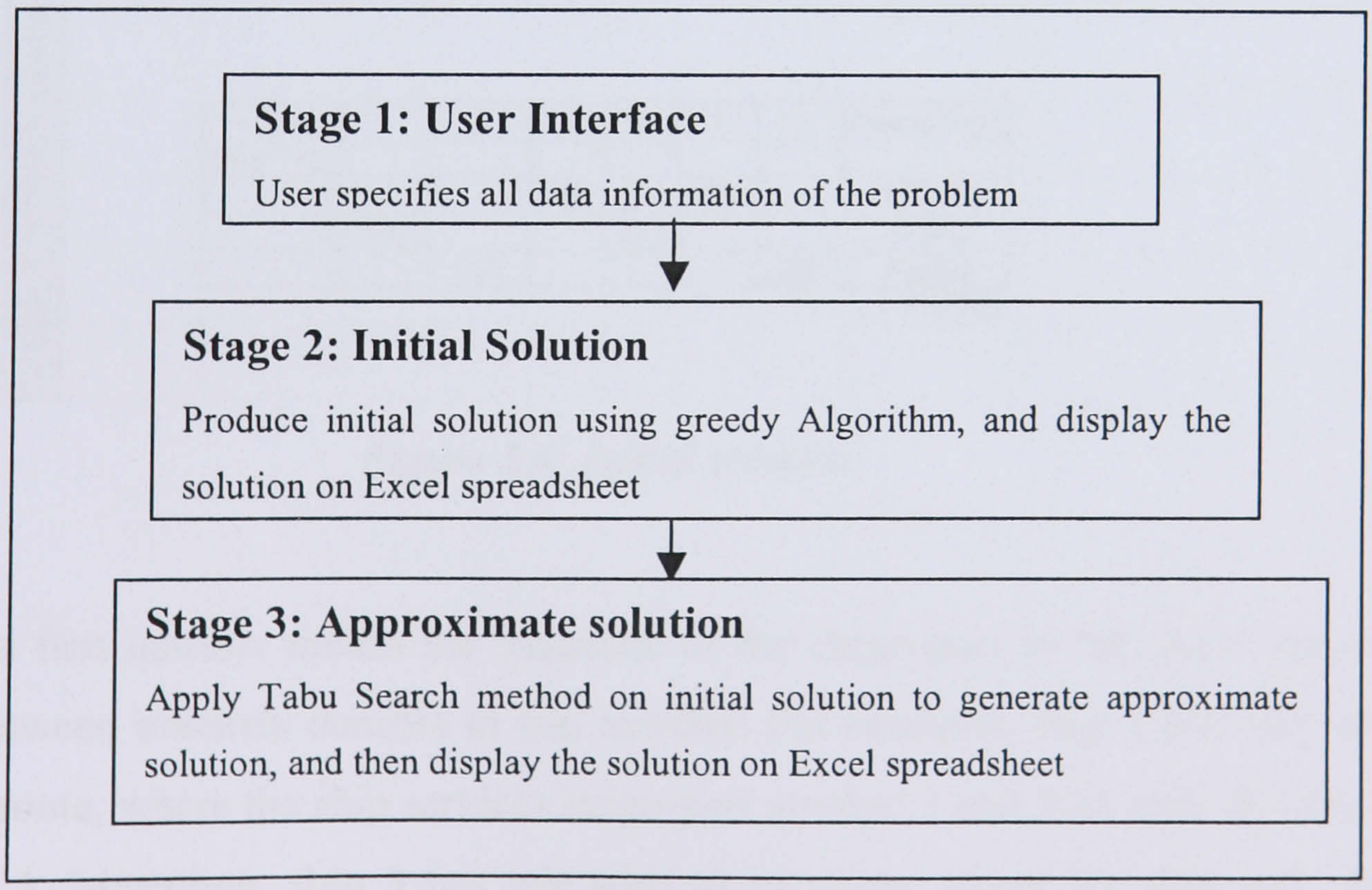
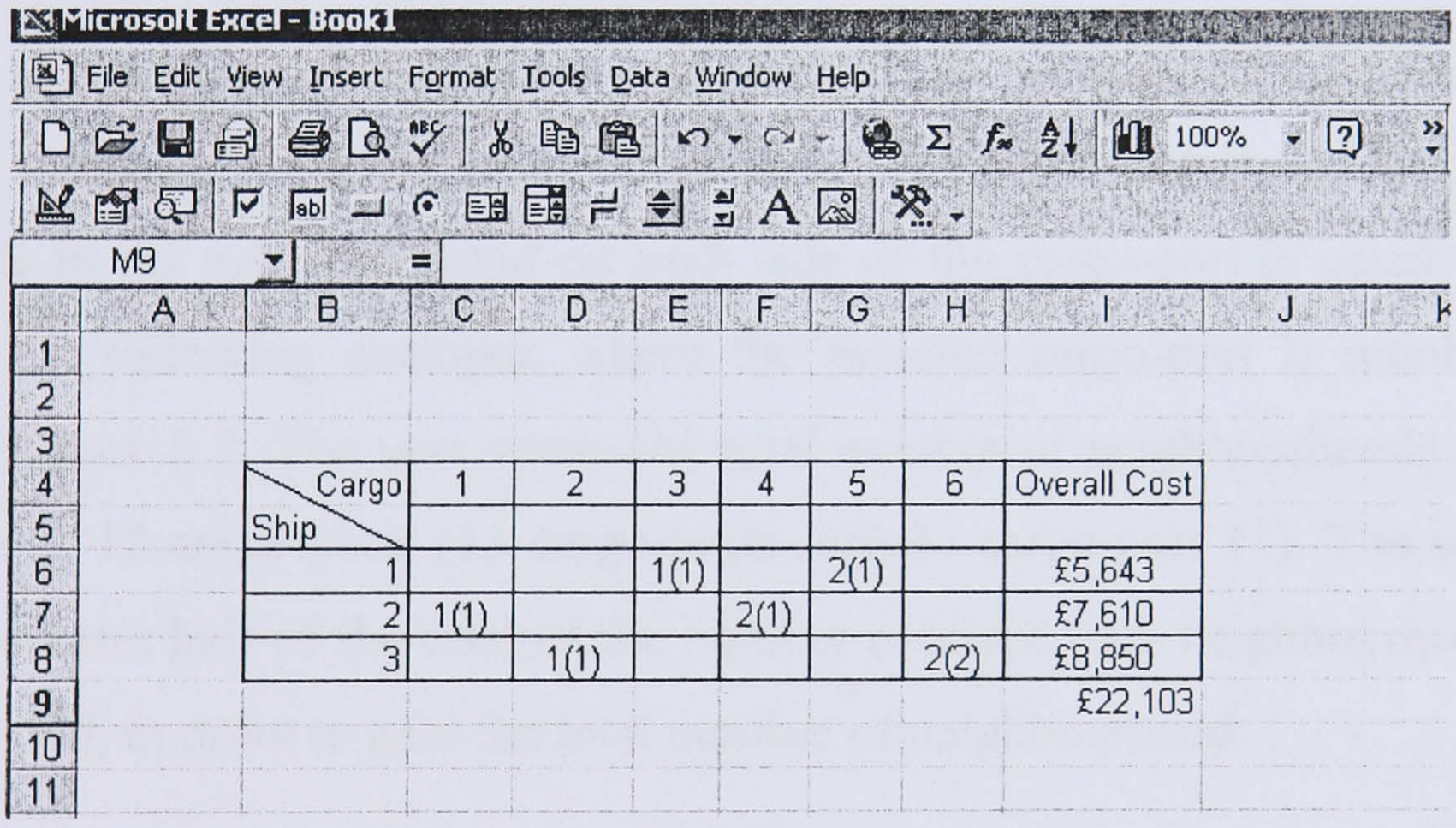


Figure 5.5: Structure of Approximate Method

Stage 1: The user specifies the requirement

Stage 2: Once all information has been entered, the user can obtain an initial solution by pressing the initial solution button. The initial solution will be obtained by applying a Greedy Algorithm as described in chapter 4. The program calls the first ship in the fleet (the cheapest day sailing expenses as determined by fuel consumption), then the process starts adding cargo-ports to the ship in order. If the cargo-port satisfies the constraints (mentioned in chapter 4) then it will assign the cargo-port to the ship. Otherwise, rejection occurs and the next cargo-port is examined. During the search operation for an initial solution, if there are insufficient ships to deliver all available cargo-ports, the user will be asked to add more ship(s). Additional ship(s) required can be obtained from the market (spot chartered). Once this stage has finished, the initial solution will be displayed on an Excel software spreadsheet. Figure 5.6 illustrates the multi-cargo model for three ships involving the delivery of six cargo-ports, where the results are displayed by an Excel software spreadsheet.



Ship \ Cargo	1	2	3	4	5	6	Overall Cost
1			1(1)		2(1)		£5,643
2	1(1)			2(1)			£7,610
3		1(1)				2(2)	£8,850
							£22,103

Figure 5.6: Initial solution

The first number means the sequence of the cargo-port in the route, and the number between brackets denotes to trip number. For example, ship 1 has only one trip in the route, where the ship services cargo-port number 3 and then sails directly to cargo-port 5. Meantime, ship 3 has two trips in its route, where the ship will visit cargo-port number 2 and then return to the origin to load cargo-port number 6. In the case of the single-cargo problem, where the ship services only one cargo-port each

time, there is no need for brackets. The cost for each route is presented in the last column, while the overall cost is displayed in the lowest cell of the column.

Stage 3: Tabu search will be applied in this stage to generate an approximate solution based on the initial solution obtained by the first stage (Greedy Algorithm). The user should enter the following six inputs:

1- Neighbourhood size (*NZ)*
2- Minimum tenure value ($MinTn$)
3- Intensification value, (Rpt_i)
4- Number of iterations needed to increase minimum tenure by one ($IncTn$)
5- Diversification value ($Divert_i$)
6- Number of iterations within the neighbourhood of cargo-port i, ($NbItr_i$)
7- Total number of iterations ($AllItr$)

The user enters *neighbourhood size* (*NZ*) which specifies the number of cargo-ports that forms the neighbourhood. *NZ* is always an even number and the number of cargo-ports in the neighbourhood on each side of the cargo-port is equal to *(NZ/2)*. Consider the following example, where the selected cargo-port is number 11, as shown in Figure 5.7. The user wants the total number of neighbourhoods for cargo-port 11 to be 12 cargo-ports (13 cargo-ports include cargo-port 11). The user in this case has to enter half of the total of the number required (one neighbourhood side of cargo-port 11), in order to gain the total number of neighbourhood.

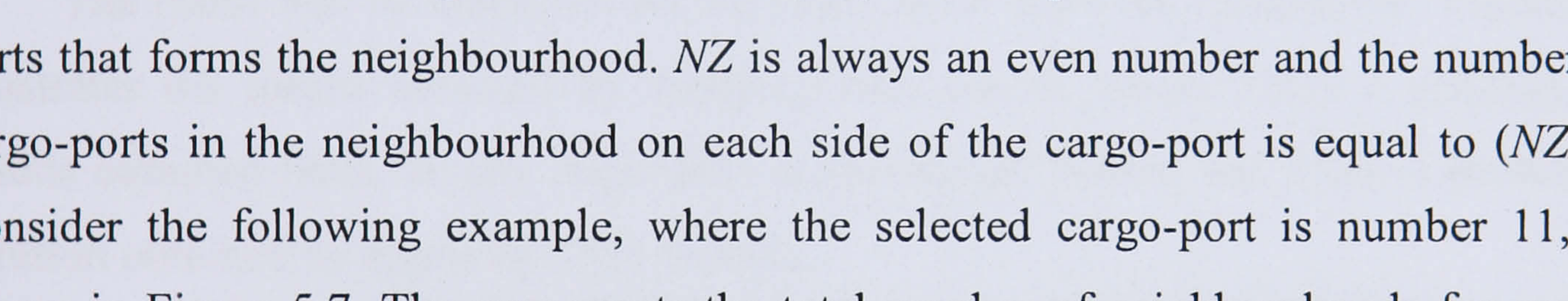


Figure 5.7: The neighbourhood of cargo-port 11

This means that the designed model will multiply 6 neighbourhoods by 2 (2 sides for cargo-port 11), which is equal to 12 neighbourhoods.

The user also enters minimum tenure value ($MinTn$). The maximum tenure value ($MaxTn$) is equal to the number of neighbourhoods.

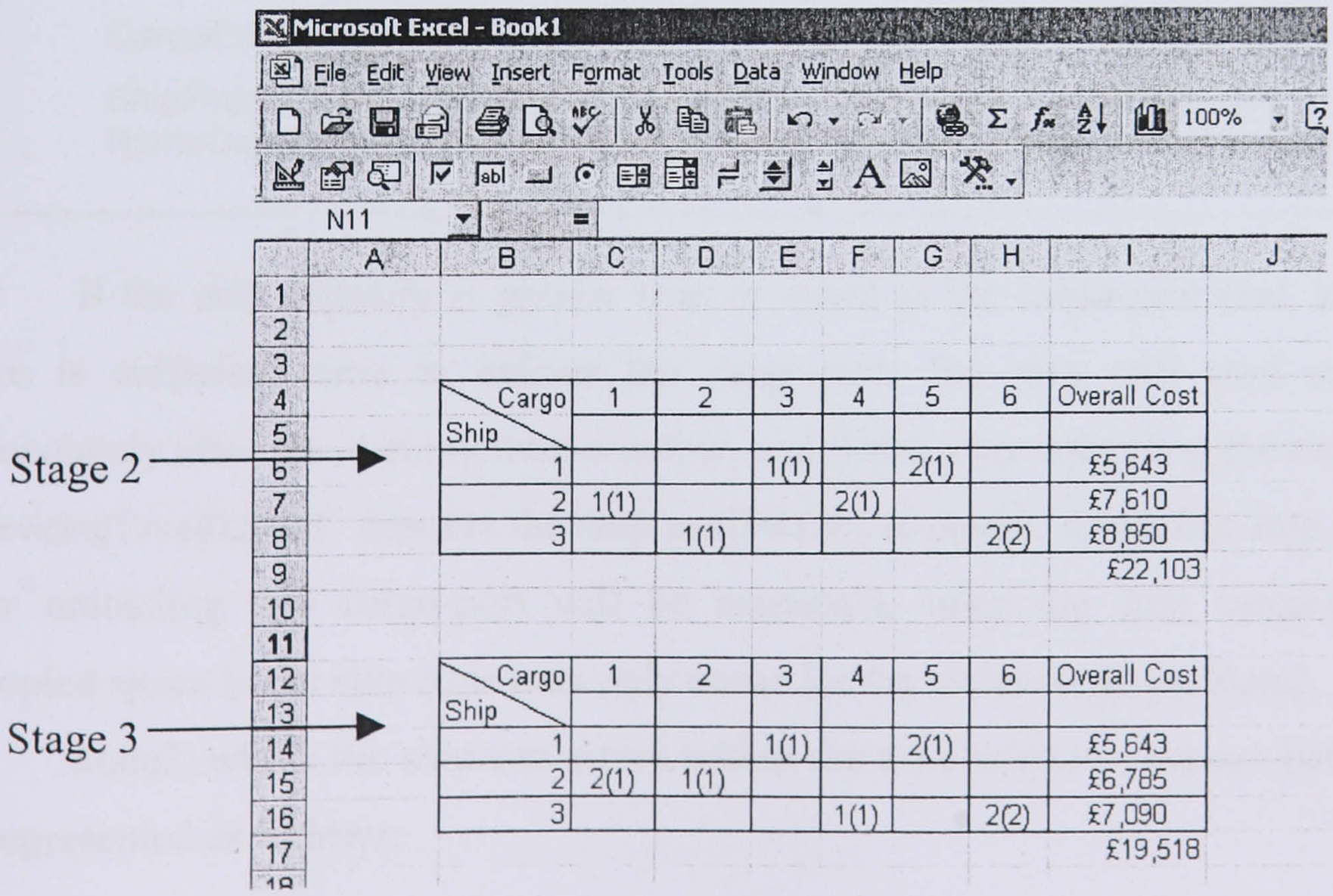
As mentioned in chapter 4, the user enters the intensification value, (Rpt_i), where the number denotes to the number of times cargo-port i has entered the tabu list. This means that, if cargo-port i has entered the tabu list Rpt_i times, then cargo-port i will remain in tabu list for a number of iterations equal to $MaxTn$.

The user also enters the number of iterations to increase minimum tenure by one ($IncTn$). This procedure will carry on until the value of tenure reach $MaxTn$.

Moreover, the user enters ($Divert_i$), mentioned in Chapter 4, which represents the number of times the specific neighbourhood will not be searched after the first search. The user also enters the number of iterations within each neighbourhood $NbItr_i$, and finally, the user enters the total number of iterations $AllItr$ for the whole problem.

When the user has entered the previous data, tabu search will start looking for good solution for this specific problem.

The result will be displayed on the same Excel software spreadsheet. Figure 5.8 illustrates the results obtained by applying tabu search, where Stage 2 displays the results obtained from Greedy Algorithm as mentioned before, and Stage 3 shows the solution obtained by applying Tabu Search.



Ship \ Cargo	1	2	3	4	5	6	Overall Cost
1			1(1)		2(1)		£5,643
2	1(1)			2(1)			£7,610
3		1(1)				2(2)	£8,850
							£22,103

Ship \ Cargo	1	2	3	4	5	6	Overall Cost
1			1(1)		2(1)		£5,643
2	2(1)	1(1)					£6,785
3				1(1)		2(2)	£7,090
							£19,518

Figure 5.8: Results obtained by applying tabu search

5.3.1 Conditions

Two major factors control the operation of assigning a specific cargo-port to a specific ship. The first factor relates to ship capacity in relation to the cargo-port size, and the second factor relates to the delivery time-window for each cargo-port in relation to ship's availability.

1. **Ship capacity**: there is a limited capacity for each ship in the fleet. This capacity allows the ship to be loaded by one or more cargo-ports, where the total size of these cargo-ports must not exceed the ship capacity.

2. **Delivery time**: there are three important states affecting ship delivery time:

   State 1 Ship can arrive before delivery opening time.

   State 2 Ship can arrive before delivery ending time, but not before delivery opening time.

   The following state occurs only with the multi-cargo case:

   State 3 There is more than one cargo-port on the ship.

State1 can be expressed as follows:

```
If ShipIntialCapacity(Ship) >= cargoAmount(cargo) Then
    If ServicingTime(0,Ship) + DistanceCustomer(0,cargo) +Loading (Trip) <=
    CargoEarlytime(cargo) Then
    ShipFinishTime(Cargo,Ship) = CargoEarlytime(cargo) + Unloading(cargo)
    RouteCapacity(Trip,Ship)=RouteCapacity(Trip,Ship) - cargoAmount(cargo)
```

If the ship capacity is greater than or equal to the cargo-port size, and since there is sufficient time to deliver the cargo-port, the ship will start unloading immediately after the opening time-window, and finish after unloading the cargo-port. "ServicingTime(0,Ship)" denotes the ship availability at origin. Available ship capacity after unloading this cargo-port will be increased, since the first cargo-port has occupied space in the ship (this case only occur for the multi-cargo problem).

State2, where the ship can arrive within the time-window, but not before, can be represented as follows:

```
If ShipIntialCapacity(Ship) >= cargoAmount(cargo) Then
    If  ServicingTime(0,Ship)  +  DistanceCustomer(0,cargo)  +Loading  (Trip)  <=
    CargoLatenesstime (cargo) Then

            ShipFinishTime(Cargo,Ship) = ServicingTime(0,Ship) +
                DistanceCustomer(0,cargo) + Unloading(cargo)

            RouteCapacity(Trip,Ship)=RouteCapacity(Trip,Ship) - cargoAmount(cargo)
```

Finally, state3 is related to loading more than one cargo-port on a specific ship. In this state, the same conditions as mentioned previously will be applied as for the first cargo-port. The next cargo-port in the trip will be in one of two situations:

1- The remaining capacity is greater than or equal to the cargo-port size and there is sufficient time to reach the port of this cargo-port before opening delivery time-window. In this case, the ship will wait at sea after unloading the first cargo-port, waiting for the opening delivery time-window for the next cargo-port. This can be presented as follows, where "PreCargo" represents the preceding cargo-port on the trip:

```
If RouteCapacity(Trip,Ship) >= cargoAmount(cargo)Then
    If ShipFinishTime (PreCargo,Ship) + DistanceCustomer(PreCargo,cargo) <=
            CargoEarlytime(cargo) Then

            ShipFinishTime(Cargo,Ship) = CargoEarlytime(cargo) + Unloading(cargo)

            RouteCapacity(Trip,Ship)=RouteCapacity(Trip,Ship) - cargoAmount(cargo)


            Slack(cargo, Ship) = CargoEarlytime(cargo) -
            [ShipFinishTime(PreCargo, Ship) +  DistanceCustomer(PreCargo, Cargo) ]
```

"Slack" represents waiting at sea. Waiting on this occasion must be counted, since there are expenses for waiting per day.

2- The remaining capacity is greater than or the equal to cargo-port size and the ship can arrive within the opening delivery time-window, but not before. This can be presented as follows:

```
If RouteCapacity(Trip,Ship) >= cargoAmount(cargo)Then
    If ShipFinishTime (PreCargo,Ship) + DistanceCustomer(PreCargo,cargo) <=
            CargoLatenesstime (cargo) Then

            ShipFinishTime(Cargo,Ship) = ShipFinishTime (PreCargo,Ship) +
                    DistanceCustomer(PreCargo,cargo) + Unloading(cargo)

            RouteCapacity(Trip,Ship)=RouteCapacity(Trip,Ship) - cargoAmount(cargo)
```

5.4 Model Validation

Validation is used for the process of ensuring that the model produces results that correspond with those of the real system.

Two methods are implemented to validate the Tabu Search model. The first method is concerned with the detail of program execution. After each iteration there are codes added to the model to test the state of the system. This is implemented by tracing each ship route and examining the cargo-port delivery time for each cargo-port by adding the sailing, loading and unloading times, and then testing if the results are correct. If there is any difference, a small window will appear in the screen with a message stating that there is an error in the model. Meanwhile, the method also examines the waiting time and the overall cost for each ship and for the whole fleet of ships.

The second method examined the model manually. In this step, the procedure is to observe and evaluate each route in the system, with particular attention to cargo-port delivery time in the route, waiting time, and overall cost for the route. This is implemented by testing the model every 1000 iterations by opening a small window which provides the user with a result summary (available in Visual Basic 6) and a comparison is made. The test is performed by tracing each route and compares the results with the results obtained from manual method.

Validation of the model designed to generate candidate feasible schedules has been examined using the second method. First, the method is applied by relaxing the constraints and testing the number of candidate feasible schedules which should be generated. The second step is to test that the number of candidate feasible schedules generated is correct. Each candidate feasible schedule is tested logically for the delivery time of the cargo-port in the route, waiting time, and the overall cost for each candidate feasible schedule. Different problem sizes were examined.

Chapter 6: Computational Experiment

6.1 Design

The major objective of the computational experiment reported here is to evaluate the performance of the proposed TS designed model in terms of the quality of the solutions and the computing time. This evaluation can be achieved by comparing the results of the two approaches: the exact method using SPP and the TS designed model. Since some companies ship their cargo-ports using single-cargo mode, as do the Kuwait Oil Tanker Company (KOTC), the second aim is to examine effectiveness of using multi-cargo mode compared to the single-cargo mode.

There are many parameters used in the TS designed model, such as the value of minimum tenure and number of iterations, which are going to be evaluated in order to reach an efficient TS designed model to solve large-scale instances of SRSP.

Since there were obstacles in obtaining some information related to Kuwait Petroleum Company (KPC), due to commercial confidentiality, random data are generated. The following section presents all data generated and their distributions, while section 6.2 presents an evaluation of TS parameter selection. Section 6.3 demonstrates the comparison between the results of solving SRSP in single-cargo and multi-cargo mode by using the TS designed model. Section 6.4 examines the gap between the exact approach and the TS designed model and the computing time for each approach.

6.1.1 Input data

In order to evaluate the performance of each approach, data is needed to carry out the evaluation. Data involved in SRSP can be presented as follows:

1- Ship details (Time availability at origin, capacity, sailing expenses, and waiting at sea expenses).
2- Distance between cargo ports (include origin).
3- Cargo-port details (delivery time-window (start and close) and quantity)
4- Loading and unloading duration, and cost.
5- Port fee due for each ship.

Table 6.1 illustrates all generated random data involved in SRSP.

Parameter		Distribution / value
CAV_v	Availability time of controlled ship v at the origin	$U[1,35]$
$ChAV_v$	Availability time of chartered ship v at the origin	$U[1,5]$
CT_v	capacity of ship v (1000 tons)	$U[70,350]$
CSP_v	sailing cost using controlled ship v ($/day)	$CT_v \times 29$
$ChSP_v$	sailing cost using chartered ship v ($/day)	$CT_v \times 29 \times U[1.5,2]$
CWP_v	waiting cost (idle in the ocean) for controlled ship v	$CT_v \times 9$
$ChWP_v$	waiting cost (idle in the ocean) for chartered ship v	$CT_v \times 9 \times U[1.5,2]$
d_{ik}	distance (in days) between cargo port i and cargo port k	$\sqrt{(a_i - a_k)^2 + (b_i - b_k)^2}$
e_i	earliest arrival time for cargo-port i	$30 + U[1,T]$
l_i	latest arrival time for cargo-port i	$e_i + U[3,20]$
Q_i	cargo-port quantity of cargo-port i (1000 tons)	$U[30,300]$
LD_{iv}	time required for loading cargo-port i onto ship v (days)	$1 \text{ if } Q_i \le 200$ $2 \text{ if } Q_i \ge 201$
UL_{iv}	time required for unloading cargo-port i from ship v (days)	$1 \text{ if } Q_i \le 200$ $2 \text{ if } Q_i \ge 201$
OC_{iv}	operating cost loading and unloading for ship v to handle cargo-port i	$Q_i \times 300$
PC_{iv}	port entrance due (fee) at cargo port i for ship v ($)	$(CT_v + Q_i) \times 22$

T: Time Horizon

Table 6-1: all generated random data involved in SRSP

The planning horizon T is defined as the number of days over which all cargo-ports are to be delivered.

For test problems, controlled ship availability, CAV_v, is generated as a uniform distribution of $U[1,35]$ (days), where $U[\alpha,\beta]$ denotes a uniform distribution over the interval $[\alpha,\beta]$. Meanwhile, chartered ship availability, $ChAV_v$ is generated as a uniform distribution of $U[1,5]$. The capacity of the smallest bulk carrier is about 70 dwt, whereas the largest is about 350 dwt (as ULCC), so the distribution for CT_v is $U[70,350]$. As was mentioned in first chapter, the fuel consumption cost is about 22 times ship capacity. For example, if ship capacity equal 300,000 dwt (CT_v=300), the

fuel consumption cost equal 6816 dollars per day, which is approximately 22 times CT_v. Meanwhile other expenses such as crew and labour cost are approximately equal to one third of fuel consumption cost. Therefore, sailing cost (CSP_v) is valued $(22+22/3)CT_v=29CT_v$. On the other hand, the charter market is unstable, so the expenses of a chartered ship are valued at between 1.5 and 2 times that of a controlled ship. Therefore, a multiplier with distribution specified as $U[1.5,2]$ is used to compute the sailing cost for a chartered ship. The same method is applied for waiting cost, since fuel consumption cost is assumed to be 710 dollars for a ship with capacity equal to 300,000 dwt ($CT_v=300$), which is approximately 2 times CT_v. Since fuel consumption costs approximately 2 times CT_v waiting and labour costs are approximately equal to 7 times CT_v, the waiting cost (CWP_v) is valued at $CT_v \times 9$. The location for each cargo port, denoted as (a_i, b_i), is randomly generated by taking a_i and b_i to be $U[3,35]$ (days). As mentioned previously, operating management should schedule their fleet of ships 30 days before the first delivery. Therefore, the start of delivery time-window, e_i, is distributed as 30 + $U[1,T]$. The end of delivery time-window, l_i, is distributed as e_i+ $U[3,20]$, where some customers impose a delivery time-window of 3 days, while others specify a window of up to 20 days. Due to the uncertainty of customer demand, cargo-port quantity Q_i is distributed as $U[30,300]$. According to official working in KOTC, loading and unloading duration is about one to two days for fully loaded ships. Therefore, if a ship is loaded with less than 200,000 tons, then it will take one day for unloading, otherwise two days. Because there is no certain information about the cost of loading and unloading, OC_{iv}, it is assumed that the distribution is based cargo-port quantity multiply by 300. According to Duisburg port in Germany, the port entrance fee is approximately 18 Euro per 1000 tonnes (about 22 $) based on the weight of the ship and cargo-port. Therefore PC_{iv} is computed as $(CT_v + Q_i) \times 22$.

6.2 Tabu Search Parameter evaluation

In this section, results for computational experiments on approximate and exact models are presented. These experiments are divided into three subsections. Subsection 6.2.1 presents all experiments related to TS parameters as follows:

1- Neighbourhood (Nbr_i) forming method (systematic or random).

2- The size of Neighbourhood (Nbr_i).

3- Intensification value.

4- Diversification value $Divert_i$.

5- Minimum tenure value *MinTn*.

6- Size of problems to be solved by TS.

Subsection 6.2.2 presents a comparison between solving SRSP by using single-cargo mode model and multi-cargo mode model. To measure the efficiency of the TS model, subsection 6.2.3 presents results for the exact and approximate models applied to the multi-cargo mode problem.

6.2.1 TS Parameters Evaluation

6.2.1.1 Selecting Nbr_i by Systematic or Random method

To evaluate the impact of the method used to select a neighbourhood on solution quality, forty randomly generated instances for three problem sizes were used. These experiments were applied, firstly by selecting Nbr_i using the systematic method and secondly using random selection (for more understanding about systematic and random selection, the reader can refer to Chapter 4 section 4.1.6). The experiments are conducted using the same values for other TS parameters (such as the value of Tenure) for the both systematic and random methods. The problem features are illustrated in Table 6.2.

Problem	No. Cargo-port	No. Ship
1	50	25
2	75	30
3	100	33

Table 6.2: problem features to Evaluate Selecting Nbr_i

The experiments show that using the systematic method yields solutions that are better than those obtained by the random method for the whole experiments. Table 6.3 illustrates the solution quality for both methods.

Problem	*The average of objective function*	
	Systematic	Random
1	3030620.4	3111539
2	4818052.8	4940431.3
3	6458454.5	6635888

Table 6.3: Average cost obtaining in the evaluation of the procedure to select Nbr_i

Since the systematic method to determine Nbr_i results in better solutions than obtained by the random method, all the following experiments will be applied to measure the influence of TS parameters on the solution and computing time based on determining Nbr_i using the systematic method.

A possible explanation for the these results is due to the chance of applying the swap move between two cargo-ports will be high if they are near each other, where, after deleting the cargo-port from the route to swap the other route, the previous empty position can be filled with the other cargo-port since the neighbourhoods are near each other. On the other hand, if the swap move is applied with two cargo-ports with a long distance between them, the chance of implementing this move will be small, since the insert to the other route could be infeasible. Since the random method may contain neighbours not near each other, the chance of obtaining a good solution in the random method is lower.

6.2.1.2 Size of Neighbourhood (*NZ*)

The size of the neighborhood is considered to be an important factor in Tabu Search. Five cases are examined with different number of cargo-ports and ships. Forty randomly generated instances for each of these problem sizes were used. Table 6.4 presents the features for each case.

Case	*Time Horizon (days)*	*No. of cargo-ports (n)*	*No. of ships*
1	120	50	18
2	150	80	25
3	180	100	30
4	220	150	40
5	350	200	50

Table 6.4: Problem features to Evaluate Selecting NZ

In this experiment, values for neighborhood size, *NZ*, start at eleven cargo-ports and increase gradually by ten cargo-ports up to the half of the total number cargo-ports in each specific case. The reason for using this method is to evaluate whether the results are influenced by considering a fixed number of cargo-ports for any problem size, or by the ratio between *NZ* and the total number of cargo-ports. Since the *MinTn* used in these experiments is less than *MaxTn* by six, where *MaxTn* as mentioned in Chapter four is equal to *NZ*, and *MinTn* is increased by one every (5**NZ*) - number five is specified by experiments- until reaching *MaxTn*, then the number of iterations for each neighborhood $NbItr_i$ is equal to (30**NZ*). The overall number of iterations is equal to $NbItr_i$ multiplied by the number of cargo-ports (*AllItr* = $NbItr_i$ **n*), where *n* is the total number of cargo-ports. Table 6.5 illustrates the number of iterations for each case.

	Iterations ($30 \times NZ \times n$)								
NZ	11	21	31	41	51	61	71	81	91
Case									
1	16500	31500	46500						
2	26400	50400	74400	98400					
3	33000	63000	93000	123000	153000				
4	49500	94500	139500	184500	229500	274500	319500		
5	66000	126000	186000	246000	306000	366000	426000	486000	546000

Table 6.5: Average number of iterations for each NZ for each case.

Table 6.6 indicates that good solutions may be obtained when the size of neighborhood *NZ* is equal to 11 or 21. In cases two and four the best average of objective function occurs when *NZ* is equal to 11 and for cases one, three and five the best average of objective function is occurred when *NZ* is equal to 21.

Case	*NZ*	*Quality* Objective Function	*Computing Time* Objective Function	
		Mean	Mean	Std. Dev.
1	11	1308851.5	25.4	5.8
	21	1306576.6	53.9	11.2
	31	1314388.5	107	16.3
2	11	2013849	72.5	14.7
	21	2013955	194	30.7
	31	2029028	282.7	44.8
	41	2038295	389.2	51.5
3	11	2414093	82.1	13.2
	21	2404989	210	41.6
	31	2423163.8	293.4	65.7
	41	2420206.8	424.7	88.6
	51	2433479.5	518.4	104.5
4	11	3483500.2	151.8	35.5
	21	3486496.2	357.2	69
	31	3490358	470.6	76.8
	41	3490358	760.8	103.6
	51	3509417	938	116.8
	61	3509103	1189.2	181
	71	3509110	1533.3	266.7
5	11	4522246	330.7	73.3
	21	4515273	483.3	97.2
	31	4520761.7	764.5	113.3
	41	4545257	1070.9	174.6
	51	4558025	1403	195.6
	61	4546985	1818	242.6
	71	4556866	2232.7	332.9
	81	4559507	2793.5	399.2
	91	4545856	3397.6	413.5

Table 6.6: the difference in quality of the objective function for all cases

Table 6.7 illustrates the percentage of times each value of *NZ* produced the best solution found for each case.

Case	*The percentage of times each value of NZ produced the best solution*								
	NZ								
	11	21	31	41	51	61	71	81	91
1	45%	35%	20%						
2	37.5%	27.5%	17.5%	17.5%					
3	20%	25%	20%	15%	20%				
4	32.5%	25%	17.5%	12.5%	0%	5%	7.5%		
5	25%	25%	17.5%	0%	5%	7.5%	10%	5%	5%

Table 6.7: The percentage of times each value of NZ produced the best solution among forty problem instances

Table 6.7 indicates that the best values of *NZ* are 11 and 21. Thus, it could be recommended to adopt a value in the range (11-21). However, reference to Table 6.6 indicates that, overall cases the average solution quality is higher with *NZ* = 21. The possible explanation for the success of the systematic method selecting the neighborhood is also could explain the results here.

6.2.1.3 Diversification ($Divert_i$)

The following experiments to evaluate other TS parameters will be applied on three different problem sizes and forty randomly generated instances for each of these problem sizes were used as illustrates in Table 6.8.

Case	*Time Horizon*	*Cargoe-ports*	*Ships*	*NZ*	*NbItr (50*NZ)*	*AllItr (NbItr*n)*
	Days	*Number*	*Number*	*Number*	*Number*	*Number*
1	100	40	20	21	1050	42000
2	120	60	23	21	1050	63000
3	150	80	25	21	1050	84000

Table 6.8: Three different problem size with number of iterations

As mentioned previously, *NZ* will be set to 21 cargo-ports. Since one of the experiment is to evaluate the value of *MinTn* (Section 6.2.1.5), where one of these values is equal to *MaxTn/2* ($21/2 \cong 11$), therefore, increasing *MinTn* by one requires ten times to reach *MaxTn*. Since *MinTn* is increased by one every (5**NZ*) – as mentioned previously - until reaching *MaxTn*, then the number of iterations for searching within each neighbourhood will be equal to 1050 iterations (50*21) as shown in Table 6.8.

The aim of diversification as mentioned in Chapter four is to steer the search to another region instead of searching in a local area. When searching a specific region has finished, a new neighbourhood will be defined. For the three cases, five different values of $Divert_i$ are evaluated. The first value of $Divert_i$ is equal to zero which means that the diversification parameter is not applied. The second value of $Divert_i$ is equal to *AllItr/4* which means that if cargo-port *i* has been selected to form a new neighbourhood, then cargo-port *i* will not be selected for at least *AllItr/4*

iterations. The other values of $Divert_i$ to be evaluated will be equal to *AllItr/2*, *3AllItr/4* and *AllItr*.

Case			$Divert_i$				
			0	*AllItr/4*	*AllItr/2*	*3AllItr/4*	*AllItr*
1	*Frequency of best solution*	*Percentage*	0%	27.5%	17.5%	15%	40%
	Quality	*Average*	2692999.7	2557731.3	2571257.4	2575080.3	2547382.4
	PC time (sec.)	*Average*	43.3	46.4	46.3	45.2	45.5
		Std. Dev	5.9	6.6	6.8	6	6.5
2	*Frequency of best solution*	*Percentage*	2.5%	20%	17.5%	15%	45%
	Quality	*Average*	4138192.1	3997969.5	3999681.9	4009363.6	3934461.1
	PC time (sec.)	*Average*	103.7	103.5	105.8	111	104.4
		Std. Dev	14.1	15.2	16.3	13.5	16.7
3	*Frequency of best solution*	*Percentage*	2.5%	32.5%	25%	22.5%	17.5%
	Quality	*Average*	5577322.2	5424004.3	5436278.5	5433320.1	5448568.7
	PC time (sec.)	*Average*	167.5	173.3	178	176.6	170
		Std. Dev	27.4	30.9	33.7	33.4	30.1

Table 6.9: Results of applying diversification on three different problem sizes

It is clear from Table 6.9 that applying diversification is beneficial. The quality of the solution and the frequency of best solutions is the lowest when $Divert_i$ equal zero. On the other hand, when $Divert_i$ is greater than zero the average quality of the solutions is close: within 1% for case one, 2% for case two and 0.5% for case three.

It can be concluded that applying diversification is essential to produce good solutions, and a value of $Divert_i$ between *AllItr/4* and *AllItr* is recommended.

When $Divert_i$ is equal to zero, then the chance to search in the same neighbourhood is high since the search will be implemented in small area. On the other hand, if $Divert_i$ is not equal to zero, then the search will be diverted to another area to be searched and this could explain why the quality of the solution is better when $Divert_i$ is not equal to zero.

6.2.1.4 Intensification

There are two methods of intensification as mentioned in Chapter four to be evaluated. In the first method, if cargo-port *i* has repeatedly entered the tabu list a

given number of times, the cargo-port will remain in the tabu list for a number of iterations equal to $MaxTn$ (cargo-port i has entered the tabu list for a number of times denoted by Rpt_i). In the second method, if the search yields an improved solution, the iteration count will be set to zero.

The first method is evaluated by examining the solutions when Rpt_i is equal to three (denote by *YES*), where number three determined by experimenters, and then by ignoring the method (denote by *NO*). Table 6.10 illustrates the results.

Case			*Rpti*	
			Yes	*No*
1	*Frequency of best solution*	*Percentage*	47.5%	52.5%
	Quality	*Average*	2544288.3	2547382.4
	PC time (sec.)	*Average*	39.2	45.5
		Std. Dev	7.1	6.5
2	*Frequency of best solution*	*Percentage*	47.5%	52.5%
	Quality	*Average*	3943770.6	3934461.1
	PC time (sec.)	*Average*	103.7	104.4
		Std. Dev	16.1	16.7
3	*Frequency of best solution*	*Percentage*	42.5%	57.5%
	Quality	*Average*	5470913.9	5448568.7
	PC time (sec.)	*Average*	176	170
		Std. Dev	34.3	30.1

Table 6.10: Results of applying first method of intensification on three different problem sizes

For the three cases, there is no advantage in applying the first method.

The second experiment is to evaluate the second method. Table 6.11 illustrates the results when the method is (*YES*) or not (*NO*). It is clear from the Table 6.11 that applying the second method where the iteration count will be set to zero when the search yields an improved solution is beneficial.

Case			*Second method*	
			Yes	*No*
1	*Number of obtaining the best solution*	*Percentage*	65%	35%
	Quality	*Average*	2547382.4	2553905.9
	PC time (sec.)	*Average*	45.5	42.9
		Std. Dev	6.5	6.3
2	*Number of obtaining the best solution*	*Percentage*	52.5%	47.5%
	Quality	*Average*	3929003.6	3934461.1
	PC time (sec.)	*Average*	104.4	100.2
		Std. Dev	16.7	15.7
3	*Number of obtaining the best solution*	*Percentage*	55%	45%
	Quality	*Average*	5435114.5	5448568.7
	PC time (sec.)	*Average*	170	163.3
		Std. Dev	30.1	29.4

Table 6.11: Results of applying second method of intensification on three different problem sizes

According to these experiments, the conclusion is to use the second method. This can be explained due to the intensive searching in each area. If the search can generate a better solution, then the counter will set to zero to search again, where the search will not stop unless no better solution is achieved.

6.2.1.5 Minimum Tenure (*MinTn*)

In this experiment, minimum tenure value (*MinTn*) is set in three possible ways. As explained in Chapter four, *MaxTn* is set equal to *NZ*, and the value of *MinTn* is set up relation to the value of *MaxTn*, as follows:

1- $MinTn = MaxTn/2$

2- $MinTn = 2MaxTn/3$

3- $MinTn = MaxTn$

Two types of experiments were applied. The first one is to evaluate the value of *MinTn,* when it is less than *MaxTn* (*MinTn = MaxTn/2* or *MinTn = 2MaxTn /3*) or equal to *MaxTn*. The second experiment is to evaluate the number of iterations needed to increase *MinTn* by one. If the first experiment where the value of *MinTn* is less than *MaxTn* produces a solution better than the case, where the value of *MinTn* is equal to *MaxTn*, then the second experiment will be implemented. However, if the first

experiment where the value of *MinTn* is less than *MaxTn* produces a solution worse than the other case, then the second experiment need not be applied.

Table 6.12 shows that when the value of *MinTn* is equal to *MaxTn/2* or *2MaxTn /3* a good solution can be achieved, where the frequency of best solution and the quality of the solution are better in all three cases. In addition, a comparison was applied between the results for the two ratios to evaluate the better value of *MinTn*. In the first case, the results show that when *MinTn* is equal to *2MaxTn /3* the solution is slightly better where the difference between the quality and frequency of best solution are very close. On the other hand, for cases two and three the results with *MinTn* equal to *MaxTn/2* perform better than when *MinTn* equal to *2MaxTn /3*, where the frequency of best solution is higher, 40%, while the quality of the solutions are very similar within 0.4% for case two and 0.2% for case three. In addition, the computing time is almost the same for all experiments.

Case			*MinTn value*		
			NZ/2	*2NZ/3*	*MaxTn*
1	*Number of obtaining the best solution*	*Percentage*	32.5%	35%	32.5%
	Quality	*Average*	2550504.8	2547382.4	2557406.4
	PC time (sec.)	*Average*	49.8	46.4	45.7.3
		Std. Dev	6.8	6.6	6.3
2	*Number of obtaining the best solution*	*Percentage*	40%	30%	30%
	Quality	*Average*	3919173.3	3934461.1	3953311.3
	PC time (sec.)	*Average*	112.6	103.5	103.9
		Std. Dev	16.3	15.2	14.7
3	*Number of obtaining the best solution*	*Percentage*	40%	27.5%	32.5%
	Quality	*Average*	5438934.4	5448568.7	5454806
	PC time (sec.)	*Average*	173.4	173.3	171.7
		Std. Dev	31.6	30.9	29.4

Table 6.12: The evaluation of MinTn

Since first experiment showed that, when the value of *MinTn* is less than *MaxTn*, performed better, then the second experiment is to evaluate the best number of iterations needed to increase *MinTn* by one. Five different numbers of iterations (required to increase *MinTn* by one) have been evaluated. Table 6.13 presents the performance of applying these different numbers of iterations (*NZ* equal to 21).

Case			*Increase MinTn every number of iterations equal*				
			NZ	*2NZ*	*3NZ*	*4NZ*	*5NZ*
1	*Frequency of best solution*	*Percentage*	15%	10%	20%	27.5%	27.5%
	Quality	*Average*	2575750	2581167.4	2562946.4	2558958.2	2550504.8
	PC time (sec.)	*Average*	47.9	52.5	45	45.4	49.8
		Std. Dev	7.1	5.9	6	6.5	6.8
2	*Frequency of best solution*	*Percentage*	20%	12.5%	15%	35%	17.5%
	Quality	*Average*	3885558.6	3897239.4	3890452.9	3868807.5	3919173.3
	PC time (sec.)	*Average*	104.1	115.2	103.4	106.9	112.6
		Std. Dev	14.3	20.1	13	17.9	16.3
3	*Frequency of best solution*	*Percentage*	22.5%	15%	25%	22.5%	15%
	Quality	*Average*	5443578.4	5463166.6	5431646	5423709.1	5438934.4
	PC time (sec.)	*Average*	177	176.9	168.3	183.1	173.4
		Std. Dev	31	29.3	33.6	34.3	31.6

Table 6.13: Problems of different MinTn

Table 6.13 shows that there is not a big difference between these different numbers of iterations. However, there is an indication that increasing *MinTn* by one every *4NZ* iterations performs better than other numbers of iterations. It is clear in case two and three the frequency of best solution and the quality are the best among other results, while in case 1, the quality is the second best.

From these experiments, a conclusion can be drawn by considering that *MinTn* equal to *MaxTn/2* or to *2MaxTn /3* generates a solution better than would be obtained with *MinTn* equal to *MaxTn*. Meanwhile, *MinTn* is equal to *MaxTn/2* generates a solution better than would be obtained with *MinTn* equal to *2MaxTn /3*. Finally, increasing *MinTn* by one can be carried out every *4NZ* iterations.

6.2.1.6 Problem size

Table 6.14 presents different size problems, start with 50 cargo-ports up to 200 cargo-ports problem. The experiment applied on forty problem instances for each problem size.

Case	*Time Horizon*	*Cargo-port*	*Ship*
	Days	*Number*	*Number*
1	80	50	18
2	100	80	25
3	150	100	30
4	180	150	40
5	250	200	50

Table 6.14: Problem features to evaluate problem size

Figure 6.1 illustrates the increase of the computing time corresponding to the increase of the problem size. Problem of fifty cargo-ports can take an average 57 seconds, while problem of 200 cargo-ports can take an average 1337 seconds. This indicates to the requirement for more time to solve large-scale problems.

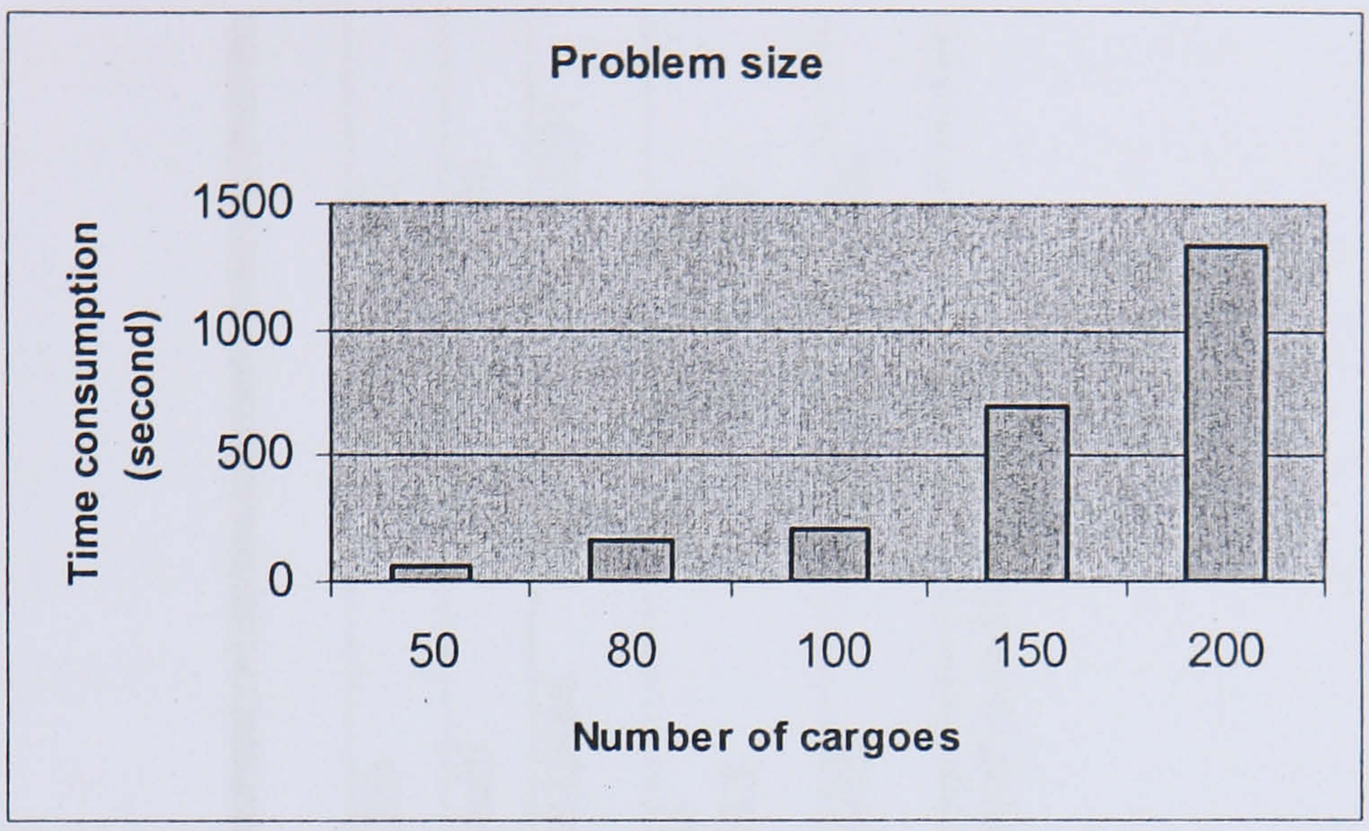


Figure 6.1: Different problem sizes

The next subsection will compare the experiment results of solving SRSP using TS between single-cargo mode and on multi-cargo mode. All TS parameters yield good solutions are shown by the previous experiments.

6.2.2 Single-cargo and Multi-cargo Comparison

Since some companies (as Kuwait Oil Tanker Company (KOTC)) prefer to use single-cargo mode to schedule their fleet of ships. Four different problem sizes and 100 randomly generated instances for each of these problem sizes was used. Table 6.15 illustrates the objective cost and the computing time for each problem. The number between brackets denotes the problem setting, where first number denotes time horizon, second denotes number of cargo-ports, and last number denotes the number of ships.

		Problems							
		(100,40,20)		(100,50,25)		(120,60,25)		(150,100,33)	
		Single	Multi	Single	Multi	Single	Multi	Single	Multi
Quality	*Average*	2829180	2503278	3546713	3088738	4329212	3935866	7257822	6505191
PC. time (Sec)	*Average*	40	46	66	50	94	85	275	262
	Std. Dev.	6	6.5	9.8	10.7	12.8	16.5	46	50.5
Percentage solved		70%	100%	74%	100%	69%	100%	72.5%	100%

Table 6.15: the objective cost and solution time for four different problem sizes

From these experiments, the quality of the objective function using multi-cargo mode is better in all four cases, where the difference of the quality between single and multi modes for the four sizes are 13%, 15%, 10%, and 12% respectively. These percentages reflect the influence of the operation cost by using multi mode instead of single mode. Meanwhile, among 100 problem instances, single mode does not have the ability to handle all 100 problems, where for example in case one single mode can solve seventy instance problems and cannot solve the rest thirty problem instances with the specified time horizon. On the other hand, solution time for single-cargo is slightly, less than multi-cargo mode.

This provides a good indication that using multi-cargo mode saves a lot of money and utilizes the available fleet of ships in an efficient way.

6.2.3 Comparison between SPP approach and TS approach

Since the previous computational results indicate that using multi-cargo mode is more efficient than single-cargo mode. This fact makes the comparison between the exact approach model and approximate approach model is unnecessary. Therefore, the following computational results are only for multi-cargo. Meanwhile, all TS parameters values, which provided good solutions in the experiments presented previously, are used in these experiments.

Table 6.16 presents the computational results for six cases. For each, forty different problem instances were generated (240 different problems are the total). The results for all cases represent schedule generations, applying Set Partitioning model, and applying Tabu Search model. Number of cargo-ports, start from twenty cargo-ports up to one hundred cargo-ports, with number of nine to forty ships involved.

The number of candidate schedules has a limit set at 900,000. Since the number of candidate schedules in cases one and two are within this limit, all candidate schedules are generated (all instances of case one and 36 out of 40 instances for case two). Therefore, it is ensured that the solutions are optimal. On the other hand, when the total number of candidate schedules generated exceeds this number, as in cases three to six, a subset of candidate schedules are selected.

		Case	1	2	3	4	5	6
Time Horizon (days)			70	100	100	120	120	150
Cargoes			20	30	40	50	75	100
Ships			9	15	17	20	30	40
Max. No. of Schedules per ship			All	All	50000	45000	30000	20000
Schedules								
	No. Schedules	*Average*	16836.6	286359	424130	551339	651790	647831
		Std. Dev.	10799	200199	109982	84688	62317	61981
	CPU-seconds	*Average*	45.5	137	131.7	106	97	91
		Std. Dev.	3	14.6	11.4	29.5	30	22.7
Set Partitioning								
	Quality	*Average*	1093621.4	1547789.7	2057987.8	2698111	3919554.2	5355456.6
	CPU-seconds	*Average*	4.88	399.6	622.4	774.3	1183	1345.2
		Std. Dev.	4.68	710	468.3	600.2	1071	1132
Tabu Search								
	Quality	*Average*	1323780.9	1918886.1	2637255.5	3382069.5	4868766.4	6311886.4
	No. of iterations		20000	52000	80000	115200	260000	500000
	CPU-seconds	*Average*	15	26.2	46	70.8	160.4	227.3
		Std. Dev.	1.7	4.7	6.3	15	18	38.5
% Gap between SPP and TS		*Average*	21 %	24 %	28 %	25.3 %	24.2 %	17.9 %

Table 6.16: SPP and TS computational results

It is expected that the solution is not optimal but may approach the optimal solutions for cases three to six, where a subset of candidate schedules are selected.

According to Table 6.16 the generation of candidate feasible schedules of case one requires CPU-time more than the time required to obtain the solution of the set partitioning problem. This occurred due to the exponential computational time, which were not occurred for the other cases.

As the problems been larger, the solutions obtained by SPP lose the property of optimality since only a subset of all feasible schedules can be included in the model. Thus, the relative quality of the TS approach approaches that of SPP approach as shown in Table 6.16, where for cases three to six the gap is shown to be 28%, 25.3%, 24.2%, and 17.9% respectively. This can be interpreted due to the number of the subset of candidate feasible schedules generated for each case, where in case three the number is large (up to 50000 schedules for each ship), while it gets smaller for next cases (the candidate feasible schedules for case six do not exceed 20000).

A conclusion can be drawn from these results of these experiments that the gap between solutions obtains by applying TS and SPP is large. However, these results required more computing time when SPP is applied, where applying TS generate the solution more quickly. This time will increase exponentially if the problem size is increased. On the other hand, the performance of TS is more predicted in term of computing time, while in SPP is not and this is clear from standard deviations provided in Table 6.16, which are large.

Meanwhile, the experiments for SPP indicate that the number of ships involved in the problem should be large to obtain subsets capable to generate feasible solutions, especially for large problem. This fact does not exist in tabu search model. Problems for more than 100 cargo-ports such as 125 cargo-ports and over were attempted to solve using the same subset method, however, infeasible solutions were achieved. This has happened in some instances in case six, where four solutions out of forty, generated infeasible solutions.

Many factors influence the number of candidate schedules. For example, time horizon, if the length of the time horizon is so wide, then the number of candidate schedules will be large. Also, the delivery time window of customers influence the number of candidate schedules, where the width for delivery time-window will allow many ships to deliver there cargo-ports if it is wide, while the opposite will restrict the number of ships. Other factor, such as ship capacities, if the capacity of the ship is so

large will allow the ship to deliver many cargo-ports in the same route. Therefore, as illustrated in Table 6.16, the number of feasible schedules is varied among the provided instance problems and this is obvious from the standard deviation. For example, in case one the average number of feasible schedules generated are 16836.6 schedules and standard deviation is 10799 schedules, which is more than 64%.

Chapter 7: Conclusions and Further Researches

The latest available sea transportation statistics show that the world shipping fleet has increased to a record level of 633,321 million gross tons in 2004, an increase of 28 million gross tons from a year earlier. This fact and many facts mentioned in this thesis, emphasize the reliance of the world economy on seaborne trade and hence highlights the need for well-organized and reliable maritime transportation systems. Routing and scheduling of ships requires a significant level of fleet management planning.

Most of the shipping companies such as (KPC), schedule their fleet of ships using little or no use of optimisation approaches. They usually schedule their fleet using ad-hoc (as in KPC) and then use a spreadsheet (excel software) to visualize the schedule.

In the past, there was a scarcity of research in this area. However, there has been an increased interest and focus on it since the review on ship routing and scheduling by Ronen in 1993 [94]. Ship routing and scheduling is considered to be an interesting area with great potential for using optimisation to improve fleet utilization.

This thesis is devoted to solving this type of problem, called Ship Routing and Scheduling Problem (SRSP). There are two classes of scheduling problem considered in this thesis. The first one is called single-cargo schedule, while the second is called multi-cargo schedule. The single-cargo schedule problem consists of routes containing trips that service only one cargo-port on each trip, while the multi-cargo can service more than one cargo-port on each trip.

Two computational approaches to handle SRSP are considered. The first is an optimisation approach based on the Set Partitioning Problem (SPP) and the second approach is an approximate method based on Tabu Search (TS). For the optimisation approach, a number of candidate feasible schedules are generated, each of which corresponds to a variable in the SPP model. A user-friendly interface is developed which allows the user to specify the data that defines a particular problem instance of SRSP. This information can be entered using an interface called "feasible schedules generation". A number of candidate feasible schedules will be generated and then transferred to software called MPL. The MPL software was used to generate the model which was solved using the commercial optimisation solver CPLEX-10. The

second approach adopted is Tabu Search. A proposed schedule for this problem will be displayed on EXCEL spreadsheet.

The major objective of the computational experiment was to evaluate the performance of the proposed TS designed model in terms of the quality of the solutions and the computing time. This evaluation was achieved by comparing the results of the two approaches: the exact method using SPP and the TS designed model.

The following parameters were evaluated in the TS designed model:

1. Neighbourhood (Nbr_i) forming method (systematic or random).
2. The size of Neighbourhood (Nbr_i).
3. Intensification value.
4. Diversification value $Divert_i$.
5. Minimum tenure value *MinTn*.
6. Size of problems to be solved by TS.

The experiments showed that forming a neighbourhood with the systematic method performed better than by using the random method. The best size of neighbourhood was formed to be 21 cargo-ports. Also, experiments emphasised that applying intensification and diversification parameters is worthwhile. The best value of *MinTn* is equal to the half of the value of *MaxTn* (*MaxTn* /2), and increasing *MinTn* by one can be carried out every *4NZ* iterations, where *NZ* is equal to 21.

Some companies (as Kuwait Oil Tanker Company (KOTC)) prefer to use single-cargo mode to schedule their fleet of ship instead of multi-cargo mode. The results indicates that using the second mode performed significantly better than the first mode. The difference of the solution quality between single and multi modes showed multi-cargo is better than single-cargo mode by at least 12%.

To measure the efficiency of the TS model, the exact approach based on SPP was applied to the multi-cargo mode problem. Computational results in Chapter six showed that the gap between solutions obtained by applying TS and SPP is up to 28% for small problems and up to 18% for large problems. On the other hand, these results required more computing time when SPP was applied, whereas the TS model could solve larger problem more quickly.

Computational results in Chapter six also indicated that the optimisation approach can solve moderate size problems by generating all feasible schedules and

some large size problems by using a method to generate a subset of feasible schedules, which is considered very close to the optimal solution. On the other hand, due to an exponential increase in the number of schedules, a very large problem will be difficult to solve. Therefore, the user in this situation can resort to TS model to solve his/her problems.

A conclusion from this thesis can be drawn, for small problems (such as thirty cargo-ports) and reasonable middle size problems (not more than one hundred cargo-ports) can be solved efficiently using the SPP model, whereas for larger problems, the user can resort to the TS model to obtain approximate solutions. The use of Tabu Search for SRSP is novel and the results indicate that it is viable approach for large problems.

Further Researches

To improve the performance of the first model (SPP) and to enhance the quality of the solution, various methods are proposed. A heuristic method to select the subset of feasible schedules could be designed. Meanwhile, reducing the computational time may be achieved by considering the waiting time (wv_{iv}). Since the waiting time for unloading the cargo-port is considered costly, all schedules that have a waiting time will be eliminated to control the number of candidate schedules.

The quality of the solution could be improved by using *Aspiration Criteria* technique, which allows one to ignore the tabu status of a move if this move leads to a value better than the best-known value found by the search so far. Another way is to merge the Tabu Search method with other heuristic methods, such as Genetic Algorithm (GA). The methodology of these hybrid methods can be achieved as follows: since a Genetic Algorithm is considered to be a very powerful heuristic method, compared to a Greedy Algorithm, the initial solution could be generated using a Genetic Algorithm, followed by applying TS to continue the search to obtain a solution. Another way, is the opposite case, where the initial solution could be obtained using Tabu Search, followed by a Genetic Algorithm to continue the search for a better solution.

Meanwhile, further work can be considered by using soft delivery time-windows. Most customers impose a delivery time-window for their cargo-ports. This delivery time-window can be violated. However, there are penalties imposed for any

violation of the delivery time-window. Therefore, ships can be loaded to their entire capacity by violating the time-window, and this approach is taken to reduce the overall cost.

References:

1. , posting date. http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html. [Online.]
2. 1988-2002. MPL Modeling System. Maximal oftware, Inc.
3. **6, v. v.** 1987-1998. Microsoft Visual Basic 6, p. 32-bit window deveploment. Microsoft Corp.
4. **Aarts, E., P. Laarhoven, J. Lenstra, and N. Ulder.** 1994. A computational study of local search algorithms for job shop scheduling. ORSA Journal on Computing **6**:118-125.
5. **Al-Basry, M.** 2006. Bunker Consumption and Price, p. Interview May. 2006. *In* K. O. T. Company (ed.), Kuwait.
6. **Al-Khayyal, F., and S.-J. Hwang.** In Press. Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, Part I: Applications and model. European Journal of Operational Research.
7. **Allahverdi, A., and F. Al-Anzi.** 2006. A PSO and a tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. Computers and Operations Research **33**:1056 - 1080.
8. **Al-Yakoop, S.** 1997. Mixed-Integer Mathematical programming Optimization Models and Algorithms for Oil Tanker Routing and Scheduling Problem. PhD. Faculty of the Virginia Polytechnic Institute and State University, Virginia.
9. **Anderson, C. A., K. Fraughnaugh, M. Parker, and J. Ryan.** 1993. Path assignment for Call Routing: An Application of Tabu Search. Annals of Operations Research **41**.
10. **Appelgren, L.** 1969. A column generation algorithm for a ship scheduling problem. Transportation Science **3**:53-68.
11. **assad, A. A.** 1980. Models for Rail Transportation. Transportation Research Part A **14A**.
12. **Baker, B. M., and M. A. Ayechew.** 2003. A genetic algorithm for the vehicle routing problem. Computers & Operations Research **30**:787-800.
13. **Barbarosoglu, G., and D. Ozgur.** 1999. A tabu search algorithm for the vehicle routing problem. Computers & Operations Research **26**:255-270.

14. **Bausch, D. O., G. G. Brown, and D. Ronen.** 1991. Elastic set partitioning: A powerful tool for scheduling transportation of oil and gas. M. Breton and G. Zaccour (eds.), advanced in Operations research in the Oil and Gas Industry, Editions Technip, Paris:151-162.

15. **Beasley, J.** 1990. distributing test problems by electronic mail. Journal of the Operational Research Society **41:**1069-1972.

16. **Bellmore, M., G. Bennington, and S. Lubore.** 1971. A multi-vehicle tanker scheduling problem. Transportation Science **5:**36-47.

17. **Ben-Daya, M., and M. Al-Fawzan.** 1998. A Tabu Search Approach for the Flow Shop Problem. ORSA Journal on Operations Research Society **109:**88-95.

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

19. **Boffey, T. B., E. D. Edmond, A. I. Hinxman, and C. J. Pursglove.** 1979. Two Approaches to Scheduling Container Ships with an Application to the North Atlantic Route. Operational Research Society Ltd **30:**413-425.

20. **Brandao, J., and A. Mercer.** 1997. A tabu search algorithm for multi-trip vehicle routing and scheduling problem. European Journal of Operational Research **100:**180-191.

21. **Breedam, A. V.** 1995. Improvement heuristics for the Vehicle Routing Problem based on Simulated Annealing. European Journal of Operational Research **86:**480-490.

22. **Breedam, A. V.** 2001. Comparing descent heuristics and metaheuristics for the vehicle routing problem. Computers & Operations Research **28:**289-315.

23. **Briskon, L. E.** 1966. Selecting delivery dates in the tanker scheduling problem. Management Science **12:**224-233.

24. **Bronmo, G., M. Christiansen, K. Fragerholt, and B. Nygreen.** In press. A multi-start local search heuristic for ship scheduling - acomputational study. Computers & Operations Research.

25. **Brown, G. G., G. W. Graves, and D. Ronen.** 1987. Sheduling ocean transportation of crude oil. Management Science **33:**335-346.

26. **Chakrapani, J., and J. Skorin-Kapov.** 1993a. Massively Parallel Tabu search for the Quadratic Assignment Problem. Annals of Operations Research **41**:327-341.

27. **Cho, S.-C., and A. N. Perakis.** 1996. Optimal liner fleet routing strategies. Maritime Policy and Management **23**:249-259.

28. **Choi, I.-C., S.-I. Kim, and H.-S. Kim.** 2003. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. Computers & Operations Research **30**:773-786.

29. **Christiansen, M., K. Fagerholt, and D. Ronen.** 2004. Ship Routing and Scheduling: status and Perspectives. Transportation Science **38**:1-18.

30. **Christofides, N., and P. Toth.** 1979. The vehicle routing problem. Combinatorial Optimization, Wiley, Chichester.

31. **Chung, C.-S., J. Flynn, and O. Kirca.** 2002. A branch and bound algorithm to minimize the total flow time for m-machine permutation flowshop problems. Int. J. Production Economics **79**:185-196.

32. **Clarke, G., and J. Wright.** 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research **12**:568-581.

33. **Cordeau, J. F., P. Toth, and D. Vigo.** 1998. A survey of optimization models for train routing and scheduling'. Transportation Science **32**:380-404.

34. **CPLEX-10.** 2005. ILOG. http://www.ilog.com/products/cplex/news/whatsnew.cfm.

35. **Crainic, T. G., and G. Laporte.** 1997. Planning models for freight transportation. European Journal of Operational Research.

36. **Crino, J. R., J. T. Moore, J. W. Barnes, and W. P. Nanry.** 2004. Solving the theater distribution vehicle routing and scheduling problem using group theoretic tabu search. Mathematical and Computer Modelling **39**:599-616.

37. **Dantzig, G. M., and D. R. Fulkerson.** 1954. Minimizing the number of tankers to meet a fixed schedule. Naval Research Logistics Quarterly **1**:217-222.

38. **Darby-Dowman, K., and J. Wilson.** 2002. Developments in linear and integer programming. Journal of the Operational Research Society **53**:1065-1071.

39. **Datz, I. M., C. M. Fixman, A. W. Friedberg, and V. A. Lewinson.** 1964. A description of the maritime administration mathematical simulayion of ship operations. Trans. SNAME:493-523.

40. **Dell'Amico, M., and M. Trubian.** 1998. Solution of large weighted equicut problems. European Journal of Operational Research **106:**500-521.

41. **Dodlin, B., A. A. Elimam, and E. Rolland.** 1998. Tabu Search in Audit Scheduling. European Journal of Operational Research **106:**373-392.

42. **Fagerholt, K.** 1999. Optimal fleet design in a ship routing problem. International. Transaction of Operational. Research. **6:**453-464.

43. **Fagerholt, K.** 2004. A computer-based decision support system for vessel fleet scheduling—experience and future research. Decision Support Systems **37:**35-47.

44. **Fagerholt, K., and M. Christiansen.** 2000. A combined ship scheduling and allocation problem. Journal of the Operational Research Society **51:**834-842.

45. **Giannelos, N. F., and M. C. Georgiadis.** 2003. Efficient scheduling of consumer goods manufacturing processes in the continuous time domain. Computers & Operations Research **30:**1367-1381.

46. **Gillett, B., and L. Miller.** 1974. A heuristic algorithm for the vehicle dispatch problem. Operations Research **22:**340-349.

47. **Glover, F.** 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. Computers & Operations Research **13:**533-549.

48. **Glover, F.** 1990. Tabu Search: A tutorial. Interfaces **20:**74-94.

49. **Glover, F., and M. Laguna.** 1997. Tabu search. Kluwer Academic Publishers, Bosten.

50. **Goldberg., D. E.** 1989. Genetic Algorithms in Search, Optimization & Machine learning. Addison-Wesley.

51. **Golden, B., E. Wasil, J. Kelly, and I. Chao.** 1998. The impact of Metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. Fleet Management and Logistics:33-56.

52. **Gopalan, R., and K. T. Talluri.** 1998. The aircraft maintenance routing problem. Operations Research **46:**260-271.

53. **Grönkvist, I.** 2006. Accelerating column generation for aircraft scheduling using constraint propagation. Computers & Operations Research **33:**2918-2934

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

55. **Han, J.** Article in press. Frequency reassignment problem in mobile communication networks. Computers & Operations Research.

56. **Ho, S. C., and D. Haugland.** 2004. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. Computers & Operations Research **31**:1947-1964.

57. **Holland, J. H.** 1975. Adaption in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

58. **Holst, O., and B. Sorenson.** 1984. Combined scheduling and maintenace planning for an aircraft fleet. Operations Research **32**:735-747.

59. **http://mathworld.wolfram.com/InteriorPointMethod.html.**

60. **http://www.cunard.com/OurShips/default.asp?Ship=QM2&main=int&sub=his.** 2004. Queen Mary 2.

61. **http://www.duisport.de/en/logistik_transport/transport_segmente/short_sea_schifffahrt/hafenentgelte/index.php.**

62. **Ioachim, I., J. Desrosiers, F. Soumis, and N. Belanger.** 1999. Fleet assignment and routing with schedule synchronization constraints. European Journal of Operational Research **119**:75-90.

63. **Jaramillo, D. I., and A. N. Perakis.** 1991. Fleet deployment optimization for liner shipping Part 2. Implementation and results. Maritime Policy and Management **18**:235-262.

64. **Jaszkiewicz, A.** 2001. Multiple objective metaheuristic algorithm for combinatorial optimization. Politechnika Poznanska, Poznan.

65. **Karp, R. A.** 1979. patching algorithm for the nonsymmertric traveling-salesman problem. SIAM **8**:561-573.

66. **Kirkpatrick, S., J. Gelatt, and M. P. Vecchi.** 1983. Optimization by Simulated Annealing. Science **220**:671-680.

67. **Kydland, F.** 1969. Simulation of liner operation. Institute for Shipping Research, Bergen.

68. **Laguna, M., R. Marti, and V. Valls.** 1997. Arc crossing minimization in hierarchical digraphs with tabu search. Computers & Operations Research **24:**1175-1186.

69. **Land, A., and A. Doig.** 1960. An automatic method for solving discrete programming problems. Econometrica **28:**497-520.

70. **Lane, D. E., T. D. Heaver, and D. Uyeno.** 1987. Planning and Scheduling for Efficiency in Liner Shipping. Maritime Policy and Management **14:**109-123.

71. **Lima, C. M. R. R., M. C. Goldbarg, and E. F. G. Goldbarg.** 2004. A memetic algorithm for heterogeneous fleet vehicle routing problem. Electronic Notes in Discrete Mathematics **18:**171-176.

72. **Lokketangen, A., and F. Glover.** 1998. Solving Zero-One Mixed Integer Programming Problems Using Tabu Search. European Journal of Operational Research **106:**624-658.

73. **Lundy, M., and A. Mees.** 1986. Convergence of an annealing algorithm. Math. Prog. **34:**111-124.

74. **Malek, M., M. Guruswamy, P. M, and O. H.** 1989. Serial and parallel simulated annealing and tabu search algorithms for travelling salesman problem. Annals of Operations Research **21:**59-84.

75. **Maniezzo, V., and A. Mingozzi.** 1999. The project scheduling problem with irregular starting time costs. Operational Research Letters **25:**175-182.

76. **Marin, A., and J. Salmeron.** 1996. A simulated annealing approach to the Railroad Freight Transportation design Problem. International. Transaction of Operational. Research. **3:**139-149.

77. **Mehrez, A., M. Hung, and B. Ahn.** 1994. An industrial ocean-cargo shipping problem. Dec. Sci. **26:**395–423.

78. **Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller.** 1953. Equation of state calculation by fast computing machines. The Journal of Chemical Physics **21:**1087-1092.

79. **Mole, R., and S. Jameson.** 1976. A sequential route-building algorithm employing a generalized savings criterion. Operational Research Quaterly **27:**503-511.

80. **Nation, U.** 2005. Review of Maritime Transportation, 2005, United Nations Conference on Trade and Development (UNCTAD).

81. **Nemhauser, G. L., and P. L. Yu.** 1972. A problem in the bulk service scheduling. Operations Research **20:**813-819.

82. **Olson, C. A., E. E. Serenson, and W. J. Sullivan.** 1969. Medium-Range Scheduling for a Freighter Fleet. Operations Research **17:**656-582.

83. **Perakis, A. N., and D. I. Jaramillo.** 1991. Fleet deployment optimization for loner shipping. Part1: background problem formulation and solution approaches. Maritime Policy and Management **18:**183-200.

84. **Pisinger, D.** 2006. Upper bounds and exact algorithms for *p*-dispersion problems. Computers & Operations Research **33:**1380-1398.

85. **Powell, B. J., and A. N. Perakis.** 1997. Fleet deployment optimization for liner shipping: an integer programming model. Maritime Policy and Management **24:**183-192.

86. **Prins, C.** 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research **31:**1985-2002.

87. **Rakshit, A., N. krishnamurthy, and G. Yu.** 1996. system operations advisor: A real-time decision support system for managing airline operations at united airlines. Interfaces **26:**50-58.

88. **Rana, K., and R. G. Vickson.** 1988. A model and Solution Algorithm for optimal Routing of a Time-Chartered Containership. Transportation Science **22:**83-95.

89. **Rana, K., and R. G. Vickson.** 1991. Routing container ships using lagrangian relaxation and decomposition. Transportation Science **25:**201-214.

90. **Rasaratnam, L., S. Nasser, and S. Chelliah.** 2006. Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups. Computers & Operations Research **33:**158-180.

91. **Reeves, C. R. (ed.).** 1995. Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill Book Company Europe.

92. **Ronen, D.** 1983. Cargo ships routing and scheduling: Survey of models and problems. European Journal of Operational Research **12:**119-126.

93. **Ronen, D.** 1986. Short-term Scheduling of vessels for shipping bulk or simi-bulk commodities originating in a single area. Operations Research **34:**164-173.

94. **Ronen, D.** 1993. Ship scheduling: The last decade. European Journal of Operational Research **71:**325-333.

95. **Ronen, D.** 2000. scheduling charter aircraft. Journal of the Operational Research Society **51:**258-262.

96. **Ronen, D.** 2002. Marine inventory routing: shipments planning. Journal of the Operational Research Society **53:**108-114.

97. **Russell, R. A., and W.-C. Chiang.** 2006. Scatter search for the vehicle routing problem with time windows. European Journal of Operational Research **169:**606-622.

98. **Salhi, S.** 2002. Defining tabu search list size and asperation criterion within tabu search methods. Computers & Operations Research **29:**67-86.

99. **Sherali, H. D., S. M. Al-Yakoop, and M. M. Hassan.** 1999. Fleet management models and algorithms for oil-tanker routing and scheduling problem. IIE Transactions **31:**395-406.

100. **Shintani, K., A. Imai, E. Nishimura, and S. Papadimitriou.** in press. The container shipping network design problem with empty container repostioning. Transportation Research Part E.

101. **Skorin-Kapov, D., J. Skorin-Kapov, and M. O'Kelly.** 1996. Tight Linear Programming Relaxations of P-Hub Median problems. European Journal of Operational Research **94:**582-593.

102. **Skorin-Kapov, J., and J.-F. Labourdette.** 1995. On minimum Congestion in Logically Rearrangeable Multihop Lightwave Networks. Journal of Heuristics **1:**129-146.

103. **Skorin-Kapov, J., and A. Vakharia.** 1993. Scheduling a Flow-Line Manufacturing Cell: A Tabu Search Approach. International Journal of Production Research **31:**1721-1734.

104. **Solomon, M. M.** 1987. Algorithms for vehicle routing and scheduling problems with time window constraints. Operational Research **35:**254-265.

105. **Subramanian, R., and R. E. Marsten.** 1993. Strategic planning and scheduling of aircraft heavy maintenance at Delta airlines. AGIFORS 33rd Annual Symposium Proceedings, Chicago, IL.

106. **Subramanian, R., R. Scheff, J. Quillinan, D. Wiper, and R. Marsten.** 1994. Coldstart : Fleet assignment at Delta Air Lines. Interfaces **24:**104-120.

107. **Sun, M., J. E. Aronson, P. G. McKeown, and D. Drinka.** 1998. A tabu search heuristic procedure for the fixed charge transportation problem. European Journal of Operational Research **106:**441-456.

108. **Taillard, E.** 1991. Robust tabu search for the quadratic assignment problem. Parallel Computing **17:**443-455.

109. **Taillard, E. D.** 1993. Parallel iterative search methods for vehicle routing problems. Networks **23:**661-672.

110. **Taillard, E. D.** 1994. Parallel Tabu Search Techniques for the Job Shop Scheduling Problem. ORSA Journal on Computing **6:**108-117.

111. **Tan, K. C., L. H. Lee, and K. Ou.** 2001. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. Artificial Intelligence **14:**825-837.

112. **Tan, K. C., L. H. Lee, Q. L. Zhu, and K. Ou.** 2001. Heuristic methods for vehicle routing problem with time windows. Artificial Intelligence in Engineering **15:**281-295.

113. **Tang, L., G. Wang, and J. Liu.** Article in press. A branch-and-price algorithm to solve the molten iron allocation problem in iron and steel industry. Computers & Operations Research.

114. **Taniguchi, E., and H. Shimamoto.** 2004. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. Transportation Research Part C **12:**235-250.

115. **Teodorovic, D., and S. Guberinic.** 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. European Journal of Operational Research **15:**178-182.

116. **Teodorovic, D., and G. Stojkovic.** 1990. Model for operational daily airline scheduling. Transportation Planning and Techniligy **14:**273-285.

117. **Thesen, A.** 1998. design and evaluation of Tabu Search Algorithms for Multiprocessor scheduling. Journal of Heuristics **4:**141-160.

118. **Toth, P., and Daniele.** 2003. The granular tabu search and its application to the vehicle routing problem. INFORMS Journal on Computing **15:**333-346.

119. **Tozkapan, A., O. Kirca, and C.-S. Chung.** 2003. A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem. Computers & Operations Research **30:**309-320.

120. **Vansteenwegen, P., and D. V. Oudheusden.** 2006. Developing railway timetables which guarantee a better service. European Journal of Operational Research **173:**337-350.

121. **Winston, W. L.** 1993. Operations Research: Applications and Algorithms, Third ed. International Thomson Publishing, California.

122. **Wu, T.-H., C. Low, and J.-W. Bai.** 2002. Heuristic solutions to multi-depot location-routing problems. Computers & Operations Research **29:**1393-1415.

123. **Yau, C.** 1988. A heuristic Method for Scheduling of Trucks Many Warehouses to Many Delivery points. Computers in Industry **11:**175-180.

124. **Zamani, M. R.** 2001. A high-performance exact method for the resource-constrained project scheduling problem. Computers & Operations Research **28:**1387-1401.

125. **Zhang, W., and R. E. Korf.** 1996. A study of complexity transitions on the asymmetric traveling salesman problem. Artificial Intelligence **81:**223-239.

126. **Zhou, X., and M. Zhong.** 2005. Bicriteria train scheduling for high-speed passenger railroad planning applications. European Journal of Operational Research **167:**752-771.

127. **Zwaneveld, P. J., L. G. Kroon, and S. P. M. v. Hoesel.** 2001. Routing trains through a railway station based on a node packing model. European Journal of Operational Research **128:**14-33.

Appendix A

A.1 SPP model using MPL.

```
TITLE
   Ship_scheduling ;

DATA
   MaxShip  := DATAFILE("ShipNumber.dat");
   MaxSched := DATAFILE("SchedNumber.dat");
   MaxCargo := DATAFILE("CargoNumber.dat");

INDEX
   ship       := 1..MaxShip ;
   schedule          := 1..MaxSched ;
   cargo     := 1..MaxCargo ;
DATA
   Schedules[ship,schedule,cargo]:=       SPARSEFILE("SCHEDULE.dat");
   cost[ship ,schedule]:=              SPARSEFILE("Costt.dat");

DECISION
  x[ship ,schedule] WHERE ( cost> 0)

MODEL
   MIN TotalCost = SUM (ship,schedule: x * cost) ;

SUBJECT TO
  condition1 [cargo] :  SUM (ship ,schedule: Schedules * x ) = 1;
   condition2 [ship] :  SUM (schedule: x )<= 1
BINARY
    x;
END
```

A.2 Generation Feasible Schedules Model

The method of using Generation Feasible Schedules model is provided in the following file name (CD provided):

ColumnGenerMulti.

The steps are as follows:

1- Open shipMulti.vbp (Visual Basic Project) icon.
2- Click on Run (Tools Box).
3- Ship Scheduling interface will appear.
4- Click on "Time Horizon, Ships & Cargoes" button.
5- Specify Time Horizon, number of Ships (controlled), number of Chartered Ship, and finally number of cargoes.
6- Once generation finishes, small icon with total number will display.
7- Go to MPL software and run the model.

A.3 Tabu Search Design Model

The method of using TS model is provided in the following file name (CD provided):

Tabu Search Multi.

The steps are as follows:

1- Open shipMulti.vbp (Visual Basic Project) icon.
2- Click on Run (Tools Box).
3- Ship Scheduling interface will appear
4- Click on "Time Horizon, Ships & Cargoes" button.
5- Enter Time Horizon, number of Ships (controlled), number of Chartered Ship, and finally number of cargoes (must be twenty cargoes and over).
6- Excel window will be opened and display initial solution.
7- Return to Ship Scheduling interface.
8- Click on "Tabu Insert & Swap" button.
9- Excel window will display the Tabu Search solution.