

**MATHEMATICAL FORMULATION OF TABU SEARCH IN  
COMBINATORIAL OPTIMIZATION**

**ZUHAIMY BIN ISMAIL**

**RESEARCH VOTE NO :  
78146**

**JABATAN MATEMATIK  
FAKULTI SAINS  
UNIVERSITI TEKNOLOGI MALAYSIA**

**2009**

UNIVERSITI TEKNOLOGI MALAYSIA

**BORANG PENGESAHAN  
LAPORAN AKHIR PENYELIDIKAN**

TAJUK PROJEK : Mathematical Formulation of Tabu Search in Combinatorial Optimization

Saya ZUHAIMY ISMAIL  
**(HURUF BESAR)**

Mengaku membenarkan **Laporan Akhir Penyelidikan** ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :

1. Laporan Akhir Penyelidikan ini adalah hakmilik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan rujukan sahaja.
3. Perpustakaan dibenarkan membuat penjualan salinan Laporan Akhir Penyelidikan ini bagi kategori TIDAK TERHAD.

4. \* Sila tandakan ( / )

SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau Kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972).

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh Organisasi/badan di mana penyelidikan dijalankan).

TIDAK  
TERHAD

\_\_\_\_\_  
TANDATANGANKETUAPENYELIDIK

**PROF.DR. ZUHAIMY ISMAIL**

Nama & Cop Ketua Penyelidik

Tarikh : \_\_\_\_\_

**CATATAN :** \*Jika Laporan Akhir Penyelidikan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan ini perlu dikelaskan sebagai SULIT dan TERHAD.

MATHEMATICAL FORMULATION OF TABU SEARCH IN  
COMBINATORIAL OPTIMIZATION

ZUHAIMY BIN ISMAIL

RESEARCH VOTE NO :  
78146

JABATAN MATEMATIK  
FAKULTI SAINS  
UNIVERSITI TEKNOLOGI MALAYSIA

2009

## **PREFACE**

This research report entitled Mathematical Formulation of Tabu Search in Combinatorial Optimization as been prepared by Professor Dr. Zuhaimy Hj. Ismail during the period January 2007 to February 2009 at the Department of Mathematic, University Technology Malaysia, Skudai Johor.

This report is submitted as the requirement for the completion of e-science research project which is fully supported by The Ministry of Science, Technology and Innovation (MOSTI). The subject of this report is to study the mathematical formulation of Tabu Search in combinatorial optimization. Model used are the CARP for solid waste collection problem. I would like to express my gratitude to MOSTI for their trust in our expertise and interest in this project.

Next, I would like to thank The Research Management Centre, Universiti Teknologi Malaysia for all the support and services provided in making this research a success. I would like to thank The Department of Mathematics and Faculty of Science for making this research a success.

I would like to thank Mr. Ahmad Kamel Ismail from the SWM Environment Sdn Bhd (formerly known as Southern Waste Management Sdn Bhd) and the management of Perniagaan Zawiyah Sdn Bhd for our collaboration on the data collection and for providing the guidelines on the processes in solid waste collection and distribution practices.

I also wish to thank my students and colleagues Irhamah, Khairil Asmani, Dr. Zaitul Marlizawati for their contribution and devotion in this work. Also. The Staff-the academic as well as the administrative-at the Department of Mathematic, Faculty of Science, UTM.

Prof. Dr. Zuhaimy Ismail  
ISDAG and ISORG, Fakulti Sains

## **ABSTRACT**

The Capacitated Arc Routing Problem (CARP) is a fundamental and well-known routing problem. This is a special form of arc routing problem which involves determining a fleet of homogeneous size vehicles and designing the routes to minimize the total cost. It is considered as CARP when the demands are located along the edges. One such problem is in the designing a tour for waste collection vehicle where each vehicle is limited in its capacity. CARP is known to be Non-deterministic Polynomial-time hard (NP-hard) where solutions are obtained through heuristic methods. Tabu Search (TS) is a heuristic method based on the use of prohibition-based techniques and basic heuristics algorithms like local search. The main advantage of TS with respect to other conventional search is in the intelligent use of past history of the search to influence its future search procedures. This study is to develop Reactive Tabu Search (RTS) heuristics for solving CARP. Our RTS algorithm allows for dynamic tabu list rather than static tabu list as being practiced in TS algorithm. The test instances involve five to 50 nodes systematically generated similar to the real world CARP. The newly modified RTS algorithm gives a better performance than TS and (Look-Ahead Strategy) LAS method.

## ABSTRAK

Masalah Perjalanan Lengkok Berkapasiti (MPLB) adalah satu masalah perjalanan yang asas dan terkenal. Ia merupakan satu masalah khas daripada masalah perjalanan lengkok yang melibatkan penentuan kenderaan bersaiz homogen dalam membina laluan-laluan yang dapat mengurangkan jumlah kos. Masalah ini dianggap sebagai MPLB sekiranya permintaan diletakkan di sepanjang lengkok. Salah satu masalah seumpama ini adalah dalam merekabentuk perjalanan bagi kenderaan pemungut sampah dengan setiap satu kenderaan mempunyai kapasiti yang terhad. MPLB diketahui menjadi Bukan berketentuan Polinomial-masa yang sukar (BP sukar), di mana kebanyakan masalah diselesaikan menggunakan kaedah heuristik. Carian Tabu (CT) merupakan kaedah heuristik berasaskan kepada penggunaan teknik-teknik larangan dan penggunaan algoritma heuristik yang asas seperti pencarian setempat. Kelebihan utama CT terhadap kebiasaan carian lain adalah penggunaan kepintarannya di dalam pencarian yang lepas untuk proses pencarian seterusnya. Kajian ini adalah untuk membangunkan kaedah heuristik Carian Tabu Reaktif (CTR) bagi menyelesaikan MPLB. Algoritma CTR membenarkan penggunaan senarai tabu yang dinamik berbanding hanya senarai tabu yang statik seperti yang dipraktikkan di dalam algoritma CT. Pengujian dilakukan ke atas lima hingga 50 titik pertemuan yang dijana secara sistematik dan menyerupai kepada MPLB sebenar. Pengubahsuaian algoritma yang baru memberikan pencapaian yang baik berbanding kaedah CT dan Strategi Pandang Depan (SPD).

## TABLE OF CONTENTS

| CHAPTER  | TITLE   | PAGE        |
|----------|---|-------------|
|          | <b>TITLE PAGE</b>   |             |
|          | <b>PREFACE</b>  | <b>i</b>    |
|          | <b>ABSTRACT</b>   | <b>ii</b>   |
|          | <b>ABSTRAK</b>  | <b>iii</b>  |
|          | <b>TABLE OF CONTENTS</b>                                  | <b>iv</b>   |
|          | <b>LIST OF TABLES</b>                                     | <b>vii</b>  |
|          | <b>LIST OF FIGURES</b>                                    | <b>viii</b> |
|          | <b>LIST OF ABBREVIATIONS</b>                              | <b>x</b>    |
|          | <b>LIST OF SYMBOLS</b>                                    | <b>xii</b>  |
|          | <b>LIST OF APPENDICES</b>                                 | <b>xiii</b> |
| <b>1</b> | <b>INTRODUCTION</b>                                       | <b>1</b>    |
|          | 1.1 Introduction  | 1           |
|          | 1.2 Problem Background                                    | 2           |
|          | 1.3 Problem Statement                                     | 3           |
|          | 1.4 Objective of the Study                                | 4           |
|          | 1.5 Scope of the Study                                    | 5           |
|          | 1.6 Significant of the Study                              | 5           |
|          | 1.7 Thesis Layout   | 6           |
| <b>2</b> | <b>LITERATURE REVIEW</b>                                  | <b>8</b>    |
|          | 2.1 Introduction  | 8           |
|          | 2.2 Capacitated Arc Routing Problem                       | 9           |
|          | 2.2.1 Real World application                              | 12          |
|          | 2.3 Metaheuristics  | 13          |
|          | 2.4 Tabu Search   | 15          |
|          | 2.4.1 Reactive Tabu Search                                | 19          |
|          | 2.4.2 Comparison: Tabu Search and Reactive Tabu Search    | 20          |
|          | 2.5 Recent Works on the Capacitated Arc Routing Problem   | 22          |
|          | 2.6 Review Solution Method on Waste Collection Management | 28          |

|          |  |           |
|----------|--|-----------|
|          | 2.7 Recent Works on Reactive Tabu Search   | 29        |
|          | 2.8 Summary  | 32        |
| <b>3</b> | <b>RESEARCH METHODOLOGY</b>  | <b>33</b> |
|          | 3.1 Introduction   | 33        |
|          | 3.2 Research Methodology   | 34        |
|          | 3.3 Data Source  | 35        |
|          | 3.4 Terminologies in Tabu Search   | 36        |
|          | 3.5 A Basic Tabu Search Procedure  | 38        |
|          | 3.5.1 The Initialization   | 39        |
|          | 3.5.2 The Forbidding Strategy  | 40        |
|          | 3.5.3 The Freeing Strategy   | 41        |
|          | 3.5.4 The Stopping Criterion   | 41        |
|          | 3.5.5 The Diversification Strategy   | 42        |
|          | 3.6 The Reactive Tabu Scheme   | 42        |
|          | 3.7 Tabu Search Implementation   | 43        |
|          | 3.7.1 The Initial Solution   | 43        |
|          | 3.7.2 Element of Tabu Search   | 46        |
|          | 3.7.3 Element of Reactive Tabu Search  | 48        |
|          | 3.8 The General Reactive Tabu Search Algorithm   | 49        |
|          | 3.9 Summary  | 50        |
| <b>4</b> | <b>LOOK-AHEAD STRATEGY AND TABU SEARCH FOR SOLVING CAPACITATED ARC ROUTING PROBLEM</b> | <b>51</b> |
|          | 4.1 Introduction   | 51        |
|          | 4.2 Look-Ahead Strategy for Capacitated Arc Routing Problem                            | 52        |
|          | 4.2.1 Basic Idea   | 52        |
|          | 4.2.2 Look-Ahead Strategy Algorithm  | 53        |
|          | 4.3 Tabu Search for Capacitated Arc Routing Problem                                    | 55        |
|          | 4.3.1 Initial Solution   | 55        |
|          | 4.3.2 Tabu List Size   | 57        |
|          | 4.3.3 Tabu Moves   | 60        |
|          | 4.3.4 Aspiration Criterion   | 61        |
|          | 4.3.5 Stopping Criterion   | 62        |
|          | 4.3.6 Tabu Search Algorithm  | 63        |
|          | 4.4 Computational Results  | 67        |
|          | 4.4.1 Look-Ahead Strategy Computational Results  | 68        |
|          | 4.4.2 Tabu Search Computational Results  | 69        |
|          | 4.4.3 Look-Ahead Strategy versus Tabu Search   | 71        |



|          |   |            |
|----------|---|------------|
|          | 4.5 Summary   | 73         |
| <b>5</b> | <b>SOLUTION APPROACH BASED ON<br/>REACTIVE TABU SEARCH</b>        | <b>74</b>  |
|          | 5.1 Introduction  | 74         |
|          | 5.2 Reactive Tabu Search Implementation                           | 75         |
|          | 5.2.1 Tabu List Size  | 76         |
|          | 5.2.2 Diversification Strategy                                    | 84         |
|          | 5.2.3 Stopping Criterion  | 88         |
|          | 5.3 Reactive Tabu Search Algorithm                                | 90         |
|          | 5.4 Computational Results   | 93         |
|          | 5.5 Summary   | 96         |
| <b>6</b> | <b>SYSTEM DEVELOPMENT FOR<br/>CAPACITATED ARC ROUTING PROBLEM</b> | <b>97</b>  |
|          | 6.1 Introduction  | 97         |
|          | 6.2 Programming with Microsoft Visual Studio                      | 98         |
|          | 6.2.1 The Visual Studio Application                               | 98         |
|          | 6.3 Waste Collection Management Computational Module              | 100        |
|          | 6.4 Program Visualization; Graphical User Interface               | 101        |
|          | 6.5 Managing the Capacitated Arc Routing Problem Program          | 105        |
|          | 6.6 Summary   | 109        |
| <b>7</b> | <b>ANALYSIS OF RESULTS, CONCLUSION<br/>AND RECOMMENDATION</b>     | <b>110</b> |
|          | 7.1 Introduction  | 110        |
|          | 7.2 Analysis of Results   | 111        |
|          | 7.3 Conclusion  | 113        |
|          | 7.4 Recommendation  | 114        |
|          | 7.5 Future Problems: The Extension of This Problem                | 115        |
|          | <b>REFERENCES</b>   | <b>116</b> |
|          | <b>Appendices A - E</b>   | <b>121</b> |

## LIST OF TABLES

| TABLE NO. | TITLE  | PAGE |
|-----------|--|------|
| 2.1       | TS Applications                                    | 18   |
| 2.2       | Comparison between TS and RTS                      | 21   |
| 2.3       | Recent Work on CARP                                | 23   |
| 2.4       | Other Solution Method for Waste Collection Problem | 28   |
| 3.1       | Terminologies in TS                                | 37   |
| 4.1       | Investigation on Tabu List Size                    | 58   |
| 4.2       | Stopping Criterion                                 | 62   |
| 4.3       | Type of Problem                                    | 67   |
| 4.4       | LAS Computational Results                          | 68   |
| 4.5       | TS Computational Results                           | 69   |
| 4.6       | Comparison between LAS and TS                      | 72   |
| 5.1       | The Differences between TS and RTS                 | 75   |
| 5.2(a)    | Investigation on Tabu List Size and Repetition     | 77   |
| 5.2(b)    | Investigation on Tabu List Size and Repetition     | 78   |
| 5.3       | Tabu List Size                                     | 82   |
| 5.4       | Investigation on Dynamic Tabu List Size            | 83   |
| 5.5       | Repetition   | 84   |
| 5.6       | Investigation on Diversification Strategy          | 86   |
| 5.7       | Summary on Diversification Strategy                | 88   |
| 5.8       | Investigation on Maximum Iteration                 | 89   |
| 5.9       | RTS Computational Results                          | 94   |
| 5.10      | Average Percentage of Improvement                  | 95   |
| 7.1       | Comparison between TS and RTS                      | 111  |
| 7.2       | Advantages and Disadvantages of TS and RTS         | 113  |

## LIST OF FIGURES

| FIGURE NO. | TITLE  | PAGE |
|------------|--|------|
| 2.1        | Graphical Network  | 10   |
| 3.1        | Research Methodology   | 35   |
| 3.2        | Network Model  | 44   |
| 3.3        | 2-opt Move   | 47   |
| 3.4        | Flow Chart of the Entire Procedure in Developing RTS                           | 50   |
| 4.1        | Flowchart of LAS   | 54   |
| 4.2        | Investigation on Tabu List Size  | 59   |
| 4.3        | 2-opt Move   | 60   |
| 4.4        | The TS Algorithm   | 65   |
| 4.5        | Comparison between IS and TS   | 70   |
| 4.6        | Percentage of Improvement for TS Computational Results                         | 71   |
| 4.7        | Comparison between LAS and TS  | 72   |
| 5.1        | Comparison on Tabu List Size and Repetition for                                | 80   |
| 5.2        | Comparison on Tabu List Size and Repetition for                                | 81   |
| 5.3        | Illustration on Diversification Strategy                                       | 85   |
| 5.4        | Comparison on a Result between Diversify and Not Diversify                     | 87   |
| 5.5        | Comparison on Percentage of Improvement between Diversify<br>and Not Diversify | 87   |
| 5.6        | The RTS Algorithm  | 91   |
| 5.7        | Comparison between IS and RTS  | 94   |
| 5.8        | Percentage of Improvement for RTS Computational Result                         | 95   |
| 6.1        | Basic IDE  | 99   |
| 6.2        | Welcome GUI  | 102  |
| 6.3        | GUI of the Program   | 103  |
| 6.4        | Features in Menu Bar   | 104  |
| 6.5        | Information Menu   | 105  |

|      |   |     |
|------|---|-----|
| 6.6  | About Menu  | 105 |
| 6.7  | Form for Insert Demand and Cost                         | 106 |
| 6.8  | Example of Complete Network in Drawing Region           | 107 |
| 6.9  | Form for Maximum Capacity of the Vehicle                | 107 |
| 6.10 | Example of the Initial Solution Produced by the Program | 108 |
| 6.11 | Message Box of Computational Time                       | 108 |
| 6.12 | Example of the Result Produced by the Program           | 108 |
| 7.1  | Comparison between TS and RTS                           | 112 |

## LIST OF ABBREVIATIONS

|        |   |  |
|--------|---|--|
| ACO    | - | Ant Colony Optimization                                |
| AI     | - | Arbitrary Insertion                                    |
| ANN    | - | Artificial Neural Network                              |
| CARP   | - | Capacitated Arc Routing Problem                        |
| CARPB  | - | Capacitated Arc Routing Problem with Backhauls         |
| CARPRP | - | Capacitated Arc Routing Problem with Refill Point      |
| CARPSD | - | Capacitated Arc Routing Problem with Stochastic Demand |
| CARPTW | - | Capacitated Arc Routing Problem with Time Window       |
| CEM    | - | Cheapest Edge Method                                   |
| CI     | - | Cheapest Insertion                                     |
| CVRP   | - | Capacitated Vehicle Routing Problem                    |
| DM     | - | Descent Method   |
| ESO    | - | Evolutionary Simulation Optimization                   |
| FI     | - | Farthest Insertion                                     |
| GA     | - | Genetic Algorithm                                      |
| GI     | - | General Insertion                                      |
| GP     | - | Grey Programming                                       |
| IDE    | - | Integrated Development Environment                     |
| IFLP   | - | Interval parameter Fuzzy Linear Programming            |
| IS     | - | Initial Solution                                       |
| LAS    | - | Look-Ahead Strategy                                    |
| LS     | - | Local Search   |
| MDCARP | - | Multiple Depot Capacitated Arc Routing Problem         |
| MDVRP  | - | Multiple Depot Vehicle Routing Problem                 |
| MINLP  | - | Mixed Integer Non-Linear Programming                   |
| MRF    | - | Markov Random Field                                    |
| NI     | - | Nearest Insertion                                      |
| NN     | - | Nearest Neighbour                                      |

|         |   |   |
|---------|---|---|
| NP-hard | - | Non-deterministic Polynomial-time hard            |
| PCARP   | - | Periodic Capacitated Arc Routing Problem          |
| PVRP    | - | Periodic Vehicle Routing Problem                  |
| RTS     | - | Reactive Tabu Search                              |
| SA      | - | Simulated Annealing                               |
| SDVRP   | - | Split Delivery Vehicle Routing Problem            |
| SWM     | - | Southern Waste Management                         |
| TS      | - | Tabu Search                                       |
| TSP     | - | Travelling Salesman Problem                       |
| UAV     | - | Unmanned Aerial Vehicle                           |
| VRP     | - | Vehicle Routing Problem                           |
| VRPB    | - | Vehicle Routing Problem with Backhauls            |
| VRPPD   | - | Vehicle Routing Problem with Pick-up and Delivery |
| VRPSD   | - | Vehicle Routing Problem with Stochastic Demand    |
| VRPTW   | - | Vehicle Routing Problem with Time Window          |
| WCM     | - | Waste Collection Management                       |

## LIST OF SYMBOLS

|               |   |   |
|---------------|---|---|
| $\mu_{ij}$    | - | Customer's demand (Quantity of garbage) |
| $\Omega$      | - | Maximum capacity of the vehicle         |
| $c_{ij}$      | - | Service cost                            |
| $C$           | - | Total cost                              |
| $x_{ij}$      | - | Edge traverse                           |
| $ T_s $       | - | Tabu List Size                          |
| $\delta_{ij}$ | - | demand/cost ratio                       |
| $n$           | - | Number of nodes                         |

## LIST OF APPENDICES

| <b>APPENDIX</b> | <b>TITLE</b>   | <b>PAGE</b> |
|-----------------|--|-------------|
| A               | Some Coding for Initial Solution (Cheapest Edge Method)              | 121         |
| B               | Some Coding for Tabu Search Implementation                           | 126         |
| C               | Some Coding for Reactive Tabu Search Implementation                  | 130         |
| D               | The Installer of Waste Collection Management Computational<br>Module | 137         |
| E               | List of Publications   | 138         |



## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 Introduction**

In recent years, many service suppliers and distributors have recognized the importance of designing efficient transportation strategies in order to improve the level of customer's service and reduce transportation costs. In a typical distribution system, vehicle such as trucks or school buses, provide delivery or customer pick-up, where a common objective is to find a set of routes for the vehicles which satisfies a set of constraints and so as to minimize the total fleet operating costs.

One of the most difficult operation problem faced by local authorities in any large city is the collection of household garbage or industrial waste. This problem is also known as a waste management problem. It is especially crucial for cities in developing countries. From our literature review, many researchers have modelled solid waste collection and distribution problem as Capacitated Arc Routing Problem (CARP) since the garbage had to be carried by vehicle with fixed capacity along the route. Some

of the researchers in this area are Lacomme *et al.* (2001), Amponsah and Salhi (2004), Mourao and Amado (2005), Chu *et al.* (2005) and Bautista *et al.* (2008). This waste collection problem can also be modelled as the Vehicle Routing Problem (VRP), but since the VRP is a node routing problem, it needs a transformation from the CARP into the VRP, which is to transform from an arc routing problem to a node routing problem to make the CARP as a special case of the VRP. Therefore CARP can be considered as a special case of the VRP (Wohlk, 2005).

This CARP will be solved using the proposed methodology which is Tabu Search (TS) and Reactive Tabu Search (RTS). The discussion of the methodology will be discussed in Chapter 3. This chapter provides problem background, problem statement, objective, scopes and significant of the study and ended with a thesis layout.

## **1.2 Problem Background**

Operations research involves “research on operations”. Thus, operations research is applied to problems that concern how to conduct and coordinate the operations (so that everything will be optimize, e.g. cost, time, space, usage etc) within organization in variety of areas such as manufacturing, transportation, construction, telecommunications, financial planning, health care, the military and so on (Hillier & Lieberman, 2005).

Combinatorial optimization is one of optimization problem in applied mathematics and computer science. It is close to operations research, algorithm theory and computational complexity theory that sit at the intersection of several fields,

including artificial intelligence, mathematics and software engineering. Combinatorial optimization algorithms solve instances of problems that are believed to be hard in general by exploring the large solution space of these instances. Most of the real world optimization problems belong to a class of “difficult to solve” problems which are known as Non-deterministic Polynomial-time hard problem, called NP-hard problem. Because of that, the problem cannot be guaranteed to be solved in reasonable time by any known polynomial-time method (Loh, 2007).

Various new optimization techniques had been discovered to solve NP-hard problems more effectively such as Artificial Neural Networks (ANN), Tabu Search (TS), Simulated Annealing (SA), Genetic Algorithm (GA), Ant Colony Optimization (ACO) and some others. This study will explore the use of TS in solving CARP and extend the traditional method of TS into Reactive Tabu Search (RTS). We explore the method of TS and compare the differences between TS and RTS. In this exploration, we use solid waste management as the CARP. The model above will be developing into computer program which is Microsoft Visual Studio 2005 Team Suit (Trial) Edition and it is written in C# language.

### **1.3 Problem Statement**

As the number of household area increases, the solid wastes generated are also increasing. In order to maintain the quality of environment, the waste generated should be properly collected and disposed. For example in Southern Waste Management in the City of Johor Bahru, currently they have a fleet of over 150 collection vehicles operating daily. A well schedule collection of these wastes is essential so that the entire vehicle

will give their optimal service in order to minimize the cost,  $C = \sum_{(i,j) \in E} \sum_{k=1}^K c_{ij} x_{ijk}$  (our proposed model). But some kind of problem arise when to collect these generated wastes. This is due to the limited arc that vehicle have to revisit it again in order to serve all the required edge while we have to minimize revisiting the edge so that the cost of traversal are also minimize. The vehicle capacity,  $Q$  also one of the problem arise because the demand for certain area are not always static. It is possible that the vehicle need to go back to the depot when the services are still on-going because the capacity has reached its limit, where  $\sum_{(i,j) \in R} q_{ij} y_{ijk} \leq Q, \quad k = 1, \dots, K, .$  Therefore, to collect these generated wastes, an appropriate method should be used in order to choose the best route traverse by vehicle so that the costs of collecting are optimal. Hence, this research tries to further the previous study on it by proposing a new metaheuristics algorithm based on RTS. The main issue in RTS is the way tabu list and tabu moves are constructed. The implementation of this algorithm will be done to solve real problem in optimizing solid waste collection.

#### 1.4 Objectives of the Study

The objectives of this study can be summarized as follows:

- i. To develop a new mathematical formulation or RTS algorithm for solving CARP.
- ii. To develop computational module for solving solid waste management problem.

## 1.5 Scope of the Study

The study is confined to solve solid waste collection problem for the city of Johor Bahru. Waste collection is divided into two types which are household waste and industrial waste. The case study will only focus on the household solid waste collection.

The scope of the study can be summarized as follows:

- Service provided by a single vehicle with a limited capacity.
- The vehicle starts and ends at a single depot node.
- Customers are represented by arc where the entire arcs form a complete graph.
- Customer's demand,  $q_{ij}$ , is considered as a non-negative integer.
- The proposed algorithm will be implemented and written using programming language to solve the problem in solid waste collection.

## 1.6 Significance of the Study

The significant of this study may be divided into two main areas. Firstly, the developments of **new RTS formulation for solving CARP**. Secondly, is the development of CARP for solving solid waste collection problem.

This proposed development contributes to the arc routing problem. This is because, the cost of vehicle routing plays an important role when the vehicle servicing an arc. Without planning a tour for the vehicle in order to give their services, the cost may form a maximum travelling cost. Due to this reason, this study is conducted. As

indicated earlier, our focus would be to work on the problem related to CARP for solid waste collection in Johor Bahru.

## **1.7 Thesis Layout**

The thesis is divided into seven chapters. The first chapter is the introduction. This chapter gives an introduction to the background of the problem, the statement of the problem, objectives and scope of the study and significance of the study.

Chapter two is the Literature Review. This chapter presents a literature review about the CARP and solution techniques for solving CARP. The literature about recent works on CARP, RTS and other solution method for solving waste collection problem are also provided in this chapter.

Chapter three is the Research Methodology which is discussed about the terminologies used in the solution technique. This chapter also discussed the development model of the initial solution and also TS and RTS procedure.

Chapter four and five in this thesis consist the discussion of the results. Chapter four is to illustrate the development of other model which is Look-Ahead Strategy (LAS) to be compared with our algorithm. Also provided in chapter four is our TS procedure that also to be used to compare the computational result with our RTS algorithm. Chapter five is our solution technique in order to solve CARP model. This chapter presenting the implementation of RTS and provide a computational results.

Next which is chapter six will be a chapter to write about developing a system. This chapter provides the information on the languages used and how to manage and use the system.

Lastly, this thesis ends up with chapter seven. At the beginning of this chapter, it discusses the comparison results produces by TS and RTS. Then a conclusion and recommendation will close up the whole thesis at the end of this chapter.

## CHAPTER 2

### LITERATURE REVIEWS

#### 2.1 Introduction

This chapter presents a literature review about the Capacitated Arc Routing Problem (CARP), solution techniques appeared in literature and also techniques which may be applied for solving CARP. The first section discusses the description and the mathematical model of CARP. The second section contains the introduction and a basic idea of **Tabu Search** (TS) and **Reactive Tabu Search** (RTS). The following section will discuss about the recent works on the CARP and review of solution method on solid waste collection management in other country. Finally, RTS are discussed overall in the last section.



## 2.2 Capacitated Arc Routing Problem

The CARP was introduced by Golden and Wong (1981), but a variant which is the quantity of demand,  $q_{ij}$  are strictly positive was investigated earlier by Christofides in 1973 (Dror, 2000). It is define on an undirected network in which a fleet of identical vehicles with limited capacity is based at a depot node. The CARP consists of determining a set of feasible vehicle trips that minimizes the total cost of traversed edges. Each trip starts and ends at the depot, each edge is serviced by one single vehicle and the total demand serviced by any trip must not exceed vehicle capacity.

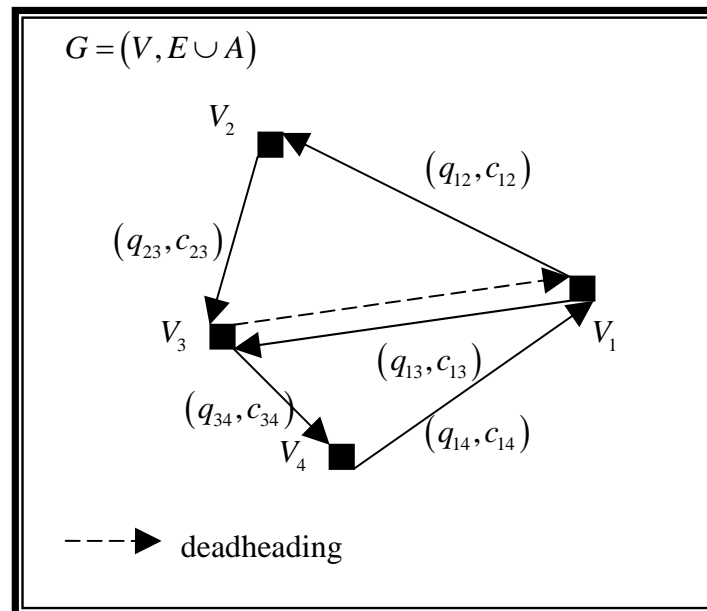
The CARP belongs to the class of problem known as Non-deterministic Polynomial-time hard (NP-hard) as proven by Golden and Wong (1981). NP-hard in computational complexity theory is the complexity class of decision problems that are intrinsically harder than those that can be solved by a nondeterministic Turing machine in polynomial time (Black, 2007). CARP is a problem that has all the characteristics of NP-hard problem as defined by Black (2007). The problem included solving a complex interaction of demands and constraints that requires non-deterministic time to solve.

Various solution procedure can be use to solve CARP such as exact algorithm, approximation algorithm, heuristics and meta-heuristics. However, when a single vehicle is able to service all the required edges, the exact methods have only been able to solve relatively small examples to optimality. While for approximation algorithm, it is designed specifically for one problem type. Therefore meta-heuristics have been proposed. In order to solve this problem, the defining of the mathematical formulation is needed. As a result, CARP formulation is used as appropriate mathematical model for solving the waste collection problem.

In the case of the CARP, the mathematical structure is a graph where each junction is denoted by a point (node) and lines (streets) are drawn connecting two nodes, called arcs or edges. Associated with every line connecting two nodes are quantity of garbage,  $q_{ij}$  and service cost,  $c_{ij}$ . When the vehicle can service every street which has some amount of garbage from a junction to another junction continually (starting and ending at the same designated node, which is the depot), then the graph is said to be complete. The total demand serviced on the route must not exceed the capacity of the vehicle,  $Q$ . When the vehicle travels over an edge without servicing it, this is referred to as deadheading (refer to Figure 2.1). In CARP, each edge in the graph can be travelled in either direction and each vertex corresponds to a road junction.

### Mathematical formulations

Different mathematical formulations have been proposed for the CARP, all of them are undirected case. However, for this problem, we formulated it as follows:



**Figure 2.1: Graphical Network**

Given a connected graph  $G = (V, E \cup A)$ , with  $V$  as the set of nodes (vertices),  $E$  set of edges ( $E \subseteq V \times V$ ) and  $A$  is a set of arcs ( $A \subseteq V \times V$ ). CARP has an additional traversal cost for each edge and arc with edge (arc) demand  $q_{ij} \geq 0$  for each edge  $(i, j)$  which must be serviced by one of a fleet of vehicles of capacity,  $Q$  (Amponsah & Salhi, 2004). The objective of the CARP is to find a minimum cost set of vehicle routes where each required edge is serviced on one of the routes.

We denote  $c_{ij}$  as the serviced cost of an edge (arc)  $(i, j) \in E(A)$  and  $x_{ijk}$  as the number of times edge (arc)  $(i, j) \in E \cup A$  is traversed in trip  $k$ ,

$$y_{ijk} = \begin{cases} 1 & \text{if the edge(arc)}(i, j) \in R \text{ is covered in trip } k, \\ 0 & \text{otherwise} \end{cases}$$

The CARP formulated by Dror and Langevin (Amponsah & Salhi, 2004) is as follows:

$$\text{Min } C = \sum_{(i,j) \in E} \sum_{k=1}^K c_{ij} x_{ijk} \quad (2.1)$$

Subject to:

$$\sum_{p \in V} x_{pik} - \sum_{p \in V} x_{ipk} = 0 \quad \forall i \in V, \quad k = 1, 2, \dots, K, \quad (2.2)$$

$$\sum_{k=1}^K y_{ijk} = 1 \quad \forall (i, j) \in R, \quad (2.3)$$

$$x_{ijk} \geq y_{ijk} \quad \forall (i, j) \in R, \quad k = 1, \dots, K, \quad (2.4)$$

$$\sum_{(i,j) \in R} q_{ij} y_{ijk} \leq Q, \quad k = 1, \dots, K, \quad (2.5)$$

$$\begin{aligned} y_{ijk} &\in \{0, 1\} \quad \forall (i, j) \in R, \quad k = 1, 2, \dots, K, \\ x_{ijk} &\in Q^+ \quad \forall (i, j) \in E, \quad k = 1, 2, \dots, K. \end{aligned} \quad (2.6)$$

Where the variables used in the CARP formulation can be described as follows:

$C$  = total cost.

$c_{ij}$  = service cost of an edge  $(i, j) \in E \cup A$ .

$x_{ijk}$  = number of times edge  $(i, j) \in E \cup A$  is traversed in trip  $k$ .

$y_{ijk} = \begin{cases} 1 & \text{if the edge(arc)}(i, j) \in R \text{ is covered in trip } k, \\ 0 & \text{otherwise} \end{cases}$

$q_{ij}$  = edge demand.

$Q$  = vehicle capacity.

The objective function is given in equation (2.1) seeks to minimize the total cost. While equation (2.2) is to ensure route continuity, and equation (2.3) states that each edge with positive demand is serviced exactly once. To guarantee that the traversal circuit  $k$  covers the edge  $(i, j) \in R$  if it delivers its demand is represent by equation (2.4), while to ensure the vehicle capacity is not violated on account is represented in equation (2.5) and integrality restrictions are given in equation (2.6).

### 2.2.1 Real World Application

CARP has a long and rich history. It can be found in many real world situations where the demand needs to be serving in every single arc with required capacity. Other than waste collection management, road network maintenance can also be modelled as the CARP, where the road markings have to be painted or repainted every year. There are special operational conditions that force the tank truck to return to the depot each time it meets the marking vehicle (Brandao & Eglese, 2008).

Other than that, some well-known examples are postal mail deliveries where the postman needs to deliver the mail in every single arc until no more mail need to be delivered and school bus routing where the bus need to serve the student and send them to school. Same goes to snow clearance, where a snow must be sweep in each road to prevent something bad happen. The truck must pass through the entire arc in order to clean it. Furthermore, this application is not limited to the routing of creatures or goods only. Interesting variants may also be found in industrial manufacturing, e.g. the routing of automatic machines that put conducting layers or component on to a printed circuit board (Greistorfer, 2003).

### **2.3 Metaheuristics**

The term metaheuristic is firstly introduced by Fred Glover, derives from the composition of two Greek words. *Heuristic* derives from the verb *heuriskein* which means “to find”, while the suffix *meta* means “beyond, in an upper level”. Before this term was widely adopted, metaheuristics were often called modern heuristics (Reeves, 1993).

A metaheuristics is a general solution method that provides both a general structure and strategy guidelines for developing a specific heuristic method to fit a particular kind of problem (Hillier & Lieberman, 2005). The most commonly used metaheuristics are Tabu Search (TS), Simulated Annealing (SA), Genetic Algorithms (GA) and Ant Colony Optimization (ACO). Nowadays, metaheuristics are widely used to solve important practical combinatorial optimization problems.

The nature of metaheuristics; it is a general kind of solution method that orchestrates the interaction between local improvement procedures and higher level strategies to create a process that is capable of escaping from local optima and performing a robust search of a feasible region. Thus, one key feature of a metaheuristics is its ability to escape from a local optimum (Hillier & Lieberman, 2005).

The evolution of metaheuristics during the past half dozen years has been widely researched. Metaheuristics in their modern forms are based on a variety of interpretations of what constitutes “intelligent” search. These interpretations lead to design choices which can be used for classification purposes. However, a strict classification of different metaheuristics is difficult, because the leading recommended of alternative methods often differ among themselves. This may be illustrated by considering the classification of metaheuristics in terms of their features with respect to three basic design choices: (1) the use of adaptive memory, (2) the kind of neighbourhood exploration used, and (3) the number of current solutions carried from one iteration to the next (Glover & Laguna, 1997).

Although metaheuristic seems like a very powerful one, but it still have their pro and con. The advantage of metaheuristic of course it provides a very efficient way of dealing with a large complicated problems due to the quickly move toward good solutions. While the disadvantage of using metaheuristic method is that there is no guarantee that the best solution found will be an optimal solution or even a nearly optimal solution.

## 2.4 Tabu Search

The basic form of TS is founded on ideas proposed by Fred Glover (Glover & Laguna, 1997) in 1986 (Salhi, 2002), (Scheuever, 2006). The word tabu (or taboo) comes from Tongan, a language of Polynesia, where it was used by the aborigines of Tonga Island to indicate things that cannot be touched because they are sacred. Based on Webster's Dictionary, the word tabu also means "a prohibition imposed by social custom as a protective measure or of something "banned as constituting a risk". These current more pragmatic senses of the word accord well with the theme of tabu search. The risk to be avoided in this case is that of following a counter-productive course, including one which may lead to entrapment without hope of escape (Wan Ibrahim, 2007).

TS can be applied directly to verbal or symbolic statements to various kinds of decision problems, without the need to transform them into mathematical formulations. Nevertheless, it is useful to introduce mathematical notation to express a broad class of these problems, as a basis for describing certain features of TS. A class of problems of TS can be characterizing as that of optimizing (minimizing or maximizing) a function  $f(x)$  subject to  $x \in \mathbf{X}$ , where  $f(x)$  may be a linear or nonlinear, and the set  $\mathbf{X}$  summarizes constraint on the vector of the decision variables  $x$ . As we know, the combinatorial optimization problem may not be easily formulated as an objective function subject to a set of constraints, so the requirement  $x \in \mathbf{X}$  may specify logical conditions or interconnections that would be cumbersome to formulate mathematically, but may be better be left as verbal stipulations that can be then coded as rules (Glover & Laguna, 1997).

### Mathematical formulations

TS is a mathematical optimization method, belonging to the class of local search technique. TS operate just like a local improvement procedure except that it may not

required that each new trial solution must be better than the preceding trial solution. The process begins by using this procedure as a local improvement procedure in the usual way, which only accepting an improved solution at each iteration to find a local optimum. This search method is extended from the Descent Method (DM) scheme which also known as hill-climbing heuristic or greedy heuristics in various ways. An obvious alternative to a DM is to accept a non-improving move and an employment of strategic memory-based evaluation criterion to escape from a local minimum (Hanafi, 2000). The general procedure of DM can be described as follows:

**[Step 1]**

- Select an initial solution, say  $x \in S$  (where  $S$  is the set of feasible solutions).

**[Step 2]**

- Choose a solution  $x' \in N(x)$  such that  $z(x') < z(x)$  (where  $N(x)$  is the neighbourhood of  $x$ ).
- If there is no such  $x'$ ,  $x$  is considered as a local optimum and the method stops.
- Else set  $x = x'$  and repeat step 2.

The basic procedure in TS does follow the basic concept of DM. A key strategy of TS is that it then continues the search by allowing non-improving moves to the best solutions that can be found in the neighbourhood of the current trial solution, the local improvement procedure is reapplied to find a new local optimum (Hillier & Lieberman, 2005). In other words, the search growth by iteratively moving from one solution to the next solution for improves the solution with the used of its memory.

For more clearly, TS is a search memory method systematically which is TS will not only remember the current and best solution. It also keep memory on the tour through the last solutions visited and such memory will be used with the purpose of



guiding the move from the current to the next solution (Castellani *et al.*, 2007). TS qualify as an intelligent heuristics due to the use of memory together with responsive exploration beyond the solution space. Responsive exploration means that it search aggressively in high quality solutions regions and then breaking away from local optima in order to explore new regions (Lim, 2007).

With this kind of method, the probability of the process to cycle right back to the same local optimum is quite high. So, to avoid this, a TS temporary forbids moves that would return to a solution recently visited. This cycling back to previously visited solutions is prevented by the use of memories, called tabu lists, which record the recent history of the search, a key idea that can be linked to Artificial Intelligence concepts (Gendreau, 2002).

TS also can incorporate some more advanced concept in order to explore another best solution, which are intensification and diversification. Intensification involves exploring a portion of the feasible region more thoroughly than usual after it has been identified as a particularly promising portion for containing very good solutions. While diversification involves forcing the search into previously unexplored areas of the feasible region to make sure that the search trajectory has not been confined to regions containing only mediocre solutions.

Variety field of problems can be solved by using the application of TS. Even sometimes the solutions provided are not close to optimality (due to NP-hard combinatorial optimization problems or problems with complex constraints), but at least the difficulties of the problem can be tackled. The application of TS in variety field of problem can be summarized as shown in Table 2.1 (Glover & Laguna, 1998).

**Table 2.1: TS Applications**

| <b>Field</b>                         | <b>Problems</b>   |
|--------------------------------------|---|
| Scheduling                           | Flow-Time Cell Manufacturing, Heterogeneous Processor Scheduling, Workforce Planning, Classroom Scheduling, Machine Scheduling, Flow Shop Scheduling, Job Shop Scheduling, Sequencing and Batching. |
| Telecommunications                   | Call Routing, Bandwidth Packing, Hub Facility Location, Path Assignment, Network Design for Services, Customer Discount Planning, Failure Immune Architecture, Synchronous Optical Networks.        |
| Design                               | Computer-Added Design, Fault Tolerant Networks, Transport Network Design, Architectural Space Planning, Diagram Coherency, Fixed Charge Network Design, Irregular Cutting Problems.                 |
| Production, Inventory and Investment | Flexible Manufacturing, Just-in-Time Production, Capacitated MRP, Past Selection, Multi-item Inventory Planning, Volume Discount Acquisition, Fixed Mix Investment.                                 |
| Location and Allocation              | Multicommodity Location/Allocation, Quadratic Assignment, Quadratic Semi-Assignment, Multilevel Generalized Assignment, Lay-Out Planning, Off-Shore Oil Exploration.                                |
| Routing                              | Vehicle Routing, Capacitated Routing, Time Window Routing, Multi-Mode Routing, Mixed Fleet Routing, Travelling Salesman, Travelling Purchaser.  |
| Logic and Artificial Intelligence    | Maximum Satisfiability, Probabilistic Logic, Clustering, Pattern Recognition/Classification, Data Integrity, Neural Network   Training and Design.  |
| Graph Optimization                   | Graph Partitioning, Graph Colouring, Clique Partitioning, Maximum Clique Problems, Maximum Planner Graphs, P-Median Problems.   |
| Technology                           | Seismic Inversion, Electrical Power Distribution, Engineering   |

|  |   |
|--|---|
|  | Structural Design, Minimum Volume Ellipsoids, Space Station Construction, Circuit Cell Placement.   |
| General<br>Combination<br>Optimization | Zero-One Programming, Fixed Charge Optimization, Nonconvex Nonlinear Programming, All-or-None Networks, Bi-level Programming, General Mixed Integer Optimization. |

### 2.4.1 Reactive Tabu Search

TS is a meta-strategy that employs computer memory structures to avoid phenomena like local minima and limit cycle. The chosen move is put in the tabu list at each iteration in order to prevent the algorithm going back to recently visited solutions. Therefore, parameter tuning is one of the main drawbacks that need to solve when dealing with TS algorithm. Tuning is often needed to obtain competitive results and requires either a deep knowledge of the problem structure or a time consuming and not always reproducible tinkering process. The most critical parameter usually is the tabu list size, which compromises between intensification and diversification strategies (Castellani *et al.*, 2007).

A fixed size of the tabu list might drive to be trapped in a cycle of length greater than the size list. But if the tabu list size is set at a high value, then the search may be restricted to certain regions, and if it is set to a low value then the search may cycle itself. Therefore, a balanced tabu list size is needed to control and run the process smoothly. Battiti and Tecchiolli developed an approach that dynamically determines the tabu list size during the search process. Their version of TS known as Reactive Tabu Search (RTS) employs two mechanisms which are feedback schemes and escape strategy (Wassan, 2006).

The first mechanism, feedback scheme, builds an automated tabu tenure that is maintained throughout the search process by the dynamic reaction to the repetitions (Wassan, 2006). This means that after one move is executed, the RTS algorithm will check whether the current searching point has already been found. Tabu list size will increase if a searching point is repeated and it will decrease if no repetitions occur during a sufficient long period (Fukuyama, 2000). Since the basic TS cannot avoid long search cycles, therefore the second mechanism which is called escape diversification strategy is also introduced. The escape strategy takes the search process out from its current position if it appears to be repeating itself excessively (Wassan, 2006).

Generally, RTS maintain the basic concept and also terminologies of TS except the tabu list size which are not static as TS. As mentioned before, this modification was introduced by Battiti and Tecchiolli in order to produce the best solution.

#### **2.4.2 Comparison: Tabu Search and Reactive Tabu Search**

There must be a much confusion on what differentiates TS and RTS. This is because the concept and procedure for both of them are actually the same. Hence, Table 2.2 describes more detail about the differences between TS and RTS.

**Table 2.2: Comparison between TS and RTS**

|                      | <b>Tabu Search</b>   | <b>Reactive Tabu Search</b>  |
|----------------------|--|--|
| Move                 | Basic move: swap, combine, shift, perturb, idle and insert.  | Same as TS.  |
| Tabu List Size       | <p>Static changes:</p> <ul style="list-style-type: none"> <li>• Can be any value such as 7, 9 or constant value such as <math> T_s  = \frac{NB}{p}</math>; where <math>\frac{NB}{p}</math> is designed rules.</li> </ul>   | <p>Dynamic changes:</p> <ul style="list-style-type: none"> <li>• Periodically changing: kept fixed for a certain number of iteration and the process is repeated for <math>k</math> times.</li> <li>• Continuously changing: depend on the change in the cost function for that selected move (change when needed).</li> </ul> |
| Aspiration Criterion | <p>i. Aspiration by objective: a tabu move is allowed to be accepted if it leads to a solution that is better than any solution found so far.</p> <p>ii. Aspiration by default (in case all move are tabu):</p> <p>a. Free the “least tabu” move: look at <math> T_s </math>, freeing the min <math> T_s </math>.</p> <p>b. Free the “least cost” move: check the objective function.</p> <p>c. Free the “least order” move: combine a &amp; b.</p> <p>iii. Soft aspiration criteria: used even though there is still non tabu moves. The idea is, the</p> | Same as TS.  |

|                    |  |   |
|--------------------|--|---|
|                    | nearly non tabu move which has produced a solution nearly as good as the best solution may be worth relaxing rather than the first non tabu move with a solution that is far from the best solution.   |   |
| Stopping Criterion | <ul style="list-style-type: none"> <li>• After a fixed number of iteration (a fixed amount of computational time).</li> <li>• After some number of iteration without an improvement in the objective function value.</li> <li>• When the objective reaches a pre-specified threshold value.</li> </ul> | <ul style="list-style-type: none"> <li>• Same as TS.</li> </ul> |

## 2.5 Recent Works on the Capacitated Arc Routing Problem

Generally, there are quite a lot of literatures on the CARP. From the literature, there are various kind of methods have been used for solving CARP. Hence, the summary of main contributions to solve CARP and similar problems is given in Table 2.3.

Table 2.3: Recent Work on CARP

| Authors  | Methods;<br>Problems                                     | Description  |
|--|--|--|
| Philippe Lacomme, Christian Prins & Wahiba Ramdane-Cherif (2001) | Genetic Algorithm [CARP]; for Municipal Waste Collection | <ul style="list-style-type: none"> <li>• Chromosomes and fitness: <ul style="list-style-type: none"> <li>▪ Evaluate chromosomes by built an auxiliary graph which each arc denotes a subsequence of <math>T</math> that can be done by one trip.</li> <li>▪ The fitness is simply the total cost of the underlying CARP solution.</li> </ul> </li> <li>• Reproduction step: <ul style="list-style-type: none"> <li>▪ Choose parents by <i>binary tournament selection</i>.</li> <li>▪ Reproduction step ends by randomly keeping only one child, <math>C</math> and discarding the other.</li> </ul> </li> <li>• Local search and mutation operator: <ul style="list-style-type: none"> <li>▪ Mutate with a fixed rate <math>pm</math> the child <math>C</math> produced by the crossover; <math>pm</math> = rate for mutation</li> <li>▪ The mutation operator is a <i>local search LS</i>, giving a <i>hybrid GA</i>.</li> </ul> </li> <li>• Stopping criteria: <ul style="list-style-type: none"> <li>▪ Stops after a maximum number of iterations, or</li> <li>▪ After a maximal number of unproductive iterations, or</li> <li>▪ When it reaches a lower bound known for some instances.</li> </ul> </li> </ul> |
| Peter Greistorfer (2003)   | Tabu Scatter Search                                      | <ul style="list-style-type: none"> <li>• Used compound neighbourhood to perform a transition from an old solution to a new one.</li> </ul>   |

|  |  |   |
|--|--|---|
|  | [CARP]; for Special Logistical Problem                 | <ul style="list-style-type: none"> <li>• Used intensification and diversification search phases (long-term TS memory).</li> <li>• The admissibility is monitored by the short-term TS memory.</li> </ul>  |
| Patrick Beullens,<br>Luc Muyldermans,<br>Dirk Cattrysse &<br>Dirk Van Oudheusden<br>(2003) | Guided Local Search Heuristic [CARP]; for general CARP | <ul style="list-style-type: none"> <li>• Neighbourhood moves (based on two types of neighbourhood move): <ul style="list-style-type: none"> <li>▪ Single vehicle moves (flip, reverse and dir-opt).</li> <li>▪ Moves between two routes (relocate, exchange and cross).</li> </ul> </li> <li>• Generates the CARP local search algorithm as indicates by the Boolean variable changed.</li> </ul>   |
| Sanne Wohlk<br>(2003)  | Simulated Annealing [CARP]; for General CARP           | <ul style="list-style-type: none"> <li>• Defining neighbourhood as: <ul style="list-style-type: none"> <li>▪ Define <math>\mu_i</math> and <math>\mu_j</math> as being neighbours if <math>\mu_i</math> equals <math>\mu_j</math> except for two element in <math>\mu_i</math> which are swapped; where <math>\mu_i</math> and <math>\mu_j</math> as a label for it neighbours.</li> <li>▪ Pick at random iteratively a neighbour solution, which is immediately accepted if the cost is lower than the current cost.</li> <li>▪ If the cost is larger, the possibility of acceptance depends on a value of temperature, such that higher temperature means higher acceptance probability.</li> </ul> </li> </ul> |
| Jose M. Belenguer &<br>Enrique Benavent (2003)   | Cutting Plane Algorithm [CARP]; for General CARP       | <ul style="list-style-type: none"> <li>• Use aggregated variables to formulate the CARP and introduce new classes of valid constraints.</li> <li>• Implement several procedures to identify constraints which are violated by the current LP solution.</li> </ul>   |



|   |   |  |
|---|---|--|
|   |   | <ul style="list-style-type: none"> <li>• Also develop a cutting plane algorithm to compute a lower bound for the CARP.</li> </ul>  |
| Amponsah, S.K. & Salhi, S. (2004)         | Look-ahead Strategy [CARP]; for Solving the Collection of Garbage | <ul style="list-style-type: none"> <li>• The idea is to examine the total demand/cost ratio on all possible temporary edges with respect to their likelihood to yield future advantage to prune away unpromising edges in the collection process and to choose edges that are most promising.</li> <li>• The algorithm proceeds from one junction (node) to one of its adjacent nodes at each stage.</li> <li>• If the quantity of garbage in a particular trip is more than or equal to a critical value <math>Q</math>, then the look-ahead strategy will shift to the least insertion cost rule.</li> </ul>   |
| Maria Candida Mourao & Ligia Amado (2005) | Heuristics [mixed CARP]; for a Refuse Collection Application      | <ul style="list-style-type: none"> <li>• Based on the Eulerian and directed network. <ul style="list-style-type: none"> <li>▪ Starts by identifying the minimum demand circuit incident to each node and from this circuit, the biggest one is chosen.</li> <li>▪ Generating a trips multigraph in second phase where it is possible to easily identify the best aggregation.</li> <li>▪ In each iteration, choose the pairs of trips to be joined together according to a matching solution in the trips multigraph, where one seeks to maximize the total savings.</li> </ul> </li> <li>• This heuristic method produces a feasible solution on a directed graph.</li> </ul> |
| Feng Chu,                                 | Heuristics  | <ul style="list-style-type: none"> <li>• Based on two insertion methods and two</li> </ul>   |

|  |  |  |
|--|--|--|
| <p>Nacima Labadi &amp; Christian Prins (2005)</p>                        | <p>[periodic CARP]; for Waste Collection</p>                                       | <p>phase algorithm.</p> <ul style="list-style-type: none"> <li>• Two insertion methods: <ul style="list-style-type: none"> <li>▪ Decreasing frequencies.</li> <li>▪ Nearest insertion heuristics.</li> </ul> </li> <li>• Two phase algorithm: <ul style="list-style-type: none"> <li>▪ <i>Lower Bound Heuristic (LBH)</i>. The first phase is guided by a lower bound to prepare a cluster of tasks for each day.</li> <li>▪ The actual trips are built in the second phase that consists of solving the single-period CARP defined by each cluster using the hybrid algorithm.</li> </ul> </li> </ul> |
| <p>Humberto Longo, Marcus Poggi de Aragao &amp; Eduardo Uchoa (2006)</p> | <p>Transformation to the CVRP [CARP]; for General CARP</p>                         | <ul style="list-style-type: none"> <li>• Transform CARP to the CVRP and solve using CVRP formulation.</li> </ul>   |
| <p>Alberto Amaya, Andre Langevin &amp; Martin Trepanier (2007)</p>       | <p>Integer Linear Programming [CARP]; Application for Road Network Maintenance</p> | <ul style="list-style-type: none"> <li>• Based on the formulation of a classical cutting plane approach by Lacomme.</li> <li>• Implement two phases of strategy: <ul style="list-style-type: none"> <li>▪ Phase 1: Finding connectivity constraints. Finding lower bound if iteration finished before visiting the 5000 nodes.</li> <li>▪ Phase 2: Finding an upper bound. Only added the components that include the depot.</li> </ul> </li> <li>• If the solution found is equal to a known lower bound, the solution is optimal.</li> </ul>   |
| <p>Jose Brandao &amp; Richard Eglese</p>                                 | <p>Tabu Search [CARP]; for</p>   | <ul style="list-style-type: none"> <li>• Neighbourhood moves (based on three types of neighbourhood move):</li> </ul>  |

|  |  |  |
|--|--|--|
| (2008)   | General CARP   | <ul style="list-style-type: none"> <li>▪ Single insertion.</li> <li>▪ Double insertion.</li> <li>▪ Swap.</li> <li>• Admissibility of moves: <ul style="list-style-type: none"> <li>▪ Tabu list is fixed.</li> <li>▪ Set to <math>N/2</math> in Phase 1 and <math>N/6</math> in Phase 2 after some experimentation in TSA; <math>N</math> =number of required edges.</li> </ul> </li> <li>• Aspiration Criterion: <ul style="list-style-type: none"> <li>▪ Tabu restriction maybe overridden if the move will produce a solution that is better than what has been found in the past (aspiration criteria).</li> </ul> </li> </ul>  |
| Joaquin Bautista, Elena Fernandez & Jordi Pereira (2008) | Ant Heuristics [CARP]; for Solving an Urban Waste Collection | <ul style="list-style-type: none"> <li>• Use two different constructive greedy heuristics: <ul style="list-style-type: none"> <li>▪ Nearest neighbour method.</li> <li>▪ Nearest insertion heuristic.</li> </ul> </li> <li>• Consider several neighbourhood for a local search: <ul style="list-style-type: none"> <li>▪ Substitution neighbourhood.</li> <li>▪ Reinsertion neighbourhood.</li> <li>▪ 3-exchange neighbourhood.</li> </ul> </li> <li>• Designed two ant heuristics that fit within the paradigm: <ul style="list-style-type: none"> <li>▪ Solution building by means of randomized constructive procedure followed by a local search.</li> <li>▪ Pheromone updating to report back information for building of new solutions.</li> </ul> </li> </ul> |
| Norhazwani   | Reactive Tabu  | <ul style="list-style-type: none"> <li>• Dynamic tabu list size.</li> </ul>  |

|                               |   |   |
|-------------------------------|---|---|
| Yunos & Zuhaimy Ismail (2009) | Search [CARP]; for solving Solid Waste Management Problem | <ul style="list-style-type: none"> <li>Considering repetition to diversify the search process and explore the solutions.</li> </ul> |
|-------------------------------|---|---|

## 2.6 Review Solution Method on Waste Collection Problem

In previous section we have seen from a literature, various solution methods used to solve CARP. Most of the problem are applied to waste collection problem which they formulated a waste collection problem into CARP. So this section provides a literature review in different solution method from the previous section that was used to solve waste collection problem itself. Table 2.4 shows summarization of solution method used to solve waste collection problem itself in last three years.

**Table 2.4: Other Solution Method for Waste Collection Problem**

| Authors  | Solution Method   | Case Study   | Objective                        |
|--|---|--|----------------------------------|
| X. Y. Wu, G. H. Huang, L. Liu & J. B.Li (2006) | Solve using interval nonlinear programming                        | Application to the planning of waste management activities in the Hamilton-Wentworth Region, Ontario, Canada | Focus to minimize the cost       |
| Julian Scott Yeomans (2007)                    | Solve using combination of Grey Programming (GP) and Evolutionary | Case study for the municipality of Hamilton-Wentworth in the   | Focus to minimize operating cost |

|   | Simulation-Optimization (ESO)  | Province of Ontario, Canada   |  |
|---|--|---|--|
| Jing-Quan Li, Denis Borenstein & Pitu B. Mirchandani (2008) | Solve using minimum cost flow problem  | Application to waste collection problem in City of Porto Alegre. Brazil                                 | Focus to minimize the total operating cost and fixed vehicle costs         |
| Y. P. Li, G. H. Huang, Z. F. Yang and X. Chen (2009)        | Solve using inexact fuzzy stochastic, Interval parameter Fuzzy Linear Programming (IFLP) | Application to the long-term planning of Municipal Solid Waste Management in the City of Regina, Canada | Focus on reducing waste flows to the landfill with a minimizes system cost |

## 2.7 Recent Works on Reactive Tabu Search

As we already know, there are a lot of literatures on the CARP. So as RTS, there are also a lot of area of research was applied to solve using this advanced heuristic method. A part of it will discuss in this section.

From the literature, the first approach using RTS was proposed by Xu *et al.* (1998) to recover epipolar geometry from a pair of uncalibrated images. By minimizing a proposed cost function with the RTS approach, the experiments on real images show that this approach is effective and fast.

O'Rourke applied the RTS for the Unmanned Aerial Vehicle (UAV) routing problem in 1999. O'Rourke used the adjustment of the tabu list size as well as a penalty coefficient. The penalties for missed time windows, exceeding vehicle capacity and exceeding vehicle range was set to the objective function. By controlling the penalty

coefficient, it forced the search process in and out of feasible regions of the solution space and acted as an additional diversification strategy. Harder (2000) and Kinney (2000) also used the same problem and method with O'Rourke. But in Harder and Kinney case, they are not only adjusted the tabu list size, they also determined how much iteration to spend in order to improve a solution in their search process (Brown, 2001).

In 2000, Fukuyama was proposed RTS for load transfer operation in distribution systems. The developed RTS algorithm showed that it can generate the most highly qualified results and realize the fastest computation for loss minimization and service restoration. As for comparison, the algorithm was compared its performance with the modern heuristic methods such as SA and GA, and he observed that RTS is the best method for load transfer operation.

The RTS also was applied to solve a pick-up and delivery problem which was proved by Nanry and Barnes (2000). Other than RTS, they also developed a hierarchical search methodology based on the average duration of a tour from the current solution and the average length of the time windows for the customers. This search methodology stated which types of moves that need to consider and when to consider them. In this research, Nanry and Barnes state that a large number of feasible solutions exist when the average time window length is large relative to the average duration of a tour. Therefore, this search methodology encourages more improvement moves in comparison to moves that add or remove tours.

In 2006, Wassan successfully implemented the RTS to solve the classical VRP. He developed a new escape mechanism strategy which manipulates different neighbourhood schemes in a very sophisticated way in order to obtain a balanced diversification and intensification continuously during the search process. In addition,

he compared his algorithm with the best methods in the literature using different sets of data.

Castellani *et al.* develop their own RTS algorithm to solve automatic selection of Markov Random Field (MRF) control parameters in the year of 2007. The core ingredient in their algorithm is the application of fitness function to measures the performance of particular parameters set and used escape mechanism if the fitness function has not increased by carrying out a random restart.

The novel approach for an integrated placement and replacement of control and protective devices in distribution network feeders was discussed in 2008 by Silva *et al.* They proposed the RTS to solve the problem which the problem was modelled through mixed integer non-linear programming (MINLP) with real and binary variables. By using RTS, the results work in the excellent performance of the algorithm.

Also in 2008, Blochliger and Zufferey introduced one more problem that can be solving using RTS. They presented a local search approach to the graph colouring problem and shown that their algorithm obtains competitive results on a large sample of benchmark graphs which are generally agreed to be difficult to colour. The scheme based on adjusting their tabu tenure itself depends not only on the graph but also on the state of the search. The algorithm was design to be easy to implement and does not need to perform an explicit check for the repetition of configurations. The determination of the tabu tenure only requires the variation of the objective function.

## 2.8 Summary

This chapter contains the important information gathered. It includes the CARP formulation, some related works from the researchers and heuristics methods which can be use to solve the CARP regarding from previous work and also previous research on RTS.



## CHAPTER 3

### RESEARCH METHODOLOGY

#### 3.1 Introduction

Nowadays, more and more municipalities, regional authorities, post office administrations, school buses operators, electricity and gas companies are applying such an arc routing system. This phenomenon is driven to be more competitive and cost-efficient. This growth has been paralleled by the development of a number of powerful optimization techniques. The two most important are probably branch-and-cut for exact optimization and mathematical formulation of Tabu Search (TS) in the area of heuristics (Dror, 2000). Thus, the data in this research will be analyzed by using one of a meta-heuristic method which is Reactive Tabu Search (RTS). The need to expand to RTS instead of TS is due to ability of the method to quickly explore an unknown domain without the need of parameter tweaking (Castellani *et al.*, 2007).

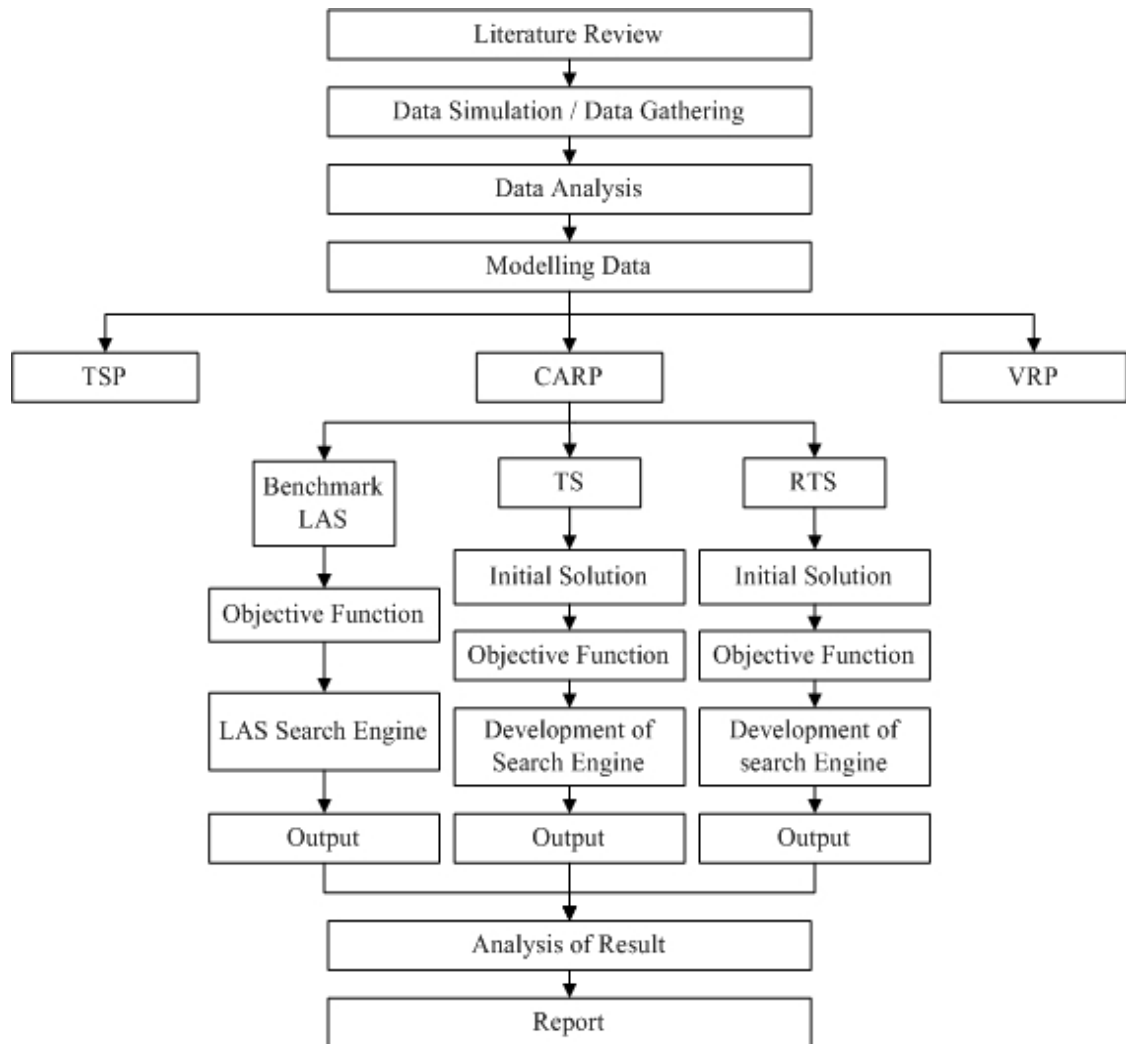
This chapter presents an overview of the methods used and the characteristic that we use to develop our newly formulated RTS algorithm. It begins with our research

framework, data source followed by explaining the terminologies in TS, a basic procedure for TS, a reactive tabu scheme, our TS implementation and the element of RTS. We also provide the general algorithm in developing our RTS algorithm and this chapter end with summary of the research methodology.

## **3.2 Research Methodology**

The research starts with a literature review on routing problem to understand the scenario of the problem. Then we gathered a set of data and simulated it into a routing problem. The data then will be analysing to model it into Travelling Salesman Problem (TSP), Capacitated Arc Routing Problem (CARP) or Vehicle Routing Problem (VRP). After modelling it, then we find out that our problem is most suitable and close to CARP. This is because, our problem is arc routing problem which need to collect the demand in every single arc but in the same time we have to minimize the total cost operation. TSP is another model for routing problem and it is only can be applied if all nodes are connected to each other. Since it can be a problem to implement our problem into TSP because some formula or rearrangement need to be made, so we decide not to choose TSP as our model.

VRP is another most popular routing problem. This model is almost similar to CARP in term of its graphical network. But for VRP, the demand is located at each of the nodes while in our problem, the demand is located along the arc. If we used VRP as our model we need to transform from an arc routing problem to a node routing problem. So to make it easy, since CARP fulfil all the criteria of our problem, then we chose CARP as our model. The entire research methodology can be summarized in Figure 3.1



**Figure 3.1: Research Methodology**

### 3.3 Data Source

There are two types of data that will be used to assess the performance of the proposed algorithm:

a. Simulation Data

Several sets of instances will be generated based on original data to simulate real life instance of CARP in solid waste collection. The variables involve in the data collection and the data generations are the demand (the quantity of waste to be collected) and the cost for each arc.

b. Primary Data

The primary data is a real life instance of CARP originating from the municipal solid waste collection. The original data was collected from Southern Waste Management Sdn. Bhd. (SWM) – Johor Bahru branch office, a provider of a diverse range of waste management services for municipal authorities, commercial and industrial sectors for the Southern Region of Malaysia. We confine our study on residential area in Johor Bahru under municipal authorities of Majlis Perbandaran Johor Bahru. The problem is to design a set of solid waste collection routes, each to be served by a truck such that the waste at each customer is fully collected and the total expected cost is minimized.

### **3.4 Terminologies in Tabu Search**

Some terminologies in TS must be defined first in order to understand how TS works by using memory structures. Then from TS, some changes can be made in order to make it RTS. The change to make TS become RTS is on its tabu list size, instead of using static tabu list size in TS, the dynamic tabu list size was applied in RTS. Diversification strategy is also one of a characteristic that not consider in TS but will be used to develop RTS. Table 3.1 shows the terminologies of TS.

**Table 3.1: Terminologies in TS**

| <b>TERMINOLOGY</b>   | <b>DEFINITION</b>  |
|----------------------|--|
| A move               | A transition from a current solution to its neighbouring (or another) solution.  |
| An attribute         | The elements that constitute the move.   |
| Tabu list            | A list of moves that are currently tabu (a list of forbidden exchanges to avoid cycling between the same solutions endlessly).                                 |
| Tabu list size       | The number of iterations for which a recently accepted move is not allowed to be reserved, $ T_s $ .   |
| Tabu tenure          | An integer number telling for how long a given move will remain tabu.  |
| Aspiration level     | A threshold (usually the best current objective function value) for which the tabu status of a move can be relaxed / override the tabu restriction.            |
| Admissible move      | A move that is nontabu or a move that is tabu active but that can produce a solution well above the aspiration level.  |
| Forbidding strategy  | The tabu condition that forbid a move from being reversed.   |
| Freeing strategy     | The conditions that allow a move to become nontabu, because either its tabu status has become not tabu or such a move satisfies an aspiration criterion.       |
| Aspiration criterion | Criterion used to identify tabu restrictions that may be overridden.   |
| Data structure       | The way to record full or partial past information, which helps avoid wasting computing time in recomputing already computed information in future iterations. |
| Neighbour solution   | One move from the current solution.  |
| Neighbourhood        | The set of all possible neighbour solution that can be reached with one move.  |

### 3.5 A Basic Tabu Search Procedure

There is a procedure to be considered in order to make computational using TS. A general outline of a TS procedure as state in Zainuddin, 2004 is as follows:

Given a feasible initial solution  $x^*$  with fitness function value  $z^*$ , let  $x = x^*$  and  $z(x) = z^*$ . While stopping criterion is not fulfilled do the following steps.

**[Step 1]**

- Select best admissible move that transform  $x$  into  $x'$  with fitness function value  $z(x')$  and add its attributes to the running list.

**[Step 2]**

- Perform tabu list management: compute moves (or attributes) to be set as tabu, i.e update the tabu list.

**[Step 3]**

- Perform exchanges:  $x = x'$ ,  $z(x) = z(x')$  ;  
if  $z(x) < z^*$  then  $z^* = z(x)$  and  $x^* = x$ .

Result:  $x^*$  is the best of all determined solution, with fitness function value  $z^*$ .

The main features to be considered in TS implementation are as follows:

- An initialization phase
- A forbidding strategy
- A freeing strategy
- A stopping criterion
- A diversification strategy (optional)

### 3.5.1 The Initialization

TS procedure requires a starting solution and it is the most important part to start a computational. At the initialization strategy, the initial solution, the tabu list size and the moves need to be define on a few attribute. The descriptions of each of them are as follows:

#### *Initial Solution*

The computational of TS algorithm has to be start with an initial solution to assign the value to the decision variables and to assess the fitness function. This initial solution can be feasible or otherwise. It can be generated randomly or via a suitable heuristic by an optimal method. Usually there are advantages to start from high quality of initial solution. However, if the initial solution is already very good, it could make out TS quite restrictive (Zainuddin, 2004).

#### *Tabu List Size*

As previously mentioned in the literature, TS is a memory search method. Thus the important parameter in TS is tabu list size ( $|T_s|$ ) which is the size of the record list of a previous moves. This size varies from iteration to iteration. The size of the tabu list represents its memory ability and it is hard to set. If  $|T_s|$  is set to a small value it may be too flexible and the probability of cycling may be high. But if  $|T_s|$  is set to a large value it may be too strict and a good solution may be missed due to the move leading to them remained tabu take so long time. This can be avoided by using dynamic tabu list size.

Dynamic tabu list size can be periodically changes over time or also can be continuously change. If the tabu list size change periodically over times, this mean that the tabu list size will kept fixed for certain number of iteration and the process is repeated for  $k$  times. While for continuously changes, the changes of tabu list size will depend on the cost function for the selected moves, or in other word it will change when needed.

### *Moves*

In each iteration, a modification to the current solutions is necessary to produce a neighbourhood solution which is known as move. This neighbourhood is constructed in order to identify the adjacent solutions that can be reached from any current solution. The size for the search neighbourhood is not limited but it must have significant influence on the result. The larger tabu list size, the better the quality of the solution but it requires more execution time.

### **3.5.2 The Forbidding Strategy**

The forbidding strategy is designed to avoid cycling problem by forbidding the moves that already been investigated. The main mechanism for using memory in TS is to classify a subset of the moves in a neighbourhood as forbidden or tabu. But it will not remain tabu forever; it only for a certain number of iterations and tabu tenure will define for how long it remains tabu.



### **3.5.3 The Freeing Strategy**

Tabu conditions may become too restrictive. Therefore it should not be inviolable under all circumstances. And so of that, aspiration criteria are introduced in the basic TS algorithm. The aspiration criterion is the rule that overrides tabu restrictions. Aspiration criterion can make a certain forbidden moves become allowable when it is satisfies the rule. It works by deleting the tabu restrictions of the solutions to reconsider in further steps of the search. The goal of the aspiration function is to avoid cycling in order to provide the ability to find an improved solution and it is organized to be compatible.

### **3.5.4 The Stopping Criterion**

Since the algorithm is open-ended, so the stopping criterion is always needed. It may run forever as the optimum is unknown. The simplest form of stopping criterion is a fixed number of iterations, such as after 1000 iterations or by using computational time, such as after 10 minutes computations. It always a trade-off to consider because maybe the computational need just two to three minutes computation, but we set it to stop after 10 minutes computation, so there is no sense of running the programme after that amount of time. Nonetheless, if the algorithm stops too early, the optimum solution may not be found yet and conversely the computation time can be wasted if the algorithm stops too late. Thus, the dynamic stopping criterion that is taking advantage of the solution changes is more suitable in most cases.

### **3.5.5 The Diversification Strategy**

Diversification provides a wider exploration of the search space and it drives the search into a new region. When there is no improvement after performing some number of iteration, it shows that either the optimum solution has already been found or the neighbourhood of the solution space being searched is not good. If it is happen, this mean that the algorithm need to terminate the search process or the algorithm need to diversify it search process. The most common diversification strategy is a random restart, solutions found by other greedy heuristics or solutions obtained using an intelligent search that take into account past information. This can help to improve the algorithm by escaping from a local optimum.

### **3.6 The Reactive Tabu Scheme**

The tabu search is one of meta-strategy and it has been shown to be an effective and efficient scheme for combinatorial optimization. It works by combining a hill-climbing search strategy based on a set of elementary moves and a heuristics which is to avoid the stops at suboptimal points and the occurrence of cycles. The fact that we already know in TS implementation is that cycle are avoided if the repetition of previous visited configuration is prohibited. But it is actually not sufficient for an effective and efficient search technique. The chaotic like attractor should be discouraged (Battiti & Tecchiolli, 1994). And so of that, Reactive Tabu Search is applying to discover of a new high quality solution.

As we already known, the reactive tabu scheme totally maintains the basic concept and also the terminologies of TS. What we try to do is to adapt the size to the problem to the current evolution of the search, also escape strategy for diversifying the search process.

### **3.7 Tabu Search Implementation**

In this section, we present our tabu search implementation. It contains the way we generate the initial solution, the element of our TS and RTS. In the element of TS we provides the way we choose the move to produces the neighbourhood and which criteria that we choose for aspiration and stopping. While in the element of RTS we state our tabu list size and how do we counter the repetition problem.

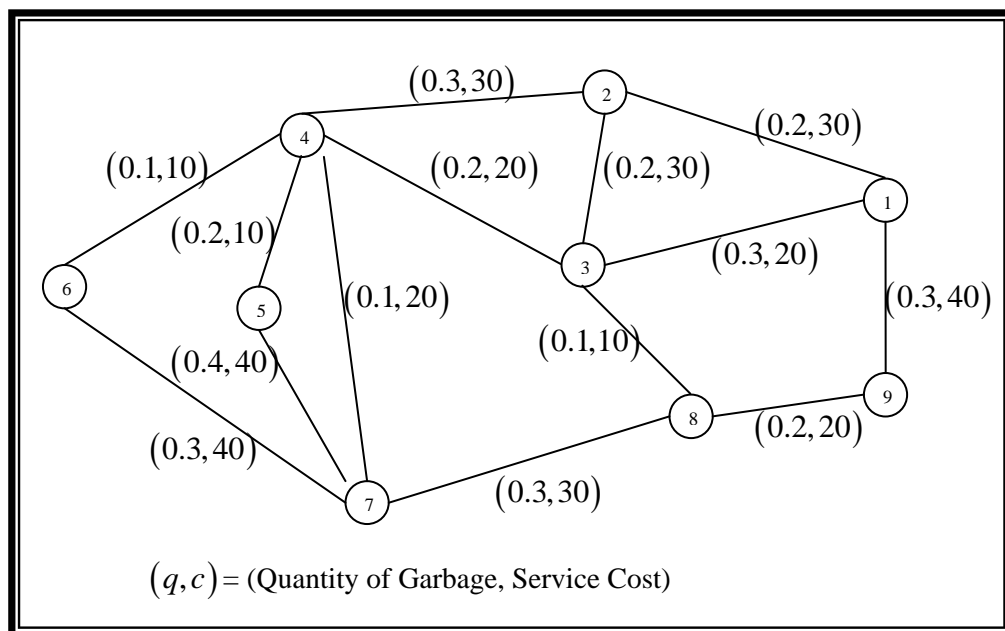
#### **3.7.1 The Initial Solution**

The RTS starts from an initial configuration where a good-quality of the initial configuration is used as a starting solution to obtain a good algorithm performance (da Silva *et al.*, 2008). The initial solutions are generated by using Cheapest Edge Method (CEM). CEM was chosen as the initial solution as it is rather a simple heuristic approach and it is similar with Nearest Neighbour (NN) method. They are different just on the move criterion, but the algorithm is almost the same. The NN used least distance to move from one node to another while cheapest edge method used least cost. NN typically does not find very high quality solutions, but it is often and successfully used

in combination with perturbative search method and it is relatively fast and easy to use compare with other tour construction such as General Insertion (GI), Nearest Insertion (NI), Cheapest Insertion (CI), Farthest Insertion (FI) and Arbitrary Insertion (AI). This is because all of them used a formula to choose the insertion of the move (Hoos & Stutzle, 2004).

By using CEM, the vehicle starts by travelling from the depot. The vehicle will move from one node to another with the required edge not yet served in a route that has the minimum cost. If there is more than one required edge not yet served with the minimum cost, then ties are broken arbitrarily. It will continue until all edge has been served. When no remaining required edges can be feasibly added to the route, the route is completed by the vehicle returning to the depot through the minimum cost of deadheading path.

A network model can be represented as shown in Figure 3.2:



**Figure 3.2: Network Model**

While the algorithm for generating initial solution is as follows and symbols used are listed as follows;

$x_{ij}$  = edge  $(i, j) \in E \cup A$  traverse.

$x_{ijk}$  = number of times edge  $(i, j) \in E \cup A$  is traversed in trip  $k$ .

$y_{ijk} = \begin{cases} 1 & \text{if the edge(arc)}(i, j) \in R \text{ is covered in trip } k, \\ 0 & \text{otherwise} \end{cases}$

$q_{ij}$  = edge demand.

$Q$  = vehicle capacity.

$c_{ij}$  = service cost of an edge  $(i, j) \in E \cup A$ .

$C$  = total cost.

**[Step 1]**

- Find  $x_{ij}$ , such that  $x_{ij} = \min[c_{ij}]$  unvisited arc.
- If no more  $\min[c_{ij}]$  unvisited arc and  $x_{ijk} < y_{ijk}$ , find  $x_{ij}$  such that  $x_{ij} = \min[c_{ij}]$  visited arc that link to the unvisited arc.

**[Step 2]**

- Check the demand;
- If  $\sum_{(i,j) \in R} q_{ij} y_{ijk} > Q$ , do not move, go to step 5 following with step 1.
- Otherwise, move and proceed to step 3.

**[Step 3]**

- Update cost,  $C = \sum_{(i,j) \in E} \sum_{k=1}^K c_{ij} x_{ijk}$ .

**[Step 4]**

- If  $x_{ijk} \geq y_{ijk}$ , then proceed to step 5.
- Otherwise, repeat step 1.

**[Step 5]**

- Return to node 1 through  $x_{ij} = \min[c_{ij}]$ , then stop.
- \* Always assign node 1 as a depot.
- \* Ties are broken arbitrarily.

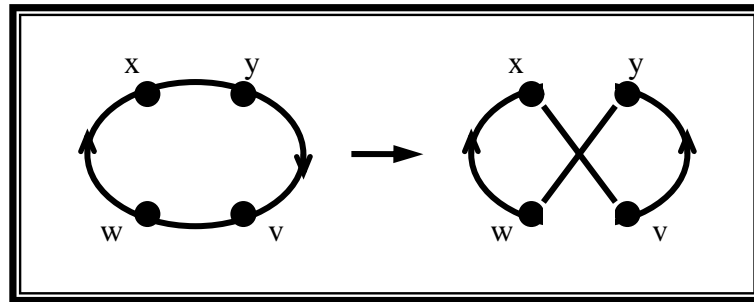
**3.7.2 Elements of Tabu Search**

There are several important elements used in the TS algorithm that need to define in order to develop our own algorithm. There are a move neighbourhood structure, aspiration criteria and stopping criteria. The descriptions of those elements are as follows:

*A Move Neighbourhood Structure:*

As other local search algorithms, the RTS algorithm starts with generating an initial solution follows by explores its neighbourhood in order to select the best move neighbour solution as the current solution. For this research, we use three types of moves in order to define their neighbourhood. There are 2-opt, insertion and deletion. For 2-opt, we only examine one combination in every iteration. Then insertion and deletion are applied if it is necessary. Its mean that, a necessary edge needs to be inserted and unnecessary edge or repeated arc need to be deleted in a new generated route to reach at a feasible route. Then the best improvement is selected with the best fitness value in the neighbour solutions.

The basic step of 2-opt is to delete two edges from a tour and reconnect the remaining fragments of the tour by adding two new edges. Figure 3.3 shows more clearly how 2-opt were operated.



**Figure 3.3: 2-opt move**

Figure 3.3 shows the initial tour is  $(x, y), (v, w)$  while after doing the 2-opt modification, it becomes  $(x, v), (y, w)$ . This 2-opt of move we use in our main neighbourhood structure to generate their neighbourhood and we also include the insertion and deletion to ensure the continuity of the route.

*Aspiration Criterion:*

As mentioned above, aspiration criterion is the rules that overrides tabu restrictions. If the aspiration criterion is satisfied, a tabu moves becomes allowed. In other words, if a move leads to a better solution, then it is chosen even if it is tabu (Castellani, 2007). There are three types of aspiration criterion as shown in Table 2.2 above. There are; aspiration by objective, aspiration by default and soft aspiration criteria. Details description as discussed in Section 2.4.2 in Table 2.2. In this research, we only consider the aspiration by default which is freeing the least cost move and its only work when all moves are tabu. This means that if there is no more move are

allowed then check to the fitness function and freed the move which have the minimum fitness function regardless of its tabu tenure.

*Stopping Criterion:*

To terminate the computational, the algorithm will stop searching right after it complete diversify on the repetition and we also confining the maximum iteration to prevent from wasting time. This is because, in case if there are too many repetitions, the algorithm may run forever until it completed diversify but in the same time the solution are not improve after such number of iteration. Since the optimum is unknown, so the maximum number of iteration is needed.

### **3.7.3 Elements of Reactive Tabu Search**

*Tabu List Size:*

The larger it size the stronger the memory. However, with a fixed list size, it is possible that the searching trajectory may form a limit cycle (if the list size is small) and it also may cause low efficiency (if the list size is too large) (Xu *et al.*, 1998). In this study, firstly we defined the tabu list size to be static changes. In the same time, we need to record the repetition of the objective function, its mean that record unimprovement of objective function in every iterations. Then if the repetition occurs for n periods of times, then apply the diversification strategies. But if new best solution found, the tabu list size need to be increase to make sure it will not cycle itself.



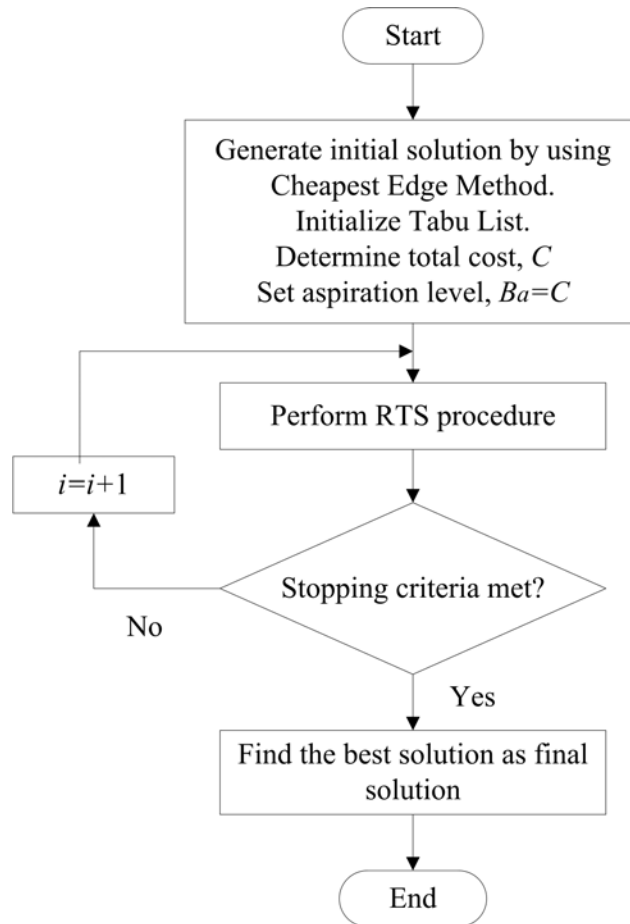
However, if all the movement already in tabu list, then we need to decrease the tabu list size to loose it and preventing from being too strict. Otherwise it might missed a good solution due to the move leading to them remained tabu take so long time.

### *A Diversification Strategy*

A diversification is needed to provide a wider exploration of the search space. In this scheme, the diversification is applied when there are too many repetitions occur. It work by undo the iteration to the first detect repeated iteration. Then search for the neighbourhood that has the same objective function value with different route and apply the move and start the search process again.

## **3.8 The General Reactive Tabu Search Algorithm**

Figure 3.4 shows the general entire procedures in developing RTS. Without performing RTS procedure in Step 3 in a flow chart as shown in Figure 3.4, the development will be just a TS procedure.



**Figure 3.4: Flow Chart of the Entire Procedure in Developing RTS**

### 3.9 Summary

In general, this chapter contains our research methodology. In this research methodology, the procedure, terms and terminologies in TS and RTS were discussed successfully. At the end of this chapter provide the general flow chart for developing RTS algorithm. Next chapter will discuss about other method that we used for comparing purposes which is Look Ahead Strategy (LAS) and the development of our TS algorithm.

## CHAPTER 4

### LOOK-AHEAD STRATEGY AND TABU SEARCH FOR SOLVING CAPACITATED ARC ROUTING PROBLEM

#### 4.1 Introduction

This chapter presents a Look-Ahead Strategy (LAS) and Tabu Search (TS) implementation for solving Capacitated Arc Routing Problem (CARP). Based on the initial worked by Ismail *et al.* (2007), a LAS computational module was developed for solving CARP. The solution generated used as a benchmark to our propose solution approaches which is the TS and Reactive Tabu Search (RTS). These two methods are used to make a comparison with our RTS algorithm. The LAS method was refer in a paper, entitled “Look Ahead Heuristics for Modelling Solid Waste Collection Problems” by Ismail *et al.* (2007). While for TS algorithm, the algorithm used to compare with RTS is RTS algorithm itself before applying it with element of RTS. Then we used the same simulated data for comparing purposes.

## 4.2 Look-Ahead Strategy for Capacitated Arc Routing Problem

Ismail *et al.*, (2007) used Look-Ahead Strategy method proposed by Amponsah in solving the arc routing problem of the collection of garbage especially for developing country. This approach conducted a case study on solid waste collection problem in the area of Johor Bahru under municipal authorities of Majlis Perbandaran Johor Bahru which is same as our area of the problem. Initially, the problem is modelled as CARP. This method work to minimize the total cost of the operation which is set to be their fitness function. It can be achieved by considering the minimum deadheading cycles through all the required edges. A deadheading in CARP cases is stand for an empty movement which means that the traversal of the vehicle without give their servicing. In other word, the vehicle just only passes an arc without collecting the garbage. More about the development of LAS algorithm will be discuss in a next section.

### 4.2.1 Basic Idea

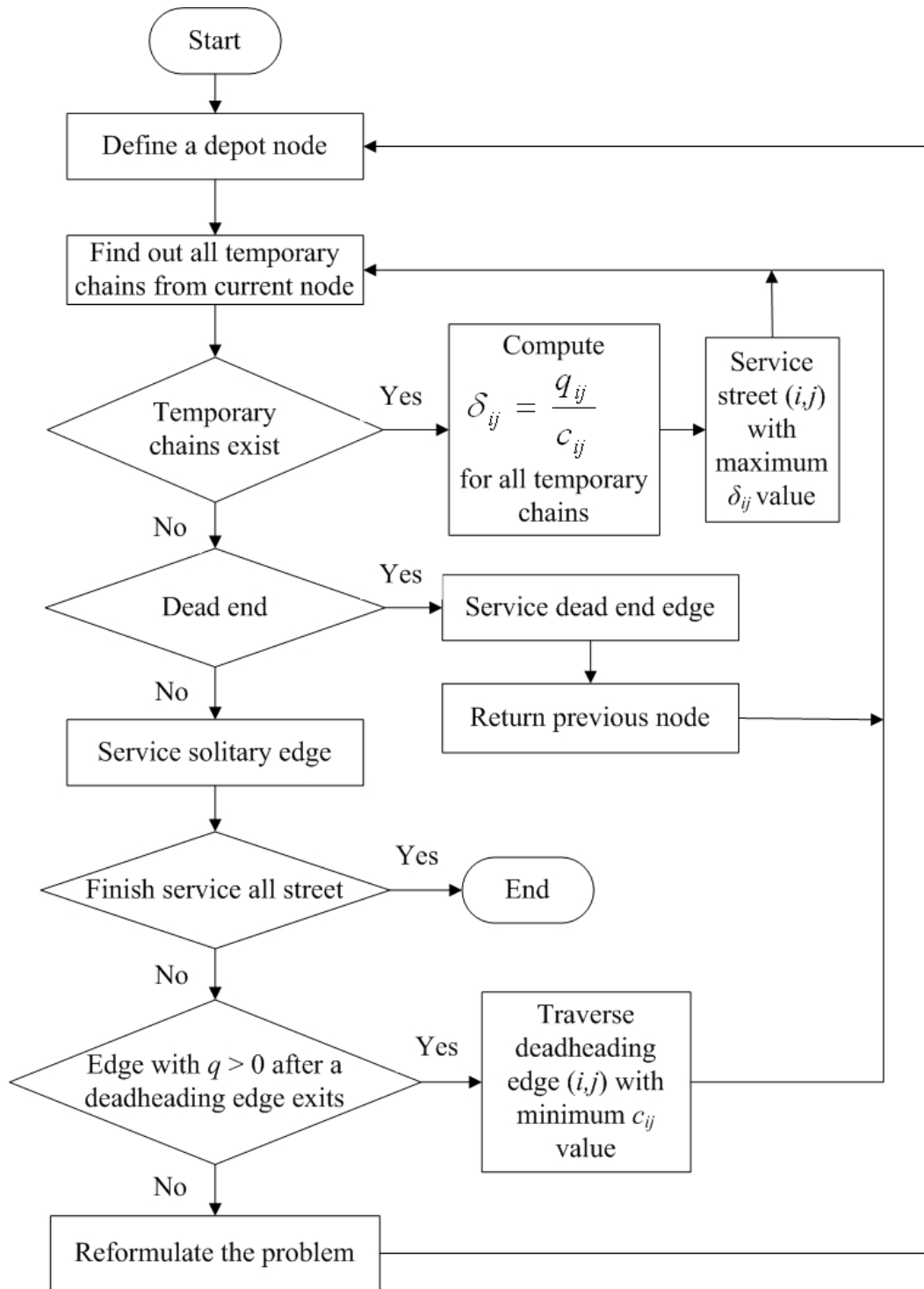
The most important point in this problem is the revisiting an empty arc again because it might increase their service cost. In other words, empty movement need to be minimized as possible so that a service cost would be minimized. They also assume that the estimation of the quantity of garbage in a particular arc is proportional to the arc distance. Therefore, the algorithm was developed by considering the quantity  $q_{ij}$  along the arc and the cost  $c_{ij}$  of servicing the arc. Hence, they used to combine both elements into the formulation, such as;

$$\delta_{ij} = \frac{q_{ij}}{c_{ij}} \quad (4.1)$$

Equation (4.1) in the fitness function used to minimize the total cost. However, this equation seems not performs well in terms of the demand/cost ratio in the entire solution. Consequently, LAS was introduced to improve this greedy method by taking into account present as well later choices in the algorithm. LAS works by inspecting the total demand/cost ratio on all possible temporary edges with respect to their likelihood to produce future advantage. It chooses the most promising edges and removes away unpromising edges in the collection process. By doing so, they can cut revisiting an empty arc again and at the same time the service cost will be reduce.

#### **4.2.2 Look-Ahead Strategy Algorithm**

Basically, the algorithm will proceeds from one junction to another which is adjacent to the related nodes in each stage. For more details, Figure 4.1 shows clearly every single step in performing the algorithm.



**Figure 4.1: Flowchart of LAS**

### **4.3 Tabu Search for Capacitated Arc Routing Problem**

In the context of TS related to the CARP, the most appropriate solution procedure need to be made so that it is suitable for our CARP model. The initial solution, the tabu list size, the tabu moves, the aspiration criterion and the stopping criterion is the main element to be defined in order to develop our own TS algorithm. It can be consider as a body of solution procedure which is quite challenging to be define.

The problem is to design a route such that the demand of each customer is fully collected. In completing the task, there are some requirements that need to be consider which is the demand must not exceeding the maximum capacity of the vehicle while in the same time all arc must be served in a minimum cost. So, the algorithm was built based on these requirements.

#### **4.3.1 Initial Solution**

Initial solution is very important in TS procedure to initialize the main search process. It is because TS will start work from one solution and explore its neighbourhoods in order to get new regions of solutions. Not just an initial solution itself is very important, the choice on defining the initial solution is also important. To generate the initial solution, we use Cheapest Edge Method (CEM). As discussed in Section 3.6.1, CEM was used for generating the initial solution because it is rather a simple heuristics approach and this will obtain a good-quality of the initial configuration. Besides, it is fast and easy to use compare with other tour construction.

The details of the algorithm for generating an initial solution can be describe as follows:

$x_{ij}$  = edge  $(i, j) \in E \cup A$  traverse.

$x_{ijk}$  = number of times edge  $(i, j) \in E \cup A$  is traversed in trip  $k$ .

$y_{ijk} = \begin{cases} 1 & \text{if the edge(arc)}(i, j) \in R \text{ is covered in trip } k, \\ 0 & \text{otherwise} \end{cases}$

$q_{ij}$  = edge demand.

$Q$  = vehicle capacity.

$c_{ij}$  = service cost of an edge  $(i, j) \in E \cup A$ .

$C$  = total cost.

**[Step 1]**

- Find  $x_{ij}$ , such that  $x_{ij} = \min[c_{ij}]$  unvisited arc.
- If no more  $\min[c_{ij}]$  unvisited arc and  $x_{ijk} < y_{ijk}$ , find  $x_{ij}$  such that  $x_{ij} = \min[c_{ij}]$  visited arc that link to the unvisited arc.

**[Step 2]**

- Check the demand;
- If  $\sum_{(i,j) \in R} q_{ij} y_{ijk} > Q$ , do not move, go to step 5 following with step 1.
- Otherwise, move and proceed to step 3.

**[Step 3]**

- Update cost,  $C = \sum_{(i,j) \in E} \sum_{k=1}^K c_{ij} x_{ijk}$ .

**[Step 4]**

- If  $x_{ijk} \geq y_{ijk}$ , then proceed to step 5.



- Otherwise, repeat step 1.

**[Step 5]**

- Return to node 1 through  $x_{ij} = \min[c_{ij}]$ , then stop.
- \* Always assign node 1 as a depot.
- \* Ties are broken arbitrarily.

### 4.3.2 Tabu List Size

For TS, a static tabu list size ( $|T_s|$ ) is used. However, a static number such as 3, 10 or 31 or even any other numbers could not be assigned as tabu list size because the size of the problem may not always be the same. Let's say if we set it to 9, then for large size of problem, let say 30, the algorithm might cycle itself due to the small size of the tabu list and it is not efficient to use. So to make the algorithm suitable for any numbers of nodes, we set tabu list size depend to the size of the problem which is equal to the number of nodes,  $n$ .

In order to choose the best value for tabu list size, a variety size in multiplication of  $n$  were investigate. In the investigation, we consider the multiplication of  $n$  because it varies from small to the large value so that the possible range will be check. Hence, Table 4.1 presents the result of the investigation of the size of tabu list.

**Table 4.1: Investigation on Tabu List Size**

| $n$ | IS   | TS          |            |            |             |             |             |             |
|-----|------|-------------|------------|------------|-------------|-------------|-------------|-------------|
|     |      | TS1         | TS2        | TS3        | TS4         | TS5         | TS6         | TS7         |
| 5   | 380  | 380         | 380        | 380        | 380         | 380         | 380         | 380         |
| 6   | 330  | 330         | 330        | 330        | 330         | 330         | 330         | 330         |
| 7   | 490  | <b>430</b>  | <b>430</b> | <b>430</b> | 490         | 490         | 490         | <b>430</b>  |
| 8   | 680  | <b>600</b>  | 620        | 620        | 680         | 680         | <b>600</b>  | 620         |
| 9   | 820  | <b>760</b>  | <b>760</b> | <b>760</b> | <b>760</b>  | <b>760</b>  | <b>760</b>  | 770         |
| 10  | 750  | 750         | 750        | 750        | 750         | 750         | 750         | 750         |
| 11  | 890  | 890         | 890        | 890        | 890         | 890         | 890         | 890         |
| 12  | 1070 | 1070        | 1070       | 1070       | 1070        | 1070        | 1070        | 1070        |
| 13  | 1170 | <b>1140</b> | 1060       | 1060       | 1050        | 1060        | <b>1140</b> | 1090        |
| 14  | 1260 | <b>1220</b> | 1260       | 1260       | <b>1220</b> | <b>1220</b> | <b>1220</b> | <b>1220</b> |
| 15  | 1370 | 1370        | 1370       | 1370       | 1370        | 1370        | 1370        | 1370        |
| 20  | 1490 | <b>1390</b> | 1490       | 1410       | 1410        | 1450        | 1470        | 1490        |
| 25  | 2330 | 2330        | 2330       | 2330       | 2330        | 2330        | 2330        | 2330        |
| 30  | 2680 | 2680        | 2680       | 2680       | 2680        | 2680        | 2680        | 2680        |
| 50  | 1103 | 1103        | 1103       | 1103       | 1103        | 1103        | 1103        | 1103        |

$$\text{TS1 : } |T_g| = n$$

$$\text{TS2 : } |T_g| = 2n$$

$$\text{TS3 : } |T_g| = 3n$$

$$\text{TS4 : } |T_g| = (n - 1)n$$

$$\text{TS5 : } |T_g| = (n - 2)n$$

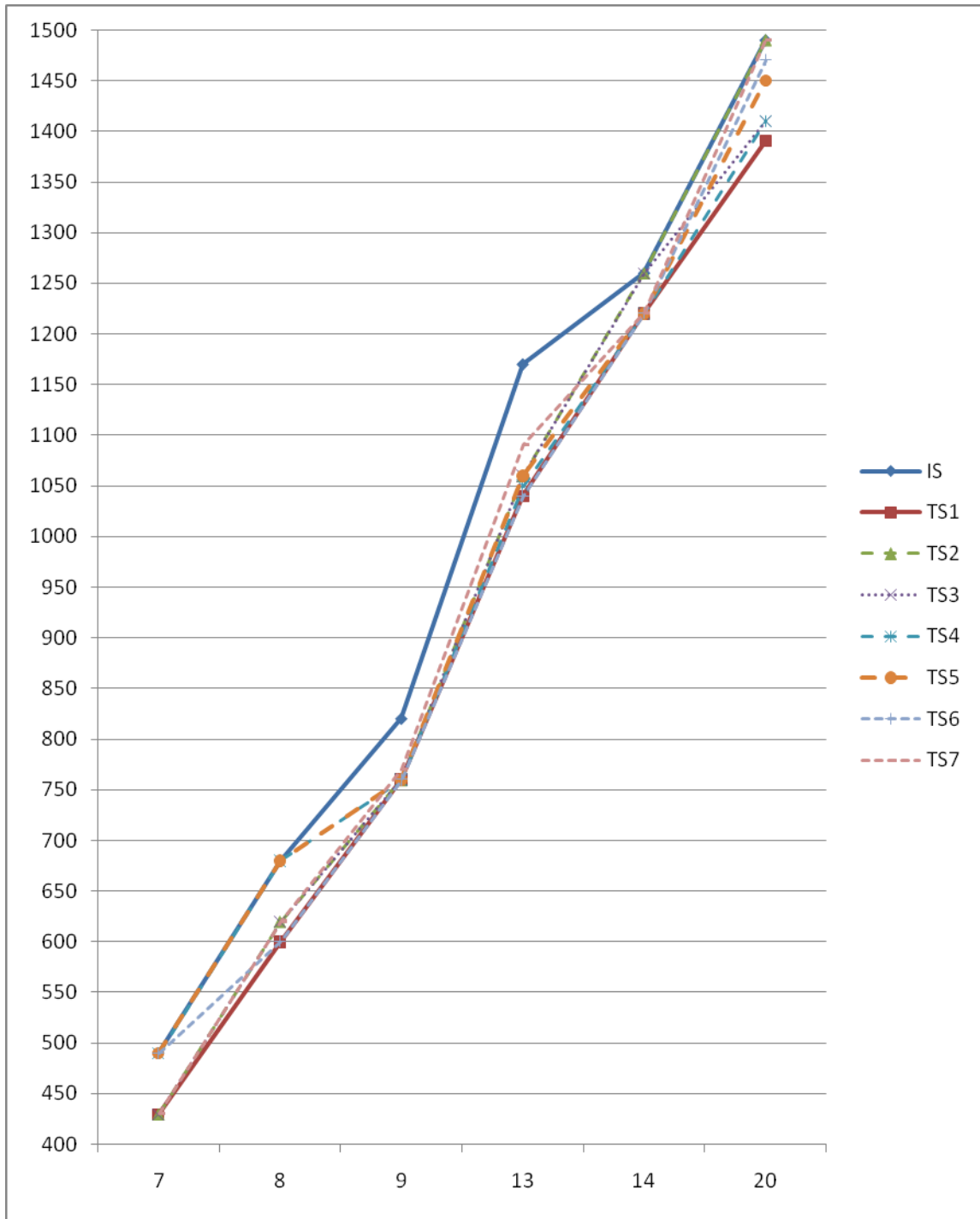
$$\text{TS6 : } |T_g| = (n - 3)n$$

$$\text{TS7 : } |T_g| = (n - 4)n$$

Where  $n = 5, 6, 7, \dots, 50$

Table 4.1 shows the experimental results generated using different sizes of tabu list for up to 50 numbers of nodes. Numbers in bold in the table shows the best results generated by the algorithm and we can see only TS1 gives the majority best fitness function among them. Hence TS1 is chosen to be the tabu list size for CARP cases in our TS algorithm. Figure 4.2 shows a specific comparison based on investigation on

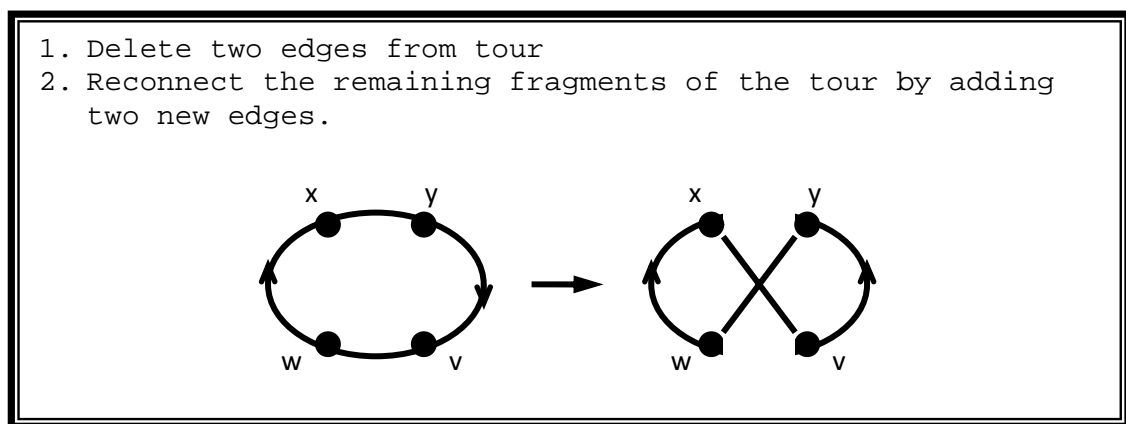
tabu list size and it shows that TS1 always give the minimum answer compared with others.



**Figure 4.2: Investigation on Tabu List Size**

### 4.3.3 Tabu Moves

The moves attribute will base on three type of move which are 2-opt, insertion and deletion. For our first step to search a neighbourhood, we explore one combination of 2-opt move. The 2-opt move can be describe as shown in Figure 4.3.



**Figure 4.3: 2-opt Move**

Figure 4.3 clearly shows the 2-opt moves. Before doing 2-opt move, there exists a connection between  $(x, y)$  and  $(v, w)$  and after 2-opt was applied in the tour, a connection between  $(x, y)$  and  $(v, w)$  has been deleted and a connection between  $(x, v)$  and  $(y, w)$  have been reconnected. The outcome of 2-opt move will be stored in the tabu list where it will stay there till some number of iteration which is based on the tabu list size.

After 2-opt move is completed, some nodes may not be connected. This is because in CARP cases, not all nodes are connected to each other. If we try to connect two nodes without considering the existing link between them, we might get infeasible answer. So to prevent from getting infeasible, then we need to insert an arc through another node to connect the remaining edge of the tour. This is called insertion which is our second step in exploring the neighbourhood.

Same concept goes to our third step of moves which is deletion. After doing 2-opt and insertion, some edge might be repeated continuously. This attribute possibly will affect our last result. Therefore to avoid the repetition of the edge, the repeated arc must be delete so that the total cost of travelling can be minimized. In a conclusion, our algorithm need three type of move which are 2-opt, insertion and deletion to explore new region of solution space.

#### **4.3.4 Aspiration Criterion**

An aspiration criterion is needed if all moves become tabu. For our TS algorithm, aspiration by default was chosen to free the tabu move so that the algorithm may escape from being trap. The move with the minimum fitness function will be freed from the tabu list. In other words, the tabu status of the move with the minimum fitness function will be overridden.

### 4.3.5 Stopping Criterion

Since the algorithm is open-ended and may run forever as an optimal solution is unknown, so the stopping criterion is needed. The TS algorithm will stop searching after some number of iteration which is determine by the experimentation that shown in Table 4.2.

**Table 4.2: Stopping Criterion**

| <i>n</i> | IS   | Maximum Iteration |          |           |
|----------|------|-------------------|----------|-----------|
|          |      | Max1=500          | Max2=750 | Max3=1000 |
| 5        | 380  | 380               | 380      | 380       |
| 6        | 330  | 330               | 330      | 330       |
| 7        | 490  | 430               | 430      | 430       |
| 8        | 680  | 620               | 620      | 620       |
| 9        | 820  | 760               | 760      | 770       |
| 10       | 660  | 750               | 570      | 610       |
| 11       | 890  | 890               | 890      | 890       |
| 12       | 1070 | 1070              | 1060     | 1070      |
| 13       | 1170 | 1140              | 1140     | 1160      |
| 14       | 1260 | 1220              | 1260     | 1220      |
| 15       | 1370 | 1370              | 1370     | 1370      |
| 20       | 1490 | 1390              | 1470     | 1330      |
| 25       | 2330 | 2330              | 2330     | 2330      |
| 30       | 2680 | 2680              | 2680     | 2680      |
| 50       | 1103 | 1103              | 1103     | 1103      |

As we can see in Table 4.2, even the algorithm is set to stop after 1000 iteration nevertheless it does not guarantees that it has reached a steady state. This is because it depends on the move. If from the beginning of the search process it has drive to an optimal solution, then a small number of iteration is sufficient enough to reach an

optimal solution. Hence, to ensure that the algorithm performs well and reach an optimal solution, then the stopping criterion is set to be maximum to 1000 iteration.

#### 4.3.6 Tabu Search Algorithm

The algorithm is needed in order to guide the process of the computational. In mathematics, an algorithm is an effective method for solving a problem using a finite sequence of instructions. Each algorithm is a list of well-defined instructions for completing a task. Starting from an initial state, the instructions describe a computation that proceeds through a well-defined series of successive states, eventually terminating in a final ending state.

For TS algorithm, the algorithm is starts with generating an initial solution. This initial solution generating by using CEM as described in Section 3.6.1 and the cost is calculate using CARP formulation. This phase is known as initial solution phase. The implementation of TS will starts in the second step. In the second step which is TS phase, the iteration number will be set and aspiration level,  $C_a$  will be initialise.

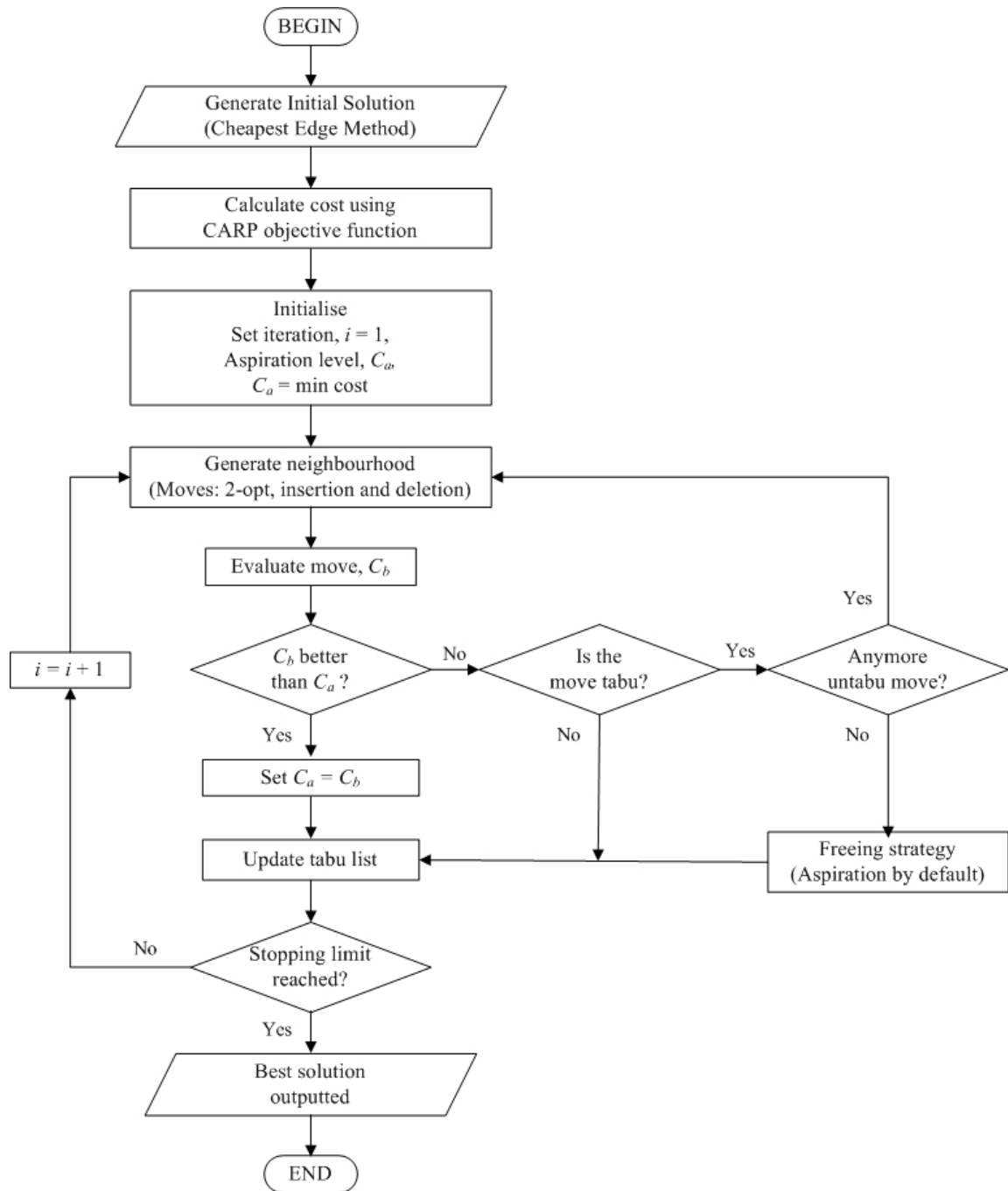
The third step is for searching the neighbourhood. In this step, the neighbourhood will be generate by performing a single combination of 2-opt move. Then an insertion and deletion procedure will be applied in order to get at a feasible route. The move will be evaluated to find the minimum cost,  $C_b$  in a next step. This cost is then will be check whether it is better than  $C_a$ . If  $C_b$  is better than  $C_a$ , then set  $C_a = C_b$ . Otherwise, this step will jump to Step 6.

After performing the fourth step and it satisfy the rule, then the process will continue with updating the tabu list size and checking for stopping criterion. If the stopping criterion is satisfied, then the iteration will go to the next iteration and the process in Step 3 will be followed again, or else, the best solution has been outputted.

Step 6 in TS algorithm is provided in order to check the move. If the move is not tabu, the process will go to Step 5 but if the move is tabu, then the process will search for another untabu move. If no more untabu move exist in the solution space, then the freeing strategy will be applied, but if untabu are still exist, the search process will back to Step 3. The process will repeatedly until the stopping criterion have been satisfied.

The algorithm will be described more clearly in a figure form which the whole process of TS algorithm can be summarized in Figure 4.4.





**Figure 4.4: The TS Algorithm**

The algorithm is given as follows:

**[Step 1]** Initial solution phase

- Generating the initial solution using Cheapest Edge Method
- Calculating the cost,  $C_a$ , using CARP formulation

**[Step 2]** TS phase

- Set iteration,  $i = 1$
- Set aspiration level =  $C_a$

**[Step 3]** Neighbourhood search

- Perform a single combination of 2-opt move
- Apply insertion and deletion procedures to reach at a feasible route
- Evaluate move by finding the minimum cost,  $C_b$

**[Step 4]** Checking the result

- Check whether  $C_b$  is better than  $C_a$
- If  $C_b$  is better than  $C_a$  then,
- Set  $C_a = C_b$
- Else go to **[Step 6]**

**[Step 5]** Updating tabu list

- Update the tabu list
- Check for stopping rule
- If continue, set  $i = i + 1$ , and go to **[Step 3]**
- Else best solution outputted

**[Step 6]** Checking the move

- If the move is tabu, then
- Search the other untabu move
- Else go to **[Step 5]**
- If no more untabu move, then
- Apply freeing strategy, accept the move and go to **[Step 5]**
- Else go to **[Step 3]**

#### 4.4 Computational Results

Both LAS and TS techniques have been tested on a set of instances generated from simulated data. A total of 45 instances within the range of five to 50 nodes have been used to test the algorithm and divided into three groups which is based on the maximum capacity of the vehicle. Table 4.3 shows all the instances used and the labelled for each of the problem according to their size of node and the maximum capacity of the vehicle capacity.

**Table 4.3: Type of Problem**

| <b>Size of Node</b> | <b>5 tonne</b> | <b>9 tonne</b> | <b>15 tonne</b> |
|---------------------|----------------|----------------|-----------------|
| <b>5</b>            | A1             | B1             | C1              |
| <b>6</b>            | A2             | B2             | C2              |
| <b>7</b>            | A3             | B3             | C3              |
| <b>8</b>            | A4             | B4             | C4              |
| <b>9</b>            | A5             | B5             | C5              |
| <b>10</b>           | A6             | B6             | C6              |
| <b>11</b>           | A7             | B7             | C7              |
| <b>12</b>           | A8             | B8             | C8              |
| <b>13</b>           | A9             | B9             | C9              |
| <b>14</b>           | A10            | B10            | C10             |
| <b>15</b>           | A11            | B11            | C11             |
| <b>20</b>           | A12            | B12            | C12             |
| <b>25</b>           | A13            | B13            | C13             |
| <b>30</b>           | A14            | B14            | C14             |
| <b>50</b>           | A15            | B15            | C15             |

#### 4.4.1 Look-Ahead Strategy Computational Results

Using the LAS approach proposed by Ismail *et al.*, (2007), the same problems have been used to generate the result. Table 4.4 shows the computational results given from LAS algorithm.

**Table 4.4: LAS Computational Results**

| <b>Problem</b> | <b>LAS</b> |
|----------------|------------|
| A1             | 400        |
| A2             | 470        |
| A3             | 460        |
| A4             | -          |
| A5             | -          |
| A6             | -          |

As we can see in Table 4.4, only 6 type of problem which only 5 tonne of vehicle capacity with the number of node less than 10 can be solved by LAS proposed by Ismail, *et al.* But unfortunately for our data, LAS failed to give the answer for problem A4, A5 and A6. This is because there exist an empty movement edge with length two or above and LAS failed to compute since its only allow an empty movement edge with length one, means the vehicle can only passes through an arc without servicing it once. A lot of an empty movement may occur in some problem due to its network. The routing for some of them may be too complicated means that the vehicle needs to pass through the same arc three to four or even five times in order to reach the unvisited arc or to complete the tour. It might because that was the only way to get the unvisited arc or it might also be in term of the cost.

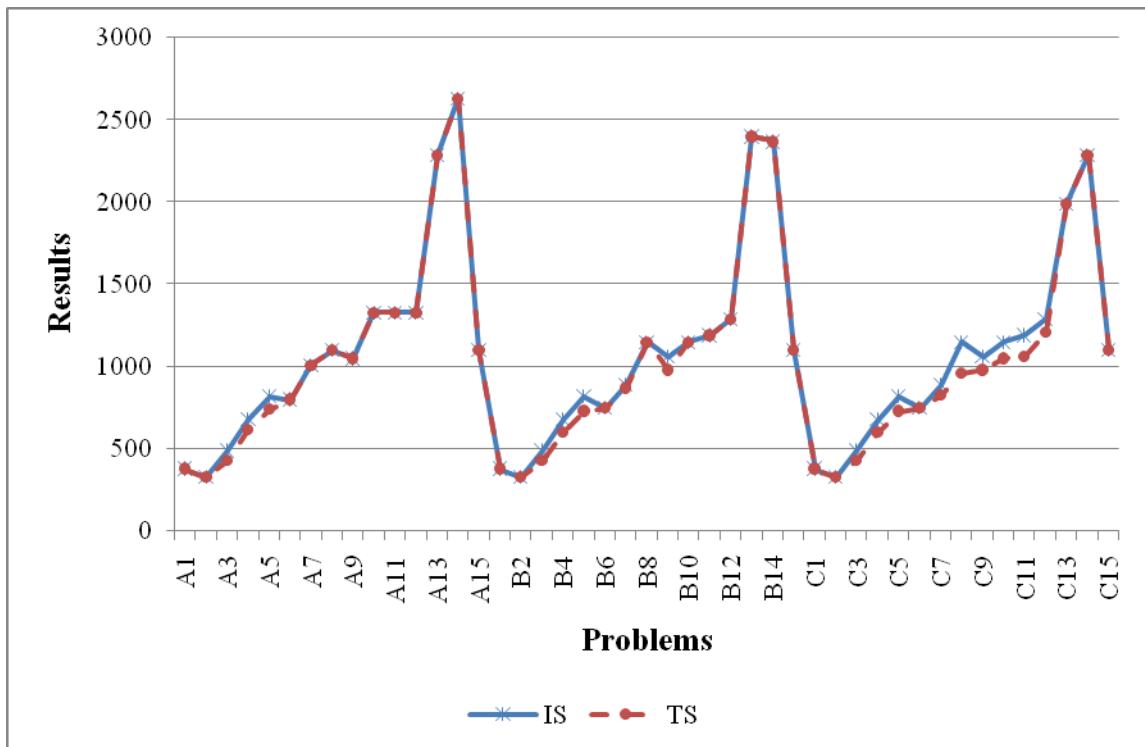
#### 4.4.2 Tabu Search Computational Results

A variety number in range five to 50 number of nodes with three size of the vehicle limit that describe earlier have been used to test the performance of TS algorithm. The results generated by TS and also percentage of improvement from initial solution are shown in Table 4.5.

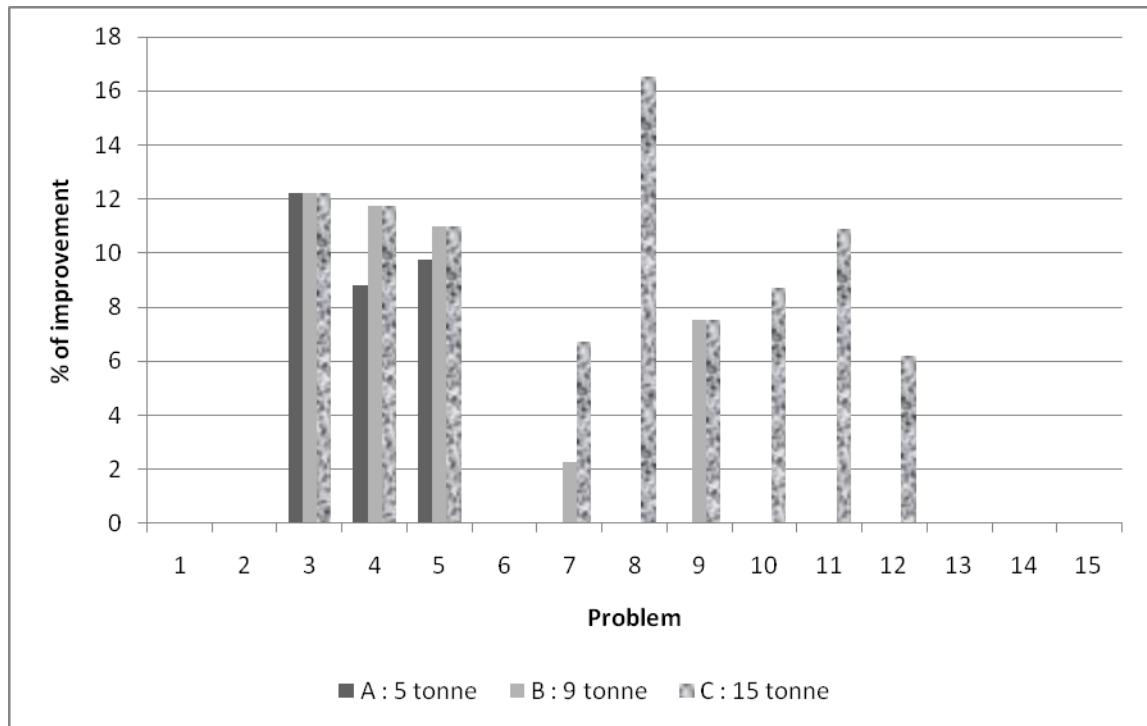
**Table 4.5: TS Computational Results**

| Problem | A : 5 tonne |      |       | B : 9 tonne |      |       | C : 15 tonne |      |       |
|---------|-------------|------|-------|-------------|------|-------|--------------|------|-------|
|         | IS          | TS   | %     | IS          | TS   | %     | IS           | TS   | %     |
| 1       | 380         | 380  | 0     | 380         | 380  | 0     | 380          | 380  | 0     |
| 2       | 330         | 330  | 0     | 330         | 330  | 0     | 330          | 330  | 0     |
| 3       | 490         | 430  | 12.24 | 490         | 430  | 12.24 | 490          | 430  | 12.24 |
| 4       | 680         | 620  | 8.82  | 680         | 600  | 11.76 | 680          | 600  | 11.76 |
| 5       | 820         | 740  | 9.76  | 820         | 730  | 10.98 | 820          | 730  | 10.98 |
| 6       | 800         | 800  | 0     | 750         | 750  | 0     | 750          | 750  | 0     |
| 7       | 1010        | 1010 | 0     | 890         | 870  | 2.25  | 890          | 830  | 6.74  |
| 8       | 1100        | 1100 | 0     | 1150        | 1150 | 0     | 1150         | 960  | 16.52 |
| 9       | 1050        | 1050 | 0     | 1060        | 980  | 7.55  | 1060         | 980  | 7.55  |
| 10      | 1330        | 1330 | 0     | 1150        | 1150 | 0     | 1150         | 1050 | 8.70  |
| 11      | 1330        | 1330 | 0     | 1190        | 1190 | 0     | 1190         | 1060 | 10.92 |
| 12      | 1330        | 1330 | 0     | 1290        | 1290 | 0     | 1290         | 1210 | 6.20  |
| 13      | 2290        | 2290 | 0     | 2400        | 2400 | 0     | 1990         | 1990 | 0     |
| 14      | 2630        | 2630 | 0     | 2370        | 2370 | 0     | 2290         | 2290 | 0     |
| 15      | 1100        | 1100 | 0     | 1100        | 1100 | 0     | 1100         | 1100 | 0     |

From the table, we can see the percentage of improvement using TS. It clearly shows that not all run will generate an improvement in the search. This is due to the already good initial solution generated using CEM method. But there are still have an improvement on the others because CEM is not the perfect method to give the optimal solution. Hence, we need to test with TS, perhaps it might have other better solution than CEM. For clearly observation, refer to Figure 4.5 and Figure 4.6. Figure 4.5 shows a comparison results between IS and TS in graph while Figure 4.6 shows a percentage of the improvement for TS computational results.



**Figure 4.5: Comparison between IS and TS**



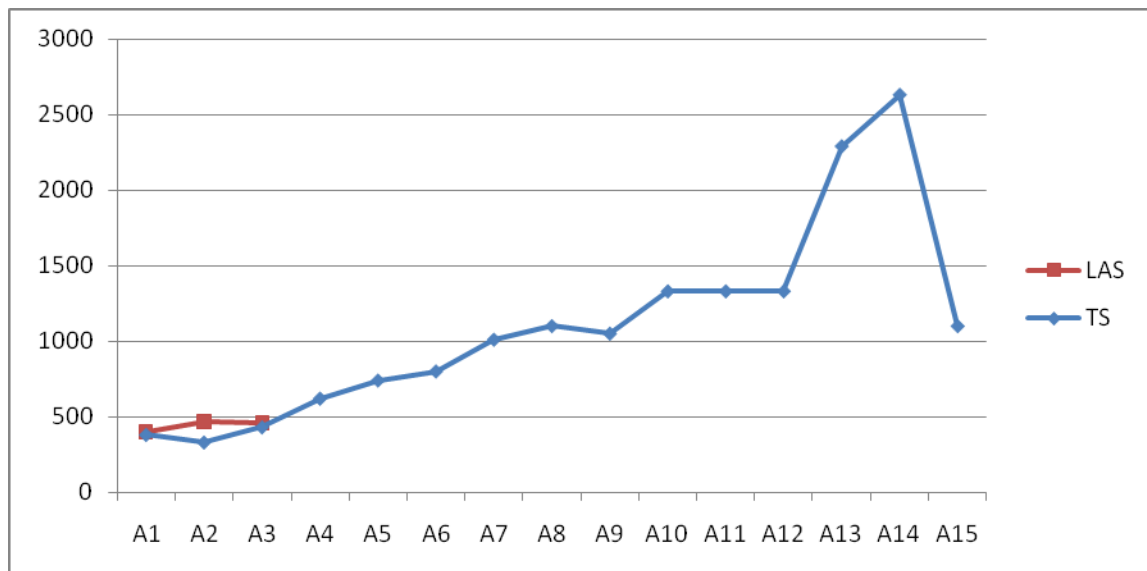
**Figure 4.6: Percentage of Improvement for TS Computational Results**

#### 4.4.3 Look-Ahead Strategy versus Tabu Search

As mention before, these two methods (LAS and TS) are used for comparing purposes. As a result, Table 4.6 shows the comparison result between LAS and TS, while Figure 4.7 illustrates the results in graph.

**Table 4.6: Comparison between LAS and TS**

| Problem | LAS      | TS   |
|---------|----------|------|
| A1      | 400      | 380  |
| A2      | 470      | 330  |
| A3      | 460      | 430  |
| A4      | $\infty$ | 620  |
| A5      | $\infty$ | 740  |
| A6      | $\infty$ | 800  |
| A7      | -        | 1010 |
| A8      | -        | 1100 |
| A9      | -        | 1050 |
| A10     | -        | 1330 |
| A11     | -        | 1330 |
| A12     | -        | 1330 |
| A13     | -        | 2290 |
| A14     | -        | 2630 |
| A15     | -        | 1100 |

**Figure 4.7: Comparison between LAS and TS**



These results clearly show the weaknesses of LAS method that it cannot compute a large number of nodes. The maximum number we can use in LAS only 10 nodes. Other than that, it also shows that results given by LAS are not the perfect one because the results produce by TS is much better. One more weaknesses on LAS is that it cannot compute the servicing cost if the vehicle need to pass through an empty edge again and again without servicing it in order to complete the route or to reach an edge that not been served. In order words, LAS can only perform well for simple network.

#### **4.5 Summary**

This chapter described a little bit about the LAS method and described clearly about the development of TS model. At the end of this chapter, we can see a computational results and a comparison between these two methods. Next, we will discuss about the development of RTS algorithm which is develop by modification from TS algorithm which describe in this chapter in certain term.

## CHAPTER 5

### SOLUTIONS BASED ON REACTIVE TABU SEARCH

#### 5.1 Introduction

Chapter five presents a Reactive Tabu Search (RTS) algorithm developed to solve a Capacitated Arc Routing Problem (CARP). RTS scheme was based on theory of dynamical systems which was introduced by Battiti and Tecchiolli (1994). This approach maintains the basic ideas of Tabu Search (TS) with changes in the features of selecting tabu size. It dynamically controls the size of tabu list. The search mechanism in TS is not sufficient to arrive at the global optimum and so of that an escape strategy for diversifying the search process was introduced. According to Wassan *et al.*, 2008, RTS is robust enough to overcome tabu list problem and has little effect of parameter changes. This chapter begins with the discussion on the implementation of RTS, followed by the discussion on the algorithm and finally a discussion on computational results.

## 5.2 Reactive Tabu Search Implementation

Generally RTS maintain the basic concept of TS algorithm. RTS modified the tabu list size and allow repetition to diversify the exploration of searching the neighbourhood in order to widen the solution space. Table 5.1 explains more clearly the differences between TS and RTS algorithm.

**Table 5.1: The Differences between TS and RTS**

| Parameter                       | TS   | RTS   |
|---------------------------------|--|---|
| Initial Solution                | Generated using Cheapest Edge Method (CEM)   | Same as TS  |
| Neighbourhood Structure (Moves) | A single combination of 2-opt move<br>Then insertion and deletion to reach at feasible route | Same as TS  |
| Tabu List Size $ T_s $          | Set to be static, which is equal to the number of the nodes, $n$                             | Set to be dynamic<br>Initially,<br><ul style="list-style-type: none"> <li>for <math>n &lt; 10</math> then <math> T_s </math> is set to <math>2n</math></li> <li>for <math>n \geq 10</math> then <math> T_s </math> is set to <math>(n - 3)n</math></li> </ul> Then,<br><ul style="list-style-type: none"> <li>if solution improve; increase <math> T_s </math> up to 20% of the current <math> T_s </math></li> <li>if all moves become tabu; decrease 20% of the current <math> T_s </math></li> </ul> |

|                          |  |  |
|--------------------------|--|--|
| Aspiration Criterion     | Aspiration by default that is freeing the least cost move (check at the fitness function and free the minimum one) | Same as TS   |
| Diversification Strategy | Not apply  | Diversification is apply on the repeated solution<br>Diversify if there is no improvement;<br><ul style="list-style-type: none"> <li>• for <math>2n</math> times (if <math>n \leq 10</math>)</li> <li>• for <math>3n</math> times (if <math>n \geq 10</math>)</li> </ul> |
| Stopping Criterion       | Confine maximum iteration up to 1000   | Two criterion for stopping:<br>1. After algorithm complete diversify the neighbourhood<br>2. Maximum iteration is set to be 500<br>The search stops whichever criterion is met first   |

The differences between TS and RTS have been described clearly in Table 5.1. From the table, the differences between both methods are tabu list size, diversification strategy and stopping criterion. These three elements will be described separately in the next sub-section.

### 5.2.1 Tabu List Size



Table 5.2(b): Investigation on Tabu List Size and Repetition

| n  | IS   | RTS         |            |            |            |            |            |             |
|----|------|-------------|------------|------------|------------|------------|------------|-------------|
|    |      | RTS9        | RTS10      | RTS11      | RTS12      | RTS13      | RTS14      | RTS15       |
| 5  | 380  | 380         | 380        | 380        | 380        | 380        | 380        | 380         |
| 6  | 330  | 330         | 330        | 330        | 330        | 330        | 330        | 330         |
| 7  | 490  | 430         | 490        | 490        | 490        | <b>430</b> | 490        | 490         |
| 8  | 680  | 600         | 670        | <b>600</b> | <b>600</b> | 620        | <b>600</b> | 670         |
| 9  | 820  | 760         | 770        | <b>710</b> | 820        | 820        | <b>710</b> | 730         |
| 10 | 660  | 570         | 660        | 660        | 660        | 660        | <b>530</b> | <b>530</b>  |
| 11 | 890  | 890         | <b>830</b> | <b>830</b> | 870        | 870        | 890        | <b>830</b>  |
| 12 | 1070 | 1060        | 1070       | 1060       | 1070       | 1070       | 1060       | <b>1000</b> |
| 13 | 1170 | 1170        | 1170       | 1170       | 990        | 1110       | 1170       | <b>940</b>  |
| 14 | 1260 | <b>1050</b> | 1260       | 1090       | 1200       | 1170       | 1220       | <b>1050</b> |
| 15 | 1370 | 1370        | 1370       | 1370       | 1370       | 1370       | 1370       | <b>1260</b> |
| 20 | 1490 | 1390        | 1490       | 1410       | 1320       | 1490       | 1450       | <b>1270</b> |
| 25 | 2330 | 2330        | 2330       | 2330       | 2330       | 2330       | 2330       | 2330        |
| 30 | 2680 | 2680        | 2680       | 2680       | 2680       | 2680       | 2680       | 2680        |
| 50 | 1103 | 1103        | 1103       | 1103       | 1103       | 1103       | 1103       | 1103        |

RTS1 :  $|T_s| = n$ , Repetition =  $n$

RTS2 :  $|T_s| = n$ , Repetition =  $2n$

RTS3 :  $|T_s| = n$ , Repetition =  $3n$

RTS4 :  $|T_s| = 2n$ , Repetition =  $n$

RTS5 :  $|T_s| = 2n$ , Repetition =  $2n$

RTS6 :  $|T_s| = 2n$ , Repetition =  $3n$

RTS7 :  $|T_s| = 3n$ , Repetition =  $n$

RTS8 :  $|T_s| = 3n$ , Repetition =  $2n$

RTS9 :  $|T_s| = 3n$ , Repetition =  $3n$

RTS10 :  $|T_s| = (n - 2)n$ , Repetition =  $n$

RTS11 :  $|T_s| = (n - 2)n$ , Repetition =  $2n$

RTS12 :  $|T_s| = (n - 2)n$ , Repetition =  $3n$

RTS13 :  $|T_s| = (n - 3)n$ , Repetition =  $n$

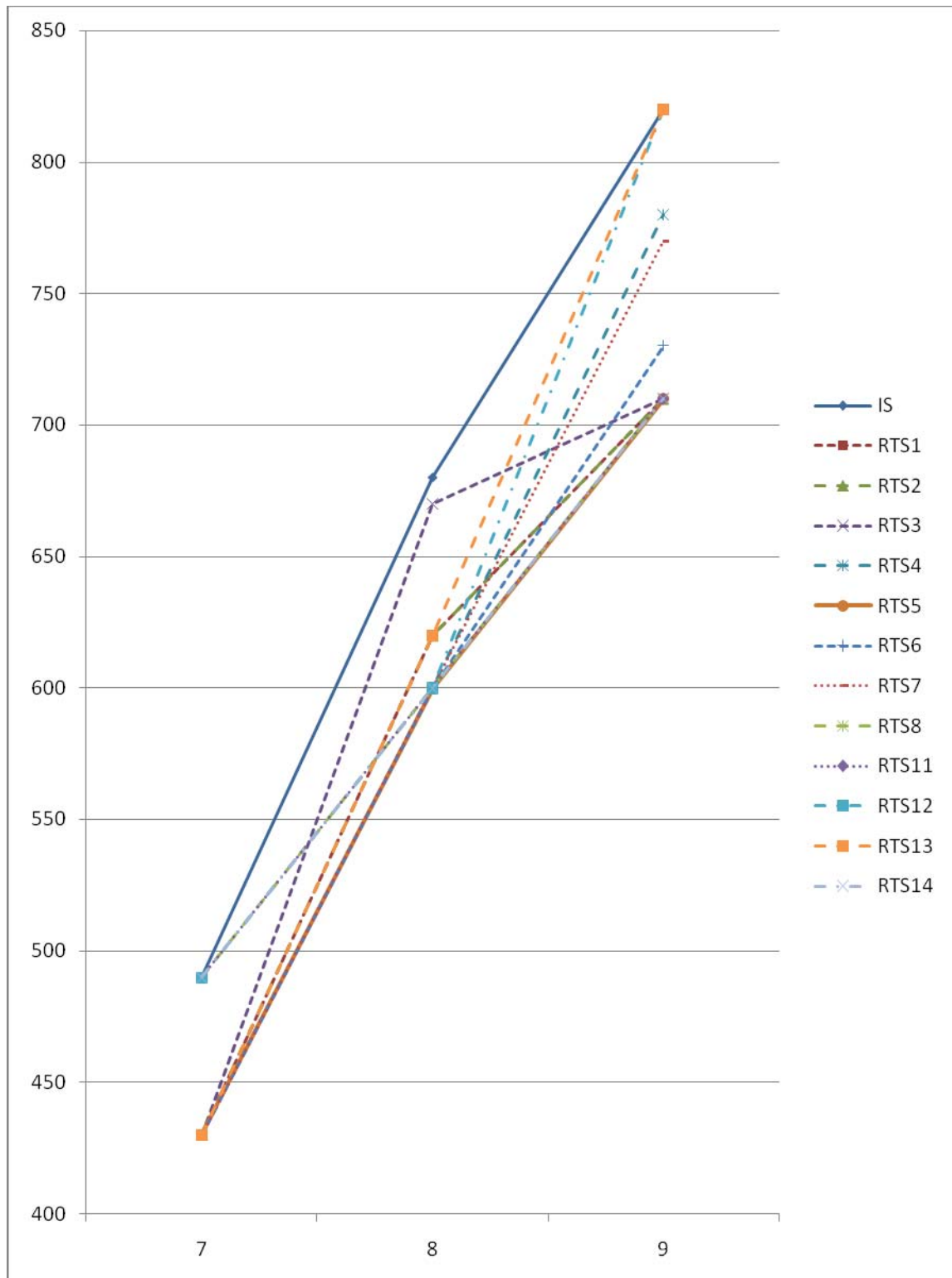
RTS14 :  $|T_s| = (n - 3)n$ , Repetition =  $2n$

RTS15 :  $|T_s| = (n - 3)n$ , Repetition =  $3n$

Where  $n = 5, 6, 7, \dots, 50$

From Table 5.2(a) and Table 5.2(b), numbers in bold shows the minimum solution that can be reached by RTS algorithm. The result shows that RTS15 gives a bigger number of solutions with the best fitness values. There are 46.67 percent instances give an improvement from an initial solution, while for other tabu list of size, the improvement is rather limited between two or four instances only. Although RTS15 give majority best solution among others, but there are only for instances with the number of node more than and equal to 10. For instances with the node less than 10, RTS15 are not be up to much in finding other solution that less than the initial solution while in other tabu list size, an improvement occurred. By considering instances with the number of node less than 10, RTS5 give majority best solution with three out of five instances give an improvement from the initial solution.

Figure 5.1 shows the specific comparison based on investigation on tabu list size for nodes less than 10. While Figure 5.2 shows the specific comparison based on investigation on tabu list size for 10 number of nodes and above. These graph show that RTS5 and RTS15 always reach at minimum solution compared to other which only some problem manage to give an improvement.



**Figure 5.1: Comparison on Tabu List Size and Repetition for  $n < 10$**



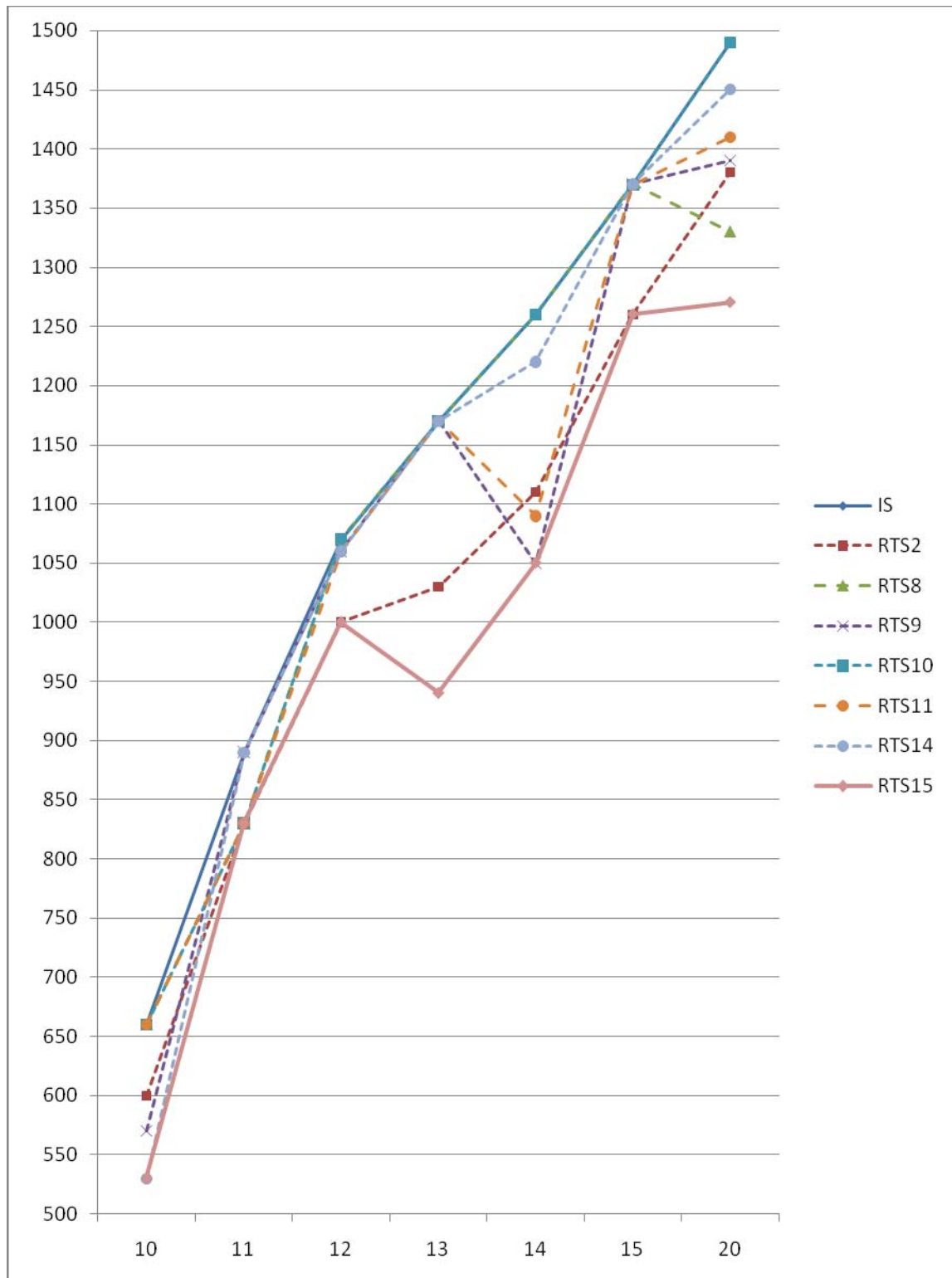


Figure 5.2: Comparison on Tabu List Size and Repetition for  $n \geq 10$

Based on this investigation, RTS5, ( $|T_s| = 2n$ ), was selected to be as tabu list size for the number of nodes less than 10 and RTS15, ( $|T_s| = (n - 3)n$ ), to be as tabu list size for the rest of it. The selections of tabu list size are given in Table 5.3.

**Table 5.3: Tabu List Size**

| Range of Nodes | Tabu List Size, $ T_s $ |
|----------------|-------------------------|
| $<10$          | $2n$                    |
| $\geq 10$      | $(n - 3)n$              |

Due to the dynamic value of the tabu list size, it changes during the search process. This list is known as a dynamic tabu list size. From this study, it shows that 20 percent of the current tabu list size will increase if there is an improvement on the solution but the list size reduces by 20 percent if all the movement becomes tabu. The values of this attribute were assigned based on the investigation tested for dynamically of tabu list size as shown in Table 5.4.

**Table 5.4: Investigation on Dynamic Tabu List Size**

| <b>n</b> | <b>IS</b> | <b>d1</b>   | <b>d2</b>  | <b>d3</b>  | <b>d4</b>  | <b>d5</b>   | <b>d6</b>   | <b>d7</b>  | <b>d8</b>   | <b>d9</b>  |
|----------|-----------|-------------|------------|------------|------------|-------------|-------------|------------|-------------|------------|
| 5        | 380       | 380         | 380        | 380        | 380        | 380         | 380         | 380        | 380         | 380        |
| 6        | 330       | 330         | 330        | 330        | 330        | 330         | 330         | 330        | 330         | 330        |
| 7        | 490       | <b>430</b>  | <b>430</b> | <b>430</b> | <b>430</b> | 490         | <b>430</b>  | <b>430</b> | 490         | <b>430</b> |
| 8        | 680       | <b>600</b>  | <b>600</b> | <b>600</b> | 620        | <b>600</b>  | <b>600</b>  | <b>600</b> | <b>600</b>  | <b>600</b> |
| 9        | 820       | <b>710</b>  | <b>710</b> | <b>710</b> | <b>710</b> | <b>710</b>  | <b>710</b>  | <b>710</b> | 820         | 820        |
| 10       | 660       | <b>530</b>  | 600        | <b>530</b> | 660        | 660         | 660         | 660        | 570         | 600        |
| 11       | 890       | <b>830</b>  | <b>830</b> | 890        | <b>830</b> | <b>830</b>  | <b>830</b>  | 890        | <b>830</b>  | <b>830</b> |
| 12       | 1070      | <b>1000</b> | 1070       | 1070       | 1070       | 1070        | 1060        | 1070       | 1070        | 1070       |
| 13       | 1170      | <b>940</b>  | 1000       | 1000       | 1170       | 1170        | 1170        | 1170       | 1170        | 1170       |
| 14       | 1260      | <b>1050</b> | 1210       | 1130       | 1200       | <b>1050</b> | <b>1050</b> | 1130       | <b>1050</b> | 1260       |
| 15       | 1370      | <b>1260</b> | 1370       | 1370       | 1370       | 1370        | 1370        | 1370       | 1370        | 1370       |
| 20       | 1490      | <b>1270</b> | 1410       | 1390       | 1370       | 1450        | 1410        | 1390       | 1320        | 1390       |
| 25       | 2330      | 2330        | 2330       | 2330       | 2330       | 2330        | 2330        | 2330       | 2330        | 2330       |
| 30       | 2680      | 2680        | 2680       | 2680       | 2680       | 2680        | 2680        | 2680       | 2680        | 2680       |
| 50       | 1103      | 1103        | 1103       | 1103       | 1103       | 1103        | 1103        | 1103       | 1103        | 1103       |

d1 : increase 20%, decrease 20%

d6 : increase 50%, decrease 70%

d2 : increase 20%, decrease 50%

d7 : increase 70%, decrease 20%

d3 : increase 20%, decrease 70%

d8 : increase 70%, decrease 50%

d4 : increase 50%, decrease 20%

d9 : increase 70%, decrease 70%

d5 : increase 50%, decrease 50%

Table 5.4 shows the results from an investigation conducted on dynamic tabu list size. The percentage of increasing and decreasing the tabu list size is based on experimentation and results produced by the RTS algorithm. As shown in Table 5.4, d1 give the best performance among others. The results produced by d1 improved up to 66.67 percent. On contrary, the results produced by others improved between 20 to 33.33 percent only. Therefore, d1 was chosen to be a size for dynamic tabu list. By selecting d1, tabu list size will increase 20 percent when a better solution found and decrease 20 percent when all the movement become tabu.

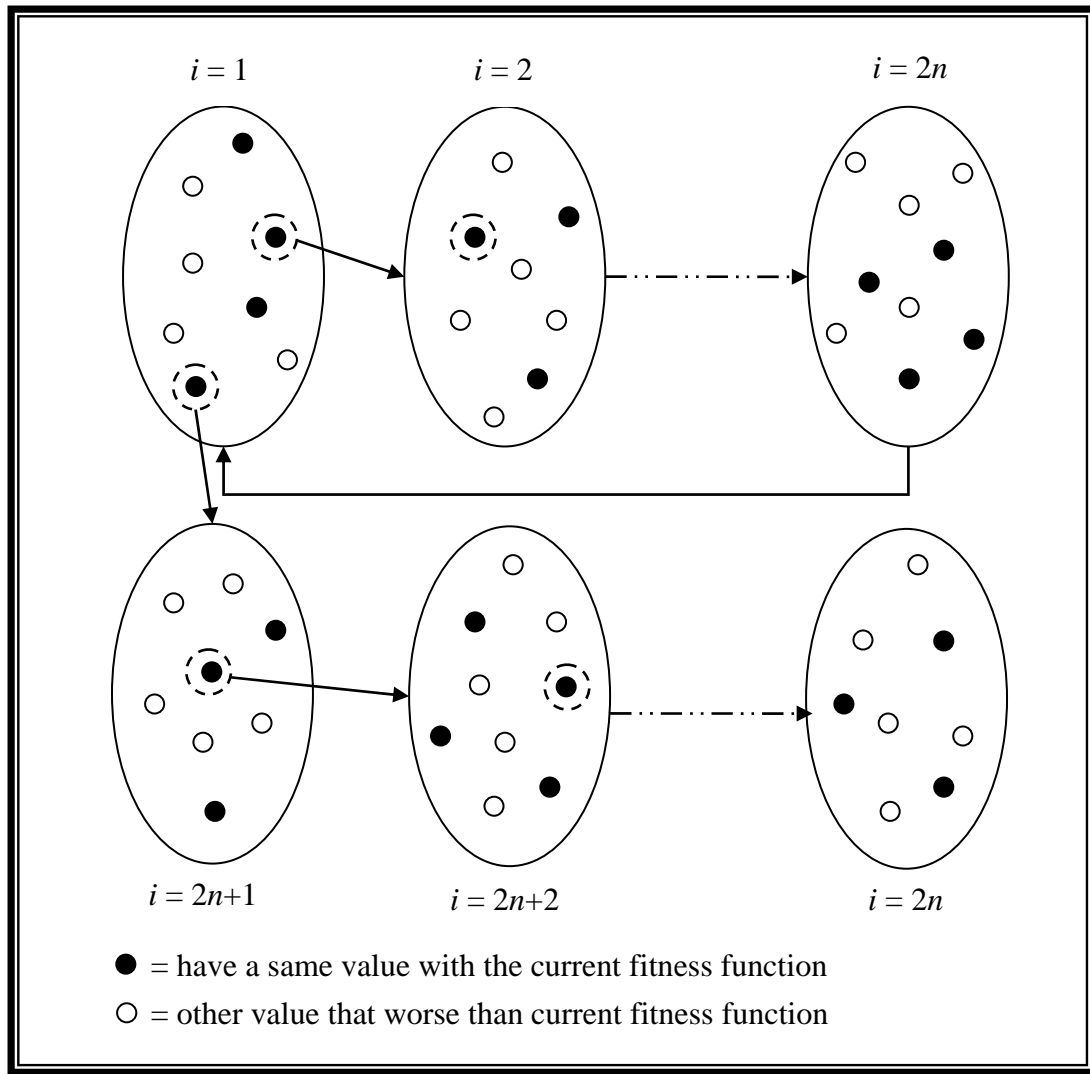
### 5.2.2 Diversification Strategy

In RTS algorithm, the diversification strategy is applied to allow for repetition to occur. The idea comes when the number of repetition occurs too many times in the solution space. A searching trajectory seeks to jump out from current solution to another solution after certain number of none improving iteration. The experimentation was carried out simultaneously with the experimentation of tabu list size which the results are provided in Table 5.2(a) and Table 5.2(b). A different number of repetition was assigned to two different groups of nodes, namely group with nodes less than 10 and group with nodes more than or equal to 10. For  $n < 10$ , the best time to diversify the search process is when there is no improvement for  $2n$  times. Mean while for  $n \geq 10$ , the search process will diversify when there is no improvement for  $3n$  times. The best time to diversify is given in Table 5.5.

**Table 5.5: Repetition**

| Range of Nodes | Repetition |
|----------------|------------|
| $<10$          | $2n$       |
| $\geq 10$      | $3n$       |

The diversification are done by undo the iteration to the first detected of non-improving moves and choose the other route that have the same fitness function for searching another trajectory to get the best possible solution. Figure 5.3 shows an illustration on how the repetition is made.



**Figure 5.3: Illustration on Diversification Strategy**

The algorithm needs to diversify if there is no improvement in the fitness function after some number of iteration. As illustrate in Figure 5.3, let say if there is no improvement on the current fitness function from the first iteration, then after  $2n$  iteration (example on number of nodes less than 10), the algorithm will undo and back to first iteration and seek for another solution that have a same value with the current fitness function and start exploring their moves. This diversification will continue until all the possible solution that have same fitness function had been explored.

Table 5.6 shows a comparing result between without the application of diversification and with the application of diversification. For addition, these results have been illustrated in Figure 5.4 and Figure 5.5.

**Table 5.6: Investigation on Diversification Strategy**

| n  | Without Diversify |      |                  | With Diversify |      |                  |
|----|-------------------|------|------------------|----------------|------|------------------|
|    | IS                | RTS  | % of improvement | IS             | RTS  | % of improvement |
| 5  | 380               | 380  | 0                | 380            | 380  | 0                |
| 6  | 330               | 330  | 0                | 330            | 330  | 0                |
| 7  | 490               | 430  | 12.24            | 490            | 430  | 12.24            |
| 8  | 680               | 620  | 8.82             | 680            | 600  | 11.76            |
| 9  | 820               | 820  | 0                | 820            | 710  | 13.41            |
| 10 | 660               | 600  | 9.09             | 660            | 530  | 19.70            |
| 11 | 890               | 890  | 0                | 890            | 830  | 6.74             |
| 12 | 1070              | 1070 | 0                | 1070           | 1000 | 6.54             |
| 13 | 1170              | 1140 | 2.56             | 1170           | 940  | 19.66            |
| 14 | 1260              | 1220 | 3.17             | 1260           | 1050 | 16.67            |
| 15 | 1370              | 1320 | 3.65             | 1370           | 1260 | 8.03             |
| 20 | 1490              | 1390 | 6.71             | 1490           | 1270 | 14.77            |
| 25 | 2330              | 2330 | 0                | 2330           | 2330 | 0                |
| 30 | 2680              | 2680 | 0                | 2680           | 2680 | 0                |
| 50 | 1103              | 1103 | 0                | 1103           | 1103 | 0                |

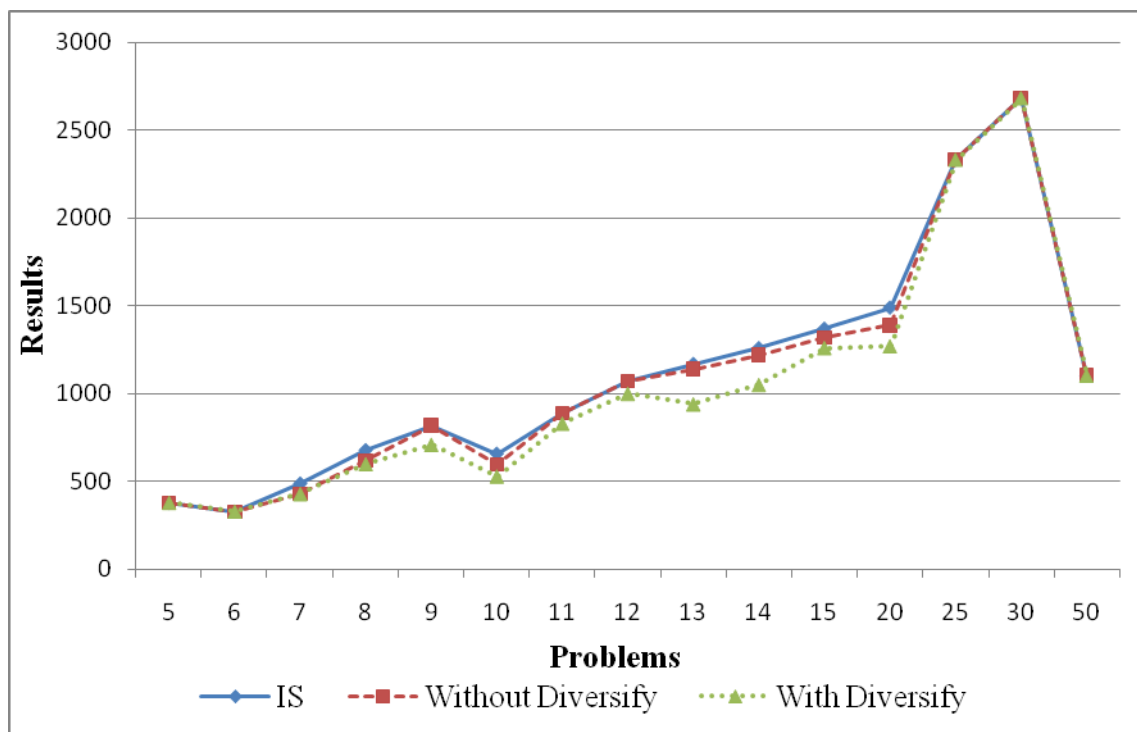


Figure 5.4: Comparison on a Result between Diversify and Not Diversify

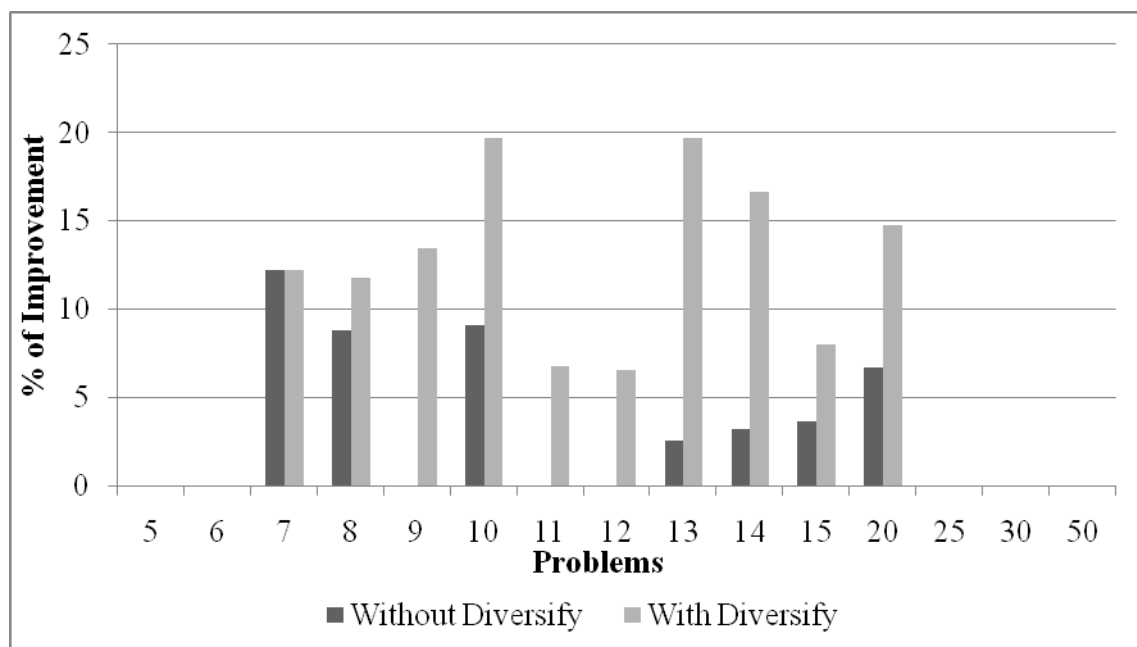


Figure 5.5: Comparison on Percentage of Improvement between Diversify and Not Diversify

Regarding the results provided in Table 5.6 and as clearly shown in Figure 5.4, results produce by the application of diversification always give the better performance compared without the application of diversification. While Figure 5.5 shows that, without the application of diversification, RTS still can present an improvement on initial solution but results with the application of diversification is much more improved. Summarization for percentage of improvement as reported in Table 5.7 shows that with the application of diversification, each node improve in average up to 8.63 percent while without the application, the improvement only at 3.08 percent. This shows that, the application of diversification is useful in exploring and searching the best solution for RTS algorithm.

**Table 5.7: Summary on Diversification Strategy**

|                          | <b>Without Diversify</b> | <b>With Diversify</b> |
|--------------------------|--------------------------|-----------------------|
| Total % of improvement   | 46.24                    | 129.52                |
| Average % of improvement | 3.08                     | 8.63                  |

### 5.2.3 Stopping Criterion

Since RTS is one of an iterative method to compute successive approximation to the solution, then a stopping criterion is needed to prevent the algorithm from running forever. Because of that, choosing the good stopping criterion is very important. In RTS, two ways to stop the algorithm had been used. First, the algorithm needs to explore the repetition in the iteration before it could stop. Mean that the algorithm has to diversify by exploring all the repetition in solution space before the stopping criterion can be applied. However, sometimes in several situations, the algorithm may running



seems like there are no ending solution. In preventing the algorithm from running forever, second criterion is used. This second criterion will ask the algorithm to stop searching if it reaches at maximum iteration which is to be 500.

Table 5.8 shows the experimental results on the number of maximum iteration. Its shows that the algorithm gives the same fitness function for different size of maximum iteration, which are 500, 750 and 1000. This means that, 500 iterations is enough for the algorithm to reach at the best fitness function.

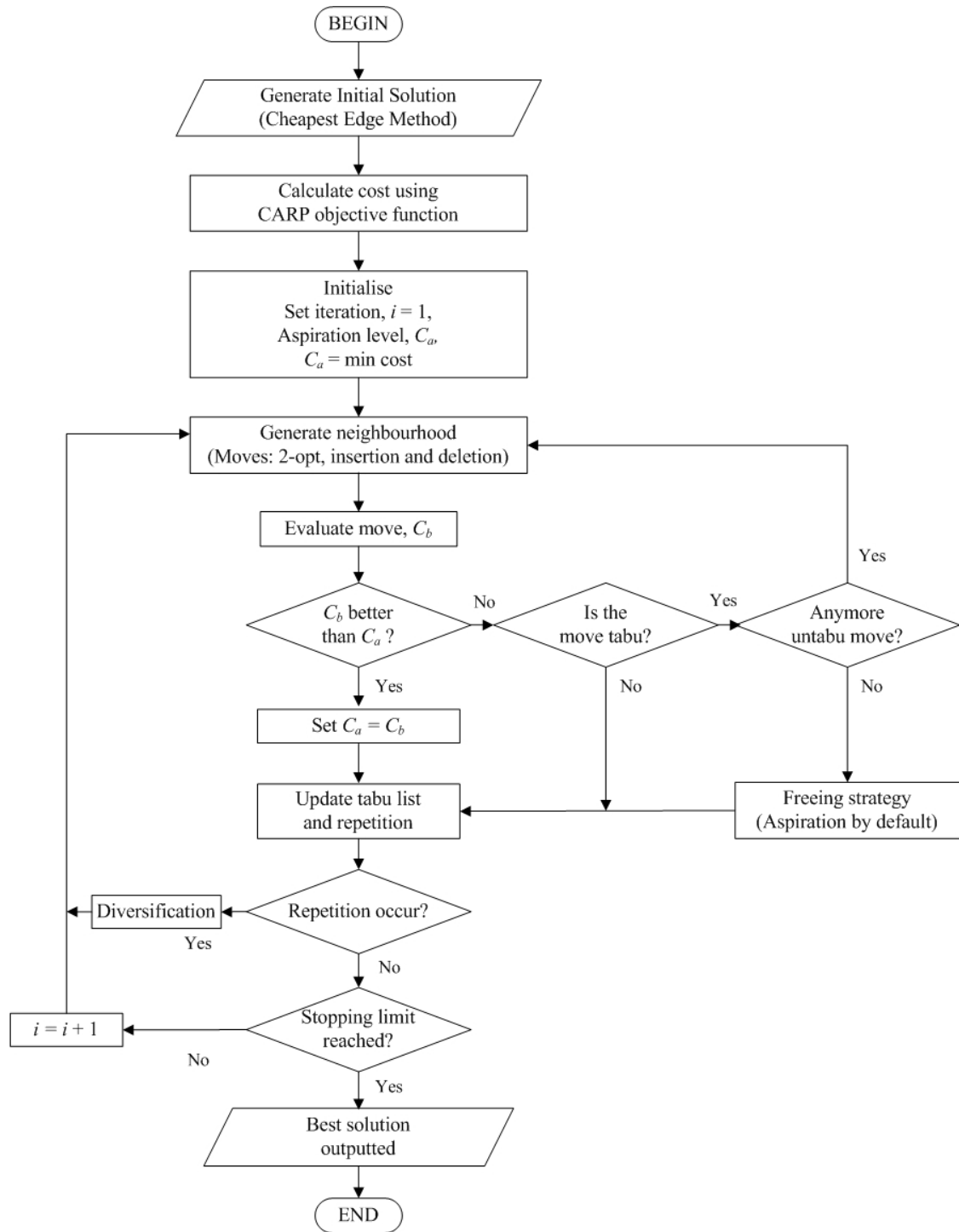
**Table 5.8: Investigation on Maximum Iteration**

| n  | IS   | Maximum Iteration |      |      |
|----|------|-------------------|------|------|
|    |      | 500               | 750  | 1000 |
| 5  | 380  | 380               | 380  | 380  |
| 6  | 330  | 330               | 330  | 330  |
| 7  | 490  | 430               | 430  | 430  |
| 8  | 680  | 600               | 600  | 600  |
| 9  | 820  | 710               | 710  | 710  |
| 10 | 660  | 530               | 530  | 530  |
| 11 | 890  | 830               | 830  | 830  |
| 12 | 1070 | 1000              | 1000 | 1000 |
| 13 | 1170 | 940               | 940  | 940  |
| 14 | 1260 | 1050              | 1050 | 1050 |
| 15 | 1370 | 1260              | 1260 | 1260 |
| 20 | 1490 | 1270              | 1270 | 1270 |
| 25 | 2330 | 2330              | 2330 | 2330 |
| 30 | 2680 | 2680              | 2680 | 2680 |
| 50 | 1103 | 1103              | 1103 | 1103 |

### **5.3 Reactive Tabu Search Algorithm**

As earlier state, RTS development is made by modification from TS algorithm. So the implementation of the algorithm is almost similar with TS. The difference is only in a part of repetition (in Step 7) where the implementation of diversification strategy is applied. The repetition is done by undo the iteration and randomly selects the different route that have the same fitness function. Then the process will turn back to Step 3 and start searching the neighbourhood again and so on.

Since the tabu list size in RTS implementation is set to dynamically change during the search process, so the additional on when to increase and decrease the tabu list size is added in Step 4 and Step 6, respectively. The complete RTS algorithm is shown in Figure 5.6.



**Figure 5.6: The RTS Algorithm**

The details descriptions of the algorithm are as follows:

**[Step 1]** Initial solution phase

- Generating the initial solution using CEM
- Calculating the cost,  $C_a$ , using CARP formulation

**[Step 2]** TS phase

- Set iteration,  $i = 1$
- Set aspiration level =  $C_a$

**[Step 3]** Neighbourhood search

- Perform a single combination of 2-opt move
- Apply insertion and deletion procedures to reach at a feasible route
- Evaluate move by finding the minimum cost,  $C_b$

**[Step 4]** Checking the result

- Check whether  $C_b$  is better than  $C_a$
- If  $C_b$  is better than  $C_a$  then,
- Set  $C_a = C_b$
- Increase the tabu list size
- Else go to **[Step 6]**

**[Step 5]** Updating tabu list

- Update the tabu list and repetition
- If repetition occur, go to **[Step 7]**
- Check for stopping rule
- If continue, set  $i = i + 1$ , and go to **[Step 3]**
- Else best solution outputted

**[Step 6]** Checking the move

- If the move is tabu, then
- Search the other untabu move
- Else go to **[Step 5]**
- If no more untabu move, then
- Apply freeing strategy, accept the move
- Decrease the tabu list size and go to **[Step 5]**

- Else go to [**Step 3**]

[**Step 7**] Repetition

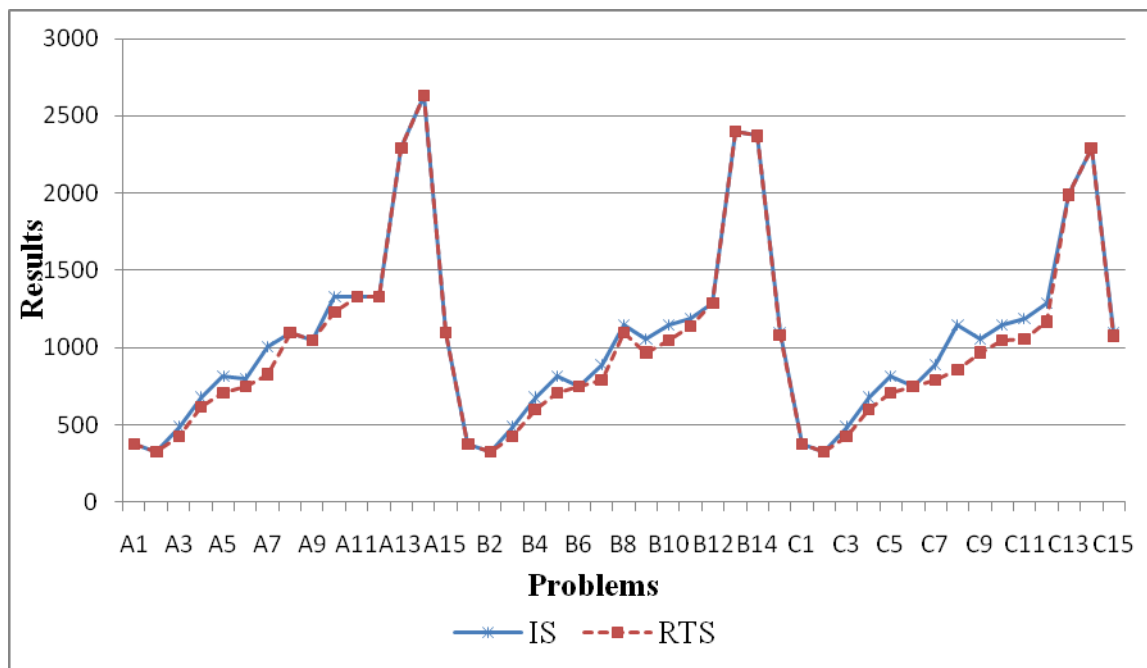
- Undo the iteration and randomly select different route with the same fitness function
- Then go to [**Step 3**]

## 5.4 Computational Results

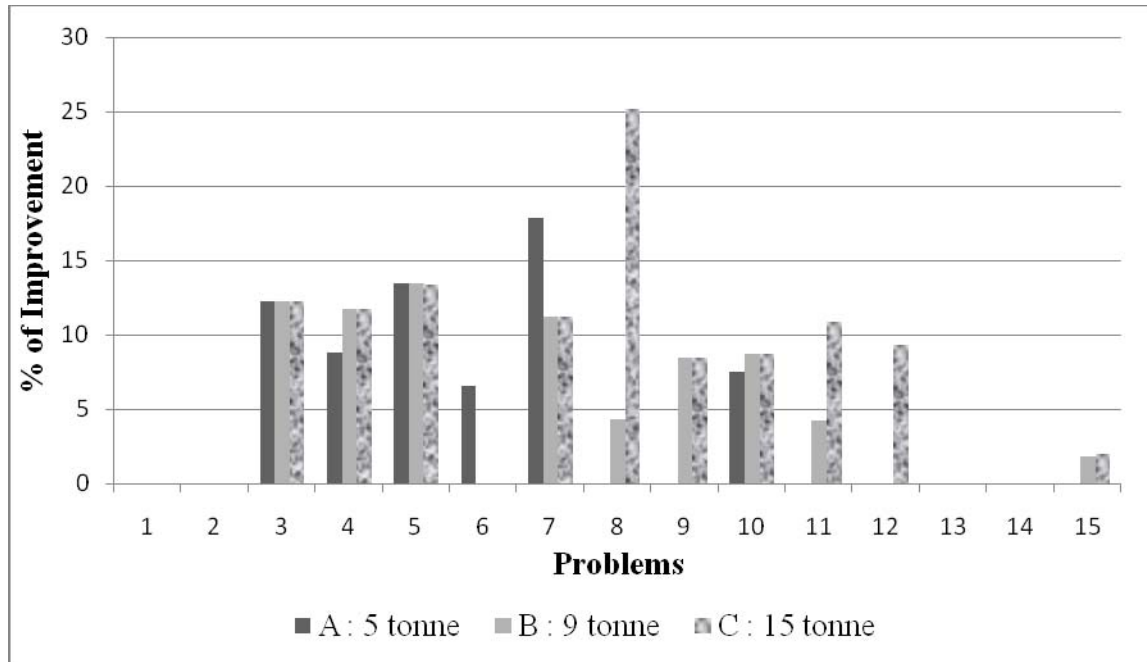
The performance of RTS algorithm have been tested and evaluated. Same instances used to test the performance of TS algorithm have been tested to RTS algorithm. The performances of the RTS algorithm are shown in Table 5.9 and have been illustrated in Figure 5.7 and Figure 5.8. Generally, some problems show no improvement on the initial solution and some of them can improve more than 25 percent. In detail, more than half of the total problem shows an improvement. As clearly shown in Figure 5.7, results produced by RTS can still reach at a better solution than IS even some of them are not really improve to a better one.

**Table 5.9: RTS Computational Results**

| Problem | A : 5 tonne |      |       | B : 9 tonne |      |       | C : 15 tonne |      |       |
|---------|-------------|------|-------|-------------|------|-------|--------------|------|-------|
|         | IS          | RTS  | %     | IS          | RTS  | %     | IS           | RTS  | %     |
| 1       | 380         | 380  | 0     | 380         | 380  | 0     | 380          | 380  | 0     |
| 2       | 330         | 330  | 0     | 330         | 330  | 0     | 330          | 330  | 0     |
| 3       | 490         | 430  | 12.24 | 490         | 430  | 12.24 | 490          | 430  | 12.24 |
| 4       | 680         | 620  | 8.82  | 680         | 600  | 11.76 | 680          | 600  | 11.76 |
| 5       | 820         | 710  | 13.41 | 820         | 710  | 13.41 | 820          | 710  | 13.41 |
| 6       | 800         | 750  | 6.52  | 750         | 750  | 0     | 750          | 750  | 0     |
| 7       | 1010        | 830  | 17.82 | 890         | 790  | 11.24 | 890          | 790  | 11.24 |
| 8       | 1100        | 1100 | 0     | 1150        | 1100 | 4.35  | 1150         | 860  | 25.22 |
| 9       | 1050        | 1050 | 0     | 1060        | 970  | 8.49  | 1060         | 970  | 8.49  |
| 10      | 1330        | 1230 | 7.52  | 1150        | 1050 | 8.70  | 1150         | 1050 | 8.70  |
| 11      | 1330        | 1330 | 0     | 1190        | 1140 | 4.20  | 1190         | 1060 | 10.92 |
| 12      | 1330        | 1330 | 0     | 1290        | 1290 | 0     | 1290         | 1170 | 9.30  |
| 13      | 2290        | 2290 | 0     | 2400        | 2400 | 0     | 1990         | 1990 | 0     |
| 14      | 2630        | 2630 | 0     | 2370        | 2370 | 0     | 2290         | 2290 | 0     |
| 15      | 1100        | 1100 | 0     | 1100        | 1080 | 1.82  | 1100         | 1078 | 2.00  |



**Figure 5.7: Comparison between IS and RTS**



**Figure 5.8: Percentage of Improvement for RTS Computational Results**

The percentage of the improvement on the initial solution is given in Figure 5.8. It is shown that some instances give no improvement and some of them can give the improvement up to 25 percent. The summarization of this improvement is given in Table 5.10.

**Table 5.10: Average Percentage of Improvement**

|                          | <b>A</b> | <b>B</b> | <b>C</b> |
|--------------------------|----------|----------|----------|
| Number of improve        | 6        | 9        | 10       |
| % number of improve      | 40       | 60       | 66.67    |
| Total % of improvement   | 66.33    | 76.21    | 113.28   |
| Average % of improvement | 4.42     | 5.08     | 7.55     |

Table 5.10 shows the summary of improvement in number of problem that categorized in different size of maximum limit of vehicle. For problem categorized in group A (5 tonnes), only six instances out of 15, which is just 40 percent that give an improvement compare to group C (15 tonnes), which is 66.67 percent from the number of instances give an improvement. This means that, the larger it size the more improvement RTS could give. This is strengthen by the total average of percentage of improvement that group A only improve 4.49 percent instead group C improve up to 7.55 percent.

## **5.5 Summary**

The development of RTS has been discussed clearly in this chapter. Also included in this chapter are the RTS algorithm and computational result on percentage of improvement which is how much RTS improve from initial solution. More analysing on TS and RTS will be discuss in a next chapter.



## **CHAPTER 6**

### **SYSTEM DEVELOPMENT FOR CAPACITATED ARC ROUTING PROBLEM MODEL**

#### **6.1 Introduction**

This chapter presents the implementation of the algorithm and methodology developed in previous chapters. In this implementation, the problem of Capacitated Arc Routing Problem (CARP) will be simulated using simple computer programming. It begins with the detail explanation of CARP's program operation developed using Microsoft Visual Studio followed by the program visualisation and end with how to manage and use the CARP solution systems.

## **6.2 Programming with Microsoft Visual Studio**

In developing a solution to CARP, a programming was developed using Microsoft Visual Studio 2005 Team Suite (Trial) Edition and it is written in Visual C# language. Visual Studio is a complete set of development tools for building ASP.NET Web applications, XML Web Services, desktop applications, and mobile applications.

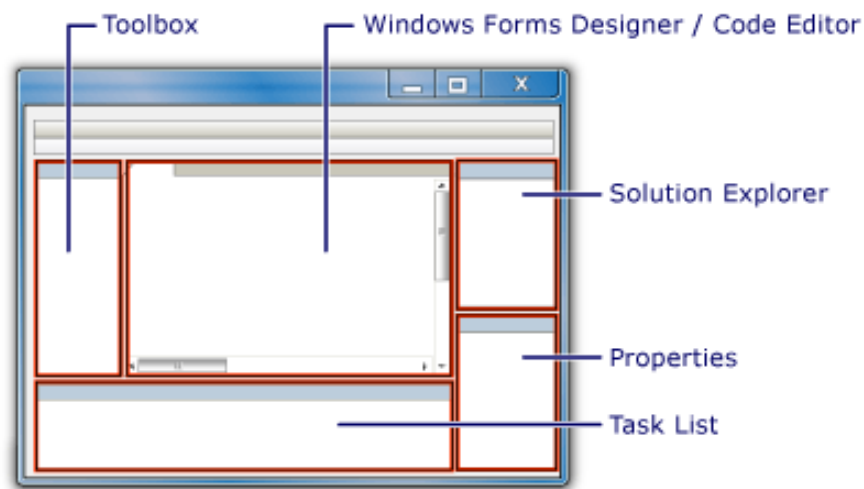
C# (pronounced "C sharp") is a programming language that is designed for building a variety of applications that run on the .NET Framework. C# is an object-oriented programming language developed by Microsoft as part of the .NET initiative. Anders Hejlsberg leads development of the C# language, which has a procedural, object-oriented syntax based on C++ and includes influences from aspects of several other programming languages (most notably Delphi and Java) with a particular emphasis on simplification. C# is simple, powerful, type-safe and object oriented. Many innovations in C# enable rapid application development while retaining the expressiveness and elegance of C-style languages. The Visual Studio supports Visual C# with a full-featured code editor, compiler, project templates, designers, code wizards, a powerful and easy-to-use debugger, and other tools. The .NET class library provides access to many operating system services, well-designed classes that speed up the development cycle significant and other useful features.

### **6.2.1 The Visual Studio Application**

One of the collections of development tools exposed through a common user interface is the Visual C# integrated development environment (IDE). Some of the tools

are shared with other Visual Studio languages and some of them such as C# compiler are unique to Visual C#. The most important tools and windows in Visual C# are as follows:

- The Code Editor, for writing source code.
- The C# compiler, for converting C# source code into an executable program.
- The Visual Studio debugger, for testing the program.
- The Toolbox and Designer, for rapid development of user interfaces by using the mouse.
- Solution Explorer, for viewing and managing project files and settings.
- Project Designer, for configuring compiler options, deployment paths, resources and more.
- Class View, for navigating through source code according to types, not files.
- Properties Window, for configuring properties and events on controls in user interface.
- Object Browser, for viewing the methods and classes available in dynamic link libraries including .NET Framework assemblies and COM objects.
- Document Explorer, for browsing and searching product documentation on a local computer and on the internet.



**Figure 6.1: Basic IDE**

The windows for most of these tools can be opened from the View menu. Figure 6.1 represent the basic IDE. The large main window is used by the Code Editor, the Windows Forms Designer or the Windows Presentation Foundation Designer which is the space needed to write the source code. Upper-right of the main window is Solution Explorer window which shows all the files in the project in a hierarchical tree view. Lower-right of the main window is the Properties window that enable user to set properties and hook up events for user interface controls such as buttons and text boxes. The Toolbox window is located on a left-side of the main window. Window below the Code Editor Window is named as Task List Window which functioning to list down the build errors when we compile the designed project in the Code Editor Window.

The C# Compiler has no window because it is not an interactive tool but we can set the compiler options in the Project Designer, while the Project Designer property pages can be accessed by right-clicking the Properties node in Solution Explorer and then clicking open. All the windows in Visual C# actually can be made dock-able or floating, hidden or visible or it can be moved to a new locations. Many other aspects of the IDE can be customizing by clicking Options on the Tools menu.

### **6.3 Waste Collection Management Computational Module**

The main purpose in developing the Waste Collection Management (WCM) computational module is to make the computational easier. The computational can be done by a manual calculation, but it will take a very long time to complete the calculation. This is because the way on how TS and RTS work. It works by iteratively

searching from one solution to another solution and this make the computational so hard to be done manually by man-power. Therefore, a best way to compute this type of problem is by using computer programming.

For this WCM computational module, the system is excellent to be used by the data with 50 numbers of nodes and less. For data with number of nodes more than 50, this WCM computational module can also be used, but the space for a drawing region in graphical user interface (GUI) need to be widen so that it can accommodate for a large number of nodes. However, the limitation is set up to 100 numbers of nodes. This WCM computational module cannot afford for a calculation more than 100 numbers of nodes. The modification in a coding is needed if the data more than 100 numbers of nodes have to be used in the calculation.

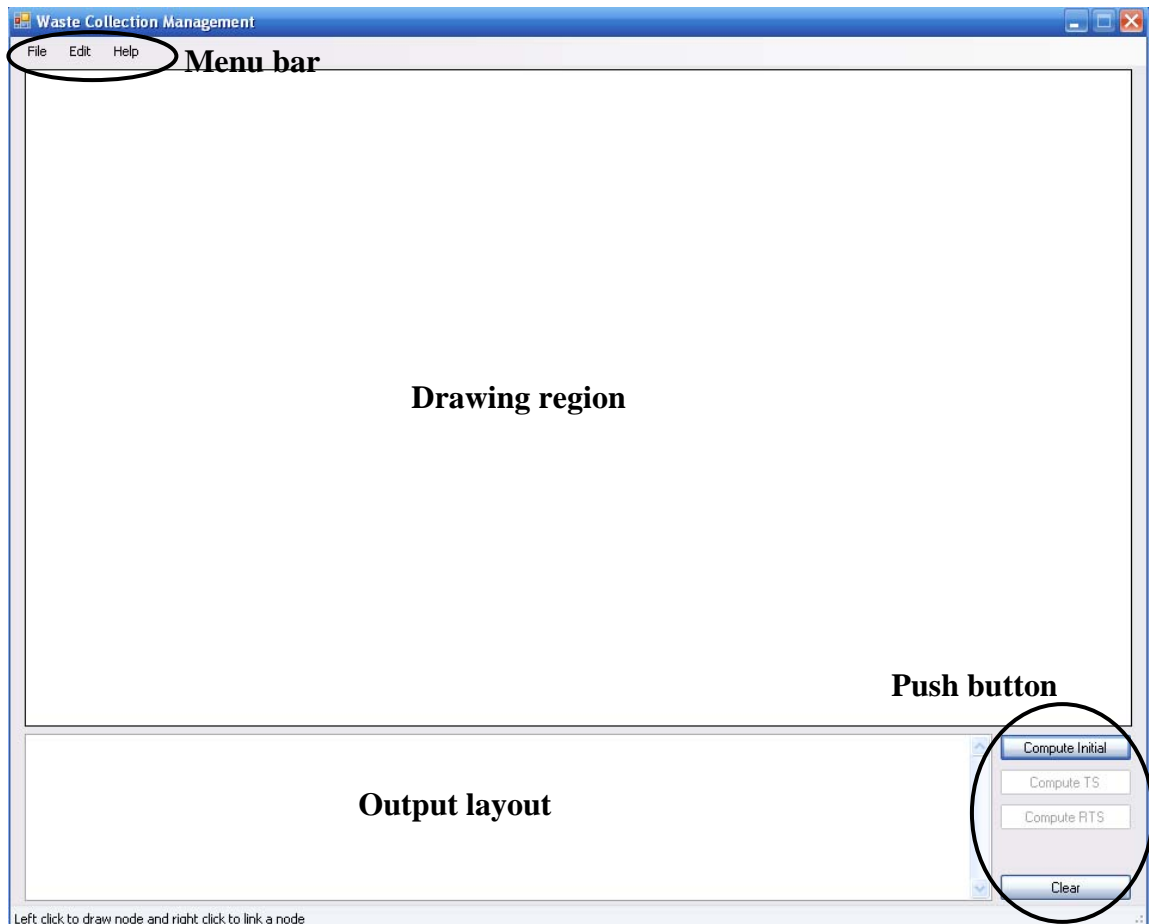
#### **6.4 Program Visualization; Graphical User Interface**

The GUI has been created with the purpose of making the program to be user friendly. The user can key in the input and design the graph as they pleased. This GUI start with a welcome message and it will display when opening the program as shown in Figure 6.2. Just click an *enter* button to go through this welcome message.



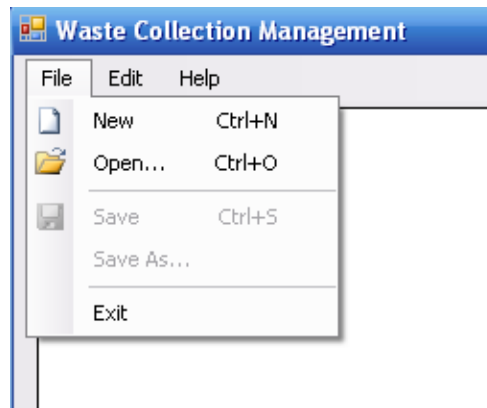
**Figure 6.2: Welcome GUI**

Image display in Figure 6.3 is our computational GUI which contains a drawing region to draw the network, a push buttons to control the behaviour of the computation and an output layout to give the result produce by the program. It also contains a menu bar and there are three pull down menus on it which are *File*, *Edit* and *Help*. Figure 6.4 shows all the features in menu bar.

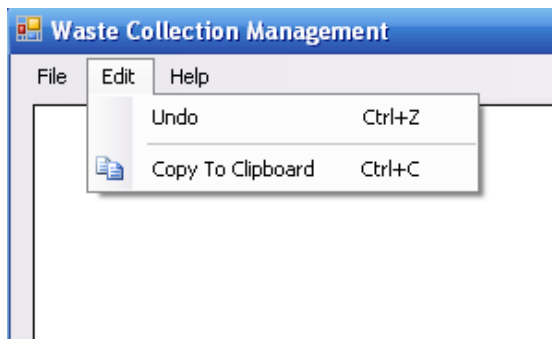


**Figure 6.3: GUI of the Program**

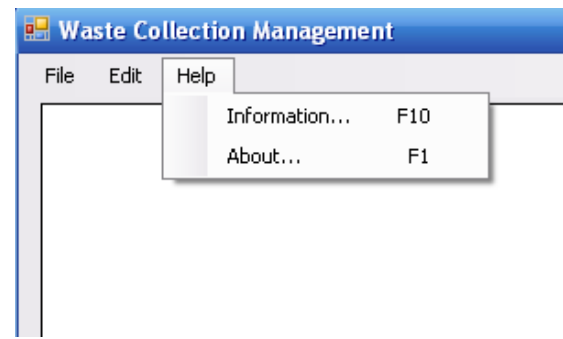
There are five items in *File* as illustrated in Figure 6.4(a) which are *New* to open a new file, *Open* to call the existing file, *Save* and *Save As* to save the network design on drawing region in *File.xml* document type and *Exit* to close the program. For the *Edit*, there are two items which are *Undo* and *Copy To Clipboard* as presents in Figure 6.4(b). The *Undo* menu allowing user to undo some of the errors user may have made while entering the data into a drawing region and *Copy To Clipboard* providing user to copy the drawing network. Figure 6.4(c) illustrates the items in *Help*. There are two of them, which are *Information* and *About*.



(a)



(b)

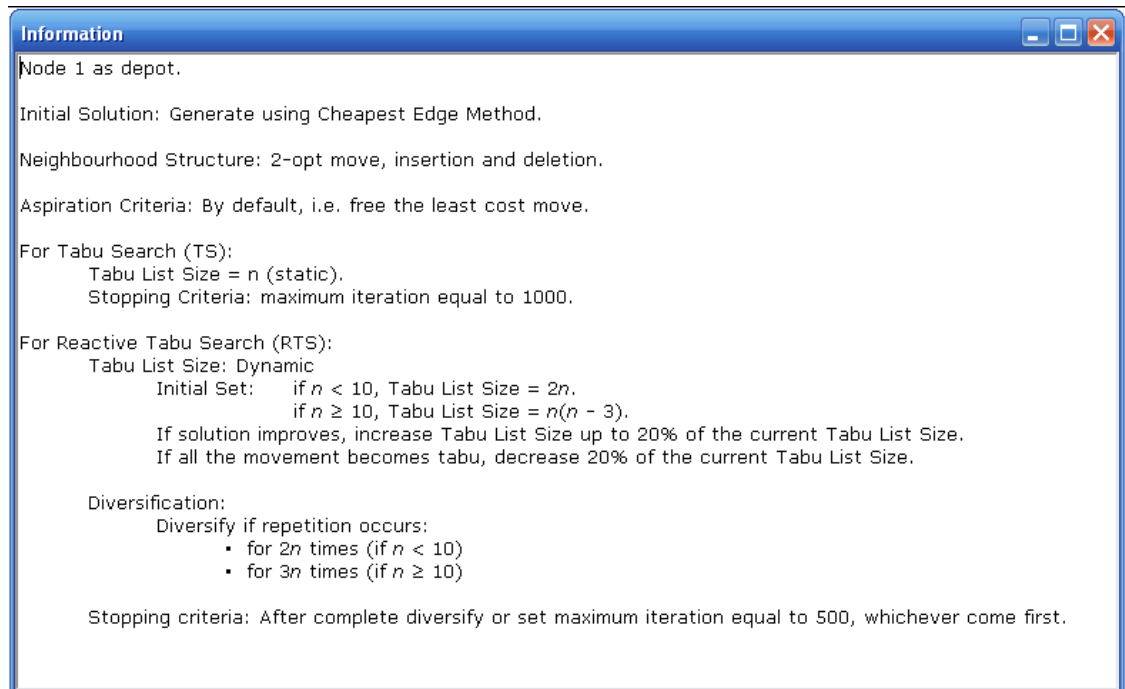


(c)

**Figure 6.4: Features in Menu Bar**

Information provide in *Information* menu are as shown in Figure 6.5 while information provide in *About* menu are as shown in Figure 6.6.





**Figure 6.5: Information Menu**



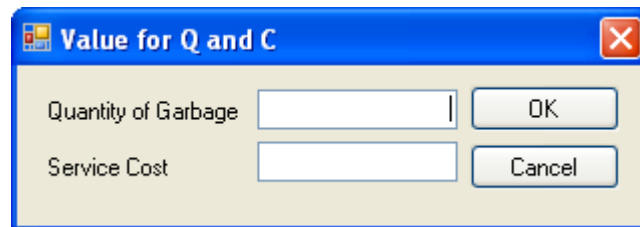
**Figure 6.6: About Menu**

## 6.5 Managing the Capacitated Arc Routing Problem Program

This program is created to be very user friendly and easy to manage. First page is just an introduction page as shown in Figure 6.2. Just click the *Enter* button to open

the computation graphical interface. Then interface as shown in Figure 6.3 will appear after clicking the enter button.

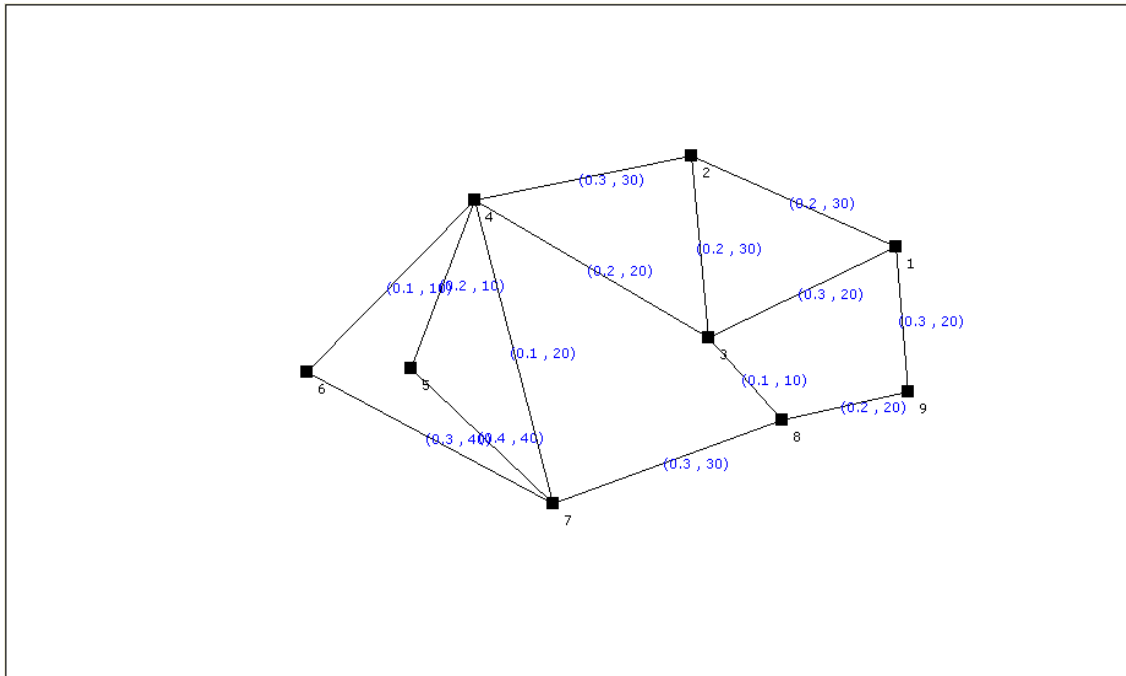
To start using this program, we need to design the input data on the drawing region or call the existing file if the network is already created. In drawing a network, use left-click mouse to create a nodes and right-click mouse to create an arc (connection) between two nodes. In connecting two nodes, a form such as shown in Figure 6.7 will appear to ask for a value of demand which is quantity of the garbage and a service cost. Key-in the value and then click *ok*.



The image shows a Windows-style dialog box with a blue title bar containing the text "Value for Q and C" and a close button (X) on the right. The main area has a light beige background. It contains two rows of input fields. The first row is labeled "Quantity of Garbage" and has a text input field followed by an "OK" button. The second row is labeled "Service Cost" and has a text input field followed by a "Cancel" button.

**Figure 6.7: Form for Insert Demand and Cost**

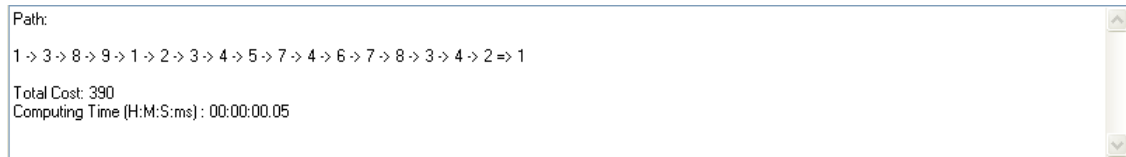
After complete drawing the network such as shown in Figure 6.8, then we can go to the computation. To start TS or RTS computation, we need to generate the initial solution first. So, click *Compute Initial* button to compute the initial solution. A form such as in Figure 6.9 will appear to ask for the maximum capacity of the vehicle; key-in the maximum limit of the vehicle can load in a time and then click *ok*.



**Figure 6.8: Example of Complete Network in Drawing Region**

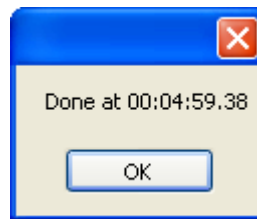
**Figure 6.9: Form for Maximum Capacity of the Vehicle**

The initial solution will produce using Cheapest Edge Method (CEM) in output layout once we click *ok* in maximum capacity form. Figure 6.10 shows the example of the result generated by the program in output region.

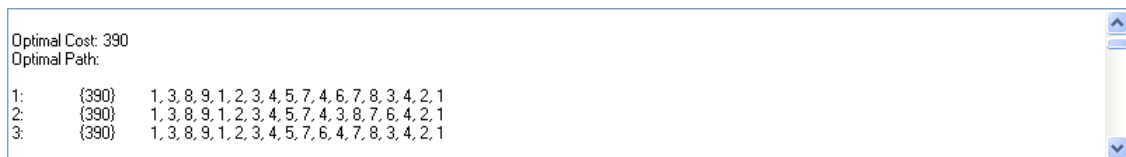


**Figure 6.10: Example of the Initial Solution Produced By the Program**

Lastly click the *Compute TS* button to get the result by using TS algorithm or click *Compute RTS* button to get the result by using RTS algorithm. After the program stop computing, a message box such as shows in Figure 6.11 will appear to tell the computational time of running the program. Click *ok* to finish it and we can see all the result produced by the program in output region. Every single route that gives the minimum cost will list out in this region. Figure 6.12 shows the example of the result produce by the program.



**Figure 6.11: Message Box of Computational Time**



**Figure 6.12: Example of the Result Produced By the Program**

## 6.6 Summary

The system development on Visual C# has been discussed widely in this chapter. Each functions in GUI have been clearly explain. We also provide in details the way to run this program. Next chapter will be the comparison analysis of the computational using TS and RTS approach.

## **CHAPTER 7**

### **ANALYSIS OF RESULTS, CONCLUSION AND RECOMMENDATION**

#### **7.1 Introduction**

In this chapter, performances of Reactive Tabu Search (RTS) algorithm will be investigated and evaluated by comparing with Tabu Search (TS) algorithm. The instances used for conducting a test on RTS algorithm are always same as TS instances. In this chapter, the conclusion and recommendation of the research is also provided. This chapter begins with the analysis of the results by comparing the results between TS and RTS. At the end of this chapter, the conclusions for the whole research and also the recommendations are given.

## 7.2 Analysis of Results

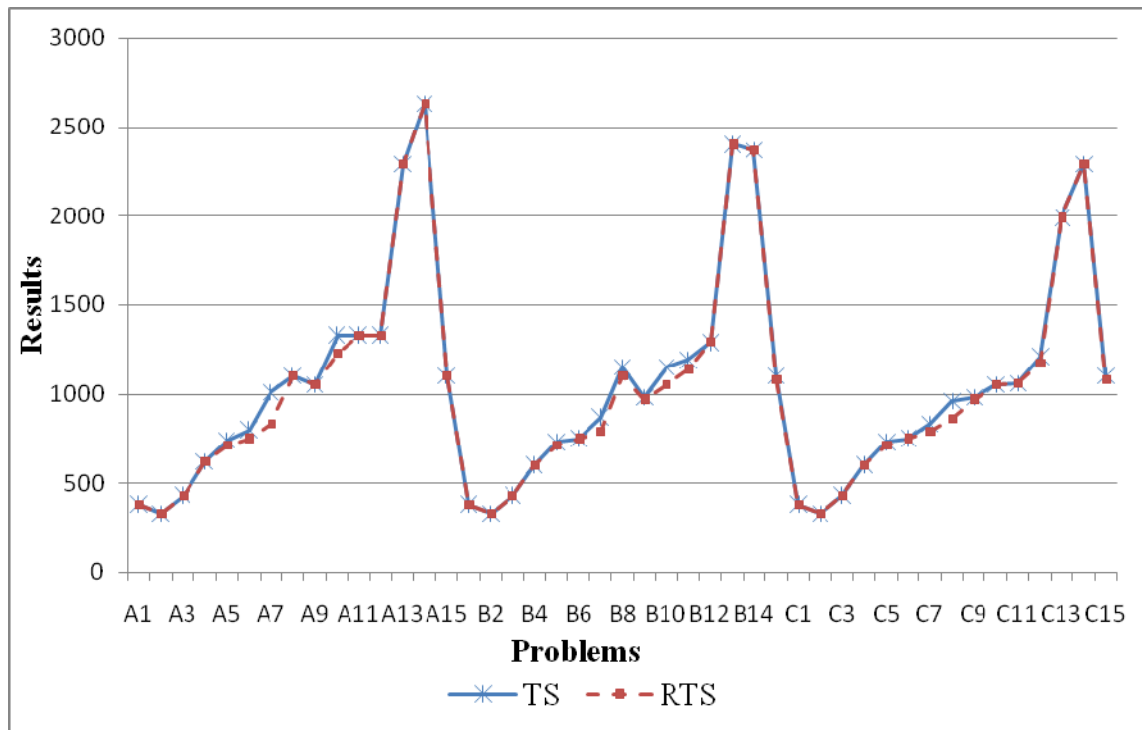
After some findings as discussed in the earlier chapters (Chapter 4 and Chapter 5), the performance of the TS algorithm and RTS algorithm will be discussed and compared in this section. Then the discussion will follow with the summarization on the advantages and disadvantages of TS and RTS.

Table 7.1 shows the details values of the fitness function produced by TS algorithm and RTS algorithm.

**Table 7.1: Comparison between TS and RTS**

| Problem | A    |             | B    |             | C    |             |
|---------|------|-------------|------|-------------|------|-------------|
|         | TS   | RTS         | TS   | RTS         | TS   | RTS         |
| 1       | 380  | 380         | 380  | 380         | 380  | 380         |
| 2       | 330  | 330         | 330  | 330         | 330  | 330         |
| 3       | 430  | 430         | 430  | 430         | 430  | 430         |
| 4       | 620  | 620         | 600  | 600         | 600  | 600         |
| 5       | 740  | <b>710</b>  | 730  | <b>710</b>  | 730  | <b>710</b>  |
| 6       | 800  | <b>750</b>  | 750  | 750         | 750  | 750         |
| 7       | 1010 | <b>830</b>  | 870  | <b>790</b>  | 830  | <b>790</b>  |
| 8       | 1100 | 1100        | 1150 | <b>1100</b> | 960  | <b>860</b>  |
| 9       | 1050 | 1050        | 980  | <b>970</b>  | 980  | <b>970</b>  |
| 10      | 1330 | <b>1230</b> | 1150 | <b>1050</b> | 1050 | 1050        |
| 11      | 1330 | 1330        | 1190 | <b>1140</b> | 1060 | 1060        |
| 12      | 1330 | 1330        | 1290 | 1290        | 1210 | <b>1170</b> |
| 13      | 2290 | 2290        | 2400 | 2400        | 1990 | 1990        |
| 14      | 2630 | 2630        | 2370 | 2370        | 2290 | 2290        |
| 15      | 1100 | 1100        | 1100 | <b>1080</b> | 1100 | <b>1078</b> |

Based on a computational result between Look-Ahead Strategy (LAS) and TS in Chapter 4, LAS gave a result slightly worse than TS and only three instances can be compared. Therefore, the comparisons are only between TS and RTS computational results. Table 7.1 shows that RTS results give an equal or even better solution than TS. For obviously evaluation, Figure 7.1 gives a clear illustration for this comparison.



**Figure 7.1: Comparison between TS and RTS**

Figure 7.1 shows that TS can reach the same minimum solution as RTS but never reach the better solution than RTS. While in some instances, RTS can give much more improvement and better solution compared to TS. This illustrates that RTS algorithm producing high-quality solution. The main reason for its improvement is the ability to dynamically change the tabu list size which allows it to escape from being trap in tabu list. In additional, a consideration of the diversification in implementing the RTS algorithm is also one of the reasons of the improvement in the initial solution. The



diversification was done by allowing a repetition in the search process in order to explore their neighbourhood widely. The summarization of advantages and disadvantages of TS and RTS are described in Table 7.2.

**Table 7.2: Advantages and Disadvantages of TS and RTS**

|            | <b>Advantages</b>   | <b>Disadvantages</b>   |
|------------|---|--|
| <b>TS</b>  | The use of memory structures allows the implementation of procedures that are capable of searching the solution space economically and effectively.   | The static and fixed size of tabu list sometimes makes the exploration getting trap. To fix this problem, RTS with dynamically changing the tabu list size have been introduced. |
| <b>RTS</b> | Dynamically change the tabu list size which allows escaping from being trap in tabu list is the strength in RTS algorithm.<br>The consideration of the repetition in the exploration also one of the advantages in RTS algorithm. | Not really suitable for the simple problems, mean the problem that not categorized in NP-hard.   |

### 7.3 Conclusion

As a result, several good works have been reported. In this investigation, a newly modified RTS algorithm presents a high quality of solution for Capacitated Arc Routing Problem (CARP). Even the development of this RTS algorithm is the simplest and straightforward implementation on it reactive scheme (without any long-term memory or any other procedure to encourage an intensification or a deep diversification), but it manage to give a very best performance compare to TS. This

basic investigation definitely provided a better understanding on the powerful of RTS that will be useful in its use to other related problem.

Subsequently for second objective, the development of a computational module was very useful and very helpful in finding a various results. Without the use of this computational module, the computational might became a big problem since automatically computational by computer are not reachable by man-power.

#### **7.4 Recommendation**

Since this RTS algorithm is the simplest and apply only the basic procedure in implementing it, so we recommend to explore a deep diversification and intensification strategy in order to wider the search exploration process. Other than that, the reactive part which is a dynamic tabu list size is also might be consider so that the future RTS algorithm is more powerful than the existing one. Rather than increasing and decreasing the tabu list size, the use of mathematical formulation can also be considered in order to make it dynamic so that it become more powerful and can be applied to any range of set of data.

## 7.5 Future Problems: The Extension of This Problem

For extending this problem, there are some types of problems lies in the area of CARP to be considered. There are Multiple Depot Capacitated Arc Routing Problem (MDCARP), Capacitated Arc Routing Problem with Stochastic Demand (CARPSD), Capacitated Arc Routing Problem with Time Window (CARPTW), Capacitated Arc Routing Problem with Refill Point (CARPRP), Periodic Capacitated Arc Routing Problem (PCARP) and Capacitated Arc Routing Problem with Backhauls (CARPB). These are the variants of CARP itself that lies in the area of arc routing that can be considered and reformulate the RTS algorithm to solve these problems.

Another problem that can be considered to solve by this RTS algorithm is in the area of node routing problem which are Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). The VRP itself also have it variants as CARP that also can be considered in future research work. There are Capacitated Vehicle Routing Problem (CVRP), Multiple Depot Vehicle Routing Problem (MDVRP), Vehicle Routing Problem with Stochastic Demand (VRPSD), Vehicle Routing Problem with Time Window (VRPTW), Vehicle Routing Problem with Pick-up and Delivery (VRPPD), Vehicle Routing Problem with Backhauls (VRPB), Periodic Vehicle Routing Problem (PVRP) and Split Delivery Vehicle Routing Problem (SDVRP).

To implement all these problem into this RTS algorithm, the formulation need to be formulated so that it suits with the types and constraints of the problems. As well as the RTS algorithm, if it does not satisfy the order or constraint, so the RTS algorithm need to modified.

## REFERENCES

- Amaya, A., Langevin, A. and Trepanier, M. (2007). *The Capacitated Arc Routing Problem With Refill Points*. *Operations Research Letters*. 35: 45-53.
- Amponsah, S. K. and Salhi, S. (2004). *The Investigation of A Class of Capacitated Arc Routing Problems: the Collection of Garbage in Developing Countries*. *Waste Management*. 24: 711-721.
- Battiti, R. and Tecchiolli, G. (1994). *The Reactive Tabu Search*. *ORSA Journal on Computing*. 6: 126-140.
- Bautista, J., Fernandez, E. and Pereira, J. (2008). *Solving an Urban Waste Collection Problem Using Ant Heuristics*. *Computer & Operations Research*. 35: 3020-3033.
- Belenguier, J.M. and Benavent, E. (2003). *A Cutting Plane Algorithm for the Capacitated Arc Routing Problem*. *Computers & Operations Research*. 30: 705-728.
- Beullens, P., Muyldermans, L., Cattrysse, D. and Oudheusden, D.V. (2003). *A Guided Local Search Heuristic for the Capacitated Arc Routing Problem*. *European Journal of Operational Research*. 147: 629-643.
- Black, P.E. (2007, Dec 17). *NP-Hard*. *Dictionary of Algorithms and Data Structure*. Retrieved August 13, 2009, from <http://www.itl.nist.gov/div897/sqg/dads/HTML/nphard.html>.
- Blochliker, I. and Zufferey, N. (2008). *A Graph Coloring Heuristic Using Partial Solutions and a Reactive Tabu Scheme*. *Computers & Operational Research*. 35: 960-975.
- Brandao, J. and Eglese, R. (2008). *A Deterministic Tabu Search Algorithm For the Capacitated Arc Routing Problem*. *Computers & Operations Research*. 35: 1112-1126.
- Brown, D.T. (2001). *Routing Unmanned Aerial Vehicles while Considering General Restricted Operating Zones*. Master Thesis, Air Force Institute of Technology, United States Air Force.

- Castellani, U., Fusiello, A., Gherardi, R. and Murino, V. (2007). *Automatic Selection of MRF Control Parameters by Reactive Tabu Search*. *Image & Vision Computing*. 25: 1824-1832.
- Chu, F., Labadi, N. and Prins, C. (2005). *Heuristics For the Periodic Capacitated Arc Routing Problem*. *Journal of Intelligent Manufacturing*. 16: 243-251.
- Corne, D., Dorigo, M. and Glover, F. (1999). *Introduction*. In: Corne, D., Dorigo, M. and Glover, F. (Eds.) *New Ideas In Optimization* (pp. 1-8). England: McGraw-Hill.
- da Silva, L.G.W., Pereira, R.A.F., Abbad, J.R. and Mantovani, J.R.S. (2008). *Optimised Placement of Control and Protective Devices in Electric Distribution Systems Through Reactive Tabu Search Algorithm*. *Electric Power Systems Research*. 78: 372-381.
- Dror, M. (2000). *Arc Routing: Theory, Solutions and Applications*. Boston MA: Kluwer Academic.
- Fukuyama, Y. (2000). *Reactive Tabu Search for Distribution Load Transfer Operation*. *Proc. IEEE Power Eng. Soc. Winter Meeting*. January, 1301-1306.
- Gendreau, M. (2002). *An Introduction to Tabu Search*. Departement d'informatique et de recherche operationnelle, Universite de Montreal, Montreal, Canada: Working Paper.
- Glover, F. and Laguna, M. (1997). *Tabu Search*. Boston: Kluwer.
- Glover, F. and Laguna, M. (1998). *Article on Tabu Search*. Boston: Kluwer.
- Golden, B.L. and Wong, R.T. (1981). *Capacitated Arc Routing Problem*. *Networks*. 11: 305-315.
- Greistorfer, P. (2003). *A Tabu Scatter Search Metaheuristic for the Arc Routing Problem*. *Computers & Industrial Engineering*. 44: 249-266.
- Hanafi, S. (2000). *On the Convergence of Tabu Search*. *Journal of Heuristics*. 7: 47-58.

- Harder, R. (2000). *A Java Universal Vehicle Router in Support of Routing Unmanned Aerial Vehicles*. Air Force Institute of Technology, United States Air Force: Master Thesis.
- Hillier, F. S. and Lieberman, G. J. (2005). *Introduction to Operations Research*. 8th. ed. New York: Mc Graw-Hill.
- Hoos, H.H. and Stutzle, T. (2004). *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann/Elsevier.
- Ismail, Z., Irhamah and Loh, S.L. (2007), “*Look Ahead Heuristics for Modelling Solid Waste Collection Problems*”, Abstract of Second International Conference on Mathematical Sciences (IcoMS2007), May 28-29, 2007 pp 78. Ibnu Sina Institute, UTM Skudai, Malaysia.
- Kinney, G. (2000). *A Hybrid Jump Search and Tabu Search Metaheuristic for the Unmanned Aerial Vehicle (UAV) Routing Problem*. Air Force Institute of Technology, United States Air Force: Master Thesis.
- Kumral, M. and Dimitrakopoulos, R. (2008). *Selection of Waste Dump Sites using a Tabu Search Algorithm*. The Journal of The Southern African Institute of Mining and Metallurgy. 108: 9-13.
- Lacomme, P., Prins, C. and Ramdane-Cherif, W. (2001). *A Genetic Algorithm For the Capacitated Arc Routing Problem and Its Extensions*. Laboratory for Industrial Systems Optimization, University of Technology of Troyes. EvoWorkshop.
- Li, J.Q., Borenstein, O. and Mirchandani, P.B. (2008). *Truck Scheduling for Solid Waste Collection in the City of Porto, Alegre, Brazil*. Omega. 36: 1133-1149.
- Li, Y.P., Huang, G.H. and Yang, Z.F. (2009). *Inexact Fuzzy-Stochastic Constraint-Softened Programming – A Case Study for Waste Management*. Journal of Waste Management. 29: 2165-2177.
- Lim, Y.F. (2007). *Reactive Tabu Search Method for Solving Travelling Salesman Problem*. Universiti Teknologi Malaysia, Malaysia: Master Thesis.
- Loh, S. L. (2007). *Vehicle Routing Problem with Stochastic Demands Using Ant Colony System Algorithm*. Universiti Teknologi Malaysia, Malaysia: Master Thesis.

- Loh, S.L. (2006). *Modelling Capacitated Arc Routing Problem: A Case Study On Solid Waste Collection*. Universiti Teknologi Malaysia, Malaysia: Degree Final Year Project.
- Longo, H., de Aragao, M.P. and Uchoa, E. (2006). *Solving Capacitated Arc Routing Problems Using a Transformation to the CVRP*. *Computer & Operations Research*. 33: 1823-1837.
- Mourao, M.C. and Amado, L. (2005). *Heuristics Method For a Mixed Capacitated Arc Routing Problem: A Refuse Collection Application*. *European Journal of Operational Research*. 160: 139-153.
- Nanry, W.P. and Barnes, J.W. (2000). *Solving the Pickup and Delivery Problem with Time Windows Using Reactive Tabu Search*. *Transportation Research, Part B*. 34, 107-121.
- O'Rourke, K.P. (1999). *Dynamic Unmanned Aerial Vehicle (UAV) Routing With a Java-Encoded Reactive Tabu Search Metaheuristics*. Air Force Institute of Technology, United States Air Force: Master Thesis.
- Reeves, C. R. (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Oxford, England: Blackwell Scientific Publishing.
- Salhi, S. (2002). *Defining Tabu List Size and aspiration Criterion Within Tabu Search Methods*. *Computer & Operations Research*. 29: 67-86.
- Scheuever, S. (2006). *A Tabu Search Heuristic for the Truck and Trailer Routing Problem*. *Computer & Operations Research*. 33: 894-909.
- Wan Ibrahim, W.R. (2007). *Travelling Salesman Problem Approach for Petrol Distribution Using Simulated Annealing and Tabu Search*. Universiti Teknologi Malaysia, Malaysia: Master Thesis.
- Wassan, N.A. (2006). *A Reactive Tabu Search for the Vehicle Routing Problem*. *Journal of the Operational Research Society*. 57: 111-116.

- Wassan, N.A., Nagy, G. and Ahmadi, S. (2008). *A Heuristic Method for the Vehicle Routing Problem with Mixed Deliveries and Pickups*. Springer Science+Business Media. 11: 149–161.
- Wohlk, S. (2003). *Simulated Annealing For the Capacitated Arc Routing Problem, Using an Online Formulation*. Department of Organization and Management, University of Southern Denmark, Odense, Denmark: Working Paper.
- Wohlk, S. (2005). *Contribution to Arc Routing: State of the Art*. University of Southern Denmark: PhD Thesis.
- Wu, X.Y., Huang, G.H., Liu, L. and Li, J.B. (2006). *An Interval Nonlinear Program for the Planning of Waste Management Systems with Economies-of-Scale Effects – A Case Study for the Region of Hamilton, Ontario, Canada*. European Journal of Operational Research. 171: 349-372.
- Xu, G., Ke, Q. and Ma, S.D. (1998). *Recovering Epipolar Geometry by Reactive Tabu Search*. Computer Vision, 1998. Sixth International Conference on 4-7 January. 233-244.
- Yeomans, J.S. (2007). *Solid Waste Planning Under Uncertainty Using Evolutionary Simulation-Optimization*. Socio-Economic Planning Science. 41: 38-60.
- Zainuddin, Z.M. (2004). *Constructive and Tabu Search Heuristics for Capacitated Continuous Location-Allocation Problem*. The University of Birmingham, England: PhD Thesis.