School of Mathematics and Statistics

Heuristic Algorithms For Routing Problems

Yen Nie Chong

This thesis is presented as part of the
requirements for the award of the
Degree of Doctor of Philosophy
of the
Curtin University of Technology

September 2001

Certification

I certify that the work presented in this thesis is my own and that all references are duly acknowledged. This thesis has not been submitted previously, in whole or in part, in respect of any academic award at Curtin University of Technology or elsewhere.

Yen Nie Chong

September 2001

Summary

General routing problems deal with transporting some commodities and/or travelling along the arcs of a given network in some optimal manner. In the modern world such problems arise in several contexts such as distribution of goods, transportation of commodities and/or people etc.

In this thesis we focus on two classical routing problems, namely the Travelling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP). The TSP can be described as a salesman travels from his home city, visits each of the other $(n-1)$ cities exactly once and returns back to the home city such that the total distance travelled by him is minimised. The VRP may be stated as follows: A set of n customers (with known locations and demands for some commodity) is to be supplied from a single depot using a set of delivery vehicles each with a prescribed capacity. A delivery route starts from the depot, visits some customers and returns back to the depot. The VRP is to determine the delivery routes for each vehicle such that the total distance travelled by all the vehicles is minimised.

These routing problems are simple to state in terms of describing them in words. But they are very complex in terms of providing a suitable mathematical formulation and a valid procedure to solve them. These problems belong to the class of $\mathcal{NP}$-hard (Non-deterministic Polynomial) problems. With the present knowledge, it is believed that the problems in $\mathcal{NP}$-hard are unlikely to have any good algorithms to arrive at optimal solutions to a general problem. Hence researchers have focused their effort on; (i) developing exact algorithms to solve as large size problems as possible; (ii) developing heuristics to produce near optimal solutions.

The exact algorithms for such problems have not performed satisfactorily as they need an enormous amount of computational time to solve moderate size problems. For instance, in the literature, TSP of size 225-city, 4461-city and 7397-city were solved using computational time of 1 year, 1.9 years and 4 years respectively (Jünger et al., 1995). Thus heuristics, in particular the probabilistic methods such as tabu search, play a significant role in obtaining near optimal solutions. In the literature, there is very little comparison between the various exact algorithms

and heuristics. (Very often the real-life problems are too large and no optimal solution can be found in a reasonable time.)

One of the problems with a probabilistic heuristic is that different implementations (runs) of the same probabilistic heuristic on a given problem may produce distinct solutions of different quality. Thus the desired quality and reproducibility of the solution cannot be ensured. Furthermore, the performance of the heuristics on the benchmark problems provide no guarantee of the quality of solutions that can be obtained for the problem faced by a researcher. Most of the documentation on the performance of heuristics in literature problems provides no information regarding the computational effort (CPU time) spent in obtaining the claimed solution, reproducibility of the claimed solution and the hardware environment of the implementation. This thesis focuses on some of these deficiencies.

Most of the heuristics for general combinatorial optimisation problems are based on neighbourhood search methods. This thesis explores and provides a formal setup for defining neighbourhood structures, definitions of local optimum and global optimum. Furthermore it highlights the dependence and drawbacks of such methods on the neighbourhood structure.

It is necessary to emphasise the need for a statistical analysis of the output to be part of any such probabilistic heuristic. Some of the statistical tools used for the two probabilistic heuristics for TSP and VRP are developed. Furthermore, these heuristics are part of a bigger class called tabu search heuristics for combinatorial optimisation problems. Hence it includes an overview of the TSP, VRP and tabu search methods in Chapters 2, 3 and 4 respectively. Subsequently in Chapters 5, 6, 7 and 8 ideas of neighbourhood structure, local optimum etc. are developed and the required statistical analysis for some heuristics on the TSP and VRP is demonstrated. In Chapter 9, the conclusion of this thesis is drawn and the direction of future work is outlined. The following is a brief outline of the contribution of this thesis.

In Chapter 5, the ideas of neighbourhood structure, local optimum, global optimum and probabilistic heuristics for any combinatorial optimisation problems

are developed. The drawbacks of the probabilistic heuristics for such problems are highlighted. Furthermore, the need to select the best heuristic on the basis of testing a statistical hypothesis and related statistical analysis is emphasised.

Chapter 6 illustrates some of the ideas presented in Chapter 5 using the GENIUS algorithm proposed for the TSP. Statistical analysis is performed for 36 variations of GENIUS algorithm based on different neighbourhood parameters, different types of insertion methods used and two types of constructions of starting solutions. The analysis is performed on 27 literature problems with size ranging from 100 cities to 532 cities and 20 randomly generated problems with size ranging from 100 cities to 480 cities. In both cases the best heuristic is selected. Furthermore, the fitting of the Weibull Distribution to the objective function values of the heuristic solutions provides an estimate of the optimal objective function value and a corresponding confidence interval for both the literature and randomly generated problems. In both cases the estimate of the optimal objective function values are within 8.2% of the best objective function value known.

Since the GENIUS algorithm proved to be efficient, a hybrid heuristic for the TSP combining the branch and bound method and GENIUS algorithm to solve large dimensional problems is proposed. The algorithm is tested on both the literature problems with sizes ranging from 575 cities to 724 cities and randomly generated problems with sizes ranging from 500 cities to 700 cities. Though problems could not be solved to optimality within the 10 hours time limit, solutions were found within 2.4% of the best known objective function value in the literature.

In Chapter 7, a similar statistical analysis for the TABUROUTE algorithm proposed for the VRP is conducted. The analysis is carried out based on the different neighbourhood parameters and tested using 14 literature problems with sizes ranging from 50 cities to 199 cities and 49 randomly generated problems with sizes ranging from 60 cities to 120 cities. In both sets of the problems, the statistical tests accepted the hypothesis that there is no significant difference in the solution produced between the various parameters used for the TABUROUTE algorithm. By fitting the Weibull distribution to the objective function values

of the local optimal solutions, the optimal objective function value and a corresponding confidence intervals for each problem is estimated. These estimates give values that are to within 6.1% and 18.3% of the best known values for the literature problems and randomly generated problems respectively.

In Chapter 8, the general neighbourhood search method for a general combinatorial optimisation problem is presented. Very often, the neighbourhood structure can be defined suitably only on a superset $\mathcal{S}$ of the set of feasible solutions S. Thus the solutions in $\mathcal{S} \backslash S$ are infeasible. Several questions are posed regarding the computational complexity of the solution space of a problem. These concepts are illustrated on the 199-city CDVRP, using the TABUROUTE algorithm.

In addition, the idea of complexity of the solution space based on the samples collected over the 140 runs is explored. Some of the data collected include the number of solutions with distance and/or capacity feasible, the number of feasible neighbourhood solutions encountered for one run, etc. Questions such as

- How many solutions are there for the 199-city problem ?
- How many numbers of local minima solutions are there for the 199-city problem?
- What is the size of the feasible region for the 199-city problem?

are answered. Finally, the conclusion is drawn that this problem cannot be used as a benchmark based on the size of the feasible region and too many local minima.

The conclusion of this thesis and directions of future work are discussed in Chapter 9. There are two appendices presented at the end of the thesis. Appendix A presents the details of the Friedman test, the expected utility function test and the estimation of the optimal objective function value based on the Weibull distribution. Appendix B presents a list of tables from Chapters 6, 7 and 8.

Acknowledgments

I would like to thank my supervisor Dr. N. R. Achuthan for his guidance and encouragement throughout the period of my study. I also thank my associate supervisor Professor L. Caccetta for his help and suggestions for writing this thesis.

Special thanks goes to my friend Dr S. Hill for encouraging and giving me support during the period of my study. He, as well as Dr. N. R. Achuthan and Professor L. Caccetta have read through the manuscript and suggested ideas in presenting the thesis.

I thank my friends K. K. Lau for helping me with my code and Dr R. Collinson for proof reading my thesis.

I am grateful to the Australian Government and Curtin University for providing the Australian Postgraduate Award (APAW). I would like to thank the School of Mathematics and Statistics for providing me with addtional financial assistance to undertake my research.

Finally, I thank my family and friends for their support.

Contents

Chapter 1

Introduction

1.1 Brief background

General routing problems deal with transporting some commodities and/or travelling along the arcs of a given network in some optimal manner. In the modern world such problems arise in several contexts. Some examples, though not exhaustive are:

- distribution of goods and services from one or more depots to several customer locations;
- production planning and distribution of products from plants/factories to retail outlets;
- inventory planning and distribution of commodities from depots to several retail outlets;
- aircraft and crew scheduling;
- transportation of semi finished products from and/or between production points and distribution points in a manufacturing or mining environment.

Such problems, dealing with the management and operational planning of vehicles and/or crews in a network, are faced on a day-to-day basis by many organisa-

tions and government. These are termed as routing problems. The origin of such routing problems can be traced to the classical **Travelling Salesman Problem (TSP)** which gave rise to another classical problem, namely the **Vehicle Routing Problem (VRP)**. For a recent review on routing problems we refer to Bramel and Simchi-Levi (1997).

The Travelling Salesman Problem is to find a route for the salesman to travel from his home city, visit each of the other $(n-1)$ cities exactly once and return back to the home city such that the total distance travelled by him is minimised. The history of TSP can be traced back to Euler (1759) (cited in Lawler et al., 1985) who discussed the problem of the knight's tour. For a detailed history on TSP we refer to Lawler et al. (1985). The TSP appears in disguise in several other problems arising from practical applications. Some examples of such applications are: drilling of printed circuit boards, Chinese postman's problem, X-ray crystallography, vehicle routing, scheduling, computer wiring, control of robot motions, etc. For some exposure to such applications see Jünger et al. (1995).

The Vehicle Routing Problems are natural generalisations of the TSP and they address some cases of the routing problems dealing with the distribution of goods. More precisely a general VRP may be stated as follows: A set of n customers with known locations and demands for some commodity is to be supplied from a set of depots using a set of delivery vehicles each with a prescribed capacity. A delivery route starts from a depot and visits some customers and returns back to the depot. The VRP is to determine the delivery routes, one for each vehicle, such that the total distance travelled by all the vehicles is minimised. Furthermore, the delivery routes of the vehicles must satisfy the following conditions:

- every customer's demand must be satisfied;
- each customer appears on exactly one vehicle route; and
- the total demand of customers appearing in a vehicle route can not exceed the capacity of that vehicle.

If the above problem concerns a single depot and a common vehicle capacity, then the problem is called capacity restricted vehicle routing problem (CVRP). Several variations of VRP can be recognised in day-to-day distribution problems. These variations are obtained by adding some more side constraints such as

- upper bound restriction on the distance travelled by a vehicle along a vehicle tour;
- customer's restriction on time windows during which a vehicle could visit the customer.

As already outlined, there are many scenarios of distribution problems involving the applications of VRP. Some specific examples include distribution and/or collection of fuel, milk, newspapers, mail, garbage, etc. (cited in Lawler et al. (1985)).

These routing problems are simple to state in terms of describing them in words. But they are very complex in terms of providing a suitable mathematical formulation and a valid procedure to solve them. The classical TSP and VRP are comparatively easy in terms of providing valid mathematical formulations. However, solution procedures are very complex in terms of the computational effort required to solve an instance of a problem. The computational complexity of a problem is determined through the analysis of the required worst case computational effort to solve various instances of the problem by the best algorithm. Normally the computational effort of an algorithm to solve a problem is quantified as a function of the size of the input instance of the problem. The computational complexity of an algorithm to solve a problem is the worst case computational effort taken over all instances of the same size of the problem and is expressed as a function of the problem size. Whenever this function is polynomial, the algorithm is considered to be good.

For many combinatorial optimisation problems such as TSP and VRP, the computational complexity of all the known exact algorithms grows exponentially as a function of the size of the problem. The literature in computational complexity is

well developed. For details of this subject, we refer to Johnson and Papadimitriou (1985). The TSP and VRP belong to the class of $\mathcal{NP}$-hard (Non-deterministic Polynomial) problems. With the present knowledge it is believed that the problems in $\mathcal{NP}$-hard are unlikely to have any good algorithm.

Thus for many combinatorial optimisation problems in the class of $\mathcal{NP}$-hard, researchers have focussed their effort on the following:

- development and implementation of exact algorithms exploiting the modern computing power so as to solve as large size problem as possible within a reasonable computational effort;
- development and implementation of heuristic algorithms so as to produce near optimal solutions to a general problem rather quickly.

More specifically, for TSP and VRP, researchers have developed exact algorithms based on partial enumeration schemes. Recent success in solving large size problems of TSP exactly is through polyhedral branch and cut methods. These methods depend on good mathematical formulations, polyhedral properties of the cutting planes and efficient coding implemented on modern fast computers. In spite of such advancements, the computational effort required to solve exactly some of the TSP are still high. For example, TSP of size 225-city, 4461-city and 7397-city were solved by using computational time of 1 year, 1.9 years and 4 years respectively (Jünger et al., 1995). Recently Applegate et al. (1998) solved a 13509-city problem using 3 months computational time on a network of 48 workstations.

The vehicle routing problem is a much more difficult problem when compared to TSP. Exact branch and cut algorithm have been developed for VRP by several authors. Largest size VRP solved exactly is a 134-city by Augerat et al. (1995) and Hill (1995).

Since the initial attempts, several heuristic algorithms have been proposed and used for both TSP and VRP. Some of these heuristics are deterministic and of polynomial computational complexity. Every implementation of a deterministic

heuristic on a given instance produces the same solution. Some authors have studied the complexity of finding a near-optimal tour for a TSP and established certain negative results (Johnson and Papadimitriou, 1985). Karp and Steele (1985) have made certain probabilistic analysis of heuristics assuming that the cities of a TSP are drawn independently from a uniform distribution over the d-dimensional unit cube. Recently a number of researchers have developed heuristics for both TSP and VRP based on probabilistic methods such as tabu search, neural networks, genetic algorithms and simulated annealing. Each implementation of a probabilistic heuristic on a given instance may produce a new solution. Some of these heuristics are of exponential computational complexity.

A typical manager faces a particular instance of a TSP and/or VRP that occurs in the day-to-day operations of his/her organisation. The concerned management is interested in solving the problem faced by them. But they get no guarantee of any desired quality solution from using any of the heuristics in the literature. For an end user, the current status of TSP and VRP can be summarised as follows:

- The exact algorithms may not solve a given instance of the problem in a reasonable time.
- There are too many heuristics available.
- Very little effort is put into assessing the quality of solutions produced by heuristics.
- Different implementations (runs) of the same probabilistic heuristic on a given input instance of a problem may produce distinct solutions of different quality. Thus desired quality and reproducibility of the solution can not be ensured.
- Most of the comparisons (available in the literature) between the performance of the various heuristics are based on the so called benchmark literature problems. Problems faced by the end user may not be similar to the benchmark literature problems. Furthermore, the performance of the

heuristics on the benchmark problems provide no guarantee on the quality of solutions that can be obtained in the problem faced by a typical end user.

- Most of the documentation on the performance of the heuristics on the literature problems provide no information regarding the computational effort (CPU time) spent on obtaining the claimed solution, reproducibility of the claimed solution and hardware environment of the implementation. Thus many of the comparisons documented are irrelevant to the end user.

- There is a lack of acceptable methods to compare various heuristics.

As noted by Bodin et al. (1983), annual distribution cost is estimated to be around $400 billion in the United States. Furthermore, surveys show that physical distribution costs account for about 16% of the sales value of an item consumed. Any reduction in these costs will be of significant benefit to consumers.

This thesis focuses on some of the above deficiencies. In the next section we provide a brief review of the contents of this thesis.

1.2 Review and Summary of Thesis

Most of the heuristics for general combinatorial optimisation problems are based on neighbourhood search methods. Hence we provide a formal setup for defining such neighbourhood structures, definitions of local optimum and global optimum. Furthermore, we highlight the dependence and drawbacks of such methods on the neighbourhood structure. Any given combinatorial optimisation problem, say

$$\text{minimise } \{f(x) : x \in S\}$$

can be reworded based on the neighbourhood structure as:

$$\text{Find } x^* \in S \text{ such that } f(x^*) = g_\theta^* = \min\{g_i^* : 1 \leq i \leq \gamma\}$$

where $g_i^*, 1 \leq i \leq \gamma$ are the distinct local minimum objective function values realised by the set of all local minimum solutions of S. Note that two distinct neighbourhood structures may lead to distinct sets of local optimum values, though the global optimum values will be same. Also for all $\mathcal{NP}$-hard problems, we have no prior knowledge of the local optimum values g_i^*, global optimum value $f(x^*)$ and the number of distinct local optimum values, γ. For a given neighbourhood structure, deterministic heuristics generate exactly one of the local optimum values. The probabilistic heuristics implemented several times on a given input data of the problem may generate some of the distinct g_i^*, but will not ensure anything about the quality of the solutions produced.

We thus emphasize the need for a statistical analysis of the output to be part of any such probabilistic heuristic. We develop and illustrate some of the statistical analysis that can be performed for the two probabilistic heuristics for TSP and VRP. Furthermore, these heuristics are part of a bigger class called tabu search heuristics for combinatorial optimisation problems. Hence this thesis includes an overview of the TSP, VRP and tabu search methods in Chapters 2, 3 and 4 respectively. Subsequently in Chapters 5, 6, 7 and 8 we develop our ideas of neighbourhood structure, local optimum etc. and demonstrate the required statistical analysis of some heuristics on TSP and VRP. In the following we briefly describe the topics discussed under each Chapter.

In Chapter 2, we present a review on exact algorithm and heuristics for the TSP. Exact algorithms are based on branch and bound and branch and cut methods. We also present a brief review of the tools such as subtour elimination constraints, cutting planes and various relaxed problems used in the exact methods. Furthermore, we present various heuristic algorithms based on tour construction procedures, tour improvement procedures and the combination of them.

In Chapter 3 we provide a review of the algorithms for two classes of VRP. We restrict our attention to CVRP and capacity and distance restricted VRP (CDVRP). After discussing a popular mathematical model for VRP, we provide a review of exact algorithms based on branch and cut methods. We also discuss

the various heuristics for the CVRP and CDVRP.

Our research relates to the tabu search methods and hence we provide an overview of the tabu search methods and its applications to TSP and VRP in Chapter 4.

In Chapter 5, we formally introduce and develop the ideas of neighbourhood structure, local optimum, global optimum and probabilistic heuristics for combinatorial optimisation problems. We highlight the drawbacks of the probabilistic heuristics for such problems. We emphasize the need for integrating the statistical analysis of local optimum values with probabilistic heuristics. Furthermore, we illustrate the need to select the best heuristic on the basis of testing of statistical hypothesis and related analysis.

Chapter 6 illustrates some of the ideas presented in Chapter 5 using the GENIUS algorithm for the TSP proposed by Gendreau et al. (1992). We present the details of the GENIUS algorithm in Section 6.2. In Section 6.3 and 6.4 we carry out statistical analysis for 36 variations of the GENIUS algorithm based on different neighbourhood parameters, different types of insertion methods used and two types of constructions of starting solutions. The analysis is performed on 27 literature problems with size ranging from 100 to 532 cities and 20 randomly generated problems with size ranging from 100 to 480 cities.

For the case of the literature problems, the result shows that 9_CGENI_ABC_US (with neighbourhood parameter $p = 9$, insertions methods type A, type B and type C and using the convex hull technique for constructing the starting solution) is the best heuristic among the 36 heuristics. For the case of randomly generated problems, 9_CGENI_AB_US (with neighbourhood parameter $p = 9$, insertions methods type A and type B and using the convex hull technique for constructing the starting solution) proved to be the best heuristic. Furthermore, the fitting of Weibull distribution gives an estimate of the optimal objective function value and a corresponding confidence interval in both the literature and randomly generated problems. The estimate of the optimal objective function values are within 8.2% of the best objective function value known for the literature problems and 6.5% for the randomly generated problems.

Since the GENIUS algorithm shows some promising results, we decide to combine the GENIUS algorithm with the branch and bound method to solve some large dimensional problems. This is discussed in Section 6.5. We test the algorithm on both the literature problems with sizes ranging from 575 cities to 724 cities and randomly generated problems with size ranging from 500 cities to 700 cities. Although the algorithm shows some unsatisfactory results, it solved the problems yielding a solution within 2.4% of the best known objective function value.

In Chapter 7, we pose similar questions for the vehicle routing problem. We present the details of the TABUROUTE algorithm for the CVRP and CDVRP in Section 7.2. Subsequently in Sections 7.3 and 7.4, we conduct a statistical analysis for four variations of the TABUROUTE algorithm based on different neighbourhood parameters. These are tested using 14 literature problems with sizes ranging from 50 cities to 199 cities and 49 randomly generated problems with sizes ranging from 60 cities to 120 cities. In both cases of the problems, the statistical tests accepted the hypothesis that there is no significant difference between the various versions of TABUROUTE heuristics.

By fitting the Weibull distribution to the objective function values of local optimal solution, we estimate the optimal objective function value and a corresponding confidence intervals, for each problem, for both the literature problems and randomly generated problems. The estimate optimal objective function values are to within 6.1% and 18.3% of the best known values for both the literature problems and randomly generated problems respectively.

In Chapter 8, we present the general neighbourhood search method for a general combinatorial optimisation problem. Note that very often, the neighbourhood structure can be defined suitably only on a superset $\mathcal{S}$ of the set of feasible solutions S. Thus the solutions in $\mathcal{S}\backslash S$ are infeasible. Furthermore the objective function $f(x)$ on S is suitably extended to $f'(x)$ on $\mathcal{S}$ such that $f'(x) = f(x)$ for every $x \in S$. Subsequently we define distinct local optimum objective function values as $f_i^*, 1 \leq i \leq \beta$ and $g_i^*, 1 \leq i \leq \gamma$ respectively for $\mathcal{S}$ and S. Several interesting questions are posed regarding the computational complexity of the

solution space of a problem. We illustrate these concepts for a 199-city CDVRP using the TABUROUTE algorithm.

We explore the idea of complexity of the solution space based on the samples such as the number of solutions with distance and/or capacity feasibility, the number of feasible neighbourhood solutions encountered for one run, etc. Such information was collected during the 140 runs. We pose the question: "how many local minimum solutions are there for the 199-city problem ?". The results show that the expected number of local minimum solutions for the 199-city problem is $4.8587 \text{x} 10^{374}$. Furthermore, the feasible region of the 199-city problem is approximately 11% of the total size of $\mathcal{S}$ based on the TABUROUTE algorithm. We conclude that this problem can not be used as a benchmark problem based on the data we have gathered.

Finally there are two appendices. Appendix A presents the detail of the Friedman test, the expected utility function test and the estimation of the optimal objective function value using fitting of the Weibull distribution. Appendix B presents a list of tables from Chapter 6, 7 and 8.

The concepts of Chapter 6, 7 and 8 can be further studied to integrate these methods into several of the recently developed probabilistic heuristics for many combinatorial optimisation problems. In Chapter 9, we summarise our conclusions on the basis of this thesis and indicate direction for future research in this areas.

1.3 Relevant Notation

In this section we present the assumptions and notation used in this thesis.

Travelling salesman problem (TSP)

Travelling salesman problem (TSP) is to find a route for the salesman to travel from his home city, visit each of $(n-1)$ other cities exactly once and return to

the home city such that the total distance travelled by him is minimised.

Let $G = (V, A)$ be a graph where $V = \{1, 2, ..., n\}$ is a set of vertices representing the cities. Without loss of generality we take city 1 as the home city of the salesman. The set of arcs can be denoted as $A = \{(i, j) : i, j \in V\}$ permitting the direct travelling by the salesman between two cities. Associated to arc (i, j), there is a cost (or distance) c_{ij} for travelling directly from city i to city j. The cost matrix $C = ((c_{ij}))$ is an nxn matrix associated to the set of arcs A. The TSP is called a symmetric TSP (STSP) if $c_{ij} = c_{ji}$ for every $(i, j) \in A$. The cost matrix C is said to satisfy the triangle inequality if and only if $c_{ij} + c_{jk} \geq c_{ik}$ for all i, j and k. A Hamiltonian tour in $G = (V, A)$ is defined as a cycle with $\mid V \mid$ edges on a graph with $\mid V \mid$ vertices.

Vehicle routing problem (VRP)

We use the following notation to describe the VRP introduced in Section 1.1.

Let $G = (V, A)$ be a graph where $V = \{0, 1, ..., n\}$, the vertices $i, 1 \leq i \leq n$ represent the n different customers and vertex '0' represents the single depot of the problem, $A = \{(i, j) : i \in V, j \in V, i \neq j\}$ represents the set of arcs permitting the direct travelling by a vehicle between two vertices of V.

Associated to arc (i, j) in A, there is a cost c_{ij} representing the distance or travel time from vertex i to vertex j, $i \neq j, i, j \in V$. Each customer i, has a demand q_i, of the commodity and service time $\delta_i, 1 \leq i \leq n$. Let m denote the number of vehicles used by the problem.

A vehicle route can be represented by a cycle $(0, i_i, i_2, ..., i_r, 0)$, starting and ending at depot '0' in graph G. Note that $\{i_i, i_2, ..., i_r\} \subseteq V$. For the case of the capacity restricted vehicle routing problem (CVRP), every vehicle route must satisfy the condition that the total demand of the customers in a vehicle route cannot exceed a common vehicle capacity denoted by Q. For the capacity and distance restricted vehicle routing problem (CDVRP), the total time travelled by a vehicle must not exceed an upperbound L.

Chapter 2

An Overview of The Travelling Salesman Problem

This chapter provides a literature overview for the Travelling Salesman Problem (TSP). TSP is one of the most widely studied problems in combinatorial optimisation. Since many applications can be modelled as a TSP or a variant, it has attracted many researchers from different fields such as Mathematics, Operations research, Artificial Intelligence and Physics. Some applications include drilling of printed circuit boards, X-ray crystallography, vehicle routing, scheduling and computer wiring. For a detailed discussions of these applications see Lawler et al. (1985). One important factor which interests researchers in the TSP is that though the problem is easy to formulate, it belongs to the class of $\mathcal{NP}$-hard ($\mathcal{NP}$ stands for non-deterministic polynomial time) problems. This class consists of problems that are computationally difficult. It is considered unlikely that polynomial time algorithms exist for TSP. A number of exact methods have been proposed to solve the TSP (Laporte, 1992). So far, the Branch and Bound method combined with polyhedral techniques provide the best approach to solving the TSP exactly. This approach is discussed in Section 2.1. Other techniques which provide near optimal solutions are called heuristics or approximate algorithms. That is, if a solution is obtained using a heuristic algorithm, it cannot be guaranteed that there is no other better solution available. Heuristics

algorithms are discussed in Section 2.2.

The concept of TSP dates back to the 1700s and the term "Travelling Salesman Problem" was introduced by Merrill Flood in 1930s (see Lawler et al., 1985). The TSP involves a salesman starting from his home city, visiting each city from the list exactly once and coming back to the home city such that the total travel distance is as small as possible. Let $G = (V, A)$ be a graph where $V = \{1, 2, ..., n\}$ is a set of vertices and $A = \{(i, j) : i, j \in V\}$ is a set of arcs. Let c_{ij} be the cost (or distance) associated with the arc (i, j) and $C = (c_{ij})$ be the cost (or distance) matrix associated with set A. The problem is symmetric if $c_{ij} = c_{ji}$ for all i and j. The cost matrix C is said to satisfy the triangle inequality if and only if $c_{ij} + c_{jk} \geq c_{ik}$ for all i, j and k. A **Hamiltonian tour** is defined as a cycle on a graph of n nodes traversing each node exactly once. The length of a Hamiltonian tour is the sum of the distances associated with the arcs in the tour (cycle). The TSP is to find a Hamiltonian tour with smallest length.

In the last two decades, the increasing power of computers, the development of polyhedral theory and efficient algorithms have contributed enormously to the progress made in solving the TSP. Recently, Applegate, Bixby, Chvátal and Cook have solved some of the "unsolved" problems in TSPLIB (Reinelt, 1995). These include a 3038-city problem solved in 1991, a 4461-city problem solved in 1993, 7397-city problem solved in 1995 and a 13509-city problem solved in 1998. Despite these successes, TSP is far from being solved satisfactorily. Some relevant points to be noted are:

1. There is no algorithm which is guaranteed to solve any instance; that is, solving a problem exactly regardless of its size, structure and characteristics.

2. Many exact algorithms developed for the TSP are not capable of solving large practical size problems in a reasonable time.

3. Heuristic algorithms or approximate algorithms proposed to provide near optimal solutions cannot ensure desired quality in terms of near optimality.

4. Very often randomness is involved in some of the heuristic algorithms. Such

heuristics cannot ensure reproducibility of solutions for a problem under different executions of the heuristic.

Despite improvements in computer technology, such as the availability of supercomputers and parallel computers, many problems still remain difficult to solve. A group of researchers from CRPC (Applegate et al., 1998) took 3 months computational time on a cluster of 48 workstations to solve the 13509-city problem. That is equivalent to 12 years computational time on a single processor machine.

For literature reviews of the TSP we refer to the book by Lawler et al. (1985), and papers by Laporte (1992) and Reinelt (1994). The rest of this chapter is divided into two sections. Section 2.1 and 2.2 discuss the exact algorithms and heuristic algorithms used recently for the TSP respectively.

2.1 Exact Algorithms

Exact algorithms used to solve the TSP are based on one of the mixed integer linear programming formulations for the TSP and/or Branch and Bound methods. Dantzig, Fulkerson and Johnson (1954) provided the following mixed integer linear programming:

Minimise

$$\sum_{i}\sum_{j\neq i} c_{ij}x_{ij} \tag{2.1}$$

subject to

$$\sum_{j\neq i} x_{ij} = 1, \quad 1 \leq i \leq n \tag{2.2}$$

$$\sum_{i\neq j} x_{ij} = 1, \quad 1 \leq j \leq n \tag{2.3}$$

$$0 \leq x_{ij} \leq 1,\ x_{ij} \text{ integer },\ 1 \leq i \leq n,\ 1 \leq j \leq n,\ i \neq j \tag{2.4}$$

$$\sum_{i,j\in S, i\neq j} x_{ij} \leq \mid S \mid -1,\ \forall S \subseteq \{1,2,...,n\},\ 2 \leq \mid S \mid \leq n-2 \tag{2.5}$$

We denote (2.1) to (2.5) by Problem $\mathcal{P}$. Next consider the relaxed linear programming problem denoted by $\mathcal{P}'$ obtained from $\mathcal{P}$ by relaxing the integer restrictions in (2.4) and a subset of constraints in (2.5). The following observations are made:

- Consider any Hamiltonian tour and correspondingly define

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ belongs to the Hamiltonian tour} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

  It is easy to check that x_{ij} satisfy (2.2) - (2.4).

- Every solution to (2.2) - (2.4), restricting attention to $x_{ij} = 1$, may not necessarily give rise to a Hamiltonian tour. In fact such solutions may give rise to non-Hamiltonian cycle called subtours (i.e. cycles not involving every node).

- The set of constraints (2.5) precisely eliminate solutions with subtours and hence are called *subtour elimination constraints.*

- Note that the number of constraints of type (2.5) increases exponentially (of the order of 2^n) with the number of cities, n. Hence for most values of n, the above model (2.1) - (2.5) cannot be solved as a linear programme even if we relax the 0-1 restrictions in (2.4).

- The relaxed problem (2.1) - (2.4) is the well known Assignment Problem and there are efficient algorithms to solve it.

- Solution x^*_{ij} to $\mathcal{P}'$ is an optimal solution to $\mathcal{P}$ if x^*_{ij} satisfies all the constraints (2.2) - (2.5).

- Furthermore, checking the feasibility of $((x^*_{ij}))$ to $\mathcal{P}$ (i.e. the constraints (2.2) - (2.5)) is equivalent to checking that $T = \{(i,j) : x^*_{ij} > 0\}$ forms a Hamiltonian tour in $G = (V, A)$.

- Given that $T \subset A$, it is very easy to check whether T forms a Hamiltonian tour. Furthermore, if T is not a Hamiltonian tour then it is very easy to find a violation subtour elimination constraints of type (2.5). Dantzig et al. (1954) called constraints (2.5) the *cutting planes.*

Based on these observations, Dantzig et al. (1954) proposed the following method to solve the problem $\mathcal{P}$.

Step 1. Define Problem $\mathcal{P}'$ as (2.1) - (2.4) without integer restriction.

Step 2. Solve $\mathcal{P}'$ and obtain $((x_{ij}))$, $T := \{(i,j) : x^*_{ij} > 0\}$.

Step 3. If T forms a Hamiltonian tour then stop; Otherwise, locate a solution S which violates constraint (2.5) and add this constraint to problem $\mathcal{P}'$. Go to Step 2.

They solved a 49-city problem which was considered large at that time. Their work showed that the cutting plane concept was relevant to solve general mixed integer linear programming problems. The method was later formalised by Gomory (1958, 1960, 1963) (cited in Lawler et al., 1985) who invented the *Cutting Plane Algorithm* for integer linear programming problems. This method can solve problems in a finite number of iterations which may be large. However the most popular method to solve TSP is the Branch and Bound method which is discussed in Section 2.1.1 and the Branch and Cut method. This approach incorporates the polyhedral approach to solving the TSP and is discussed in Section 2.1.2.

There are a vast number of papers published in the literature for the TSP and we present here some of the important algorithms for the Symmetric TSP (STSP) and Asymmetric TSP (ATSP) in Tables 2.1 - 2.2.

Author/(s)	*Method(s)*	*Comments*
Dantzig et al. (1954)	Integer linear programming formulation	Provided the first formulation and solved a 49-city problem
Held and Karp (1970, 1971), Christofides (1970)	1-tree relaxation	Proposed the 1-tree relaxation using Branch and Bound method for TSP
Bellmore and Malone* (1971)	Assignment Problem (AP) relaxation	Proposed the Branch and Bound method using AP relaxation. Both symmetric and asymmetric problems are tested.
Helbig Hansen and Krarup (1974)	1-tree relaxation	Improved the algorithm of Held and Karp (1971)
Smith and Thompson (1977)	1-tree relaxation	Used less memory than the Karp and Held's algorithm
Bazaraa and Goode* (1977)	1-tree relaxation	Extended the Held and Karp and introduced the stepped fathoming scheme in the Branch and Bound method
Miliotis (1976,1978)	Branch and cut method with subtour elimination inequalities	Provided a competitive approach to solving the TSP
Houck et al. (1980)	n-path relaxation	Produced weaker bound than 1-tree relaxation and required longer computation time
Padberg and Hong (1980)	Branch and cut method with subtour elimination and 2-matching inequalities	Solved 74 problems with sizes ranging from 15 to 318 cities

Table 2.1: Exact algorithm for the STSP. (The * indicates that the proposed algorithm is used for both ATSP and STSP)

Author/(s)	*Method(s)*	*Comments*
Crowder and Padberg (1989)	Branch and cut method with subtour elimination and comb constraints	Solved problems of sizes ranging from 48 to 318 cities
Carpaneto et al. (1989)	Additive bounding procedure	Randomly generated test problems of size up to 200 vertices were solved
Malik and Fisher (1990)	1-tree relaxation	A dual ascent algorithm to compute the multipliers of lagrangean relaxation and required less computational time
Descrochers and Laporte (1991)	Integer linear programming	Improved the MTZ subtour elimination constraints
Grötschel and Holland (1991)	Branch and cut method with subtour elimination, 2-matching, comb and clique tree inequalities	Solved problems up to 1000 cities
Padberg and Rinaldi (1991)	Branch and cut method with subtour elimination, 2-matching, comb and clique tree inequalities	Solved 42 symmetric TSPs with sizes ranging from 48 to 2392 cities
Applegate et al. (1995)	Branch and Cut method with clique-tree-like inequalities	Solved some unsolved problems from TSPLIB with sizes ranging from 225 to 7397 cities
Jünger and Störmer (1995)	Parallel implementation on the Branch and Cut method by Jünger et. al. (1994)	Solved some literature problems from TSPLIB
Applegate et al. (1998)	Branch and Cut with various facet-defining inequalities	Solved the 13509 city problem from the TSPLIB with a network of 48 workstations

Table 2.1: (Cont.) Exact algorithm for the STSP. (The * indicates that the proposed algorithm is used for both ATSP and STSP)

Author/(s)	*Method(s)*	*Comments*
Smith et al. (1977)	AP relaxation	Made two improvements: lower bounds were calculated using the cost operators based on the assignment solution from parent node; and a last in first out (LIFO) branching strategy was adopted to save the computer memory. Problems of sizes up to 200 nodes were tested.
Smith (1978)	1-tree relaxation	Found the reason that AP relaxation is more efficient for the STSP
Carpaneto and Toth (1980)	AP relaxation	Proposed a new branching rule and a new way to calculate lower bound. The algorithm solved up to 240 cities problems
Balas and Christofides (1981)	Lagrangean relaxation based on AP and Branch and Bound method	Made various improvements including the lower bound calculated from the Lagrangean problem from AP, new branching and bounding procedures were used. The proposed algorithm required less computational time and smaller search tree. 120 randomly generated problems were solved with maximum size up to 325 nodes
Fischetti and Toth (1989)	Additive bounding and Branch and Bound method	Proposed different boundings and problem of size 2000 was solved in 8329 seconds
Miller and Pekny (1991)	Dual of AP	Used a method which quickly pruned a large number of solutions. Problems with various matrix structures were tested
Pekny et al. (1991)	Dual of AP	Used an algorithm for Directed Hamiltonian Cycle (DHC) for ATSP so that the problem produced a Hamiltonian tour or that the tour did not exist. Randomly generated problems of size up to 3000 nodes were solved
Pekny and Miller (1992)	Dual of AP	Parallel implemented the Miller and Pekny (1991) algorithm by converting the cost matrix to a sparser matrix so that the optimality of the final solution was retained. Various classes of problems were tested with sizes ranging from 250 to 3000 cities
Carpaneto et al. (1995)	AP relaxation and Branch and bound method	Improved the Carpaneto and Toth (1980) algorithm and solved up to 2000 vertices on DEC station 5000/240 computer in less than 3 minutes

Table 2.2: Exact algorithm for the ATSP

2.1.1 Branch and Bound methods

The Branch and Bound method can be described as a partial enumeration method to which a search tree is associated. The root of the search tree corresponds to the entire set of feasible solutions. Every node of the search tree corresponds to a subset of solutions with an associated lower bound on the objective function value of the solutions from this subset. The lower bound is often obtained by solving a corresponding relaxed linear programming problem. While searching for an optimal solution at a node of the search tree, the associated subset of solutions can be partitioned into smaller subsets with corresponding descendent nodes of the search tree. This method of creating new search paths is called branching. In the search, under certain conditions nodes can often be discarded. The worst case involves a complete enumeration of the search tree. The process stops when every node is either discarded or searched.

The concept of Branch and Bound method was used by Dantzig et al. (1954) in solving the 49 cities problem. However, Eastman (1958) was the first who attempted to solve the TSP by Branch and Bound (cited in Lawler et al., 1985) and the term Branch and Bound was first introduced by Little et al. (1963).

We outline the Branch and Bound algorithm in the following:

Step 1. (Initialisation) Set the upper bound $z^* = \infty$ and put TSP on the list, Q.

Step 2. (Node selection, Subproblem and Branching rule) If Q is empty and $z^* < \infty$ then TSP tour is optimal; If Q is empty and $z^* = \infty$ then there is no solution. If Q is not empty, choose the problem (or subproblem) $\mathcal{P}'$ from Q according to **subproblem selection** and remove it from Q.

Step 3. (Lower bound) Solve the relaxation of subproblem $\mathcal{P}'$ and let $\underline{z}$ be its optimal cost. If $\underline{z} \geq z^*$, go to **Step 2**. If $\underline{z} < z^*$ and the optimal solution of $\mathcal{P}'$ forms a hamiltonian tour, store the solution and set $z^* = \underline{z}$. Otherwise, apply a **branching rule** to define new subproblems as descendants of the subproblem $\mathcal{P}'$ and store them in Q. Go to **Step**

2.

Different **Subproblem Selection Rules**, **Branching Rules** and **Relaxation** are used by different authors. For details refer to Lawler et al. (1985) and Laporte (1992) and the references there. A number of authors have used different relaxations of the linear programming formulation for TSP such as the Assignment Problem (AP), 1-tree relaxation and Lagrangean relaxation based on AP. The various formulations and relaxations for the ATSP and the 1-tree relaxation formulations for the STSP are described in the following.

Assignment Problem

The Assignment Problem (AP) can be described as assigning n jobs to m machines and the total cost $\sum_i \sum_{j \neq i} c_{ij} x_{ij}$ is minimum. This is one of the first relaxations used for the TSP. Note that (2.1) - (2.4) provide a formulation of the AP. Thus it is a relaxation of the TSP formulation (2.1) to (2.5) provided by Dantzig, Fulkerson and Johnson (1954). Note that a lot of the algorithms are based on the AP relaxation are for asymmetric TSP. Since our work is concentrated on the symmetric TSP, the AP relaxation is briefly discussed here. Some of the effective algorithms are proposed by Smith et al. (1977), Carpaneto and Toth (1980), Balas and Christofides (1981), Miller and Pekny (1991) and Carpaneto et al. (1995). For detail of the subject, we refer to Balas and Toth (1985) and the references above. In the following we discuss some of the recent work by various authors.

Carpaneto and Toth algorithm (1980)

Carpaneto and Toth (1980) proposed an algorithm based on the lowest-first branch and bound (branching is always done on the node with lowest objective function value) for the ATSP. At each node, h say, the problem is solved by the Modified Assignment Problem (MAP_h) defined by constraints (2.1) - (2.4) and additional constraints associated with excluded arcs subset E_h and included

arcs subset I_h defined as follows:

$$E_h = \{(i,j) \in A : x_{ij} \text{ is fixed to } 0\},\ I_h = \{(i,j) \in A : x_{ij} \text{ is fixed to } 1\}.$$

Suppose the optimal solution of MAP_h does not produce a hamiltonian tour and the optimal objective function value of MAP_h is less than the current best known solution, then m subproblems are generated from node h according to the new branching rule (derived from Bellmore and Malone (1971)) described below:

Let $G_q, q = 1, ..., d$, be the subtours produced from MAP_h and consider a subtour $G_q = (V_q, A_q)$ with $V_q = \{r_{q1}, ..., r_{qe_q}\}$ and arcs $A_q = \{(r_{q1}, r_{q2}), ..., (r_{qe_q}, r_{q1})\}$ where $e_q = \mid V_q \mid = \mid A_q \mid$ for the qth subtour. The subtour G_p which has the minimum number of not-included arcs in the set I_h such that

$$m = e_p - \mid A_p \cap I_h \mid = min_{\{q=1,...,d\}}\{e_q - \mid A_q \cap I_h \mid\}$$

is chosen for branching.

Let $\overline{A} = \{(s_1, t_1), ..., (s_m, t_m)\}$ be the subset of not-included arcs of A_p. The subset of the excluded and included arcs associated with jth descending node $g(j), j = 1, ..., m$ of node h is

$$E_{g(j)} = E_h \cup \{(s_j, t_j)\},$$
$$I_{g(j)} = I_h \cup \{(s_i, t_i) : i = 1, ..., j-1\}.$$

The proposed algorithm was able to solve 20 randomly generated problems with sizes ranging from 40 to 240 cities in average CPU time of 3-36 seconds on CDC 6600.

Carpaneto, Dell'amico and Toth algorithm (1995)

Carpaneto et al. (1995) proposed a lower first, branch and bound algorithm based on AP relaxation and a subtour elimination branching scheme. The algorithm improves from the algorithm of Carpeneto and Toth (1980) in the following four aspects:

1. At the root node (node 0), the reduction procedure removes any arcs which cannot belong to an optimal tour. The original matrix C is transformed into $\overline{C}$ where each entry of C is reduced by u_i and v_j i.e. $\bar{c}_{ij} = c_{ij} - u_i - v_j \geq 0$ where u_i and v_j are the optimal solution of the dual problem associated with the AP. $\bar{c}_{ij}$ represents a lower bound on the increase of the optimal solution value of AP corresponding to the inclusion of arc (i, j) in the solution AP. Let $\underline{z}$ be the lower bound obtained from the dual problem of AP. If the feasible solution, say z^*, is known, then each arc $(i, j) \in A$ such that $\bar{c}_{ij} \geq z^* - \underline{z}$ is removed since the inclusion of the arc will not lead to solution less than z^*. Hence the original graph G has been transformed into a sparse one, $\tilde{G} = (V, \tilde{A})$ where $\tilde{A} = \{(i, j) \in A : \bar{c}_{ij} < z^* - \underline{z}\}$.

2. The MAP is solved at each node and the efficiency of the algorithm depends greatly on solving MAPs. At node h, a parametric technique is used to find only one *shortest augmenting path* instead of solving the MAP_h from scratch and arc (s, t) is excluded from the solution MAP_k (k is the parent node of h). To obtain a solution from MAP_k, we only need to consider constraints (2.2) for $i = s$ and constraints (2.3) for $j = t$ in the reduced cost matrix c_{ij}. The resulting time complexity to solve each MAP is $O(\mid \tilde{A} \mid logn)$.

3. Effective data storage is introduced to store the information associated with the node of the decision tree, avoiding the updating of the unchanged information.

4. Consider a node h which has several optimal solutions. A connecting procedure which decreases the number of subtours is applied repeatedly using the following:

   Given two subtours $G_a = (V_a, A_a)$ and $G_b = (V_b, A_b)$, consider a pair of arcs $(i_a, j_a) \in A_a$ and $(i_b, j_b) \in A_b$ such that $\bar{c}_{i_a,j_a} = \bar{c}_{i_b,j_b} = 0$, then subtours G_a and G_b can be connected to form a unique subtour $G_a = (V_a \cup V_b, A_a \cup A_b$ $((i_a, j_a) \cup (i_b, j_b)) \cup ((i_a, j_b) \cup (i_b, j_a))$.

   If a hamiltonian tour is found, it is the optimal solution to node h.

Using the improved algorithm, the randomly generated problems of sizes up to 2000 nodes were solved on a DEC station 5000/240 computer in less than 3 minutes.

Miller and Pekny (1991)

Miller and Pekny (1991) considered the duality of the AP which can be formulated as follows:

Maximise

$$\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \tag{2.7}$$

subject to

$$c_{ij} - u_i - v_j \geq 0, \;\; i, j \in V, \;\; i \neq j \tag{2.8}$$

To effectively solve a problem by avoiding examining a large number of solutions, they created a simpler problem which has the same optimal solution to the original problem and can be solved faster. Consider a modified cost matrix c'_{ij} for the ATSP', associated with a relaxation AP' and its dual-AP':

$$c'_{ij} = \begin{cases} c_{ij} & \text{if } c_{ij} \leq \lambda \\ \infty & \text{otherwise.} \end{cases}$$

Let the optimal value of ATSP and AP be $v(\text{ATSP})$ and $v(\text{AP})$ respectively. Consider the following proposition (Miller and Pekny, 1991):

An optimal solution z^* for ATSP' is an optimal solution for ATSP if

$$v(\text{ATSP}) - v(\text{AP}) \leq \lambda + 1 - u'_i - v'_{max} \tag{2.9}$$

and

$$\lambda + 1 - u'_i - v'_{max} \geq 0, \;\; \forall i \in V \tag{2.10}$$

where (u', v') is an optimal solution to dual AP' and v'_{max} is the maximum element of v'. Note that $\lambda + 1 - u'_i - v_{max}$ is the smallest reduced cost of the discarded matrix element which guarantees that no excluded element can lead to better solution.

A feature is added to their branch and bound method, to quickly find the solution for ATSP from the set of optimal solutions to AP. This feature (Hamiltonian cycle problem reduction) is to reduce the solution of ATSP to the solution of a Hamiltonian cycle problem on a given graph.

Let an admissible graph be defined as $\overline{G} = (V, \overline{A})$ where V is a set of vertices and $\overline{A} = \{(i,j) \mid c_{ij} - u_i^* - v_j^* = 0\}$ where u_i^* and v_j^* are the optimal dual-AP variables. A hamiltonian cycle on $\overline{G}$ is an optimal solution to ATSP. If $\overline{G}$ does not have a Hamiltonian cycle, then the lower bound AP can be increased by the smallest nonzero reduced cost. This feature is incorporated in their branch and bound method.

Five different classes of randomly generated problems were tested using the algorithm suggested by Miller and Pekny (1991) and the problem with 500000-city was solved on a Cray2 supercomputer which required 12,623 seconds computing time. Furthermore, Pekny and Miller (1992) implemented the above method using parallel implementation. Three different classes of randomly generated problems were tested and problems of size up to 10000-city have been solved.

This concludes the review for the ATSP. The 1-tree relaxation for the STSP is discussed in the following.

1-tree Relaxation Approach

Consider that an undirected graph $G = (V, A)$ where $V=\{1, ..., n\}$ is a set of vertices and A is a set of edges. Let c_{ij} be the edge weight from node i to j. A tree is a connected graph without a cycle. Given the weighted graph G, the minimum spanning-tree problem is to find a spanning tree in G having the smallest edge sum weight. A 1-tree consists of a connected graph with two distinct edges joining to vertex 1 and the graph restricted to $V - \{1\}$ is a tree. A 1-tree is a tour if and only if each vertex has degree 2. The following formulation of the minimum 1-tree problem is due to Christofides (1970) and Held and Karp (1970, 1971):

Minimise

$$\sum_{i<j} c_{ij}x_{ij} \tag{2.11}$$

subject to

$$\sum_{j} x_{1j} = 2, \tag{2.12}$$

$$\sum_{1\leq i<j\leq n} x_{ij} = n, \tag{2.13}$$

$$\sum_{i,j\in S, i<j} x_{ij} \leq \mid S \mid -1, \ S \subset \{2,3,...,n\}, \tag{2.14}$$

$$0 \leq x_{ij} \leq 1, \ x_{ij} \text{ integer.} \tag{2.15}$$

Assuming x_{ij}'s to be 0-1 variables, we can note the following. Constraints (2.12) ensure that vertex 1 has degree two. Constraint (2.13) ensures that the edge set $\{(i,j) : x_{ij} = 1\}$ forms a tree on vertices of $V - \{1\}$. Constraints (2.14) are the subtour elimination constraints which prevent the formation of subtours. Held and Karp (1970) established that the extreme points of the polyhedron generated by (2.12) - (2.15) has one to one correspondence with the set of 1-trees on the vertex set V. A solution x_{ij} to (2.12) - (2.15) representing a 1-tree will form a TSP tour if it further satisfies

$$\sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2, \ 2 \leq i \leq n-1. \tag{2.16}$$

Note that (2.16) ensures that each vertex has degree two. Thus (2.11) -(2.16) provide another formulation of the TSP, where the minimum weight 1-tree problem is a relaxation of the TSP. Furthermore, if a minimum weight 1-tree given by x^*_{ij} forms a TSP tour then x^*_{ij} is an optimal tour for the TSP.

Held and Karp (1970) used Lagrangian relaxation techniques to solve the TSP (2.11) - (2.16). They added the equation (2.16) to the objective function (2.11) and defined the Lagrangian relaxation problem as follows:

$$L(\pi) = min \left\{ \sum_{i<j} c_{ij}x_{ij} + \sum_{i} \pi_i \left[\sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} - 2 \right] : x_{ij} \text{ satisfies (2.12)} - \text{(2.15)} \right\} \tag{2.17}$$

that is,

$$L(\pi) = min\left\{\sum_{i<j}(c_{ij}+\pi_i+\pi_j)x_{ij} - 2\sum_i \pi_i : x_{ij} \text{ satisfies (2.12)} - \text{(2.15)}\right\}. \tag{2.18}$$

The strongest lagrangean relaxation is given by $\pi = \pi^*$ such that

$$L(\pi^*) = \max_{\pi}\{L(\pi)\} \tag{2.19}$$

and this problem is called the **lagrangean dual** of the TSP. The Lagrangian relaxation problem $L(\pi^*)$ provides a lower bound for the optimal tour length of the TSP. Subgradient optimisation method is used to solve (2.19) and further details can be found in Nemhauser and Wolsey (1988). Problems of sizes 20 to 64 nodes were solved.

Other authors who have contributed to this topic include Helbig Hansen and Krarup (1974) who improved the Held and Karp algorithm by considering different methods for calculating π in the subgradient method. Their algorithm is 25 times faster than the Held and Karp's algorithm. The problems tested range from 10 to 80 nodes. Held and Karp used the breadth first search method where a list of subproblems is created which require a large memory space for storage. The disadvantage of this is it limits the sizes of the problems solved. To overcome this, Smith and Thompson (1977) proposed the last-in-first-out (LIFO) implicit enumeration search algorithm which requires less memory space. Problems of sizes up to 100 nodes were solved. Bazaraa and Goode (1977) extended the work of Held and Karp and solved the dual problem of the original formulation. In the case of STSP, three literature problems of sizes up to 57 cities and 10 randomly generated problems of sizes up to 80 cities were solved. In the case of ATSP, 8 randomly generated problems of sizes up to 30 to 60 cities were solved. Malik and Fisher (1990) proposed the dual ascent for finding the multipliers of the Lagrangean relaxation which is faster and used less CPU time compared with the subgradient method used by Held and Karp in which ten literature problems of size up to 100 cities were solved.

2.1.2 Branch and Cut methods

Branch and cut is an approach combining the Branch and Bound method and polyhedral theory to find strong cutting planes or constraints describing the feasible solution space of the TSP. This method has been successful in solving some of the large dimensional hard problems in the literature. Some polyhedral theory terminology is described below:

Recall that a problem $\mathcal{P}$ can be defined as $\mathcal{P} : min\{cx$ subject to $x \in S\}$ where S is defined as a set of feasible solutions. The relaxation of $\mathcal{P}$ can be defined as $\mathcal{P}' = min\{cx$ subject to $Ax \leq b\ , x \in \mathbf{R}^n\}$. The **polyhedron** of $\mathcal{P}'$ can be described by an intersection of a set of a system of linear inequalities i.e. $\mathcal{P}' = \{x \in \mathbf{R}^n | Ax \leq b\}$. A bounded polyhedron is called a **polytope**. An inequality $a^T x \leq a_0$ is **valid** for $\mathcal{P}' \subseteq \mathbf{R}^n$ if $\mathcal{P}' \subseteq \{x \in \mathbf{R}^n | a^T x \leq a_0\}$. A subset F of the polyhedron $\mathcal{P}'$ is called a **face** of $\mathcal{P}'$ if there exists a valid inequality $a^T x \leq a_0$ with respect to $\mathcal{P}'$ such that $F = \{x \in \mathcal{P}' | a^T x = a_0\}$. A convex hull of $\mathcal{P}'$ is the smallest convex set containing all point in $\mathcal{P}'$ and is denoted by conv($\mathcal{P}'$). So a **facet-defining** valid inequality is a valid inequality describing conv($\mathcal{P}'$). Facet-defining inequalities can be used to give a strong relaxation of $\mathcal{P}$.

The idea of Branch and Cut method was first introduced by Dantzig et al. (1954) in 1954 to solve a 49-city problem. The relaxation problem $\mathcal{P}'$ is solved and a solution x is obtained. If x lies outside of S, then x can be separated from S by a cutting plane which satisfies all the points of S and violates the solution x. This is added to the problem which results in a tighter relaxation. This process is continued until a solution in S is found. However, the challenge of the Branch and Cut method is to find the facet-defining inequalities which describe the convex hull of the feasible solutions of the TSP. Over the years, some facet-defining inequalities for TSP have been developed in the literature. The first facet-defining inequality i.e. **subtour elimination constraint** was discovered by Dantzig et al. (1954) for their integer linear programming formulation. Grötschel and Padberg (1979a, 1979b) introduced two classes of inequalities called **2-matching** and

comb which can be used as cutting planes. Grötschel and Pulleyblank (1985) introduced another class of inequalities called **clique tree inequalities**. These inequalities have been used successfully in solving the TSP in recent years for large dimension problems. It should be noted that other classes of facet defining inequalities (see Naddef and Rinaldi, 1991, 1993) have been found but have not proven computationally useful. The above four families of the inequalities will be discussed briefly in the following. For further details refer to Grötschel and Padberg (1985).

Consider a graph $G = (V, A)$, an edge $(i, j) \in A$ is an ordered pair (i, j) of nodes of V. Let S be a subset of nodes in V and define $E(S) = \{(i, j) \in A \mid i \in S, j \in S\}$. The four families of facet defining inequalities are described in the following:

(a) **Subtour elimination constraints** (Dantzig et al. (1954), Grötschel and Padberg (1979a, 1979b)) were first identified by Dantzig et al. (1954). This constraint is to prevent the formation of the subtours and hence the name 'subtour elimination'.

$$\sum_{i,j \in S, i \neq j} x_{ij} \leq \mid S \mid -1, \;\; S \subset V, \;\; 2 \leq \mid S \mid \leq n-2. \tag{2.20}$$

Another form of (2.20) is given in the following:

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 1, \;\; S \subset V, \;\; \bar{S} = V \setminus S, \;\; 2 \leq \mid S \mid \leq n-2. \tag{2.21}$$

Constraint (2.21) can be interpreted as there is at least one edge from S connected to $\bar{S}$.

(b) **2-matching inequalities** (Grötschel and Padberg (1979a, 1979b)) make sure that every vertex contains exactly two edges. The formulation is defined as

$$\sum_{(i,j) \in E(H)} x_{ij} + \sum_{(i,j) \in A'} x_{ij} \leq \mid H \mid + \frac{1}{2}(\mid A' \mid -1) \tag{2.22}$$

for all $H \subset V$ and all $A' \subset A$ satisfying

(i) $\mid \{i, j\} \cap H \mid = 1$ for all $(i, j) \in A'$;

(ii) $\{i_k, j_k\} \cap \{i_l, j_l\} = \emptyset, k \neq l$ and $(i_k, j_k), (i_l, j_l) \in A'$;

(iii) $\mid A' \mid \geq 3$ and odd.

The set H is called the *handle* and the edges of set A' are called the *teeth.*

(c) **Comb inequalities** (Grötschel and Padberg (1979a, 1979b)) are defined as:

$$\sum_{(i,j)\in E(H)} x_{ij} + \sum_{k=1}^{s} \sum_{(i,j)\in E(T_k)} x_{ij} \leq \mid H \mid + \sum_{k=1}^{s} (\mid T_k \mid -1) - \frac{1}{2}(s+1) \qquad (2.23)$$

for all $H, T_1, ..., T_s \subseteq V$ satisfying

(i) $\mid T_k \cap H \mid \geq 1, k = 1, ..., s,;$

(ii) $\mid T_k \setminus H \mid \geq 1, k = 1, ..., s,;$

(iii) $T_k \cap T_l = \emptyset, 1 \leq k < l \leq s,;$

(iv) $s \geq 3$ and odd.

The set H is called the handle and sets T_k are called the teeth of the comb. If (i) is satisfied with equality the comb inequality is called simple.

(d) **Clique tree inequalities** (Grötschel and Pulleyblank, 1986) are defined as:

$$\sum_{k=1}^{r} \sum_{(i,j)\in E(H_k)} x_{ij} + \sum_{k=1}^{s} \sum_{(i,j)\in E(T_k)} x_{ij} \leq \sum_{k=1}^{r} \mid H_k \mid + \sum_{k=1}^{s} (\mid T_k \mid -t_k) - \frac{1}{2}(s+1) \quad (2.24)$$

for all $H_1, ..., H_r \subseteq V$ and $T_1, ..., T_s \subseteq V$ are called the handles and teeth of the clique tree. The number of handles intersected by the tooth T_k is defined by t_k. A clique tree is a connected subgraph of K_n whose cliques satisfy the following properties:

(i) The cliques are partitioned into two sets namely the set of handles and the set of teeth;

(ii) no two teeth intersect;

(iii) no two handles intersect;

(iv) each tooth contains at least 2 and at most $n-2$ nodes and at least one node not belonging to any handle;

(v) each handle intersects odd numbers (≥ 3) of teeth;

(vi) if a tooth T and a handle H have a nonempty intersection, then $H \cap T$ is an articulation set of the clique tree. An articulation set is referred to a subset of vertices, say A, in a connected graph, G, such that graph G is disconnected by removing the set A.

Recall from previous section that while solving TSP using the Branch and Bound method the lower bound must be calculated as close to the upper bound as possible. The advantage of using facet defining cuts is that they will usually result in stronger lower bounds being obtained. Weyl (1935) (cited in Padberg and Rinaldi, 1990a) established that the convex hull of the set of all TSP tours can be described by a system of linear inequalities

$$lx \leq l_0, \forall (l, l_0) \in \mathcal{L} \tag{2.25}$$

where $\mathcal{L}$ is a finite family of linear inequalities such that each inequality of $\mathcal{L}$ induces a different facet of the convex hull. Note that $x = (x_{ij}) \in \mathrm{R}^{n(n-1)}$. Only a proper subfamily $\mathcal{L}'$ of the family of $\mathcal{L}$ is known and $\mid \mathcal{L}' \mid$ increases exponentially in terms of the number of cities n. So to solve the TSP using the Branch and Cut method, the following relaxed problem using the cutting plane procedure is solved.

Minimise

$$\sum_i \sum_{j \neq i} c_{ij} x_{ij} \tag{2.26}$$

subject to

$$\sum_j x_{ij} + \sum_j x_{ji} = 2, \quad 1 \leq i \leq n, \tag{2.27}$$

$$lx \leq l_0 \ \forall (l, l_0) \in \mathcal{L}', \tag{2.28}$$

$$0 \leq x \leq 1. \tag{2.29}$$

The cutting plane procedure is given below. Define $\mathcal{L}_0$ to be a set of known inequalities.

Step 1. Set $\bar{\mathcal{L}} = 0$.

Step 2. Solve (2.26), (2.27), (2.28) for $\mathcal{L}' = \bar{\mathcal{L}}$ and (2.29) and let the solution be $\bar{x}$.

Step 3. Find one or more inequalities in $\mathcal{L}_0$ violated by $\bar{x}$.

Step 4. If none is found, then stop; Otherwise add the inequalities to $\bar{\mathcal{L}}$ and go to Step 2.

This method depends on efficient ways of finding inequalities of $\mathcal{L}_0$ in Step 3. This problem is referred to as the identification or separation problem. Exact methods are usually unavailable or too slow and hence heuristic procedures are used.

The term Branch and Cut was first used by Padberg and Rinaldi (1987) in their algorithm to solve a 532-city problem which essentially has four components: a heuristic to find an upper bound, a set of procedures to identify the violated inequalities, an LP solver and a procedure which combines branching and cutting plane techniques. The exact methods for defining subtour elimination constraints and 2-matching inequalities were found by Gomory and Hu (1961) and Padberg and Rao (1982) (cited in Padberg and Rinaldi, 1990a). Improved methods are discussed in Padberg and Hong (1980), Padberg and Rinaldi (1990a, 1991) and Applegate et al. (1995). In particular, Padberg and Rinaldi (1991) proposed efficient reduction techniques to speed up the convergence of the exact methods.

With the development of powerful computers and elegant Branch and Cut algorithms, large size TSP's have been solved to optimality. Crowder and Padberg (1980) solved problems from 48 to 318 cities with the use of subtour elimination and comb constraints. Padberg and Hong (1980) solved a total of 74 problems with sizes ranging from 15 to 318 cities using an exact procedure for finding subtour elimination constraints and a heuristic for the 2-matching constraints. Grötschel and Holland (1991) solved a 1000 cities problem. Padberg and Rinaldi (1987) solved the 532-city problem and the term Branch and Cut was first defined in their paper. A breakthrough came in 1991 when they solved a 2392-city problem, the largest problem solved at the time. The success of their work was mainly due to the following factors:

- A combination of procedures, heuristics and exact procedures ((Padberg and Rinaldi, 1990a, 1990b) and (Padberg and Rao, 1982)) to identify the subtour elimination inequalities, 2-matching inequalities, comb inequalities and clique-trees inequalities are used in their algorithm.

- Faster computers, CYBER 205 and IBM 3090/600, are used to handle larger

dimension problems to reduce the running time. Table 2.3 shows the comparison of the running time with different computers.

	Problems	
	1002	2392
CYBER 205	7hours 18mins	27hours 20mins
IBM 3090/600	3 hours 10 mins	2hours 40mins

Table 2.3: Comparison of running for larger dimension problems with different computers

- Two types of LP solvers are used: XMP was used on CYBER 205 and OSL was used on IBM 3090/600. The type of LP solver can significantly affect the running time of the algorithm. Table 2.4 shows the comparison of the time used by different LP solvers.

	Problems	
	1002	2392
CYBER 205	19104	79425
IBM 3090/600	5225	5890

Table 2.4: Comparison of the total CPU time (in second) spent in LP solver

- An efficient heuristic is required to find a good upper bound. The Lin and Kernighan (1973) heuristic was used.
- The best node search is adopted where the node with the lowest objective function value is selected to branch. This allows the keeping of the smaller search tree.
- One way to increase the efficiency of the LP solver is to keep the constraint matrix sparse. To do this reduction procedures are used to reduce the size of the support graph. The support of an inequality is the partial subgraph that is spanned by the edge having nonzero coefficients in the inequality.

Exact methods for subtour elimination and 2-matching constraints are used. Heuristic procedures are used to find comb inequalities and clique tree inequalities.

A group of researchers Applegate, Bixby, Chvátal and Cook solved some of the problems from TSPLIB which include 3038 cities , 4461 cities in 1993, 7397 cities in 1995 and 13509 cities in 1998. The success of solving the 7397-city problem was due to the following factors

- A combination of efficient cut finding techniques were used including a few new ways of finding cuts such as gluing together old constraints into a conglomerate cut, the checking for the consecutive ones properly and a new way of finding comb inequalities based on dominos and necklaces.
- The availability of a network of UNIX workstations.

Recently, the 13509-city problem was solved by Applegate et al. (1998), which is the largest problem solved to date. The main factor for their success is due to the new trick of projecting S and a given point x^* in $\mathbf{R}^n$ into a lower dimensional space. The algorithm was run on a network of 48 workstations including DigitalAlphas, Intel Pentium IIs and Pentium Pros and Sun UltraSparcs.

This concludes the section of exact algorithm. The heuristics or approximate algorithms are discussed in the next section.

2.2 Heuristic Algorithms

One of the drawbacks of exact algorithms for the TSP is that they usually take a long time to solve the problem and thus can only handle smaller size problems. Hence heuristic algorithms, or approximate algorithms, are developed to find the 'near-optimal' solutions for larger dimension problems within a reasonable CPU time.

Heuristic algorithms for the TSP can be divided into three groups:

- **tour construction procedure** - gradually builds up a tour by adding a city at each step;

- **tour improvement procedure** - improves the tour by exchanging the position of the cities such that the cost of the tour is improved;
- **composite algorithm** - combines the above two procedures.

The heuristic algorithms and the major development for TSP using heuristic algorithms are discussed in Tables 2.5 and 2.6.

Author(s)	*Method(s)*	*Comments*
Lin (1965)	3-Opt	Proposed one of the earliest edge exchanging techniques
Lin and Kernighan (1973)	Improvement of 3-Opt by Lin (1965)	Proposed a further improvement of the 3-Opt algorithm by Lin (1965)
Or (1976)	Or-Opt algorithm	Proposed a modification of the 3-Opt procedure
Golden and Stewart (1985)	CCAO (Convex-hull, Cheapest insertion, Angle selection and Or-Opt)	Proposed a composite algorithm
Gendreau et al. (1992)	GENIUS	Proposed a heuristic algorithm and claimed to outperform other heuristics in solution quality and CPU times
Chatterjee et al. (1996)	Genetic algorithms	Proposed a one parent crossover scheme for their GA. Problems from TSPLIB ranging from 100 to 666 cities were tested and the results were within 3.2% of the optimal solution
Sun et al. (1993)	Hierarchical strategy	Proposed that the algorithm was able to produce a rough solution in a short time period
Homaifar et al. (1993)	GA	Proposed a GA with matrix crossover. Problems tested include 25 to 318 cities and yield solution to within 1% of the optimal solution

Table 2.5: Heuristic algorithms for TSP

Author(s)	*Method(s)*	*Comments*
Chardaire et al. (1995)	Simulated annealing	Proposed a heuristic based on the SA approach.
Poon and Carter (1995)	Genetic algorithms	Proposed the tie-breaking crossover and union crossover 2 for their GA. Four problems of sizes from 20 to 30 are tested.
Budinich (1996)	Neural Network	Proposed a NN algorithm and tested with problems with sizes ranging from 10 to 1000. The results showed that it performs better than SA for problems with more than 500 cities.
Somhon et al. (1997)	Neural Network	Proposed an algorithm based on self-organising approach. Problems from TSPLIB ranging from 51 to 1400 were tested and the deviation is to within 2.2% of the best known solutions.
Meeran and Shafie (1997)	Convex hull and local search algorithm	compared the proposed algorithm with three other TSP heuristics and the results indicated that the proposed algorithm performed better.
Nagata and Kobayashi (1997)	Genetic algorithm	Proposed the edge assembly crossover (EAX) for their GA. Problems tested ranging from 101 to 3038 cities from the TSPLIB.
Tsubakitani and Evans (1998a)	Jump search	Indicated in the results that jump search outperforms tabu search and iterative local search on the literature and randomly generated problems

Table 2.5: (Cont.) Heuristic algorithms for TSP

2.2.1 Tour construction procedure

In the 1960s and 1970s various types of tour construction algorithms were developed with the hope of finding near-optimal tours for large size problems. These methods are outlined in the following. For further details of the method, refer to Lawler et al. (1985).

- *Nearest addition procedure* - Insert a city k not yet in the tour between i and j such that the cost c_{kj} is minimum and either i and j or both are in the tour.
- *Nearest insertion procedure* - Similarly as above except k is inserted in the *best* place. This improves Nearest addition procedure by inserting k in the

Author(s)	*Comparison of different heuristics*	*Types of comparison*	*What the author(s) claimed?*
CCAO (Golden & Stewart, 1985)	Four insertion procedures and two post-optimisation procedures	Percentage deviation from lower bound, CPU time and Friedman test	That CCAO outperformed all other algorithms
Gendreau et al. (1992)	GENIUS, GENI, CCAO, Lin (1965), SA, TS, Lin & Kernighan (LK) (1973)	Compared with random generated problems and literature problems from 100 - 500 and 100 - 532 cities using CPU time and solution cost	That GENIUS outperformed other heuristics for TSP in terms of solution quality and computational times
Perttunen (1994)	Random initial solution and Clark-Wright as initial solution and Lin (1965), LK and Or-Opt for post-optimisation procedure	Compared with randomly generated problems of size 100 and 1200 using CPU time and the percentage deviation from the best known solution	That the use of construction procedure improved the performance of the heuristic
Tsubakitani and Evans (1998a)	Jump search (the proposed), TS with 2-Opt and 3-Opt	Compared 7 literature problems from 33 to 105 and randomly generated problems from 50 to 150 cities using CPU time, number of moves made and percentage deviation from the best known solution	That the jump search showed more effective and quicker converging then TS

Table 2.6: Types of comparisons for the heuristic algorithms by different authors

Author(s)	*Comparison of different heuristics*	*Types of comparison*	*What the author(s) claimed?*
Yagiura and Ibaraki (1996)	Genetic DP (the proposed), LK, Or-opt and 3 other local search methods	Compared in CPU time and error rate (%). Problems used are unknown	That the proposed algorithm genetic DP attained better solutions than other heuristics except the Genius algorithm
Lin et al. (1993)	SA, Genetic-Annealing (GA) (the proposed)	Compared using randomly generated problems from 16 to 144 cities and literature problems from 30 to 75 cities by the number of moves made, error rate (%) and CPU time	That the improved SA solved some large scale $\mathcal{NP}$-hard problems
Somhon et al. (1997)	Proposed algorithm, Matsuyama (1992), Guilty net (Burke and Damany, 1992) and Elastic net (Durbin and Willshaw, 1987)	Compared 20 literature problems from 51 to 1400 cities using CPU (single run) and percentage deviation from optimal	That the proposed algorithm obtained a solution to within 1.3% of the optimal compared with other heuristics which can only be obtained to within 3%
Budinich (1996)	Proposed algorithm, Elastic net (Durbin and Willshaw, 1987) and SA (Kirtpatrick et al., 1983)	Compare random problems from 10 to 1000 cities by taking average tour length of 10 runs for the proposed algorithm over bound by SA and percentage deviation from the solution of Durbin and Willshaw	That NN algorithm was competitive with SA algorithm for problems more than 500 cities

Table 2.6:(Cont.) Types of comparisons for heuristic algorithms by different authors

best minimum place instead of inserting it next to j.

- *Cheapest insertion procedure* - Choose a city k such that it gives the smallest cost over all other cities not yet in the tour i.e. $\min c_{ik}$ for all k not in the tour.

- *Farthest insertion procedure 1* - Choose a city k not in the tour and j in the tour such that $c_{kl} = max_k\{min_j c_{kj}\}$ and insert k between i and j such that $c_{ik} + c_{kj} - c_{ij}$ is minimum.

- *Farthest insertion procedure 2* - Choose a city k not in the tour such that it is farthest from any city in the subtour.

- *Arbitrary insertion procedure* (Rosenkrantz, Stearns and Lewis, 1977) - Arbitrary select a city k not yet in the tour and insert it between i and j such that $c_{ik} + c_{kj} - c_{ij}$ is minimum.

- *Nearest neighbour insertion* - Let city j be the last node in the tour, then insert city k to the tour such that city k is closest to city j.

- *Convex hull insertion procedure* (Stewart, 1977) - Choose a city k not in the tour such that $c_{ik} + c_{kj} - c_{ij}$ is minimum and insert k^* between i and j such that $(c_{ik^*} + c_{k^*j})/c_{ij}$ is minimum.

- *Greatest angle insertion procedure* (Norback and Love, 1977, 1979) - Insert k not yet in the tour such that it forms the largest angle with the edges (i, k) and (k, j) where i and j are cities in the tour.

- *Ratio times difference insertion procedure* (Or, 1976) - Insert k between i and j such that $(c_{ik} + c_{kj} - c_{ij})(c_{ik} + c_{kj})/c_{ij}$ is minimum.

Depending on the type of procedure, the complexity of the algorithms are between $O(nlogn)$ and $O(n^3)$. According to Golden and Stewart (1985), these algorithms produced solutions to within 5% to 7% of the optimal and are used as a starting point or initial tour for other tour improvement procedures and composite procedures.

2.2.2 Tour improvement procedure

The earlier work on tour improvement procedures was developed by Croes (1958), Lin (1965) and later improved by Lin and Kernighan (1973) and Or (1976). Recent approaches for the tour improvement procedures include simulated annealing, tabu search, genetic algorithm and neural networks. These methods are discussed in the following. For further details of the work, refer to Lawler et al. (1985) and Johnson and McGeoch (1997). The discussion of tabu search is left to Chapter 4.

The r-Opt algorithm

The r-Opt algorithm is one of the best known edge exchanging procedures which was originally proposed by Croes (1958) for $r = 2$ and later improved by Lin (1965) for $r = 3$ for the symmetric TSP. In a consideration of a feasible tour, the algorithm is to delete r edges and replace them by a second set of edges to form a new tour with reduced cost. The process is repeated until no further improvement can be made and when this happens, the solution is said to be r-optimal. Figure 2.1 shows the 2-opt procedure.

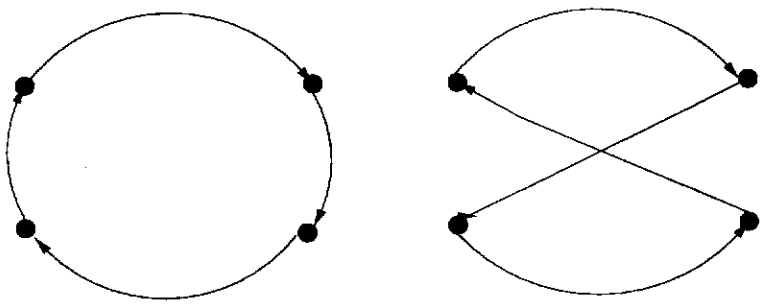


Figure 2.1 : 2-Opt

Lin-Kernighan method

Lin and Kernighan (1973) improved Lin's r-Opt algorithm by considering different values of r throughout the algorithm. The Lin-Kernighan algorithm is more

difficult to code than the Lin's r-Opt algorithm because the r value needs to be determined at each iteration. If r edges are considered, then a series of tests need to be done to decide if $r+1$ edges are needed to be used in the next iteration. This method is reported to outperform 2-Opt and 3-Opt for the TSP. See Johnson and McGeoch (1997) for details.

Or-Opt method

An improved 3-Opt method has been proposed by Or (1976), whose algorithm only considered a small number of exchange edges and works extremely well. The algorithm is referred to as the Or-Opt algorithm. This algorithm is to remove r ($r = 3$, then 2) vertices in the tour and tentatively insert the r vertices between two other cities such that the tour length is reduced.

Simulated Annealing

The idea of Simulated Annealing (SA) was introduced by Metropolis in 1953 in an algorithm to simulate the cooling of material in a heat bath. In this process a solid material is heated until it is melted and then cooled back into a solid state. The structure of the cooled solid depends on the rate of cooling. The algorithm is controlled by the Boltzman distribution

$$f(e) = \frac{1}{Z(T)} exp\left(\frac{-E}{kT}\right) \tag{2.30}$$

where E denotes the energy at temperature T and k refers to the Boltzman constant. A probability of the form

$$p(\Delta E) = exp\left(\frac{-\Delta E}{kT}\right) \tag{2.31}$$

was used to control the behaviour of the system of particles in thermal equilibrium at temperature T. For a given time T, a perturbation mechanism is applied to the current state of the system and transformed to the next state. The resulting energy change is called ΔE. If the energy decreases, then the system moves to a new state. If the energy increases the state is accepted according to probability

(2.31). The process is repeated until the system reaches the frozen state. This approach was applied by Kirkpatrick et al. (1983) thirty years later by relating the physical cooling process of Metropolis to combinatorial optimisation problems. In this case the cost or objective function value corresponds to the energy in the physical system, the feasible solutions correspond to the states of the system, the neighbouring solution corresponds to the change of the state of the system and the heuristic solution corresponds to the frozen state of the system. The objective is to search for the feasible solutions to a problem so that it converges to an optimal solution. Such a method has been applied successfully to areas such as computer design, artificial intelligence and TSP. For further details of Simulated Anealing (SA), refer to Aarts and Korst (1989), Otten and van Ginneken (1989) and Dowsland (1993).

The general outline of the SA algorithm for the TSP can be described as follows. Note that the variation in implementing the algorithm differs in generating the initial solution, updating the temperature, defining the frozen state and the definition of the neighbour.

Step 1. An initial TSP tour τ is obtained. Initialise the temperature $T > 0$. Set $\tau^* = \tau$.

Step 2. Make a random movement from the current tour τ to a new tour τ' (a neighbour of τ) according to a suitable method. Calculate the cost $\Delta = c(\tau') - c(\tau)$.

Step 3. If $\Delta < 0$, set $\tau = \tau'$. If $c(\tau) < c(\tau^*)$ set $\tau^* = \tau$. Otherwise, compute a random number $x \in [0, 1]$.

Step 4. If $x < \exp(-\Delta/T)$ set $\tau = \tau'$.

Step 5. Repeat Steps 2,3 and 4 for a pre-determined number of times.

Step 6. Update temperature T and repeat Steps 2, 3, 4 and 5 for a pre-determined number of times. Stop.

Kirkpatrick et al. (1983) were the first to apply SA to the TSP. However, they

only applied it to small size problems and running times were not reported. Furthermore, Johnson and McGeoch (1997) made a comparison between the SA algorithm, 2-Opt, 3-Opt and Lin-Kernighan algorithm on randomly generated Euclidean problems of sizes of 100, 316 and 1000 cities. The results showed that the SA algorithm produced on average worse tours than 3-Opt and the Lin-Kernighan algorithm and required longer CPU time. One approach for the SA to produce better quality tours is to increase the number of steps of temperature. The disadvantage of this method is that it requires longer computation time than other methods (3.5 days running time for the SA algorithm compared with 0.77 seconds for the Lin-Kernighan algorithm for the 1000-city problem (Johnson and McGeoch, 1997)). So to speed up the algorithm, Nahar et al. (1985) and Golden and Skiscim (1986) (cited in Johnson and McGeoch, 1997) tried to reduce the steps of the temperature (i.e. using fewer temperatures), the results of which indicate that this produces a worse solution. Fortunately there are ways to speed up the SA algorithm by (1) pruning the neighbourhood (that is, by avoiding introducing a city with a long edge which makes the tour worse); and (2) starting with a lower temperature. Combining them has improved the performance of the SA drastically. The improved SA has produced better quality tours than Lin-Kernighan, 2-Opt and 3-Opt but unfortunately still requires longer CPU time as reported by Johnson and McGeoch (1997).

Other authors who have proposed heuristics based on the SA and include TSP as their test problems, include Lin et al. (1993), Lin & Hsueh (1994) and Chardaire et al. (1995). In all cases, they showed that their heuristics produce good results.

One of the attractions of the SA is its simple structure which can easily be combined with other algorithms. In general, the SA produces good quality solutions but requires a long CPU time. For a quality solution and CPU time, one tends to use the Lin-Kernighan method for TSP.

Genetic Algorithms

Genetic algorithms (GA) were invented by John Holland in the early 1970s to simulate some difficult problems involving natural evolution on a computer. The early applications of GA lie in the field of Artificial Intelligence such as game playing and pattern recognition. Recently GA have been applied to combinatorial optimisation problems using the TSP as a benchmark problem.

GA are a search algorithms based on mechanics of natural selection. They work with a population of solutions and attempt to improve the solution using the 'best fit' principle. A solution is called a **chromosome** which consists of binary digits of 1's and 0's . Each chromosome in the population is given a fitness score. A number of the 'fittest' chromosomes are selected to produce the next generation. Two parents from the fittest group are randomly selected to create two children based on the operations which alter the chromosomes. These operations include **crossovers**, **mutations** and **inversions**. Each operation is briefly described in the following and for convenience a graphical explanation based on the concept of TSP is given in Table 2.7.

Crossover is a technique of combining the features of two parents to produce offspring. A *One-point crossover* is a standard technique which occurs when parts of two parents' chromosomes are swapped at a randomly chosen point and produce two children. **Mutation** is a process that occurs after crossover when a single bit of an offspring is flipped. **Inversion** occurs when the order between two chosen elements is inverted and this alters the location of the genes of the chromosomes. These new generated children are then sent to become the new

Type	*Parent(s)*	*Child(ren)*
Crossover	1-2|3-4-5-6	1-2-4-3-6-5
	2-1|4-3-6-5	2-1-3-4-5-6
Mutation	2-4-3-1-5-6	6-4-3-1-5-2
Inversion	2-1-3-4-5-6	2-4-3-1-5-6

Table 2.7: Examples of crossover, mutation and inversion

generation until there are n new chromosomes in the new generation. The new generation, in general, will be fitter than the old generation. The process of generating a new generation is continued and the algorithm terminates when the parents cannot produce any further different children. The following is an outline of the GA algorithm for TSP:

Step 1. Obtain an initial set τ of TSP tours.

Step 2. Combine two parents from τ and produce new tours. Add the new tours to τ.

Step 3. Reduce the set τ according to some rules.

Step 4. Repeat Steps 2 and 3 until the stopping criteria is satisfied. Output the best tour and stop.

The first attempt to apply GA to TSP was done by Goldberg and Lingle (1985) (cited in Reeves, 1993) in which an optimal solution was found for a 33-city problem. Different crossover schemes are used by different authors in their GA namely partially mapped crossover (PMX) by Goldberg and Lingle (1985), cycle crossover (CX) by Oliver et al. (1987), order crossover (OX) by Davis (1985), matrix crossover (MX) by Homaifar et al. (1993) and edge recombination is investigated by Grefenstette et al. (1985) and Whitley et al. (1989). Table 2.8 gives a brief description of the crossover methods used by different authors. A one parent crossover scheme is suggested by Chatterjee et al. (1996) which is discussed in more detail. The edge assembly crossover (EAX) proposed by Nagata and Kobayashi (1997) has optimally solved problems ranging from 101 to 3038 cities using a 200MHz Pentium. The computational time for 3038 cities is approximately 2.5 hours.

Chatterjee et al. (1996) introduced a one parent crossover (*asexual* reproduction) scheme which can be applied to a permutation of n solutions of a TSP. They defined MUT3 as a 3-cut on a tour which cuts the tour in 3 places and rearranging the tour pieces. For example, a MUT3 operation at the third, sixth and eighth locations for a solution is given by 1-2-|3-4-5-6|7-8|9-10 and the tour is arranged

Crossover types	*Author(s)*	*Description of method/examples*
Edge recombination	Whitley et al. (1989)	Consider the edge map for all the cities from parent 1 and 2. Select the element with the fewer edges to create the first element for the offspring. The rest of the places of the offspring are randomly picked so that the edge map has fewer edges.
Cycle crossover	Oliver et al. (1987)	A cycle refers to a common subset of cities in the subtour. The cycle from the two parents is identified and then swapped to produce two offspring.
Matrix crossover	Homaifar et al. (1993)	Present the 2 parents in the matrix form with 1 if there is an edge connected between city i and j. If not the matrix is set to 0. Randomly pick 2 points for crossover and the resulting matrix may have duplication of 1s in some rows and no 1s in others. Remove the 1s from the duplicating row to the row with no 1s. If the resulting matrix contains a cycle, then some edges are deleted and added such that a legal tour is formed.
Tie-breaking crossover	Poon and Carter (1995)	The Tie-Breaking Crossover #1 can be described as follows. 2 parents and two random points are chosen for crossover (this results in ties in the offspring). A string of size n's called a Crossover Map is generated using integers between 0, 1, ..., $n-1$. Each of the element in the offspring is then multiplied by n and the corresponding number is added in the Crossover Map. The lowest element in the string is replaced by 1, the next lowest element by 2 and so on. This gives the new offspring. We refer to the Tie-Breaking Crossover #2 to the authors.

Table 2.8: Various crossover methods for the TSP

as 1-2-7-8-3-4-5-6-9-10. Rank is used to represent the legal tour and inversion and mutation work directly with the rank. For example, given a solution S, 6-8-7-4-2-9-1-5-10-3, the inversion of S^I between third and sixth position is given 6-8-9-2-4-7-1-5-10-3. The mutation of S^M between third and eighth position is given by 6-8-5-4-2-9-1-7-10-3. They pointed out that the operations of inversion, MUT3 and mutation represent the cutting of the tour at 2, 3 and 4 places which can be generalized as a r-cut operation ($r = 2, 3, 4, ...$) and is similar to the r-Opt by Lin (1965). The number of cuts is decided by a value p between 0.35 and 0.45. Six problems of sizes ranging from 100 to 666 cities were tested and the

Crossover types	*Author(s)*	*Description of method/examples*
Order crossover	Davis (1985)	Two cut points are randomly generated. A section between the cut points of parent 1 is copied to the offspring. The rest of the places of the offspring are filled with other elements that have not yet occurred.
Partially mapped crossover	Goldberg and Lingle (1985) (cited in Reeves, 1993)	Two cut points are generated and the cut out section represents the swapping of elements between parents 1 and 2 to generate the offspring. The rest of the places of the offspring are filled with an element or elements from parent 2.
Edge assembly crossover	Nagata and Kobayashi (1997)	Two parents A and B are chosen. A graph G' consisting of the edges of both parents A and B is generated. Several cycles called *AB-cycles* are generated by tracing the edges of parent A and B on G'. A set of subtours is generated from *AB-cycles* by choosing the effective edges and applying them to parent A. The set is called the *Exchangeable Edge Sets (E-sets)*. The immediate individual is obtained from *E-sets* and applied to parent A. This results in several subtours and is then modified to form a valid tour.

Table 2.8: (Cont.) Various crossover methods for the TSP

solutions were found to within 3.5% of the optimal solution. Both the Euclidean and non-Euclidean randomly generated problems were tested on the proposed algorithm.

Neural Networks

Neural Networks (NN) is inspired by the functioning of the human brain. A set of neurons are connected by a certain network. A neuron receives an input, computes the output and sends signals to other neurons. The application of NN is used mostly in the area of pattern recognition. For details of NN and its applications refer to Peterson and Söderberg (1993).

In general, Neural Networks algorithms for the TSP can be divided into two classes. The neurons in the first class are organised in the integer programming formulation and in the second class, the neurons are viewed as points in space which move towards the position of the cities.

The first application of NN for the TSP was proposed by Hopfield and Tank (1985) (cited in Johnson and McGeoch, 1997) based on the integer programming formulation as follows.

Minimise

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}(x_{in}x_{j1} + \sum_{k=1}^{n-1} x_{ik}x_{j(k+1)}) \tag{2.32}$$

subject to

$$\sum_{i=1}^{n} x_{ik} = 1, \ \ 1 \leq k \leq n, \tag{2.33}$$

$$\sum_{k=1}^{n} x_{ik} = 1, \ \ 1 \leq i \leq n, \tag{2.34}$$

$$x_{ik} \in \{0,1\}, \ \ 1 \leq i, \ \ k \leq n. \tag{2.35}$$

In this formulation when $x_{ik} = 1$ in a solution then it corresponds to the situation that city i is in in the k position of the tour. Constraint (2.33) describes that each position in the tour contains exactly one city and constraint (2.34) describes that each city is in one position of the tour. Each x_{ij} is viewed as a neuron which is connected in a network such that the constraints are satisfied and the cost is minimum. The algorithm, did not perform well and failed to converge to a feasible solution for the 30-city problem. Improvements were made by others and larger sized problems were tried, however the results were not impressive. Johnson and McGeoch (1997) pointed out that the results of a single run of 3-Opt performed better in solution quality and time than the above approaches.

The second class of NN for the TSP is quicker and can handle larger size problems. The algorithm starts with a ring or a cycle of m neurons at the centre of all the cities. At each iteration, a city is chosen at random and a neuron which is closest to the city is declared as the winner. According to predetermined rules, the winner neuron and neighbours which lie "close" by are moved to a position closer to the city. The algorithm is continued until every city is in the tour. There are two variants of this class; the **elastic net** (Durbin and Willshaw, 1987) and **self-organising map** (Kohonen (1988), Budinich (1996), Somhon et al. (1997)). The elastic net approach is able to handle 1000-city problems but

the comparison of the results between the elastic net, 2-Opt and Lin-Kernighan showed that 2-Opt and Lin-Kernighan give faster and better solutions (Johnson and McGeoch, 1997). The self-organising map is able to handle large instances which include a 2392-city problem, solved to 10% above the optimal by Fritzke and Wilke (1991) (cited in Johnson and McGeoch, 1997). Other results include a 30000-city problem which is solved to within 7% of the average of the Held and Karp lower bound by Amin (1994) (cited in Johnson and McGeoch, 1997) and a 1400-city problem which requires 11,493 seconds CPU time by Somhon et al. (1997). These results not only take longer CPU time but also produce less optimal solutions than the result by 2-Opt or 3-Opt.

This concludes the heuristic algorithms. In the following section, the composite algorithms are discussed.

2.2.3 Composite algorithms and other heuristic algorithms

Composite algorithms refer to the combination of the tour construction procedures and the tour improvement procedures. Several efficient composite algorithms have been developed and they include the CCAO algorithm by Golden and Stewart (1985), GENIUS algorithm by Gendreau et al. (1990), Jump search by Tsubakitani and Evans (1998a) and an algorithm by Meeran and Sharfie (1997). The algorithm by Sun et al. (1993) based on a hierarchical strategy will be examined. All the heuristics are designed for the symmetrical Euclidean TSP. Refer to Chapter 6 for the GENIUS algorithm and the rest of the algorithms are discussed in the following.

The CCAO algorithm (Golden and Stewart, 1985)

The CCAO heuristic refers to Convex hull, Cheapest insertion, Angle selection and Or-Opt method. It starts by defining a partial tour which forms a convex hull of vertices. For each vertex not in the partial tour, say k, two adjacent cities i and j are identified such that the cost calculated by $c_{ik} + c_{kj} - c_{ij}$ is the minimum.

Select a vertex k such that the angle between edges (i, k) and (k, j) is as large as possible and insert k between i and j. The cheapest insertion and angle selection are repeated until a Hamiltonian tour is obtained. The Or-Opt method is then applied to improve the tour.

Jump Search (Tsubakitani and Evans, 1998a)

The Jump Search is a powerful and yet simple metaheuristic for the TSP proposed by Tsubakitani and Evans (1998a). The proposed algorithm is shown to be more effective than the tabu search on the benchmark and the randomly generated problems. The idea of Jump Search comes from the fact that there are some good solution regions and there are some poor solution regions. If there was a search procedure which could jump from a good plateau to a better one, then better solution regions could be searched for more efficiently. The algorithm starts by generating a set of jump destinations (starting points) whose neighbourhoods are disjointed. A local search heuristic is applied to the best jump destination. If a better solution (local optimum) is found, then a local search is applied to the second best jump destination. The process is continued until the iteration time is reached or all jump destinations are used. Six different tour constructions heuristics are used to generate a set of diversify jump destinations. Two local search heuristics i.e. 2-Opt and 3-Opt are used to compare the proposed algorithm with the tabu search and the iterative local search.

Seven literature problems of size ranging from 33 to 105 cities and three randomly generated test problems of size ranging from 50 to 150 cities are tested. The results indicate that the jump search outperforms the tabu search and iterative local search for both sets of test problems. In particular, the jump search with 3-Opt is found to perform well on the literature problems. Five out of seven optimal solutions were found. When directly comparing the jump search and the tabu search, the results show that the jump search is able to find better solutions more quickly. The reason is that the jump search is able to make more moves than the tabu search in a given time period. The tabu search requires time to check

if the candidate move is tabu. Tsubakitani and Evans found that the number of moves the tabu search made is less than half of that of the jump search.

The Meeran and Shafie Algorithm (1997)

Meeran and Shafie (1997) proposed an algorithm for optimum path planning based on the convex hull method and the local search heuristic. The term "optimum path planning" is used in areas such as machine layout, motion planning and mechatronics. In the computing/mathematics literature, optimum path planning is traditionally addressed as a "Travelling Salesman Problem". This algorithm uses the Graham scan algorithm to find an initial sub-tour for a given set of points. A local search heuristic is then applied successively until all the points are in the tour. The complexity of the algorithm is $O(n^2)$. The algorithm is compared with simulated annealing, Hopfield networks and the TSPotts algorithm and the results indicate that the Meeran and Shafie's algorithm is better than the others.

The following is the description of Meeran and Shafie's algorithm:

The algorithm starts by creating a convex hull i.e. a subtour of a given set of points by using the Graham Scan algorithm. A set of neighbourhoods are created by clustering groups of points within circles. The circle is created by the edge of the convex hull. For example, let AB be the edge of a convex hull boundary, then the circle is created using the diameter of AB. All the points which lie inside the circle AB are therefore regarded as the neighbourhood. For those points which lie outside the circle or within the "intersection" of two circles, the cost of inserting the point to neighbourhood A and B is calculated such that the point is inserted into the neighbourhood with the lowest insertion cost. The neighbourhood optimisation heuristic is then applied to each neighbourhood. A mid point of the edge x is calculated and the points in the neighbourhood are sorted according to the mid point x. The first three sorted points are examined by the triangle rule where the longest edge of the triangle is deleted and the shortest edge is accepted into the tour. The process is repeated for all sorted points. The

intersecting edges are deleted and reconnected by the non-intersecting edges. The neighbourhood optimisation step is used for all the neighbourhoods. In the end a neighbourhood linking procedure is called to link all the "sub-tour" in each neighbourhood and hence give an optimised tour.

Hierarchical Strategy for the TSP

The algorithm proposed by Sun et al. (1993) is based on a hierarchical strategy which is widely used in the areas of social science, information processing, administration and management. The algorithm is very efficient and it takes 15 to 25 seconds to get a solution for 15,000 cities on a Hewlett Packard 700 work station. However the solution is not close to the global optimum and no systematic way is known to refine the solution. The algorithm was tested on two sets of data: four different groups of randomly and uniformly distributed data with sizes ranging from 1,000 to 16,385 cities and 25 literature problems from TSPLIB with problems' sizes ranging from 70 to 3,038 cities. The solutions for the four groups of data were obtained within 29 seconds computing time and the solutions for the literature problems are within 3% to 25% of the optimal solutions. The CPU time for obtaining the solutions for the literature problems were not given.

A detailed description of the algorithm is given below.

Step 1. Divide the area into four districts and hence all cities are separated into four groups. The position of the district is calculated by taking the average position of the corresponding group of cities. The shortest route between the four districts is obtained exactly.

Step 2. Each district is subdivided into 4 sub-districts and the average position of the sub-districts is calculated according to Step 1.

Step 3. The tour is modified by calculating the shortest path between the sub-districts and joining the tour with three other districts.

Step 4. Repeat Step 2 and 3 until there is only one city inside each of the sub-districts.

This concludes the overview of the exact and heuristics algorithms for the TSP. In the next Chapter, an overview of the exact and heuristics algorithms to solve the VRP is discussed.

Chapter 3

An Overview of The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) can be described as the delivery or collection of goods from one or several depots to a set of customers or cities so that the travel distance or cost is minimised. Usually, some side constraints are imposed such as restrictions of the vehicle capacity and/or length of travel, the period of time the customers has to be serviced etc. Practical applications arise in the physical distribution of goods for example delivery of fuel, newspapers, grocery, pet foods to customers and etc. Furthermore, considerable financial resources are spent on distribution, for example, Bodin et al. (1983) report that the annual distribution cost is approximately $400 billion in the US and £15 billion in the UK. Hence numerous distribution costs can be saved by properly studying and implementing an efficient model. Recently, a Swiss company reduced its distribution costs by 10 - 15% by modelling the problem to a VRP model and using an appropriate heuristic algorithm to solve the model (Semet and Taillard, 1993).

The VRP was first introduced by Gavin et al. (1957) forty years ago to distribute gasoline to service stations using a fleet of vehicles of various capacities. Extensive studies have been conducted on this problem and various algorithms including the exact algorithms and heuristics algorithms have been proposed to

solve the problem in order to reduce the distribution costs. The VRP can be viewed as an extension of the TSP with more than one vehicle and with additional side constraints. The VRP is much more difficult to solve with the same number of customers or cities than for the TSP. The largest problem solved for the capacity restricted VRP (CVRP) is the 134 customers problem using exact methods (Augerat et al., 1995 and Hill, 1995). The literature problems with up to 200 customers for both the CVRP and capacity and distance restricted VRP (CDVRP) are tackled using heuristics.

There are various names used in the literature which refer to the VRP. These include vehicle scheduling, truck dispatching and delivery problems (cited in Christofides, 1985). For different aspects of the VRP, refer to Bodin and Golden (1981), Bodin et al. (1983) and the book by Golden and Assad (1988).

In the following two sections, an overview of the algorithms developed in the literature to solve the VRP are discussed. Section 3.1 and Section 3.2 discuss exact algorithms and heuristic or approximate algorithms respectively. Note that this thesis focuses on the symmetric problems only.

A simple symmetric VRP can be formulated as follows (Laporte et al., 1985).

Minimise

$$\sum_{i \in V} \sum_{i<j} c_{ij} x_{ij} \tag{3.1}$$

subject to

$$\sum_{j \in V} x_{0j} = 2m \tag{3.2}$$

$$\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 2, \;\; i \in V \tag{3.3}$$

$$\sum_{i,j \in S} x_{ij} \leq \mid S \mid - l(S), \;\; S \subset V, \;\; 3 \leq \mid S \mid n - 2 \tag{3.4}$$

$$x_{ij} = \begin{cases} 1 & \text{if a vehicle travels between } i \text{ to } j; \\ 2 & \text{if a vehicle makes a single trip from } i = 0 \text{ to } j; \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

$$m \geq 1 \text{ and integer} \tag{3.6}$$

where m represents the number of vehicles used in the solution and can be either fixed or free. For $S \subseteq V$, $l(S)$ represents a lower bound on the number of vehicles required to serve the customers of S in an optimum solution. Constraints (3.2) and (3.3) specify the degree of the depot and the customer locations. Constraints (3.4) are called the subtour elimination constraints which prevent the formation of subtours disconnected from the depot and subtours connected to the depot which do not satisfy the capacity, or other restrictions.

3.1 Exact Algorithms

The approaches used to solve the VRP exactly are extensions of the algorithms from the TSP. It can be divided into three categories, namely direct tree search, integer linear programming and dynamic programming. For a survey of the exact methods for the VRP, refer to Christofides (1985), Laporte and Nobert (1987) and Laporte (1992). The integer linear programming can be classified into three sections: set-partitioning formulation, vehicle flow formulations and commodity flow formulations (Laporte and Nobert, 1987). The formulations can be solved by the branch and bound or the branch and cut methods. However, the problem is difficult. By 1985, problems with up to 60 customers were solved. Larger size problems have been solved recently due to the success of the polyhedral theory and the improvement in computer power. In particular, Augerat et al. (1995) and Hill (1995) solved the 134-customer problem for the CVRP.

Table 3.1 present the various formulations and methods used in the literature to solve the CVRP and/or CDVRP. Section 3.1.1 discusses the algorithm by Fisher (1994) based on the minimum K-trees which solves the 100-customer benchmark to optimal. The polyhedral approach for the VRP is discussed in Section 3.1.2. This includes the discussion of the algorithm by Laporte et al. (1985), the improvements by Achuthan et al. (1996a, 1996b, 1996c and 1998) and Augerat et al. (1995).

Author/(s)	*Method(s)*	*Comments/Results*
Christofides & Eilon (1969)	Direct tree search methods	Proposed algorithm solved two problems with 6 and 13 customers
Christofides, Mingozzi & Toth (1981a)	Tree search algorithm based on minimum k-degree center tree and q routes	Proposed algorithm solved 6 out of 10 problems with sizes ranging from 10 to 25 customers
Christofides, Mingozzi & Toth (1981b)	State-space relaxation	Proposed algorithm solved 10 problems with customers from 10 to 25
Laporte et al. (1984)	Integer linear programming based on branch and bound method and cutting plane method	Proposed two exact algorithms to solve DVRP on Euclidean and non-Euclidean problems with sizes ranging from 20 to 60 customers
Laporte, Nobert and Desrochers (1985)	Integer linear programming	Solved 83 out of 84 randomly generated problems with sizes ranging from 15 to 50. Proposed a branch and cut algorithm for the CVRP. Problems include the euclidean and non-euclidean with size 15-50 customers and 15-60 customers.
Gavish and Srikanth (1986)	Integer linear programming	Tested the large sized problems up to 500 cities
Agarwal, Mathur and Salkin (1989)	Set Partitioning formulation	Tested seven problems from Christofides et al. (1981a) with sizes ranging from 15 to 25. Their algorithm was 13 times faster than Christofides et al.

Table 3.1: Exact algorithms for the VRP.

3.1.1 The Minimum K-Tree Approach

Fisher (1994) defined a K-tree to be a set of $n+k$ edges that span the graph G with $n+1$ nodes. The VRP is modelled in such a way as to find the minimum cost K-tree with degree $2K$ on the depot subject to: the capacity restrictions and that each node is visited only once.

Let x be a set of unordered pairs of edges between each node $i, i \in V$ and between the depot '0'. Define X to be a set of x which defines a K-tree satisfying $\sum_{i=1}^{n} x_{0i} = 2K$. So the formulation is as follows:

Author/(s)	*Method(s)*	*Comments/Results*
Achuthan and Caccetta (1991)	Mixed integer linear programming	Provided an MILP formulation for the capacity and distance VRP which overcame the error by Kulkarni and Bhave (1985)
Cornuejols and Harche (1993)	Graphical vehicle routing problem (GVRP)	Demonstrated the use of other facet inequalities such as subtour elimination and comb inequalities to solve CVRP
Araque et al. (1994)	Path-Partitioning formulation	Discussed various cutting planes. Problems tested included 21-60 customers.
Fisher (1994)	minimum k-trees	Found optimal or improved solutions of the 7 out of 12 problems tested with customers ranging from 25 to 199. Solved 100 customer benchmark problem
Mingozzi, Christofides & Hadjiconstantinou (1994)	Set partitioning formulation	Tested problems for CVRP with sizes ranging from 21 to 75
Augerat et al. (1995)	Vehicle flow formulation and branch and cut method	Solved 9 out of 13 problems ranging from 22 to 134 customers and found optimal solution to the 134 customers problem
Achuthan, Caccetta & Hill (1996a)	Branch and cut method	Proposed a new subtour elimination constraint to solve CVRP. A total of 1590 randomly generated problems were tested with customers ranging from 15 to 100 and 10 literature problems of sizes from 10 to 25 customers

Table 3.1: (Cont.) Exact algorithms for the VRP

$$\min_{x \in X} \sum_{i,j \in V \cup \{0\}} c_{ij} x_{ij} \tag{3.7}$$

subject to

$$\sum_{j \in V \cup \{0\},\ j \neq i} x_{ij} = 2, \ \forall i \in V \tag{3.8}$$

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 2l(S), \ \forall S \subset V \text{and} \mid S \mid \geq 2 \tag{3.9}$$

where $l(S) = \left\lceil \frac{q(S)}{Q} \right\rceil$ is a lower bound on the minimum number of vehicles

required to service S, $q(S) = \sum_{i \in S} q_i$ and $\bar{S} = V \cup \{0\} - S$. Let $u_i, i \in V$ and $v_S \geq 0$ for $S \subseteq V, \mid S \mid \geq 2$ be the lagrange multipliers for (3.8) and (3.9). So the Lagrange relaxation of (3.7) - (3.9) can be defined as

$$L(u,v) = \min_{x \in X} \sum_{i,j \in V \cup \{0\}} \bar{c}_{ij} x_{ij} + 2 \sum_{i=1}^{n} u_i + 2 \sum_{S \subseteq V} v_S l(S) \tag{3.10}$$

where $u_0 = 0$ and $\bar{c}_{ij} = c_{ij} - u_i - u_j - \sum_{(i \in S, j \in \overline{S}) \text{or} (i \in \overline{S}, j \in S)} v_S$.

The capacity constraints can be tightened as follows:

For any $S \subset V$, let

$$S' = \left\{ j \in \overline{S} \mid j \geq 1 \text{ and } q_j > l(S) - \sum_{j \in S} q_j \right\} \tag{3.11}$$

$$e_j = \begin{cases} 0, & j \in S \\ 0, & j \in S' \text{ and } \mid S' \mid \leq 2 \\ \frac{l(S)}{l(S)+1}, & j \in S' \text{ and } \mid S' \mid > 2 \\ 1, & j \in \overline{S} - S' \end{cases} \tag{3.12}$$

So the tightened capacity constraints are

$$\sum_{j=0}^{n} e_j \sum_{i \in S} x_{ij} \geq 2l(S) \quad \text{for all } S \subset V, \mid S \mid \geq 2. \tag{3.13}$$

The Lagrangian relaxation is solved by the subgradient method. Heuristics are developed to identify violating capacity constraints. At each subgradient iteration three heuristics are used to obtain feasible solutions.

This algorithm was tested on six benchmark problems with sizes ranging from 50 to 199 customers. These problems were taken from Christofides and Eilon (1969) and Christofides et al. (1979). The algorithm was also tested on six "real problems" with sizes ranging from 25 to 134 customers. The optimal solution was found for the 100-customer benchmark taken from Christofides et al. (1979).

3.1.2 Polyhedral Approach For VRP

The success of the polyhedral approach to solving the TSP has inspired investigation to apply the theory to the VRP. Study of the facet inequalities is carried out by Laporte et al. (1985) and Achuthan et al. (1996c). Further improvements are given in Achuthan et al. (1996a, 1996b). Furthermore, Cornuejols and Harche (1993) defined the comb inequalities for the CVRP and Araque et al. (1994) explored the polyhedral structure for the unit demand VRP and generated strong cut based on multistars, partial multistars and subtour elimination constraints. Other classes of valid inequalities such as comb and extended comb inequalities, generalised capacity inequalities and hypotour inequalities have been investigated by Augerat et al. (1995). Their algorithm has solved some literature problems including the 134-customer problem, which is the largest CVRP solved optimally in the literature. The problem is also solved by Hill (1995).

Cornuejols and Harche (1993)

Cornuejols and Harche (1993) followed the approach used in the study of graphical TSP. This approach aimed to find facets for the graphical relaxation of TSP which lead to facets for the TSP (Naddef and Rinaldi, 1991). The graphical vehicle routing problem (GVRP) is a relaxation of CVRP which is concerned with constructing k tours such that each customer must be in at least one of the routes and that the capacity restrictions are satisfied.

The comb inequalities for the STSP have been proved by Cornuejols and Harche (1993) to be valid facets for the CVRP and are defined in the following:

For a complete graph G with $k \geq 2$, let $W_0, W_1, ..., W_s \subseteq V$ satisfy

- $| W_i \setminus W_0 | \geq 1, \quad i = 1, ..., s$
- $| W_i \cap W_0 | \geq 1$
- $| W_i \cap W_j | = 0, \quad 1 \leq i < j \leq s$

- $s \geq 3$ and odd

The comb inequality is

$$\sum_{a=0}^{s} \sum_{i,j \in W_a} x_{ij} \leq \sum_{a=0}^{s} \mid W_a \mid - \frac{3s+1}{2} + \alpha(k-1) \tag{3.14}$$

where

$$\alpha = \begin{cases} 0, & \text{if } 0 \notin \bigcup_{a=0}^{s} W_a, \\ 1, & \text{if } 0 \in W_0 \setminus \bigcup_{a=1}^{s} W_a \text{ or } 0 \in W_b \setminus W_0 \text{ for some } b = 1, ..., s, \\ 2, & \text{if } 0 \in W_b \cap W_0 \text{ for some } b = 1, ..., s. \end{cases} \tag{3.15}$$

In addition to the above conditions, if $0 \in W_1 \setminus W_0$, the comb inequality can be strengthened as follows:

$$\sum_{a=0}^{s} \sum_{i,j \in W_a} x_{ij} \leq \sum_{a=0}^{s} \mid W_a \mid - \frac{3s+1}{2} + k - R(V \setminus W_1) \tag{3.16}$$

where $R(S)$ is defined to be the smallest integer t such that $S_1, ..., S_t, ..., S_k$ is a partition of $\{V \setminus 0\}$ satisfying $\sum_{i \in S_\alpha} q_i \leq Q, 1 \leq \alpha \leq k$ and $S \subseteq \bigcup_{\alpha=1}^{k} S_\alpha$.

The subtour elimination constraints and comb inequalities are used to solve the four examples. First the relaxed problem is solved and then the LP is strengthened by adding the facets inequalities which are generated by hand and added to the LP. This process is repeated. Three 18-customer problems and the 50-customer benchmark problem were solved.

The Laporte, Nobert and Desrochers Algorithm (1985)

In the paper by Laporte et al. (1985), the branch and bound methods based on the integer linear programming for the CVRP and CDVRP are proposed. This work is an extension of the initial work by Laporte and Nobert (1983) and Laporte et al. (1984).

Recollect that the formulations (3.1) - (3.6) are suitable for both the CVRP and CDVRP where $l(S), S \subseteq V$ is suitably defined. The subtour elimination constraint (3.4) is a direct generalisation of the corresponding constraint $\sum_{i,j \in S} x_{ij} \leq | S | -1$ for the TSP. Hence the number of constraints is of the order of 2^n. Furthermore, the subtour elimination constraints of the CVRP and CDVRP include a further difficulty of estimating an appropriate value for $l(S), S \subseteq V$.

In the branch and bound method developed by Laporte et al. (1985), the subproblem associated with a node of the search tree can be defined as follows:

- The reduced problem is obtained from (3.1) - (3.3), (3.5), (3.6) and some additional subtour elimination constraints of type (3.4). Some variables may also be fixed.

- Using

$$l(S) = \begin{cases} \lceil \frac{1}{Q} \sum_{i \in S} q_i \rceil & \text{for CVRP,} \\ \max\{\frac{1}{Q} \sum_{i \in S} q_i, \frac{1}{L} \sum_{i,j \in S} c_{ij} x_{ij}\} & \text{for CDVRP.} \end{cases} \tag{3.17}$$

A relaxation of the subproblem is obtained by relaxing the integer restrictions of (3.5). Whenever this relaxation does not produce an optimal solution to the subproblem, a violated subtour elimination constraint from the optimal solution of the relaxed problem needs to be identified. In the case of the TSP, there is an efficient method of recognising the subtour elimination constraint. But in the case of the VRP, there is no efficient method for recognising a violation of (3.4) (Augerat et al., 1995). Laporte et al. (1985) proposed a simple search heuristic to find the violated constraint of type (3.4). However, Achuthan et al. (1996b) give a counter-example for the CDVRP when using $l(S)$ defined by (3.17). The estimate for $l(S)$ may eliminate an optimal solution for the CDVRP because the distance bound is incorrect.

This algorithm was tested on both the Euclidean and non-Euclidean randomly generated problems with sizes ranging from 15 to 50 cities for the Euclidean case and 15 to 60 cities for the non-Euclidean case.

Achuthan, Caccetta and Hill (1996a)

The work of Achuthan et al. (1996a) is an improvement of Laporte et al. (1985). They proposed another subtour elimination constraint as follows:

$$\sum_{i,j\in S, i<j} x_{ij} + \sum_{i\in S} x_{0i} - m \leq \mid S \mid - l(\overline{S}) \tag{3.18}$$

for $S \subseteq V$ with $1 \leq \mid S \mid \leq n-2$.

Achuthan et al. (1996c) used both (3.4) and (3.18) in their algorithm to search for violating constraints. A comparison of their result with that of Laporte et al. (1985) on a range of randomly generated problems, showed that Achuthan et al. (1996c) algorithm produced better lower bound and used less branches. Further improvements were made by Achuthan et al. (1996a) by developing three new cutting planes which were useful in eliminating infeasible or non-optimal solutions in their branch and cut algorithm to solve the CVRP. The new algorithm was tested on the randomly generated problems from 15 to 100 customers. A comparison between their new algorithm with the algorithm developed by Laporte et al. (1985) and their earlier algorithm showed that the new algorithm not only produced a smaller search tree and superior bounds but also required less CPU time.

Achuthan et al. (1996b) also applied the above algorithm, and proposed a procedure for generating a good lower bound on the number of vehicles needed to service a subset of customers for the CDVRP. This method is described in the following:

For a subset $S \subset V$, at node k of the search tree, let $F(k, v, S)$ be the set of feasible solution of CDVRP at node k obtained by (3.2) - (3.4) and other cutting planes in the LP by restricting the constraints in node k to $S \cup \{0\}$ and set $\sum_{i,j\in S} x_{ij} = 2v$ where v is a specified positive number. Some variables in the LP maybe fixed. At the node k, three problems can be defined. For further details refer to their paper.

A lower bound for the number of vehicles can be described in the following:

Define

$$a(S) = \left\lceil \frac{\sum_{j \in S} q_j}{Q} \right\rceil, \tag{3.19}$$

and let

$$b_k(S) = \min \left[\left\lceil \frac{H_k(v,S)}{L} \right\rceil : a(S) \leq v \leq \mid S \mid \right] \tag{3.20}$$

where $H_k(v,S)$ is a lower bound on the distance travelled between customers in S and the depot. Then the lower bound $l(S)$ of node k is defined as $l_k(S) = \max [a(S), b_k(S)]$. Note that the lower bound $l(S)$ is obtained by solving a second LP obtained from the LP subproblem by modifying the objective function.

A total of 4590 problems were tested which included the CDVRP Euclidean problem with sizes ranging from 15 to 50 customers and non-Euclidean problems with sizes ranging from 15 to 60 customers. Three out of eight literature problems were solved but some large size problems ($n > 100$) could not be completed within a 1-hour limit.

Achuthan et al. (1998) proposed eight cutting planes for the CVRP. They also used a Bin Packing lower bound proposed by Martello and Toth (1990) for the values of $l(S)$ in the subtour elimination constraints. Tighter lower bounds were produced compared with their earlier work and the results by Fisher (1994). However, when comparing the lower bound to the work by Augerat et al. (1995), Achuthan et al. (1998) produced the lower bound within 1.57% of the best known upper bound on average whereas Augerat et al. (1995) produced lower bounds to within 1.23% of the best known upper bound on average. However, Achuthan et al. (1998) used less total CPU time to generate the initial lower bound by a factor of 5.76 times compared with Augerat et al. (1995).

Augerat, Belenguer, Benavent, Corberan, Naddef and Rinaldi (1995)

Augerat et al. (1995) proposed a branch and cut algorithm for the CVRP based on procedures which extensively search for the violations of the subtour elimination constraints, the use of the comb inequalities, the Bin Packing lower bound and

the Hypotour inequalities. This algorithm was tested on 13 problems with sizes ranging from 22 to 134 cities. Nine problems were solved optimally including the 134 cities problem.

This concludes the review for the exact algorithms for the VRP. In the next section we will discuss the heuristic algorithms for the VRP.

3.2 Heuristic Algorithms

Heuristic algorithms developed for the VRP can be classified into two classes: the classic and modern heuristics. The classic heuristics involve construction and improvement of the routes. Some of these heuristics include the saving method by Clarke and Wright (1964), the Sweep algorithm by Gillet and Miller (1974) and the two-phase algorithm developed by Christofides et al. (1979). A recently developed heuristic along the lines of the classic ones is the petal heuristic by Renaud et al. (1996). These heuristics will be discussed in Section 3.2.1. The modern heuristics include the genetic algorithm (GA), simulated annealing (SA), neural networks (NN) and tabu search (TS). In particular, heuristic algorithms based on TS have produced good quality results. Parallel implementation of the TS heuristic as shown by Taillard (1993) not only reduced the CPU time but was also able to produce results for randomly generated problems up to 1024 customers in 123.7 seconds. The SA, NN and GA will be discussed in Section 3.2.2. The TS algorithms for the VRP are discussed in Chapter 4. Table 3.2 shows a list of major developments of the classic heuristics for the CVRP and/or CDVRP. This thesis focuses on the VRP with capacity restriction (CVRP) and capacity and distance restrictions (CDVRP) only. For additional details on this subject, refer to Gendreau et al. (1997) and Laporte et al. (1999).

Author/(s)	*Method(s)*	*Comments/Results*
Clarke and Wright (1964)	Saving method	Proposed the earliest heuristic to solve VRP
Gillet and Miller (1974)	Sweep algorithm	Tested the proposed algorithm on problems with 21 to 250 customers
Christofides, Mingozzi and Toth (1979)	Two phase method	Proposed a competitive algorithm which tested on the 14 benchmark problems for CVRP and CDVRP with customers ranging from 50 to 199
Paessens (1988)	Saving algorithm	Improved the saving method by using less CPU time and memory
Bachem et al. (1996)	Simulated trading heuristics	Proposed a parallel implemented iterative search algorithm which is competitive with the Tabu search for CVRP, CDVRP and VRP with time windows
Renaud et al. (1996)	Petal heuristics	Proposed a competitive heuristic with less CPU time which provided the solutions to within 2.38% of the best known solutions with customers from 50 to 199
Breedam (1995)	SA	Proposed a SA algorithm with three types of improved edge exchange method. Fourteen CVRP and CDVRP were tested
El Ghaziri (1991)	NN	Proposed the first NN algorithm for CVRP
Matsuyama (1991)	NN	Solved the VRP of size 532 customers derived from TSP by adding a capacity constraint
Alfa, Heragu and Chen (1991) (cited in Gendreau et al. (1997))	SA	Proposed a SA algorithm and tested the algorithm on 3 problems ($n = 30, 50, 75$). The results were poor
Torki, Somhon and Enkawa (1997) (cited in Somhon et al. (1997))	NN	Proposed a self-organizing NN algorithm for VRP which obtained solutions to within a few percent of optimal
Bramel & Simchi-Levi (1995)	Location based heuristic	Proposed an algorithm based on the capacitated concentrator location problem (CCLP) and extended the idea to the CVRP. Tested the algorithm on 7 literature problems and the results showed that the proposed heuristic is competitive with the other heuristic algorithms
Osman (1993)	SA	Found 13 out of 26 better solutions for the literature and randomly generated problems with customers ranging from 29 to 199

Table 3.2: Major development of the heuristics for the CVRP and/or CDVRP.

3.2.1 Classic heuristic algorithms for VRP

In the following we briefly discuss a few of the earliest heuristic algorithms proposed for the VRP. In addition, a recent algorithm, petal heuristic, an improvement procedures are discussed.

Saving Method (Clarke and Wright, 1964)

The saving method developed by Clarke and Wright (1964) is one of the earliest heuristics for solving the VRP. The algorithm can be described as follows:

- Create n vehicle routes as $(0, i, 0)$, $i = 1, ..., n$.
- Calculate the saving $s_{ij} = c_{0i} + c_{0j} - c_{ij}$, $i, j = 1, ..., n$ where s_{ij} is the saving in cost which produces the route $(0, i, j, 0)$ instead of $(0, i, 0)$ and $(0, j, 0)$.
- Order the saving in a list in descending order. Two routes are merged if the resulting route is feasible.
- Repeat this until no more routes are in the list.

Others who have proposed a variation of this method include Tillman and Cochran (1968), Gaskell (1967) and Yellow (1970). The improved algorithm which reduced the computational time was shown by Golden et. al. (1977) and Paessens (1988) (cited in Laporte, 1992).

Sweep algorithm (Gillet and Miller, 1974)

The sweep algorithm was developed by Gillet and Miller (1974) to solve medium to large size CDVRP. However, the origins of this work are found in Wren (1971) and Wren and Holiday (1972) for CVRP (cited in Laporte, 1992).

Each vertex is located in terms of the polar coordinates (r_i, θ_i) for $i = 1, ..., n$ with the depot at $r_0 = 0$ and $\theta_0 = 0$. The coordinates are arranged in ascending order according to θ_i. The algorithm can be described as follows:

- Choose an unused vehicle k.
- Start with the vertex with the smallest angle and assign the vertex to vehicle k if the capacity is not exceeded. Repeat the process until all the vertices are in the routes.
- Optimise each vehicle route using one of the TSP algorithms (exact or heuristic).

The Two-Phase method (Christofides et al., 1979)

The two phase method was developed by Christofides et al. (1979) for solving the CVRP and CDVRP. The algorithm is divided into two sections which can be described as follows:

Phase 1:

Step 1. Set $k = 1$.

Step 2. Choose an unrouted customer as a seed, s say, for building up the route R_k. Calculate δ_i for all the unrouted customers $i \neq s$ where $\delta_i = c_{0i} + \lambda c_{is}$, $\lambda \geq 1$.

Step 3. Select and insert i^* into route k (R_k) if $\delta_{i^*} = \min [\delta_i]$ and feasibility holds. Optimise R_k using r-opt and repeat Step 3 until no more customer can enter R_k.

Step 4. Set $k = k + 1$ and repeat Step 2 (with new route R_k) and Step 3 until all customers are in the routes.

Phase 2:

Step 1. Let h be the number of routes obtained from Phase 1 where $\bar{R}_r = (0, i_r, 0)$ and $r = (1, ..., h)$ for the seed customer i_r of $R_r, 1 \leq r \leq h$. Set $K = (\bar{R}_1, ..., \bar{R}_h)$.

Step 2. For each $\bar{R}_r \in K$ and for each unrouted node j, calculate $\epsilon_{rj} = c_{0j_r} + \mu c_{ji_r} - c_{0i_r}$ where $\mu \geq 1$ and $\epsilon_{r^*j} = \min_{\bar{R}_r \in K}[\epsilon_{rj}]$.

Step 3. Choose $\bar{R}_r \in K$ and set $K = K \setminus \bar{R}_r$. For each j compute $\bar{\delta}_j = \epsilon_{r'j} - \epsilon_{rj}$ where $\epsilon_{r'j} = \min_{R_r \in K}[\epsilon_{rj}]$.

Step 4. Select and insert j^* into $\bar{R}_r$ where $\delta_{j^*} = \max[\delta_j]$. Optimise $\bar{R}_r$ using r-opt and repeat Step 3 until no more vertices can be inserted.

Step 5. If $K \neq 0$, go to Step 2. Otherwise if all vertices are in the route, then stop. Otherwise, go to Step 2 in Phase 1.

An Improved Petal Heuristic (Renaud et al., 1996)

The petal heuristic was first proposed by Foster and Ryan (1976) for the VRP and later extended by Ryan et al. (1993) (cited in Renaud et al., 1996). The improved petal heuristic was proposed by Renaud et al. (1996) to generate a set of routes using the petal method and the optimal selection was made by solving the set partitioning problem using a column generation procedure. This heuristic produced near optimal solutions and used less computational time. The 1-petal heuristic and 2-petals heuristic were used to generate routes which can be described in the following:

1-petal heuristic constructs a Hamiltonian tour for S customers. An initial tour is formed and the remaining vertices are inserted into the partial tour. Finally the tour is improved by 4-opt edge exchange scheme.

2-petal heuristic starts with 2 seed vertices which are farthest apart to create two initial tours. The remaining vertices are inserted using the cheapest feasible insertion procedure. The tours are reoptimised by 4-opt according to a parameter γ. If any uninserted vertices remain, then six operations (similar to λ-interchange by Osman (1993)) are applied to improve the partial tours. As soon as one of the operations is feasible, the move is implemented and the insertion procedure starts again. When all vertices are in the routes, reoptimise each of them with 4-opt. The algorithm can be described as follows.

Step 1. Label all vertices in increasing order according to the polar angle of each position of the vertices with the depot. If ties occur then the vertex with the smallest radius is selected first. Set $i = 0$.

Step 2. Set $i = i + 1$, if $i > n$ stop.

Step 3. Let $S = \{v_i\}$ and record the cost $c = 2c_{0i}$. Set $j = i + 1$ and $S = \{v_i, v_j\}$. If the route is infeasible go to Step 4. Otherwise update S and c.

Step 4. Set $j = j+1$ and $S = \{v_i, ..., v_j\}$, if $\sum_{k=i}^{j} q_k > Q$, go to Step 5. Otherwise apply 1-petal heuristic with S. If no feasible route is defined, go to Step 5. Otherwise record solution S and cost c. Repeat this step.

Step 5. If $\sum_{k=i}^{j} q_k > 2Q$, go to Step 6. Otherwise apply 2-petal heuristic. If no feasible solution is found, go to Step 6. Otherwise record solution S and cost c. Set $j = j + 1$ and repeat this step.

Step 6. If $j = 2$, go to Step 2. Otherwise consider h to be the last vertex in S where $h = j, j - 1, ..., 3$. Let c_h be the cost of S with vertex h. If $c_{h-1} \leq c_h$, remove v_h from S and go to Step 2.

The solutions generated from the above heuristic are either embedded or intersecting routes which are then solved by the set partitioning problem defined as follows.

$$\text{Min} \sum_{l \in L} c_l x_l \tag{3.21}$$

subject to

$$\sum_{l \in L} a_{kl} x_l = 1, \quad k = 1, ..., n \tag{3.22}$$

$$x_l = 0 \text{ or } 1 \tag{3.23}$$

where L is a set of candidate 1-petal and 2-petal solutions, c_l is the cost of routes and $a_{kl} = 1$ if v_k belongs to route l. All pairs of vehicle routes are checked to determine if they can be combined. The combined route is then reoptimised by the 1-petal heuristic.

The algorithm was tested on 14 benchmark problems from Christofides et al. (1979) for the CVRP and CDVRP. The average results lie within 2.38% of the best known solution with running time between 0.43 to 11.70 minutes on a Sun Sparc 2 workstation.

Improvement Heuristics

There are two types of tour improvement procedures for VRP. One of the **Edge exchanges for a single route** based on the concept of k-Opt by Lin (1965) where k edges are replaced by another set of k edges. The other is called the **edge exchanges for multiple routes** where vertices or edges are exchanged between two routes. There are 3 basic exchanges which include **relocation, exchanges** and **crossover**. Similar operations are also developed by Van Breedam (cited in Laporte et al., 1999). For detail of this subject refer to Kindervater and Salvesbergh (1997).

This concludes the classic heuristic algorithms for the VRP. In the next section, modern heuristic algorithms developed for the VRP are discussed.

3.2.2 Modern heuristic algorithms for VRP

The modern heuristics developed recently for VRP are based on genetic algorithm (GA), neural networks (NN), tabu search (TS) and simulated annealing (SA). These methods explore the solution space more thoroughly by combining special procedures or functions. In general, these methods produce better solutions than the classic heuristic algorithms, but requiring longer CPU time. In some cases hybridization methods based on these new local search methods are proposed to solve the VRP. Note that a few hybridization methods have been proposed to solve the VRP with time window (VRPTW). However, there is no hybridization method for solving the CVRP and CDVRP. In this section, the methods based on GA, SA and NN for CVRP and CDVRP are discussed. For a review on TS for solving the VRP refer to Chapter 4.

In order to compare various algorithms developed for CVRP and CDVRP, 14 benchmark problems from Christofides et al. (1979) are used for testing the algorithms. The way of comparing the algorithm is based on the deviation of the solution obtained by the proposed algorithm with the best known solution in the literature and the CPU time required to run the problems. A table of solutions for the 14 benchmark problems from Christofides et al. (1979), solved by different authors with different heuristics, is presented in Table 3.3.

Note that the literature of GA for solving the VRP is limited. No research has been done on either CVRP and/or CDVRP. However, there are a few papers reporting on the VRP with time window (VRPTW) with/without capacity constraints since this is a more practical problem.

Problem	*Types*	*EG*[1]	*Alfa*[2]	*B*[3]	*RT*[4]	*T*[5]	*GHL*[6]	*Os*[7]	*Os*[8]	*Renaud*[9]
50	C	586	556.0	521	-	524.61	524.61	528	524	524.61
75	C	-	892.3	841	-	835.26	835.32	838	844	854.09
100	C	-	-	830	-	826.14	826.14	829	835	830.40
150	C	-	-	1063	-	1028.42	1031.07	1058	1044	1054.62
199	C	-	-	1360	1291.45	1298.79	1311.35	1376	1334	1354.23
50	C,D	-	-	548	-	555.43	555.43	555	555	560.08
75	C,D	-	-	920	-	909.68	909.68	909	911	922.75
100	C,D	-	-	870	-	865.94	865.94	866	866	877.29
150	C,D	-	-	1197	-	1162.55	1162.89	1164	1184	1194.51
199	C,D	-	-	1462	1395.85	1397.94	1404.75	1418	1422	1470.31
120	C	1133	-	1042	-	1042.11	1042.11	1176	1042	1109.14
100	C	836	-	821	-	819.56	819.56	826	819	824.77
120	D	-	-	1568	-	1541.14	1545.94	1545	1545	1585.20
100	D	-	-	867	-	866.37	866.37	890	866	885.87

Table 3.3: Comparison of the solutions obtained for the modern heuristic for CVRP and CDVRP using 14 benchmark problems from Christofides et al. (1979)

1: El Ghaziri (1991)
2: Alfa et al. (1991) (cited in Gendreau et al. (1997))
3: Breedam (1995)
4: Rochat & Taillard (1995)
5: Taillard (1993)
6: Gendreau et al. (1994)
7: Osman (1993): From SA algorithm
8: Osman (1993): Best solution from TS between the FBA and BA
9: Renaud et al. (1996)

Simulated Annealing and VRP

There are three algorithms based on SA for VRP by Alfa et al. (1991) (cited in Gendreau et al., 1997), Osman (1993) and Breedam (1995). The following describe the last two papers and omit the SA algorithm by Alfa et al. (1991) since it produced poor results on the small to medium size problems.

The SA algorithm developed by Osman (1993) was based on a scheme used to search the neighbourhood called λ-interchange generation mechanism. Let $S = \{R_1, ..., R_p, ..., R_q, ..., R_m\}$ be the feasible VRP solution. Select two routes R_p and R_q from S and let S_p and S_q be the set of nodes replaced between the routes R_p and R_q using the λ-interchange where $S_p \subseteq R_p$ and $S_q \subseteq R_q$. The size $\mid S_p \mid \leq \lambda$ and $\mid S_q \mid \leq \lambda$ where λ equals 1 or 2. This process consists of interchanging S_p and S_q between the routes or shifting the customers from one route to another if either S_p or S_q is empty. The new routes are $R_p' = (R_p - S_p) \cup S_q$ and $R_q' = (R_q - S_q) \cup S_p$ and the new VRP solution becomes $S = \{R_1, ..., R_p', ..., R_q', ..., R_m\}$. The following is an outline of the SA algorithm by Osman:

Step 1. Obtain an initial solution using the Clark and Wright saving algorithm.

Step 2. Perform a neighbourhood search of the initial solution using a λ-interchange generation mechanism without implementing the moves to obtain Δ_{max} and Δ_{min}, the largest and smallest value change in the objective function values. Let $S^* = S$ be the current solution. Set $R = 3$, $k = 1$, $T_r = \Delta_{max}$ and β be the number of feasible exchanges.

Step 3. Select a solution S' from the neighbourhood.

If $f(S') > f(S)$ and $\exp^{-(\frac{f(S')-f(S)}{T_k})} \geq \theta$ where θ is a uniform random parameter $0 < \theta < 1$, accept S'.

If $f(S') < f(S^*)$, update $S^* = S'$ and $T^* = T_k$.

Otherwise retain S.

Step 4. Update the temperature according to the following:

Normal decrement rule:

$$T_k = \frac{T_k}{1+\gamma T_k} \text{ where } \gamma = \frac{\Delta_{max} - \Delta_{min}}{(n\beta + n\sqrt{k})\Delta_{max}\Delta_{min}}$$

Apply occasional increment rule when a cycle of search is completed without accepting any 1-interchange move:
$T_r = \max(\frac{T_r}{2}, T^*)$ and set $T_k = T_r$. Update $k = k+1$.

Step 5. Stop if $k = R$ and report the best solution S^*. Otherwise go to Step 3.

A total of 26 problems including both the literature problems from 29 to 199 customers and randomly generated problems from 50 to 100 customers, were tested on the algorithm for the CVRP and CDVRP. The algorithm found 8 out of 14 better solutions from the problems of Christofides et al. (1979).

Breedam (1995) proposed three types of routes improvements for the VRP based on SA algorithm. One is the String Relocation method which locates a string of 'a' customers from one route to another, another is the String Exchange method exchanges a string of 'a' customers from one route to another with a string of 'b' customers and the final method is called the String Mix method which is a mixture of the above. The results of this algorithm are shown in Table 3.3.

Neural networks and VRP

Neural networks (NN) consists of a set of neurons which are interconnected and composed of weights that give some information about the networks. Starting from random weights, these weights are adjusted using a learning algorithm to improve the neuron's ability to perform a task.

The literature of NN for VRP is limited. Applications of NN based on the model of self-organising maps for the CVRP can be found by El Ghaziri (1991) and Matsuyama (1991). The following describes the algorithm by El Ghaziri (1991)

- Several different rings (or routes) are defined with weight vectors i.e. the initial position of the units in the ring.

- An input vector is considered and the output value is calculated (the weighted sum of its inputs). The winning output, say j^*, is the unit with maximal output.

- The weight vectors are then modified by $w_j = w_j + f(j, j^*)(I - w_j)$ where $w_j = (w_{ij}), i = 1, ..., n$ is the weight vector of unit j and f is a function of j and j^*.

- The unit of weight vectors are gradually modified until there is a weight vector that is close to each vertex. These rings are gradually moved to the vertices and form a set of VRP tours.

This algorithm was tested on three problems (50, 100, 120 customers) from Christofides et al. (1979) but the results were not good. Matsuyama (1991) employed a similar idea to solve the CVRP with 532-city problem derived from the TSP literature problem by adding a capacity constraint.

In this Chapter, a review of the exact and heuristic algorithms to solve the VRP in the literature is presented. The comparison of the algorithms is based on the deviation between the obtained solutions with the best known solutions for the literature problems. In particular, the problems from Christofides et al. (1985) are used as the benchmark problems to test various algorithms. In Chapter 5, the techniques or methods to compare various algorithms based on the use of the statistical tools, are proposed. In the next Chapter, the tabu search method and its applications to the TSP and VRP are discussed.

Chapter 4

Tabu Search

This chapter is devoted to Tabu Search (TS), a heuristic which has achieved widespread success in solving practical optimisation problems over the last few years. The origin of the TS went back to the 1970s and the modern form of TS was derived independently by Glover (1986) and Hansen (1986). The hybrids of the TS have improved the quality of solutions in numerous areas such as scheduling, transportation, telecommunication, resource allocation, investment planning. The success of the TS method for solving optimisation problems was due to its flexible memory structures which allowed the search to escape the trap of local optima and permitted to search the forbidden regions and explored regions thoroughly.

In Section 4.1 a brief overview of the TS algorithm will be presented. For details of the TS method refer to Glover (1990, 1995, 1996) and Glover and Laguna (1995, 1997). TS is still in the early stages of development with majority of its applications occurring since 1989. Much success has come from tackling the Vehicle Routing Problem (VRP) and the Travelling Salesman Problem (TSP). Sections 4.2 and 4.3 discuss the development of TS method with reference to the TSP and the VRP respectively.

4.1 An Overview of the Tabu Search method

Tabu search is a meta-heuristic which is designed to cross the boundaries of feasibility and search beyond the space of local optimality. The use of flexible memory based structures is the centre strategy of the TS method. It can be differentiated in long and short term memory. This feature allows the TS method:

1. To use the information already evaluated to search the region thoroughly;
2. To control the process, by using the memory structure to free or to continue the search process using techniques such as **tabu restriction** and **aspiration criteria** embedded in TS; and
3. To allow the use of **intensification** and **diversification strategies** for searching the region thoroughly and to search a new region using the Long and short term memory.

The structure of the TS method is similar to the local search (or neighbourhood search). Let X be the set of all solutions and for each solution $x \in X$, let $N(x) \subset X$ be an associated neighbouring set. At each iteration, the best solution x' is chosen from the neighbouring set $N(x)$, i.e. $f(x') < f(x)$ where $f(x)$ is the objective function value of the solution x. Then an operation called a **move** can be used to reach from x to x'. However, TS goes beyond local search by maintaining a record of history during the search. This determines which solution can be reached from the current solution and hence modify the neighbourhood set, denoted by $N'(x)$. In some large problems a subset of the neighbourhood set called the **candidate list** denoted by $cand\_N(x)$, is created because it is costly to examine all the elements in the neighbourhood set. The TS method is outlined as follows.

Step 1. Select a solution $x \in X$. Initialize the cost function $f(x)$ and the best solution x^*.

Step 2. Determine the candidate list, $cand\_N(x)$, from the modified neighbourhood set $N'(x)$. Choose a solution $x' \in cand\_N(x)$ which improves the

cost function.

Step 3. Update the new solution x' and x^*. Repeat Steps 1 and 2 until a termination criterion is satisfied.

4.1.1 Short Term Memory

Short term memory is the core of the TS method and the most commonly used structure which keeps the attributes of the recently changed solution. It is also called **recency-based memory**. Recency-based memory keeps a record of the recently changed solutions by assigning them to a tabu-active list. In fact, those solutions which are tabu-active are declared as **tabu**. This feature prevents the solutions which are tabu from being in the neighbourhood set $N(x)$ and also being revisited again for a certain number of iterations. The primary goal is to allow the process to go beyond the local optimal and still make good quality moves.

Tabu list is used to manage the recency-based memory where it keeps a record of solutions which are tabu-active. The duration for which a solution remains tabu is measured in terms of iterations and is called **tabu tenure**. Tabu tenure can vary in different stages of the search process or period of time. Hence this creates a different trade-off for the long term and short term memory strategies which results in a dynamic and robust technique for the TS method. The size of the tabu list can have a great effect on the search process. If the list is too short, the process may return to the same local optima which prevents the search to explore other solution space. In contrast, if the tabu list size is too long, longer computational time is needed to search the tabu list to determine if the move is tabu. Tsubakitani and Evans (1998b) have researched in this area by testing the symmetric TSP using the 2-opt and 3-opt procedures to find a reasonable size for the tabu list. Their results showed that the tabu list size should be as small as possible but long enough to allow the heuristic to move away from the local optimal. The more powerful the search method is, the smaller the tabu list size should be.

Another important element in TS is the **aspiration criteria** which over-rules

the tabu status of a move's attributes if a certain condition is met. That is, a move which is classified as tabu is considered to be admissible if it improves the objective function value.

In most of the situations when the problems are large, examining all the neighbourhood sets is expensive and hence TS used the **candidate-list strategy** which restricts the number of solutions to be examined. As a result reduce the computation effort. Glover (1995) suggested a list of selecting candidate-list strategies: subdivision strategy, aspiration plus strategy, elite candidate list strategy, bounded change candidate list strategy and sequential fan candidate list strategy. For details of these strategies refer to Glover (1995) and Glover and Laguna (1997).

In order to choose the best admissible candidate from the candidate list, each move is evaluated based on the change of the objective function value of the solution before and after the move. Tabu status is then checked for admissibility. If the move is not tabu, it is accepted; otherwise, the aspiration criteria is applied to override the tabu status if the solution is of better quality. The best move, in terms of solution quality, is chosen.

The short term memory structure can be described as follows.

Step 1. A solution $x' \in N(x)$ is chosen and removed from the candidate list.

Step 2. Check the tabu status of the solution. If x' is tabu, proceed to Step 3. Otherwise go to Step 4.

Step 3. (Aspiration criteria) If the objective function value $f(x')$ improves the solution quality, go to Step 4. Otherwise a large penalty is added to the $f(x')$.

Step 4. If x' is the best solution among all the candidate list, update the record.

Step 5. If the candidate list is empty, implement the best solution x' recorded and stop. Otherwise go to Step 1.

4.1.2 Long Term Memory

In some applications short term memory is sufficient to produce a good quality solution, but the TS method becomes stronger when long term memory components are used. When using long term memory, the neighbourhood set consists of a selection of elite solutions, i.e. high quality local optimal obtained at various stages of the search process. Some of these elite solutions can be identified in a cluster of a region by the intensification strategy or in different clusters by the diversification strategy.

The intensification and diversification strategies counterbalance and reinforce one another. They make use of the frequency-based memory, which provides information obtained from the recency-based memory and broadening the selection of solutions. Two types of frequency-based memory are recorded: **transition frequency** which identifies the number of times a particular attribute moves and **residence frequency** which identifies the number of time the attribute resides in the solution.

Intensification Strategy

Intensification strategy focuses the search on good regions and good solution features. The approach of the intensification strategy can be described as follows.

Step 1. The short term memory is applied.

Step 2. Apply an **elite solution strategy**, i.e. select a list of high quality local optima.

Step 3. If the list is empty, stop; Otherwise choose a solution and remove it from the elite solution list.

Step 4. Apply the chosen solution using steps which describe short term memory structure from Section 4.1.1. If a good quality solution is found during the search, add to list.

Step 5. Repeat Steps 2, 3 and 4 until the iteration limit is reached.

In the literature there are three variants of the intensification strategy which have been proven to be useful.

- A diversification measure is introduced such that the solutions recorded each time differ from another. The short term memory is then erased before starting from the best solution recorded;
- A bounded sequential list is created which adds the new solution at the end of the list if the solution is better than any previous solutions. The current last element in the list is chosen as the basis for starting the search. The short term memory for this solution is recorded so that the move which was previously taken to be forbidden is kept and different moves can be created in order to create new solutions; and
- To begin the search from unvisited neighbours which have been previously generated.

Another type of the intensification strategy is called intensification by decomposition. This strategy placed restrictions on some parts of the solution structure to generate a form of decomposition which focuses more on other parts of the solution structure.

Diversification Strategy

Diversification strategy explores new regions and is based on a modifying choice rule by bringing the attributes which are infrequently used into the solution. The timing for applying the diversification step is important and only applies at the local optima. In order to identify appropriate moves so that the search procedure can jump out of the local optima, a memory function can be used to identify the relative attractiveness of the moves based on the move distance. This is based on the idea that the greater the distance involved in the move, the greater the cost is compared to a smaller shift. Hence large distance moves are generally unattractive and rarely chosen. Furthermore, historical information is used to

identify moves which are infrequent and with a large distance. This provides a good technique for diversification (Glover, 1990).

Another form of diversification is to restart the solution process from different solutions generated randomly or by a set of starting heuristic solutions. This approach was applied to TSP by Malek et al. (1989) and quadratic assignment problem by Skonic-Kapov (1989) (cited in Glover, 1990). Rochat and Taillard (1995) implemented this approach by producing a set of different initial solutions so that different regions can be explored.

A variant of the TS method called **probabilistic tabu search** keeps track of tabu evaluation during the process with the move chosen probabilistically from the set of evaluated moves. Rochat and Taillard (1995) applied this approach to their algorithm which will be discussed in Section 4.3.2.

4.1.3 Strategic Oscillation

Strategic oscillation provides an effective interplay between intensification and diversification strategies for the intermediate to long term memory. It operates by moving the solution until it reaches a boundary, which can be represented by feasibility, a point where normally the method would stop. However the method does not stop here, the neighbourhood definition is extended or the condition of selecting the moves is modified in order to allow the boundary to be crossed. It then moves for a specified depth beyond the boundary and turns around. The boundary is again crossed from the opposite direction, approaching a new turning point. The process is continued by crossing the boundary from different directions which creates the oscillation pattern. This pattern is controlled by modifying the evaluations and the rules of movement depending on the region and the direction of search. The strategy has been applied to multidimensional knapsack problems and the graph theory problems. For further details of this subject refer to Glover and Laguna (1997).

4.1.4 Path Relinking

Path relinking is an approach which generates new solutions by investigating the path that connects the elite solutions. It starts from a solution called **initiating solution** and a path is generated in the neighbourhood space which leads to a second solution called **guiding solution.**

This approach chooses moves which incorporate the attributes of the guiding solutions to create a "good attribute composition" in the current solution. The composition of each step is determined by choosing the best move based on a set of criteria, from a restricted set of moves that incorporate a maximum number of attributes of guiding solutions. However aspiration criteria can be overridden by allowing other moves of high quality to be considered.

After a collection of one or more elite solutions are identified, weights are assigned to the attributes of these guiding solutions. Larger weights are assigned to attributes occurring greater number of times which emphasised on the solutions with high quality.

The intensification strategy based on the path relinking may choose solutions to be the elite solutions which lie in the common region or have similar attributes. The diversification strategy based on path relinking on the other hand chooses solutions which are from different regions or have different features. Refer to Glover and Laguna (1997) for further details.

4.2 Tabu Search and The Travelling Salesman Problem

The first TS algorithm for TSP was suggested by Glover in 1986 (cited in Johnson and McGeoch, 1997). Similar work was also done by Malek et al. (1989), Knox (1994) and Fiechter (1994). The algorithms proposed by these authors adopted the 2-edge exchange strategy. Different tabu list size and aspiration criterion were suggested. Smaller size problems from 25 to 105 cities were tested by Malek et al.

(1989) and Knox (1994). Fiechter (1994) combined long and short term memory of TS which allowed the algorithm to find near-optimal solutions for some large problems from 500 to 100 000 cities. Parallel implementations of the algorithm were proposed to speed up the CPU time.

Malek et al. (1989) also implemented a parallel version of the TS algorithm and their results showed that the parallel algorithm outperformed the serial one in terms of CPU time and solution quality.

The following discusses the TS algorithm by Knox (1994).

4.2.1 The Knox algorithm (1994)

Knox proposed a TS method together with the 2-edge exchanges for the symmetric TSP. The 2-edge exchange can be described as deleting 2 non-adjacent edges from the current tour and adding 2 other edges to obtain a feasible tour. The author pointed out that there is a relationship between the number of searches made and the length of each search which will influence the quality of the best solution identified by the tabu search method. Tests were done on six literature problems ranging from 25 to 75 cities and the results showed that 4 searches and $(0.0003*n^4)$ iterations per search are required to produce approximately 95% best solutions. A comparison of the TS method and the 2-opt and 3-opt on the six problems showed that TS yields better quality solutions. This result, however, was only based on smaller size problems (less than 75 cities).

The following outlines the TS algorithm by Knox.

Step 1. Construct an initial tour either randomly or by some initial tour construction procedures.

Step 2. Identify a set of 2-edge exchange moves and select the best admissible candidate.

Step 3. Implement the best admissible candidate.

Step 4. Update the tabu list, aspiration list, other variables and the local best

tour found so far. If the stopping criterion is reached, go to Step 5. Otherwise go to Step 2.

Step 5. Update the global best tour if the recent tour found is the best one found so far.

Step 6. If the number of searches is reached, output the best tour and stop. Otherwise go to Step 1.

4.3 Tabu Search and The Vehicle Routing Problem

In this section, the applications of the TS method for the VRP are discussed. Due to the limited success of exact methods to solve the VRP of more than 100 customers, researchers have denoted considerable effort towards developing efficient heuristic algorithms that provide good solutions to large-scale problems. One recent approach is the tabu search method. This section will focus on the capacity restricted vehicle routing problem (CVRP) and capacity and distance restricted vehicle routing problem (CDVRP). Table 4.1 shows a list of the development of the TS method with reference to the VRP. The TS has shown some promising results in solving the VRP. The method not only demonstrates some new results for the problems in the literature but also solves some real-life problems. In solving two real-life cases using the TS method, cost was reduced approximately by 15% as reported by Semet and Taillard (1993) and Rochat and Semet (1994). Parallel implementations of the TS algorithm are able to handle larger size problems as suggested by Taillard (1993). He proposed two decomposition methods for solving the VRP for uniform and non-uniform problems. The method is to partition the problem into several subproblems and solves them independently. This approach is tested on the uniform problems by Christofides et al.(1979). The results indicated that better solutions were found for 5 out of 14 problems. The 120-city problem from Christofides et al. (1979) and the 385-city problem were tested for the nonuniform problems. The algorithm showed some improvement

Author/(s)	*Special Features*	*Comments/Results*
Osman (1993)	Special data structures	Obtained 13 better solutions out of the 17 literature problems. Also tested on 9 randomly generated problems
Taillard (1993)	Decomposition method and TS	Obtained 5 better solutions out of 14 problems from Christofides et al.(1979). Parallel implementation for TS method and randomly generated problems were tested from 64-1024 cities
Semet & Taillard (1993)	Real life VRP with TS method	Reduce 10-15% of the distribution costs for the company in Switzerland
Gendreau et al.(1994)	GENIUS+TS	Obtained 11 better solutions out of the 14 problems from Christofides et al.(1979)
Rochat & Semet (1994)	Real life VRP with TS method	Found good solutions in a reasonable amount of CPU time and saving approximately 17% costs
Rochat & Taillard (1995)	Probabilistic TS	Improved on the solutions for several problems from the literature
Xu & Kelly(1996)	Network flow based TS method	Solved 5 out of 7 problems from the literature solutions

Table 4.1: Development of the TS methods for the VRP

in the solutions. Furthermore, randomly generated problems ranging from 64 to 1024 cities were also tested.

Real-life VRPs and TS

Solving real-life problems are different from solving problems from the literature. In the case of real-life problem a variety of restrictions need to be considered. Two real-life VRPs using the TS method were considered by Semet and Taillard (1993) and Rochat and Semet (1994).

Semet and Taillard (1993) considered delivering groceries order (70-90 orders) to 45 different grocery stores in Switzerland. The vehicles used by the company consists of trucks and trailers. The problem was to minimise the transportation costs while satisfying the following constraints:

- The order must be delivered within a time window;

- The carrying capacity of a vehicle must be satisfied; and
- Each store can only be reached by a subset of trucks and trailers.

The TS method was used to solve the problem and the results indicated a 10-15% reduction in distribution costs.

Rochat and Semet (1994) considered the delivery of pet food and flour for a major firm in Switzerland. The problem was to reduce the transportation costs since it represented a large percentage of the total costs of the company. A few constraints were considered in the problem such as customers, fleet and crew requirements. Two sets of customers were considered: a small set representing the farmers and a large set which consists of wholesalers, bakers and retailers. Each customer was visited by one vehicle and was reached by a subset of vehicles. A variety of delivery locations were considered such as village, isolated farm and centre of the city and the time windows for delivering the products needed to be satisfied. A heterogeneous fleet consisting of 14 trucks was used and each was characterised by the volume and weight capacity of the truck. Each vehicle started and ended at the depot and the carrying capacity of the truck must be satisfied. The total length of a route which included travel time, service time, waiting time and access time (which was the time taken for finding the way to the customer from the village centre and parked the vehicle) cannot exceed 10 hours 15 minutes. According to the Swiss federal law of work and rest for the professional drivers, the following needs to be followed:

- At least an hour break needs to be taken by the driver after 4 hours of uninterrupted driving time or 5.5 hours of uninterrupted work.

Uninterrupted work or driving time was referred to as not interrupted by a 30-minute break. The results showed that good solutions with fewer vehicles than those used in previous were produced within a reasonable CPU time.

4.3.1 The Osman algorithm (1993)

The TS method is based on the use of memory structure. The data structures use to store information require in the design must be such that the time taken to search the information is minimum. In the TS algorithm proposed by Osman, a special data structure is used to reduce the computational time. Two matrices named BSTM and RECM with dimensions mxm and $m(m-1)$ are used in the algorithm so that the required information can be quickly scanned. The top triangular part of the BSTM(p,q) ($1 \leq p < q \leq m$) is used to store the change in the objective function value of the best move of customers i and j from routes p to q or if the move is not allowed, a large arbitrary number is assigned. The lower part of the BSTM(q,p) is used to store an index l which is associated with routes p and q for the matrix RECM. The matrix RECM is used to store the attributes of the best move, for example, RECM(l, 1)=i and RECM(l, 2)=j.

In the TS method, parameters such as the tabu list size[1] (i.e. how long the move is tabu for?), the stopping criteria (i.e. when to stop?) will need to be determined. Osman used the regression technique to determine such parameters based on the number of cities, n, the number of vehicles used, m, and the demand for each customer versus the vehicle capacities. The tabu list data structure used by Osman is a $m(n+1)$ matrix called TABL. It has n rows, one for each customer and the extra row is used to store the shift process information (the shift process is either (0,1) or (1,0) which can be described by shifting one customer from one route to another) and m column for the vehicles. TABL(i, p) records the iteration number where customer i is removed from route p.

In the algorithm, two selection strategies were used: the *best-admissible strategy* (BA) where the best-admissible move is chosen from the neighbourhood set and *first-best-admissible strategy* (FBA) where the first admissible move which improves the objective function value is selected. If there is no improvement then the best non-improvement move is chosen. The two selection strategies were

[1]Osman refers to tabu list size which is denoted by $\mid Ts \mid$ as the period of iterations the move is tabu.

tested for a range of literature problems and randomly generated test problems. The algorithm with FBA strategy obtained 13 better solutions out of the 17 literature problems with sizes ranging from 29 cities to 199 cities and 6 better solutions out of the 9 random test problems with sizes ranging from 50 cities to 100 cities. The algorithm with the BA strategy found 12 better solutions for the literature problems and 3 better solutions for the random test problems.

The following outlines the TS algorithm by Osman:

Step 1. Obtain a solution S by the savings method. Initialise the tabu list size $\mid Ts \mid$, the tabu list data structure TABL, the stopping criteria M. Set k=1, k_b=0 and the best solution so far, $S_b = S$. Initialise the two matrices BSTM and RECM if BA strategy is used. Initialise the best solution S_b.

Step 2. Choose a feasible and admissible move S' from the neighbourhood based on BA or FBA strategies. Update TABL with the new acceptable move. Update the solution $S = S'$ and set $k = k+1$. If $f(S') < f(S_b)$, update S_b and set $k_b = k$. Update the data structure RECM and BSTM if BA is used.

Step 3. If $(k - k_b)$ >M go to Step 4; otherwise go to Step 2.

Step 4. Stop. Report the best solution S_b.

4.3.2 The Rochat and Taillard algorithm (1995)

The algorithm proposed by Rochat and Taillard (1995) is based on two components of the long term memory of TS method, i.e. intensification and diversification strategies. The approach of their algorithm comes from probabilistic tabu search which uses the information generated by the search history and measures of attractiveness, to evaluate probabilities for selecting the next attributes. Those attributes with higher evaluation are in favour.

Their TS algorithm can be described as follows.

Step 1. Generate I different initial tours with the local search method.

Step 2. Remove any tours with only one customer. Sort the tours in increasing order according to the cost/distance and call this new set of tours T.

Step 3. Set $T' = T$. Let S be a set of tours extracted from T and set $S = \emptyset$. Choose $t \in T'$ probabilistically based on the evaluation of the objective function value. Set $S = S \cup \{t\}$. Remove from T' all the customers belong to t.

Step 4. Construct a feasible solution S' using the partial solution S and include some customers not belong to S. Improve the solution quality of S' with a local search method.

Step 5. Remove tours with one customer from the improved tour S' and insert the remaining tours to T. Repeat Step 2.

Step 6. Repeat Steps 3 to 5 until stopping criterion is reached.

In Step 1, several different initial solutions are generated and the aim is to explore diverse regions of the solution space and hope that a solution of high quality will exist in this set of solutions. As the process progresses, the set of tours, T, will grow and the partial solutions are becoming more and more complete. The process will automatically intensify the search after several iterations of Step 3 to 5 because the worst solutions are removed from T and the best tours are extracted more frequently and hence the search changes from a diversification to intensification process.

One of the disadvantages of the local search algorithm is the large computational effort because it processes sequentially and hence parallel implementation is difficult. However, the TS algorithm proposed by Rochat and Taillard, can be parallelized easily.

Rochat and Taillard (1995) applied their algorithm to the basic VRP (i.e. one depot and identical vehicles) and VRP with time windows (VRPTW). The results showed that solutions have been improved for the basic VRP problems for 134

cities from Fisher (1994), 199 cities from Christofides et al. (1979) and 385 cities from Taillard (1993); and for the VRPTW 27 problems out of the 56 Solomon's VRPTW problems were found to improve in the quality of the solution.

4.3.3 The Xu and Kelly algorithm (1996)

Xu and Kelly approached the VRP using the TS method with the network flow-based model. Their work is an extension of Glover (1992) and Stewart et al. (1994) (cited in Xu and Kelly, 1996). The purpose of this approach is to exchange the customers between routes such that the distance and feasibility are simultaneously considered.

The network model can be divided into four levels of arcs. The first level consists of a source node S with an input flow of c units which is connected with m vehicles. On level 2, all m vehicles are connected with n customers. The flows on these arcs represent the corresponding customer being removed from its current route. On level 3, all n customers are connected with m vehicles and the flows on these arcs represent the corresponding customer being inserted into a particular route. On level 4, the flow represents the number of customers in each route and all arcs are joined to a sink node T with the output flow of c units.

Let f_{ij}^1 be the flow along the arc from customers i to j on level 1, C_{ij}^1 be the cost on that arc, L_{ij}^1 and U_{ij}^1 be the lower and upper bound on that arc. The mathematical representation of the model is given below.

Minimise

$$\sum_{j=1}^{n}\sum_{i=1}^{m} C_{ij}^2 f_{ij}^2 + \sum_{j=1}^{n}\sum_{k=1}^{m} C_{jk}^3 f_{jk}^3 \tag{4.1}$$

subject to

$$\sum_{i=1}^{m} f_{si}^1 = c \tag{4.2}$$

$$f_{si}^1 - \sum_{j=1}^{n} f_{ij}^2 = 0, \;\; i = 1, ..., m \tag{4.3}$$

$$\sum_{i=1}^{m} f_{ij}^2 - \sum_{k=1}^{m} f_{jk}^3 = 0, \;\; j = 1, ..., n \tag{4.4}$$

$$\sum_{j=1}^{n} f_{jk}^{3} - f_{kt}^{4} = 0, \;\; k = 1, ..., m \tag{4.5}$$

$$\sum_{i=1}^{m} f_{it}^{4} = c \tag{4.6}$$

$$L_{si}^{1} \leq f_{si}^{1} \leq U_{si}^{1} \;\; i = 1, ..., m \tag{4.7}$$

$$L_{ij}^{2} \leq f_{ij}^{2} \leq U_{ij}^{2} \;\; i = 1, ..., m, \;\; j = 1, ..., n \tag{4.8}$$

$$L_{jk}^{3} \leq f_{jk}^{3} \leq U_{jk}^{3} \;\; k = 1, ..., m, \;\; j = 1, ..., n \tag{4.9}$$

$$L_{kt}^{4} \leq f_{kt}^{4} \leq U_{kt}^{4} \;\; k = 1, ..., m \tag{4.10}$$

In the algorithm the search oscillates between the swap moves and the network flow moves. The swap move is where two customers are exchanged between the two routes. While the network flow move dominates the search, the swap move is executed every s_1 iterations or if *iter_swap* $\geq$ iteration count for the algorithm where *iter_swap* is a pre-determined integer number. If the move is tabu, the associated costs are changed to a large number and when the tabu status is released, then the associated cost is recalculated.

The 3-opt procedure is used to improve the tour where three edges are deleted from the current tour and reconnected them in different ways such that the objective function value is improved. A TS component which is called TSTSP tries to reduce the gap between the heuristic solution and the optimal route for larger TSP. This approach uses two different moves: ejection and swap. The ejection move ejects one customer to another position while the swap exchanges two customers' positions. During the search process, all eligible moves are evaluated and the best move is selected. If a move is made which leads the search back to the previous solution, then the move is declared *tabu* for a number of iterations. But the tabu status can be overridden if the move leads to a new solution.

Let λ and ϵ be the pre-defined tolerances of the solution values, n_1 be the pre-determined iteration counter and *iter* be the iteration count for the algorithm. Let the objective function value of the current solution and best feasible solution be *current_solution* and *best_solution* respectively. The following outlines the algorithm.

Step 1. Generate an initial solution using the sweep method.

Step 2. If ($iter$ mod $s_1 = 0$) or ($iter\_swap \geq iter$) evaluate the swap moves procedure. Otherwise evaluate the network flow model using the CPLEX LP solver. Update the elite solution and select the best move as *current_solution*.

Step 3. For every n_1 iterations or if (*current_solution* - *best_solution* $\leq \lambda$) improves the current solution using 3-opt. If (*current_solution* - *best_solution* $\leq \epsilon$) apply TSTSP.

Step 4. Update the variables, penalty parameters, *iter_swap* and tabu list. If (*current_solution* $<$ *best_solution*) update the best tour. Set $iter = iter + 1$.

Step 5. Repeat Steps 2 to 4 until the maximum number of iterations is reached. Otherwise stop.

Tests were carried out using the capacity restricted problems from Christofides et al. (1979) and Fisher (1994). The sizes of the problems were ranging from 44 to 199 cities. The results showed that the algorithm was competitive or outperformed recent heuristics on both real and integer distances.

This concludes the literature review for the TS method for the TSP and the VRP. The following chapter discusses the drawbacks faced by an end user while solving the TSP and VRP and presents a formal definition of the neighbourhood structure.

Chapter 5

Neighbourhood Structures and Heuristics

The reviews of the literature in Chapters 2, 3 and 4 reiterate the following drawbacks faced by researchers while solving a TSP or VRP:

- The exact algorithms may not solve a given instance of the problem in a reasonable time;
- There are too many heuristics available;
- Very little effort is put into assessing the quality of solutions produced by heuristics;
- Different implementations (runs) of the same probabilistic heuristic on a given input instance of a problem may produce distinct solutions of different quality. Thus desired quality and reproducibility of the solution cannot be ensured;
- Most of the comparisons available in the literature between the performance of the various heuristics are based on the so called benchmark literature problems. Problems faced by the researchers may not be similar to the benchmark literature problems. Furthermore, the performance of the heu-

ristics on the benchmark problems provide no guarantee on the quality of solutions that can be obtained for the problem faced by a researcher;

- Most of the documentation on the performance of heuristics on literature problems provide no information regarding the computational effort (CPU time) spent on obtaining the claimed solution, reproducibility of the claimed solution and hardware environment of the implementation. Thus many of the comparison documented on heuristics are irrelevant to the researcher; and
- The lack of accepted method of comparisons between the various heuristics.

These deficiencies emphasise the need for a proper statistical evaluation of any probabilistic heuristic proposed for a combinatorial optimisation problem. In this chapter we develop a scheme for performing the required statistical analysis. In Section 5.1, we introduce three different combinatorial optimisation problems and demonstrate different types of neighbourhood structures used in the literature for these problems. We also highlight the dependency of the heuristic algorithm on the exact neighbourhood structure used.

In Section 5.2 we present a formal abstract definition of neighbourhood structure, local minimum and global minimum solutions for a combinatorial optimisation problem. Furthermore, we discuss the use of such structure in a probabilistic search method. Subsequently we introduce the statistics (that is indicators of measure of performance of the heuristics) that could be analysed to evaluate the heuristics. These techniques are developed with the intention of selecting the best heuristic or the best set of parameters for a given heuristic in the context of solving a problem. Some of the concepts of this chapter appear in Achuthan and Chong (1999). Subsequently in Chapter 8 we further extend the investigation of neighbourhood search method on the basis of the complexity of the solution space for a given instance of a problem.

5.1 Background

Without loss of generality, a combinatorial optimisation problem can be defined as, the problem of minimising an objective function defined over a set of discrete feasible solutions. The problem can be formally stated as

$$\text{Minimise } \{f(x) : x \in S\} \tag{5.1}$$

where f is a cost function and S represents the set of feasible solutions of the problem and x represents a typical feasible solution. There are various classes of combinatorial optimisation problems such as the scheduling problem, TSP and VRP. The method used to recognise the solution $x \in S$ varies from problem to problem. In the following we describe a few examples of combinatorial optimisation problems.

1. **Scheduling Problem**

   The general scheduling problem normally involves a set of n jobs and a set of m machines. Each job has to be processed on some machines. There may be precedence relations between jobs and also between processing a particular job on two different types of machines. The processing times are given. The problem is to schedule the jobs on the machines satisfying the given restrictions and minimising the total elapsed time between the start of the first processing job and the completion of the last processing job. To provide a suitable mathematical model for such general scheduling problem is not easy. More specifically, let us consider the general scheduling problem described below

   - m machines, denoted by k, $1 \leq k \leq m$.
   - n jobs, denoted by $i, 1 \leq i \leq n$.
   - r_i refers to the arrival time of job $i, 1 \leq i \leq n$.
   - d_i refers to the due date, that is the time by which all operations of the job i must be completed, $1 \leq i \leq n$.

- Job i consists of a set of g_i operations: $\{m_{i,1}, p_{i,1}\}, ..., \{m_{i,g_i}, p_{i,g_i}\}$ where the integer $m_{i,j}$ refers to the identification number of the machine that is required to perform jth operation of job i, $1 \leq m_{i,j} \leq m$. The processing time $p_{i,j}$ refers to the amount of time required for machine $m_{i,j}$ to perform this operation.

In such a general scheduling problem, a feasible schedule x will be representing collection of intervals associated to each operation $(m_{i,j}, p_{i,j})$ such that the total sum of the intervals associated to a operation $(m_{i,j}, p_{i,j})$ is $p_{i,j}$. The set S will be the collection of all feasible schedules x satisfying the given restrictions on the jobs, their operations and their precedence relations. The objective is often to find a feasible schedule x^* which minimises the total elapsed time over the set S.

In the following we describe the model of the flow-shop scheduling problem with two machines.

Two-machine Flow Shop

Two-machine flow shop problem is a special case of the above general scheduling problem where the number of machines $m = 2$ and the technological ordering of the machines is same for all the jobs. In other words, $m_{i,1} = 1$ and $m_{i,2} = 2$ for all i, $1 \leq i \leq n$. Furthermore we assume that $r_i = 0$ for all i, $1 \leq i \leq n$.

To simplify the notation, define

$A_i = p_{i,1}$ be the processing time of the ith job on the first machine.

$B_i = p_{i,2}$ be the processing time of the ith job on the second machine.

$F_i =$ the time of completion of ith job.

Each job consists of a pair (A_i, B_i) and the followings are assumed:

- Each machine can work on only one job at a time.
- Each job can be processed on only one machine at a time. For a job i, A_i has to be finished before starting B_i.

For a two-machine flow shop problem, while minimising the total elapsed time, it is enough to consider the permutation schedules with same permutation on both machines. Therefore a feasible schedule can be represented by a permutation $\pi = (\pi(1), \pi(2), ..., \pi(n))$ of the n jobs. Furthermore, the total elapsed time under the schedule π can be established as

$$F_{max}(\pi) = \max_{1 \leq i \leq n} F_{\pi(i)} = \max_{1 \leq r \leq n} [\sum_{i=1}^{r} A_{\pi(i)} + \sum_{i=r+1}^{n} B_{\pi(i)}]. \qquad (5.2)$$

Thus the two machines flow-shop problem can be reworded as

$$\text{minimise } \{F_{max}(\pi) : \pi = (\pi(1), ..., \pi(n)) \in S\}$$

where S is the set of all permuations of the n jobs, $1 \leq i \leq n$ and $F_{max}(\pi)$ is given by (5.2). A permutation of the n jobs, $1 \leq i \leq n$ is denoted by $(\pi = (\pi(1), ..., \pi(n))$. We refer to Conway et al. (1967) for detail of the two-machine flow shop problem.

Three-machine Flow Shop

In this case, $m_{i,j} = j$, where $1 \leq j \leq 3$ and $1 \leq i \leq n$. Similarly in the case of the three-machine flow shop problem we can restrict attention to the permutation schedules and seek to minimise the total elapsed time expressed as

$$F_{max}(\pi) = \max_{1 \leq i \leq n} F_{\pi(i)} = \max_{1 \leq r \leq s \leq n} [\sum_{i=1}^{r} A_{\pi(i)} + \sum_{i=r}^{s} B_{\pi(i)} + \sum_{i=s}^{n} D_{\pi(i)}]. \qquad (5.3)$$

Note that A_i, B_i and D_i denote the processing times of job i on machines 1, 2 and 3 respectively. Thus again the three machines flow-shop problem can be reworded as

$$\text{minimise } \{F_{max}(\pi) : \pi = (\pi(1), ..., \pi(n)) \in S\}$$

where S is a set of all permutations of the n jobs, $1 \leq i \leq n$ and $F_{max}(\pi)$ is given by (5.3)

2. **The Travelling Salesman Problem (TSP)**

From the review given in Chapter 2, we can reword the TSP as

$$\text{minimise}\{f(\pi) : \pi \in S\} \tag{5.4}$$

where $S = \{\pi = (1, i_2, ..., i_n, 1)$ is a tour such that $(i_2, ..., i_n)$ is a permutation of the $(n-1)$ cities, $2 \leq i \leq n\}$. The objective function value is $f(\pi) = \sum_{r=1}^{n} C_{i_r i_{r+1}}$ and $i_1 = i_{n+1} = 1$. Recall that '1' is the home city of the salesperson.

3. **The Vehicle Routing Problem (VRP)**

From the review provided in Chapter 3, we can reword a CVRP as follows using the following notation:

- Let $\pi_k = (0, \tau_1^k, ..., \tau_{r_k}^k, 0)$ be a vehicle tour of kth vehicle if $(0, \tau_1^k, ..., \tau_{r_k}^k, 0)$ forms a cycle in the graph $G = (V, A)$.
- Let $T = (\pi_1, ..., \pi_m)$ be a set of m valid vehicle tours if
  - every pair of vehicles u and v visit disjoint set of customers, that is, we have $\{\tau_1^u, ..., \tau_{r_u}^u\} \cap \{\tau_1^v, ..., \tau_{r_v}^v\} = \emptyset$ for every $u \neq v, 1 \leq u, v \leq m$; and
  - the total demand of the customers in every vehicle tour does not exceed the vehicle capacity, that is $\sum_{j=1}^{r_u} q_{\tau_j^u} \leq Q$ for every $u, 1 \leq u \leq m$.
- S is the set of feasible solutions to CVRP can be written as

$$S = \{T = (\pi_1, ..., \pi_m) : T \text{ is a set of} m \text{ feasible vehicle tours }\}. \tag{5.5}$$

- The objective function $f(\pi)$ representing the total time travelled by all m vehicles can be written as

$$f(T) = \sum_{k=1}^{m} d(\pi_k) \tag{5.6}$$

where $d(\pi_k) = \sum_{u=0}^{r_k} C_{\tau_u^k \tau_{u+1}^k}$, $\tau_0^k = \tau_{r_k+1}^k = 0, 1 \leq k \leq m$. Note that $d(\pi_k)$ is the time travelled by kth vehicle.

Again the CVRP is reworded using the above notation as

$$\text{minimise } \{f(T) : T \in S\}$$

where S is a set of feasible solutions of CVRP given by (5.5) and $f(T)$ is given by (5.6)

In the next section we provide an insight into the dependency of the heuristic on the definition of neighbourhood structure.

5.1.1 Neighbourhoods

A suitable definition of neighbourhood structure plays an important part in local search methods for optimisation problems. How well an algorithm solves a problem depends on how well the neighbourhood structure is defined. There are optimisation problems where a suitable definition of the neighbourhood structure ensures that a global optimum can be easily obtained by a local search method. For example consider the two-machine flow shop problem which can be solved efficiently by Johnson's algorithm. This can be interpreted as travelling through a neighbourhood structure defined as follows. Two permutations π and π' are neighbours if they differ exactly in two adjacent positions. In other words, if $\pi = (\pi(1), ..., \pi(i-1), \pi(i), \pi(i+1), \pi(i+2), ..., \pi(n))$ then $\pi' = (\pi(1), ..., \pi(i-1), \pi(i+1), \pi(i), \pi(i+2), ..., \pi(n))$.

Thus in the case of the two-machine flow shop problem,

- the set of all feasible solutions is represented by the set of all permutations; and
- two feasible solutions π and π' are neighbours if π' can be obtained from π (by interchanging two consecutive positions) and vice versa.

Due to the following theorem by Smith (1956), this neighbourhood structure ensures that starting from any feasible solution π, one can reach the optimal solution π^* by using a standard local search method.

Smith's Theorem (1956)

Let f be a real-valued function on a set of permutations of n jobs. A sufficient condition that $f(\pi^*) \leq f(\pi)$ for all permutation is that:

- There is a real-valued function g defined on ordered pairs of jobs such that if π is a permutation and π' is a permutation obtained from π by interchange of the ith and $(i+1)$th element in π then $f(\pi) \leq f(\pi')$ if $g(\pi(i), \pi(i+1)) \leq g(\pi(i+1), \pi(i))$.
- The optimal solution π^* is such that ith job precedes jth job if $g(i,j) \leq g(j,i)$.

In fact Johnson's algorithm can be derived using this theorem where $g(i,j) = \min(A_i, B_j)$ and if $g(i,j) \leq g(j,i)$ i.e. $\min(A_i, B_j) \leq \min(A_j, B_i)$ then job i is scheduled before job j. Johnson's algorithm can be implemented in $O(nlogn)$ time. The same neighbourhood structure does not produce any efficient algorithm for the three-machine flow shop problem. In fact the three machine flow-shop problem is known to be $\mathcal{NP}$-complete and hence it will be unlikely that we can find any neighbourhood structure that can yield a good algorithm. It is very unlikely that one will be able to identify a good neighbourhood structure yielding efficient local search methods for the hard combinatorial optimisation problems. However, the literature has focused significantly on developing a variety of neighbourhood structures and local search methods for such hard problems.

Neighbourhood structures are defined differently for different classes of combinatorial optimisation problems. Furthermore different algorithms used to solve a particular problem will define the structure differently. In the following we briefly describe the neighbourhood structure defined for TSP and VRP.

Travelling Salesman Problem (TSP)

One important neighbourhood structure used in many algorithms to solve TSP is based on the k-opt procedure developed by Lin (1965). k-opt can be viewed as a

tour improvement procedure where the cost is calculated based on deleting k edges and then reconnecting the vertices back into a single tour. In the literature, $k = 2$ or 3 are widely used. Figures 5.1 and 5.2 show the 2-opt and 3-opt procedures.

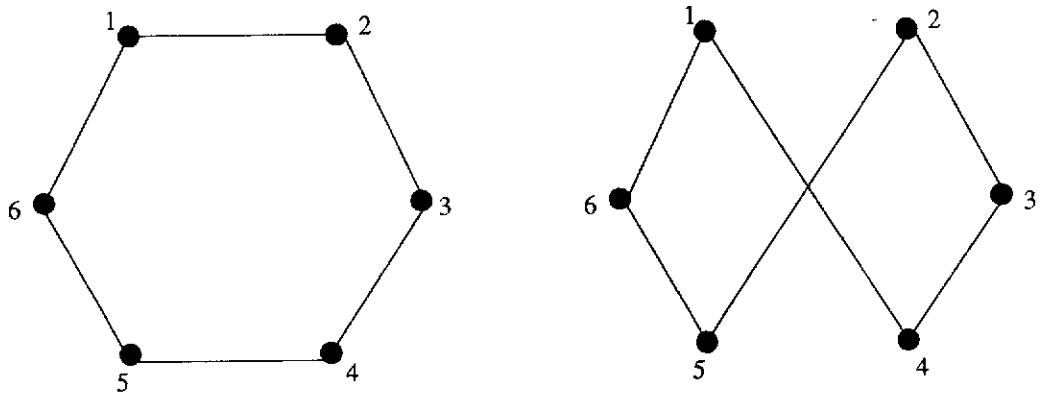


Figure 5.1: 2-Opt

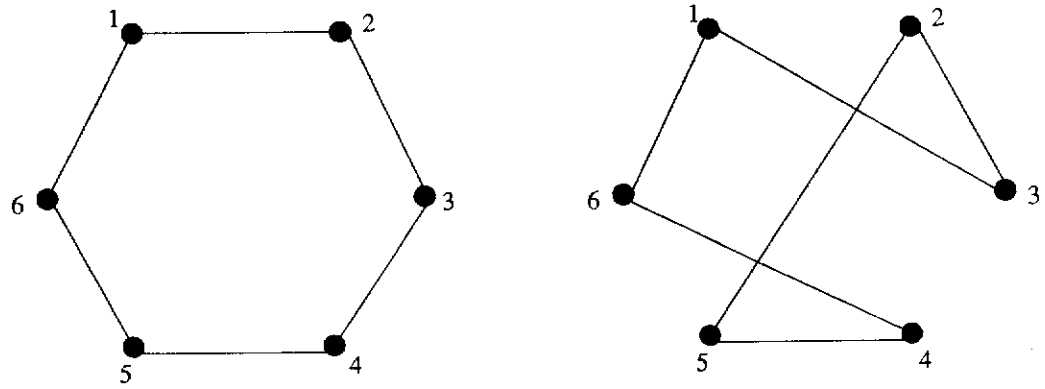


Figure 5.2: 3-Opt

For 2-opt procedure, 2 edges (1,2) and (4,5) are deleted and replaced by (2,5) and (1,4). Similarly for 3-opt procedure, three edges (1,2), (3,4) and (5,6) are deleted and replaced by (1,3), (2,5) and (4,6).

For the algorithm, the local optimum depends on the neighbourhood structure. For example, for the 2-opt procedure, if the edge (1,2) is chosen for deletion, we can choose the other edge to be deleted as (4,5) or (3,4) or (5,6) and hence obtain respectively the three neighbours as the three solutions shown by figures 5.1, 5.3a and 5.3b. Thus under 2-opt procedure, three neighbours of TSP tour τ=(1,2,3,4,5,6,1) are depicted in Figure 5.1, 5.3a and 5.3b corresponding to tours

τ'=(1,4,3,2,5,6,1), τ''=(1,3,2,4,5,6,1) and τ'''=(1,5,4,3,2,6,1) respectively.

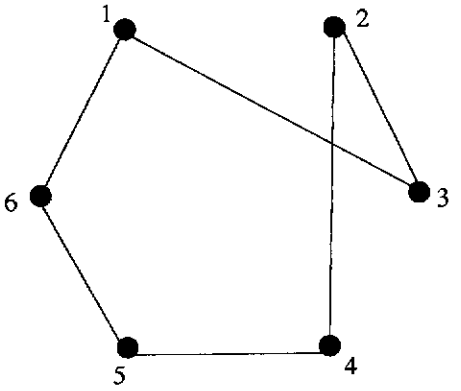


Figure 5.3a

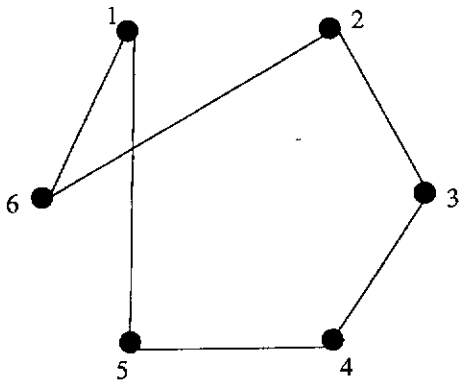


Figure 5.3b

Different neighbourhood structures can be obtained from different procedures for the same problem. Similarly under 3-opt procedure, three neighbours of TSP tour τ=(1,2,3,4,5,6,1) are depicted in Figure 5.2, 5.4a and 5.4b corresponding to tours τ'=(1,3,2,5,4,6,1), τ''=(1,5,4,2,3,6,1) and τ'''=(1,4,5,3,2,6,1) respectively. These neighbours are obtained by deleting the same edges (1,2), (3,4) and (5,6) but by inserting different sets of edges. Note that these neighbours cannot be obtained by 2-opt procedures, for the given τ=(1,2,3,4,5,6,1). For a given solution π, let the neighbourhood sets under 2-opt and 3-opt procedures be denoted respectively by $N_2(\pi)$ and $N_3(\pi)$. The relationship between $N_2(\pi)$ and $N_3(\pi)$ is not easy to study. In fact they may be unrelated and may be even disjoint. Hence the local minimum obtained by these two procedures may be different.

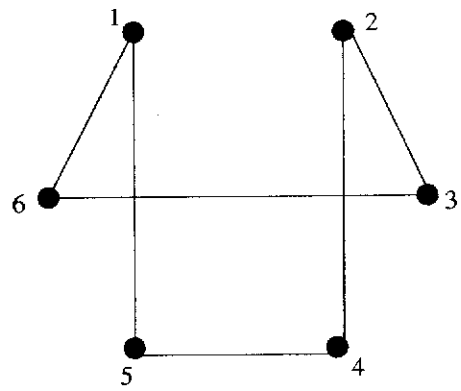


Figure 5.4a

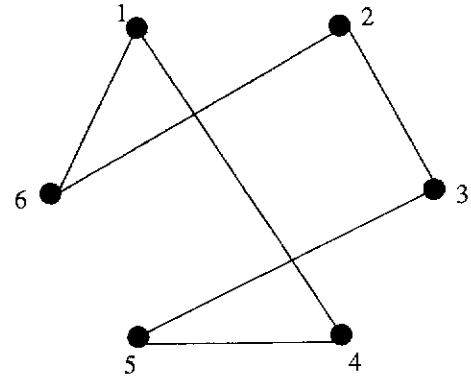


Figure 5.4b

There are variations of the k-opt used in the literature such as the Lin-Kernighan

algorithm (Lin and Kernighan, 1973), Or-opt (Or, 1976) and Genius algorithm (Gendreau et al., 1992). The Lin-Kernighan algorithm is a variation of the 2-opt and 3-opt procedures where different value of k at different iterations of the algorithm need to be determined. This procedure is more powerful than 2-opt and 3-opt procedures. Or-opt is another variation of k-opt. Genius algorithm uses a similar idea of deleting and inserting three edges and four edges which are called Type I and Type II unstringing and stringing procedures. The procedures calculate the cost of deleting and inserting the closest neighbours of a node, v say, based on a predetermined number p. The best tour is chosen to implement.

Table 5.1 shows a list of some recent local search methods based on neighbourhood structures as a variation of k-opt procedure for the TSP.

Author/(s)	*Algorithm*	*Comments*
Gendreau et al. (1992)	Genius	Procedures similar to 3-opt and 4-opt called Type I and Type II stringing and unstringing are used in the algorithm.
Malek et al. (1989)	TS	Used 2-opt as the basic move in their algorithm.
Fiechter (1994)	TS	Used 2-opt in the parallel algorithm.
Knox (1994)	TS	Used 2-opt in their algorithm.
Charttejee et al. (1996)	GA	The operations used in their procedures such as the inversion, MUT3 and mutation are similar to k-opt where k=2,3 and 4.
Tsubakitani and Evans (1998a)	TS	Used 2-opt and 3-opt in their algorithm.

Table 5.1: Examples of variations of k-opt as improvement procedures for the TSP

Vehicle Routing Problem

The neighbourhood structure of VRP can be viewed as an extension of that of TSP. There are two types of neighbourhood structures which can be defined for VRP.

1. The first one is based on exchanging edges on the same route. Procedure k-opt (normally k=2) is used to improve the tour.

2. The second type is based on exchanging edges on different routes. Three techniques are defined which include relocation, exchange and crossover.

   - A relocation can be described as a node deleted from a route and inserted into another.
   - An exchange is when two nodes from two different routes are deleted and inserted into the opposite respective routes i.e. swapping of nodes between the two routes.
   - A crossover is when the crossing links of the two routes are replaced by two non-crossing links to produce two new routes.

Table 5.2 shows a list of local search methods used by various authors in the literature where various neighbourhood structures are used. In these problems

Author/(s)	*Algorithm*	*Comments*
Christofides et al. (1979)	2-Phase	k-opt is used to improve on single route.
Ryan et al. (1993)	Petal heuristic	4-opt is used to improve on the single route.
Osman (1993)	SA	An exchange called λ-interchange by Osman is used where the number of customer is 1 or 2.
Garcia et al. (1994)	TS	Used 2-opt and Or-opt in their algorithm.
Gendreau et al. (1994)	Taburoute	Used Genius algorithm to improve the tour. This can be on the same route or different routes depending on the neighbour of node to be deleted or inserted.
Breedam (1995)	SA	Used relocation and exchange in the algorithm.
Xu and Kelly (1996)	TS	Used 3-opt to improve tour.

Table 5.2: Examples of the neighbourhood structures used for VRP

different types of neighbourhood structures have been used. Relationship between

the neighbourhood structures for a given problem is not studied. In the next section we provide a formal setup of the neighbourhood structure in an abstract form. In Chapter 8, we extend these ideas to study the complexity of the solution space in terms of a given neighbourhood structure and the corresponding local search method.

5.2 Formal Definition of Neighbourhood Structures

Recall that a combinatorial optimisation problem can be defined as (5.1). Very often it is difficult to describe the elements of S. However it can be easy to describe elements of a superset $\mathcal{S}$ of S where $\mathcal{S}$ is finite and discrete. Note that a typical element $x \in \mathcal{S}$ can be represented using mathematical notation of finite size, such as $x \in B^n$ or a system of linear equations, etc. In most real life problems, for a given $x \in \mathcal{S}$, it is easy to check the feasibility of x i.e. whether $x \in S$. The definition of neighbourhood structures we describe here are based on the VRP case. The neighbourhood structures for the TSP case can easily be derived from this. Note that for the case of TSP, the elements represented by both $\mathcal{S}$ and S are the same when we represent a solution of TSP by a permutation π of the $(n-1)$ cities excluding the home city. However, for the capacity restricted VRP (CVRP), the superset $\mathcal{S}$ represents set of all valid tours, namely feasible and infeasible. The set S represents the set of all feasible solutions. As we already discussed the set $\mathcal{S}$ can be represented by system of linear equation given by (2.2) - (2.5).

For a given solution $x \in \mathcal{S}$, we define a subset, $N(x)$, as the neighbours of x. The neighbourhood structure must satisfy the following:

Assumption 1 : $x' \in N(x)$ if and only if $x \in N(x')$ for all $x \neq x' \in \mathcal{S}$.

Assumption 2 : For any given pair of solution x and x', there exist a sequence of distinct solutions such that $x = x_1, ..., x_k = x'$ where $x_i \in N(x_{i-1})$ for $2 \leq i \leq k$.

For a well defined objective function $f(x)$ over $\mathcal{S}$ and neighbourhood structure $N(x)$ for $x \in \mathcal{S}$, we can introduce a direction between x and every $x' \in N(x)$ as follows: there exists a directed edge from x to x' if $f(x) > f(x')$. This edge indicates the direction of the required move operation for improvement in the objective function.

The solution x^* is defined to be the **local minimum of** $\mathcal{S}$ if $x^* \in \mathcal{S}$ and $f(x^*) \leq f(x)$ for all $x \in N(x^*)$. This means that there is no edge leading from x^* to x for every $x \in N(x^*)$. The solution x^* is defined to be the **local minimum of** S if $x^* \in S$ and $f(x^*) \leq f(x)$ for all $x \in N(x^*) \cap S$.

Let $f_1^*, f_2^*, ..., f_\beta^*$ be the distinct local minimum objective function values realised by the set of all local minimum points of $\mathcal{S}$. In many real life problems we have no prior knowledge regarding the size of β. A combinatorial optimisation problem with smaller β may be considered as a solvable problem, whereas a problem with larger β certainly has to depend upon other characteristics of the function f, if it is to be solved efficiently.

First note that some of the local minimum points of $\mathcal{S}$ may not be in S and hence are not feasible. Restricting our attention to S, we denote by $g_1^*, g_2^*, ..., g_\gamma^*$, the distinct local minimum objective function values realised by the set of all local minimum points of S. The given original problem, minimise $\{f(x) : x \in S\}$, can be reworded as:

$$\text{Find } x^* \in S \text{ such that } f(x^*) = g_\theta^* \text{ where } g_\theta^* = min\{g_i^* : 1 \leq i \leq \gamma\}.$$

The effectiveness of an algorithm in solving the above problem lies in how well the neighbourhood structures can be defined and explored. However, this also depends on how difficult the problem is. Thus one is interested in the performance of the algorithm on a range of problems and in particular, on 'bad' problems. In the following section we discuss the ideas of performance of the algorithms.

5.2.1 Performance of Heuristics

Different heuristics have different neighbourhood structures, running time, complexity and may even produce different solutions. Some heuristics run faster but may not produce optimal solutions often. Others take a longer time and usually produce near-optimal solutions. So one of our interests is to find out the performance of these heuristics. One way to do this is to find the bound or the ratio of the obtained solution to the optimal solution. For some heuristics, the bounds can be represented by a function of the number of nodes and for some other heuristics, the bounds can be a constant. Note that there are two types of heuristics: deterministic and probabilistic. For a given problem, a deterministic heuristic H produces the same solution f_H^* each time the heuristic is applied. Examples of these heuristics include the nearest neighbour and cheapest insertion algorithms. For the probabilistic heuristic, each application produces different solutions. Examples of these heuristics include the Genius algorithm (Gendreau et al., 1992) and Taburoute algorithm (Gendreau et al., 1994).

Since the optimality of the solution for a given heuristic algorithm H say, cannot be ensured, we are interested in finding out how close the obtained solution f_H^* is to the optimal objective function value $f(x^*)$ for a given problem. In one run of a heuristic we can obtain any one of the set of distinct objective function values $g_1^*, g_2^*, ..., g_\gamma^*$ (the values corresponding to the local optimal solutions). Very often, γ is unknown and the set of values $g_1^*, g_2^*, ..., g_\gamma^*$ is also not known. Hence the optimal objective function value $f(x^*)$ cannot be determined in the absence of $g_1^*, g_2^*, ..., g_\gamma^*$.

In order to evaluate the performance of a heuristic and to answer questions like how close is the heuristic solution to the optimal, the measures used are the magnitude of $\mid f(x^*) - f_H^* \mid$ or

$$\frac{\text{heuristic solution}}{\text{optimal solution}} = \frac{f_H^*}{f(x^*)}. \tag{5.7}$$

Table 5.3 shows a list of algorithms, the complexity and the bounds on the ratio of the heuristic solution and the optimal solution i.e. $\frac{f_H^*}{f(x^*)}$. The upper bound

on the table is with reference to expression (5.7). This bound is measured over a set of all problems i.e.

$$\max_{\{\text{set of all problems}\}} \frac{f_H^*}{f(x^*)}.$$

The sharpness of the upper bound is shown in column 3 and is sometimes referred to the worst-case performance of the heuristics. These bounds are derived from a theoretical analysis of the heuristic algorithms.

Algorithm	*Upper Bound*	*Sharp Upper Bound*	*Complexity*	*Reference*
Lin and Kernighan	-	-	$O(n^{2.2})$	Papadimitriou(1992)
nearest neighbour	$\frac{1}{2}\lceil log_2 n \rceil + \frac{1}{2}$	$\frac{1}{3}log_2(n+1) + \frac{4}{9}$	$O(n^2)$	Lenstra and Rinnooy Kan (1981)
nearest insertion	2	2	$O(n^2)$	Lenstra and Rinnooy Kan (1981)
cheapest insertion	2	2	$O(n^2 log_2 n)$	Lenstra and Rinnooy Kan (1981)
nearest addition	2	2	$O(n^2)$	Lenstra and Rinnooy Kan (1981)
convex hull	-	-	$O(n^2 log_2(n))$	Golden et al. (1980)
farthest insertion	$2ln(n) + 0.16$	-	$O(n^2)$	Golden et al. (1980)
Christofides	$\frac{3}{2}$	-	$O(n^3)$	Golden et al. (1980)
k-opt	-	-	$O(n^k)$	Lenstra and Rinnooy Kan (1981)
k optimality for $k \leq \frac{n}{4}$	-	$2(1 - \frac{1}{n})$	-	Lenstra and Rinnooy Kan (1981)

Table 5.3: Heuristic algorithms for the TSP: Complexity and upper bound on the ratio of the heuristic solutions to the optimal solution

There are situations when we need to compare the performances of two or more heuristics. Suppose we have a probabilistic heuristic H where randomness is involved and n different implementations of H provide solutions x_i^H with objective

function value $f(x_i^H)$, $1 \leq i \leq n$. The work of Gendreau et al. (1992) compares two heuristics H_1 and H_2 through the mean objective value of several trials, that is

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i^{H_1}) \text{ and } \frac{1}{n}\sum_{i=1}^{n} f(x_i^{H_2}).$$

We believe that such sample mean is not a good indicator since in any implementation of the heuristic the user will obviously prefer the best known solution, that is

$$\min_i f(x_i^{H_1}).$$

Some statistical methods used in the literature to compare the performance of various heuristic include the non-parametric tests such as Wilcoxon or Friedman tests. These tests are based on testing of hypothesis where the null hypothesis is such that 'All heuristics are performed equally well on all problems' versus the alternative hypothesis that 'All heuristics are not performing equally well on all problems'. Golden and Stewart (1985) carried out such study based on some heuristics for the TSP. The drawback of this test is that it only accepts or rejects the null hypothesis and does not give information about the heuristic that performs the best. This problem can be resolved by the expected utility function (Golden and Assad, 1984) that ranks the heuristics that perform well on an average. They applied the method for the TSP.

For a probabilistic heuristic, on a given problem over r runs, let $f_H^1, ..., f_H^r$ be the local optimal solution obtained by the r runs. Note that over the r runs, some of these solutions may appear more than once. We are interested to assess the number of times or the frequency of the occurance of the distinct local optima, $g_i^*, 1 \leq i \leq \gamma$. The performance of the heuristic can be measured by the magnitude of

$$\min_{1 \leq i \leq r} \mid f_H^i - f(x^*) \mid$$

or the magnitude of the ratio

$$\frac{\min_{1 \leq i \leq r} f_H^i}{f(x^*)} \tag{5.8}$$

as is done for the deterministic case. However (5.8) can be viewed as a random variable and for a given problem, one can estimate the objective function values of an optimal solution from the set of objective function value $f_H^i, 1 \leq i \leq r$ obtained by the r runs of the heuristic. Furthermore, a confidence interval for the objective function value of the optimal solution can be evaluated. This method was investigated by various researchers to predict an optimal solution for the combinatorial problems.

In the next chapter we demonstrate these ideas on the GENIUS algorithm applied to the TSP. In Chapter 7 we explore the use of these tools on the TABUROUTE algorithm applied to the CVRP and CDVRP.

Chapter 6

Statistical Evaluation of the GENIUS Algorithm for the Symmetric TSP

This chapter provides a statistical evaluation to validate the comparison of the performance of the heuristics discussed in Chapter 5. We investigated the ideas based on the GENIUS algorithm (Gendreau et al., 1992) developed for the TSP. The GENIUS algorithm is chosen because it is claimed to outperform known heuristics to solve TSP in terms of solution quality and computing time. An overview of the GENIUS algorithm is given in Section 6.1. The statistical tests include the Friedman test (Golden and Stewart, 1985), expected utility approach (Golden and Assad, 1984) and fitting of the Weibull distribution to the obtained solutions (Golden, 1978) discussed in Chapter 5 are extended to the 36 variations of heuristics for the literature problems and the randomly generated problems. The variation of the heuristics are based on various neighbourhood parameters, insertion methods and construction initial solution procedures. From the combination of the 36 heuristics, four new algorithms are produced. This is discussed in Sections 6.2 and 6.3. For details of statistical methods we refer to Appendix A. Some of the results of this chapter are part of our published work in Achuthan and Chong (1998).

6.1 The GENIUS algorithm

The Genius algorithm is developed by Gendreau, Hertz and Laporte (1992) for the symmetric TSP. The algorithm can be divided into two parts: GENI and US which can be described in the following.

The Generalised Insertion Procedure (GENI)

The Generalized Insertion (GENI) procedure starts with a partial tour of 3 cities and at each iteration a randomly selected city is inserted into the partial tour until a Hamiltonian tour is constructed. The notation and terminology is introduced as follows.

Given a partial (or complete) tour τ', a city v and an integer p, define

$$K_p(v,\tau') = \{u: \text{ city } u \text{ is one of the } p \text{ closest cities,in tour } \tau, \text{ to the given city } v \}. \tag{6.1}$$

Note that if $\mid \tau' \mid \geq p$ then $\mid K_p(v,\tau') \mid = p$; if $\mid \tau' \mid < p$ then $K_p(v,\tau') = \mid \tau' \mid$.

Parameter p is called the neighbourhood parameter and its value determines the extent of the search imposed by the algorithm. The larger the value of p, the more extensive the search for new tours will be. Many of the heuristics and algorithms developed for TSP and VRP use $K_p(v,\tau')$ as a basic neighbourhood set of vertex v, where elements of $K_p(v,\tau')$ play a key role in extending or searching for new tours from τ'.

Given an orientation of a partial tour τ' and a city v_i of τ' , let v_{i+1} be the next city in τ' to which the given orientation forces to travel from v_i. Given an orientation of a partial tour τ' , v not in τ' and $v_i, v_j \in K_p(v,\tau')$, let v_k be a vertex on the path from v_j to v_i and let v_l be a vertex on the path from v_i to v_j such that $v_k \in K_p(v_{i+1},\tau')$ and $v_l \in K_p(v_{j+1},\tau')$. Three types of insertion methods to insert v between v_i and v_j in for a partial tour τ' is defined as follows.

Type A Insertion: For $v_k \neq v_i$ and $v_k \neq v_j$, the set of arcs $\{(v_i,v_{i+1}), (v_j,v_{j+1})$ and $(v_k,v_{k+1})\}$ is replaced by $\{(v_i,v), (v,v_j), (v_{i+1},v_k)$ and $(v_{j+1},v_{k+1})\}$. The

orientation of the two paths $(v_{i+1}, ..., v_j)$ and $(v_{j+1}, ..., v_k)$ are reversed. Type A insertion is illustrated in Figure 6.1. For the special case when $v_{i+1} = v_j$ and $v_{j+1} = v_k$, Type A insertion reduces to the standard insertion procedure as illustrated in Figure 6.2.

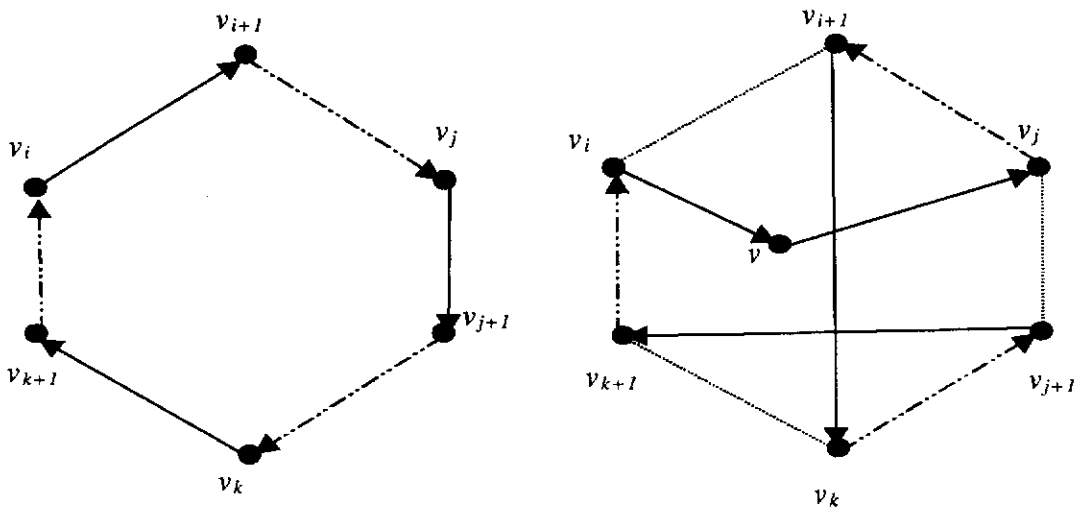


Figure 6.1: Type A Geni Insertion

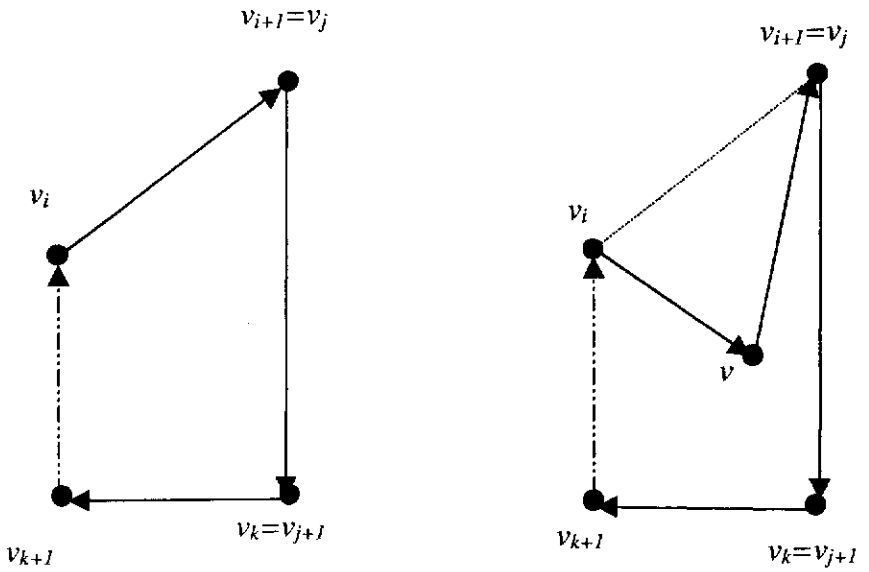


Figure 6.2: Standard insertion when $v_{i+1} = v_j$, $v_k = v_{j+1}$

Type B Insertion: For $v_k \neq v_j$ and $v_k \neq v_{j+1}$; $v_l \neq v_i$ and $v_l \neq v_{i+1}$, the set of arcs $\{(v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1})$ and $(v_{k-1}, v_k)\}$ is replaced by $\{(v_i, v)$,

(v, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) and $(v_{i+1}, v_k)\}$. The orientation of the two paths $(v_{i+1}, ..., v_{l-1})$ and $(v_l, ..., v_j)$ are reversed. This scheme of insertion is illustrated in Figure 6.3.

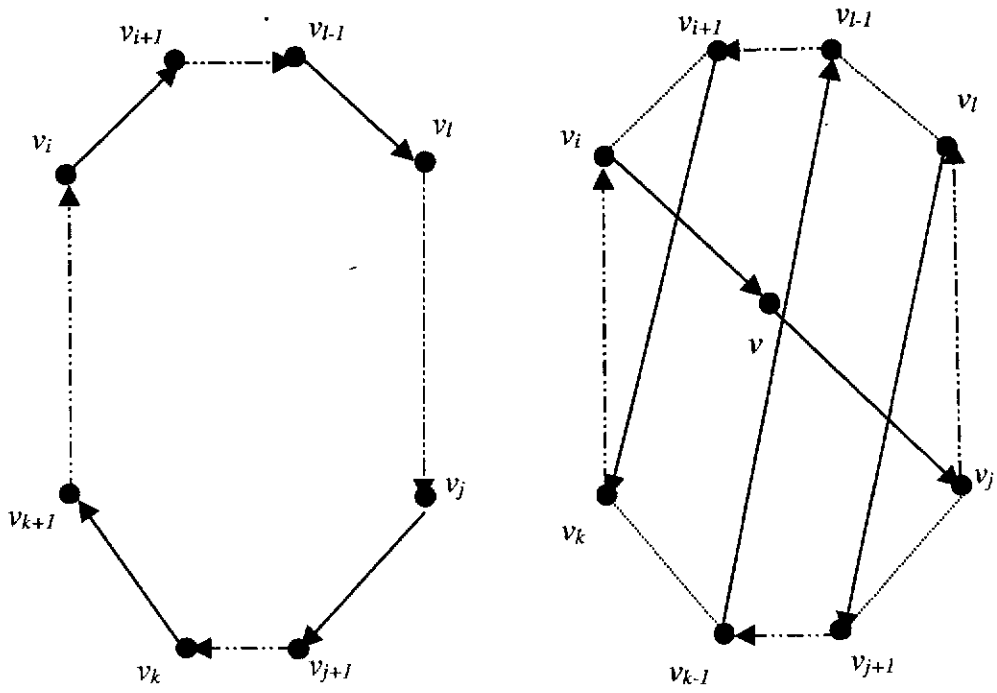


Figure 6.3: Type B Geni Insertion

Type C Insertion: We call the scheme of inserting v between v_i and v_{i+1} when $v_i \in K_p(v, \tau')$ as **Neighbour insertion**. For notational convenience Neighbour insertion by Type C insertion. This includes the special case of Type A insertion as illustrated in Figure 6.2.

The flow chart of the GENI algorithm is presented in Figure 6.4.

Unstringing and Stringing Procedure (US)

The post optimisation procedure proposed by Gendreau et al. (1992) is to remove a vertex from the tour and insert it back. The removal of vertex v_i from the tour can be accomplished by two methods called Unstringing procedures defined below.

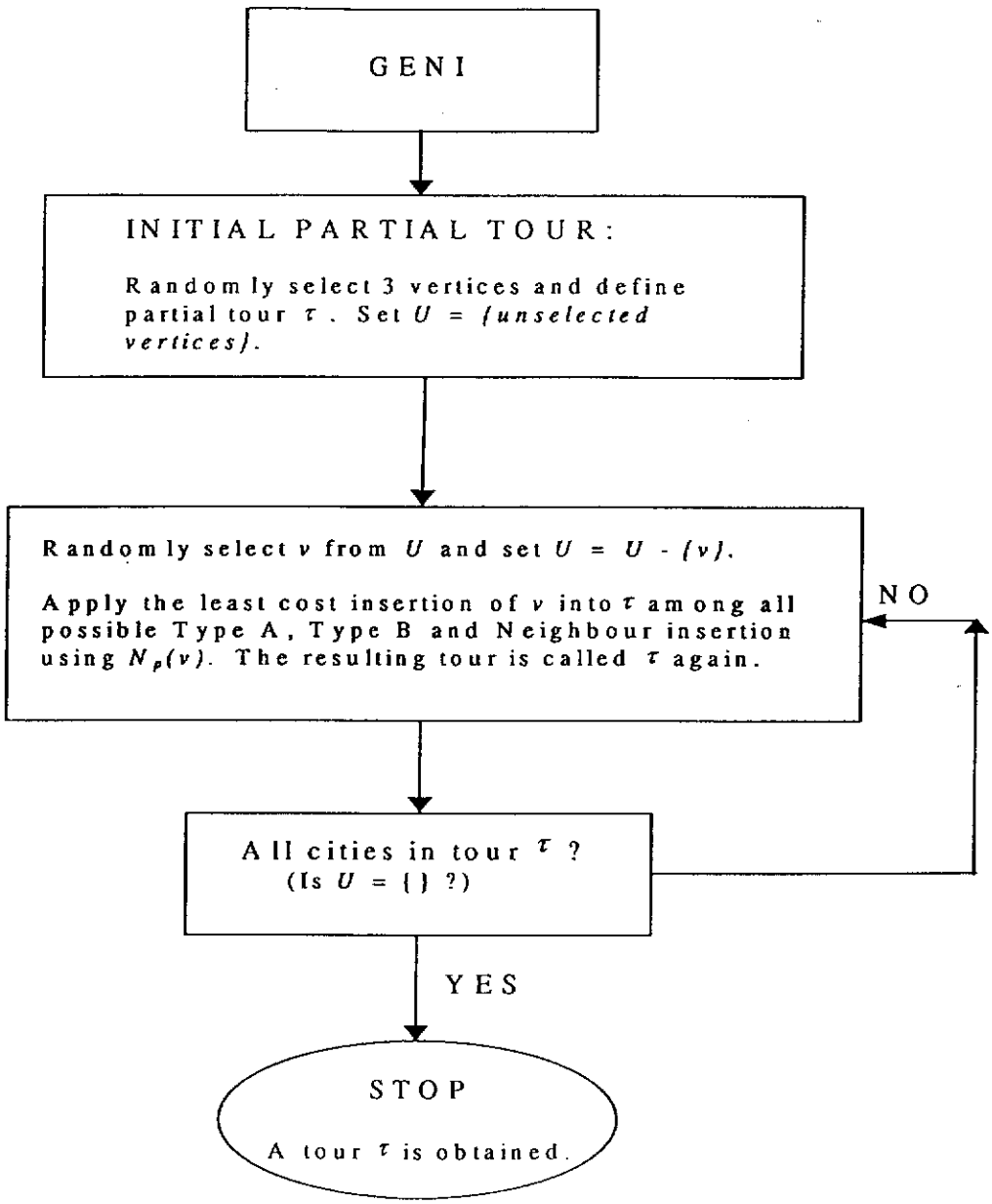


Figure 6.4: GENI Algorithm

The insertion is called the Stringing procedure that is similar to type A, type B and type C of GENI.

Type I Unstringing: For a given tour τ and its orientation, suppose that vertex v_i (or v) is to be removed, let $v_j \in K_p(v_{i+1}, \tau)$ and $v_k \in K_p(v_{i-1}, \tau)$ be a vertex on the path $(v_{i+1}, ..., v_{j-1})$. The arcs $\{(v_{i-1}, v_i), (v_i, v_{i+1}), (v_k, v_{k+1})$ and $(v_j, v_{j+1})\}$ are deleted and replaced by $\{(v_{i-1}, v_k), (v_{i+1}, v_j)$ and $(v_{k+1}, v_{j+1})\}$. The two paths $(v_{i+1}, ..., v_k)$ and $(v_{k+1}, ..., v_j)$ are reversed. The result is a new partial tour τ' without involving vertex v_i. This is shown in Figure 6.5.

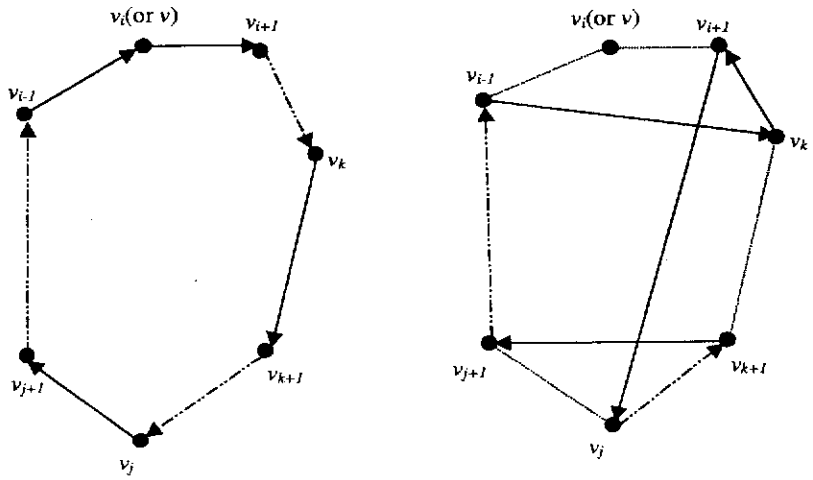


Figure 6.5: Type I Unstringing

Type II Unstringing: For a given tour τ and its orientation, let v_i (or v) be the vertex to be removed. Let $v_j \in K_p(v_{i+1}, \tau)$ and $v_k \in K_p(v_{i-1}, \tau)$ be a vertex on the path $(v_{j+1}, ..., v_{i-2})$ and let $v_l \in K_p(v_{k+1})$ be a vertex on the path $(v_j, ..., v_{k-1})$. The arcs $\{(v_{i-1}, v_i), (v_i, v_{i+1}), (v_{j-1}, v_j), (v_l, v_{l+1})$ and $(v_k, v_{k+1})\}$ are deleted and replaced by $\{(v_{i-1}, v_k), (v_{l+1}, v_{j-1}), (v_{i+1}, v_j)$ and $(v_l, v_{k+1})\}$. The two paths $(v_{i+1}, ..., v_{j-1})$ and $(v_{l+1}, ..., v_k)$ are reversed. The resulting partial tour τ' does not involve vertex v_i. This is shown in Figure 6.6.

Unstringing $(\tau, v_t; \tau', C(\tau'))$: Given a complete tour τ and a specified city v_t,

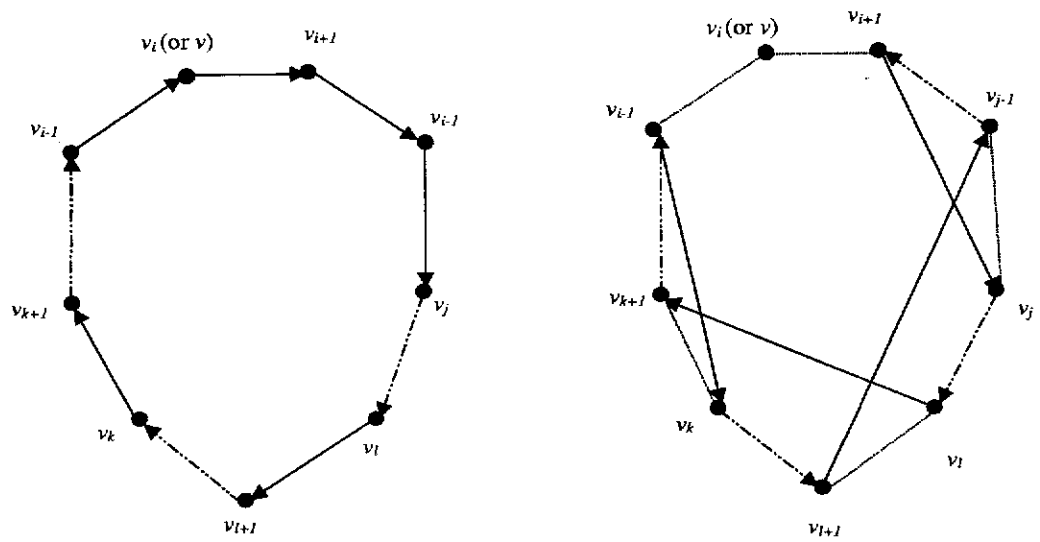


Figure 6.6: Type II Unstringing

this procedure applies the type I and type II Unstringing methods to the tour τ with city v_t for both possible orientations of τ. Then the best partial tour τ', in the sense of least cost, $C(\tau')$, is chosen.

Stringing $(\tau', v_t; \tau'', C(\tau''))$: Given a partial tour τ' and a specified city v_t not in τ', this procedure applies the type A, type B and type C insertion methods to τ' and a complete tour τ'' with least cost, $C(\tau'')$ is obtained.

The flow chart of the US procedure described in Figure 6.7 uses the above Unstringing and Stringing procedures.

Given a TSP tour τ, its neighbourhood can be considered as $K(\tau) = \{\tau' : \tau'$ can be obtained from τ by a suitable unstring and then string operation$\}$. More precisely select a customer v from τ and then obtain a partial tour τ' by deleting a customer v by either Type I unstringing or Type II unstringing. The string operation is then performed on τ' to obtain a complete permutation τ. Note that the neighbourhood structure $K(\tau)$ is understood from the context on the basis of the stringing types, unstringing types and the neighbourhood parameter p.

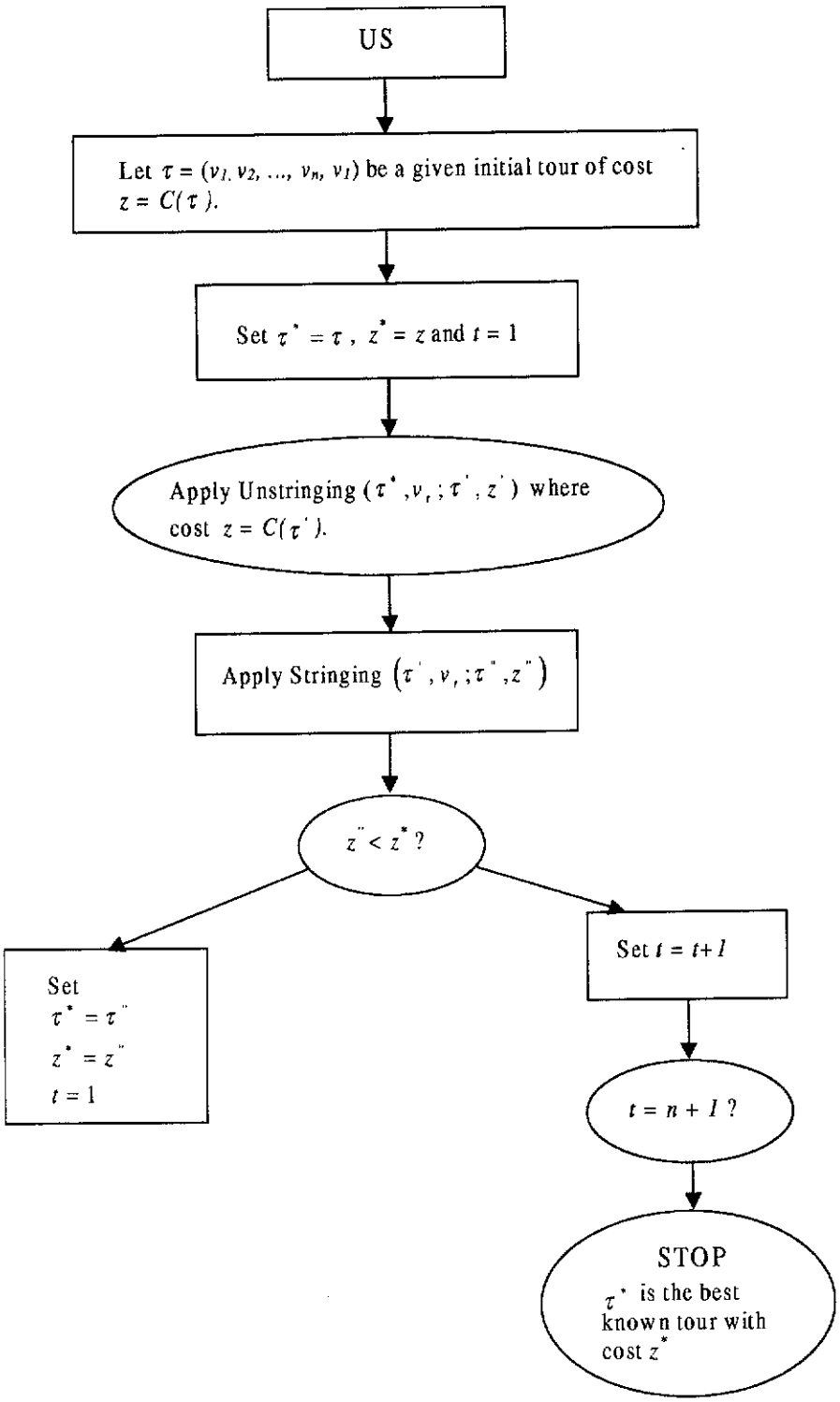


Figure 6.7: US Algorithm

The initial partial tour of GENI is made up of three randomly selected cities. In contrast to this initial partial tour, one can start GENI with any other suitable initial partial tour. Gendreau et al. (1992) have implemented GENI with the initial partial tour generated by the convex hull of the cities. This version of GENI is denoted by CGENI. The convex hull technique of generating a partial tour is used by Golden and Stewart (1985) while implementing a heuristic denoted by CCAO (convex hull, cheapest insertion, angle selection and Or-opt) (see Section 2.2 for detail). The US procedure can be applied to any given complete tour τ. Gendreau et al. (1992) also applied US procedure to the tour obtained by CGENI and named this combination as CGENIUS. Furthermore, we replace the Or-opt procedure of CCAO with US and name the new combination as CCAUS.

Several versions of the heuristics GENIUS, CGENIUS and CCAUS can be obtained by varying the Stringing types, Unstringing types and the neighbourhood parameter p. Both the Unstringing types were used in the US procedure of the different heuristics considered in this section. The various combinations of Stringing types used in the heuristics are ABC, AB, AC and BC. Note that Gendreau et al. (1992) used all three types of insertion methods in their algorithm. The different values considered for the neighbourhood parameter p are 3, 5, 7 and 9. A given heuristic GENIUS with neighbourhood parameter p and Stringing types AB will be denoted by p_GENI_AB_US. Similarly a given heuristic CGENIUS with neighbourhood parameter $p = 7$ and Stringing types ABC will be denoted by 7_CGENI_ABC_US. Thus GENIUS and CGENIUS give rise to 32 different heuristics in total. Furthermore the given heuristic CCAUS with neighbourhood parameter p is denoted by p_CCAUS. Note that p_CCAUS is a deterministic procedure in the sense that every execution of this procedure on a given problem will yield the same solution. There are 4 such heuristics. The variations of GENIUS and CGENIUS are not deterministic procedures and hence we execute them on a given problem for 100 times and take the least cost solution. All programmes for these heuristics were written in C and were executed on Silicon Graphic INDY machine with a single 100 MHZ IP22 processor with 32 Mbytes of memory size and run speed of 87.3 MIPS.

In the following sections, we carry out the statistical tests to compare these 36 heuristics.

6.2 Statistical Evaluation on the Literature Problems

In this section we carry out an analysis of the heuristics using 27 literature problems chosen from Krolak et al. (1971) and Reinelt (1995). The sizes of these problems range from 100 cities to 532 cities and the best known lower bounds are shown on Table 6.1. In order to compare the heuristics we have executed each of the 36 heuristics on every problem. For each heuristic of the types *p*_GENI_*types*_US and *p*_CGENI_*types*_US we execute it 100 times on a given problem. The least objective function value out of the 100 runs is chosen as the objective function value associated to that heuristic on the given problem. Table B.1 shows the best objective function value obtained over 100 runs for various heuristics Hj on the given problem i, i.e. $\min_{1\leq r\leq 100}(f^r_{Hj})_i$ (see Appendix B).

Note that for 3 problems on Table 6.1 we obtained similar tour but different objective function values from the literature. These solutions are presented in the bracket on Table 6.1. This may be due to different versions of C compiler and truncation of the number. We use our solution (in the bracket) for carrying out the statistical analysis.

In the following, we carried out the Friedman test to compare the 36 heuristics. Friedman test is an extension of the Wilcoxon test and it is used when comparing three or more heuristics. It is a non-parametric test similar to the ANOVA test of homogeneity.

Problem	*size n*	*Best known lower bound* $c(\tau^*)$	*Reference*
a280	280	2579	Reinelt (1995)
bier127	127	118282	Reinelt (1995)
ch130	130	6110* (6099)	Reinelt (1995)
ch150	150	6528	Reinelt (1995)
d198	198	15780	Reinelt (1995)
eil101	101	629	Reinelt (1995)
gil262	262	2378	Reinelt (1995)
lin105	105	14379* (14380)	Reinelt (1995)
lin318	318	42029	Reinelt (1995)
pr107	107	44303	Reinelt (1995)
pr124	124	59030	Reinelt (1995)
pr136	136	96772	Reinelt (1995)
pr144	144	58537	Reinelt (1995)
pr152	152	73682	Reinelt (1995)
pr226	226	80369	Reinelt (1995)
pr264	264	49135	Reinelt (1995)
pr299	299	48191	Reinelt (1995)
pr439	439	107217	Reinelt (1995)
rat195	195	2323	Reinelt (1995)
ts225	225	126643	Reinelt (1995)
tsp225	225	3919	Reinelt (1995)
u159	159	42080	Reinelt (1995)
k24	100	21282	Krolak et al. (1971)
pcb442	442	50778	Reinelt (1995)
rd100	100	7910* (7911)	Reinelt (1995)
rd400	400	15281	Reinelt (1995)
att532	532	27686	Reinelt (1995)

Table 6.1: Problems and their best known lower bounds in the literature

6.2.1 Friedman Test

Comparison of the k heuristics denoted by $H_1, ..., H_k$ can be made through a non-parametric test suggested by Friedman. We refer the details of this approach to Appendix A. We carried out the Friedman test on 3 cases:

Case 1. Compare all 36 heuristics.

Case 2. Compare the 16 *p*_CGENI_*types*_US heuristics.

Case 3. Compare the 16 *p*_GENI_*types*_US heuristics.

Our null hypothesis is that all 36 heuristics are all equal in performance. Tables B.2, B.4, and B.6 (see Appendix B) provide the ratio of the obtained heuristic solutions over the best known solution, i.e. $r_{ij} = \frac{x_{ij}}{c(\tau^*)}$ where $c(\tau^*)$ = the best known solution of problem j for various cases. For a given problem j, the ratios r_{ij} are arranged in increasing order and ranked from 1 to k. The rank of r_{ij} is

denoted by R_{ij}. Those ratios giving rise to a tie are given the average rank of the associated indices. Tables B.3, B.5, B.7 , (see Appendix B) show the ranking of the various cases and calculation of $R_j = \sum_{i=1}^{n} R_{ij}, \sum (R_{ij})^2$ and R_j^2.

The summary of the calculations for the Friedman test for the 3 cases listed above are given in Table 6.2. The table includes the calculation of the statistic A_F, B_F and the test statistic T_F with $\alpha = 0.05$ level of significance. Note that the null hypothesis that the heuristics are equally accurate is rejected for all 3 cases as $T_F > F_\alpha$. Thus we need to locate the best heuristic and further investigation of the heuristics through the expected utility approach will assist us towards this goal.

	Case 1	*Case 2*	*Case 3*
Hypothesis:	All 36 heuristics are equally accurate.	All 16 *p_CGENI_types_US* heuristics are equally accurate.	All 16 *p_GENI_types_US* heuristics are equally accurate.
	$n = 27, k = 36$	$n = 27, k = 16$	$n = 27, k = 16$
$A_F = \sum_{i=1}^{n} \sum_{j=1}^{k} (R_{ij})^2$	416181	37853	38020
$B_F = \frac{1}{n} \sum_{j=1}^{k} R_j^2$	364610.26	32258.46	32380.37
$T_F = \frac{(n-1)(B_F - (nk(k+1)^2)/4)}{A_F - B_F}$	16.10	4.86	5.38
$F_\alpha((k-1),(n-1)(k-1))$	$F_{0.05}(35, 910) = 1$	$F_{0.05}(15, 390) = 1.67$	$F_{0.05}(15, 390) = 1.67$
Conclusion:	**Reject** the null hypothesis that all 36 heuristics are equally accurate at $\alpha = 0.05$.	**Reject** the null hypothesis that all 16 *p_CGENI_types_US* heuristics are equally accurate at $\alpha = 0.05$.	**Reject** the null hypothesis that all 16 *p_GENI_types_US* heuristics are equally accurate at $\alpha = 0.05$.

Table 6.2: Computational result on Friedman test on three cases.

6.2.2 Expected Utility Approach

The Friedman test is a procedure testing of location i.e. mean or median of the distribution and nothing about the shape or the dispersion of the distribution. So the results can be unsatisfying. Another method to compare the heuristics is called the Expected Utility Approach (Golden and Assad, 1984). This approach

can be used to answer question such as which heuristic is the most accurate? We extend the idea to the 36 heuristics for the TSP. We refer details of this approach to Appendix.

Let x be the percentage deviation of the heuristic solution and the best known solution of a particular heuristic on a given problem i.e.

$$x = \frac{\text{heuristic solution - best known solution}}{\text{best known solution}} \text{x100\%}.$$

Empirically we fit a gamma function to x. The heuristic which yields the largest expected utility function is the most accurate. Golden and Assad (1984) suggested that the risk averse decreasing utility function is of the form:

$$u(x) = \gamma - \beta \exp^{tx}$$

where $\gamma, \beta, t > 0$. The expected utility function can be derived and is in the form:

$$\gamma - \beta(1 - \tilde{b}t)^{-\tilde{c}}$$

where $\tilde{b}$ and $\tilde{c}$ are the estimated parameters of the Gamma function. We select the utility function where $\gamma = 500$, $\beta = 100$ and $t = 0.05$, where t measures the risk of aversion for the utility function and $t < 1/\tilde{b}$. Note that γ and β are chosen arbitrarily.

Tables B.8 (see Appendix B) show the percentage deviation, x, from the best known solution and Table 6.3 shows the summary of the calculation required for the expected utility function. The notations used in Column 2 to 5 of Table 6.3 are $\bar{x} = \frac{1}{27}\sum_{j=1}^{27} x_j$, $s^2 = \frac{1}{27}\sum_{j=1}^{27}(x_j - \bar{x})^2$, $\tilde{b} = \frac{s^2}{\bar{x}}$ and $\tilde{c} = (\frac{\bar{x}}{s})^2$ respectively. The last column provides the rank 1 to 36 of the entries $\gamma - \beta(1 - \tilde{b}t)^{-\tilde{c}}$ of Column 6 corresponding to the largest to smallest. The results indicated that 9_CGENI_ABC_US is the most accurate of 36 heuristic followed by 5_CGENI_AB_US based on the 27 literature problems. p_CCAUS does not perform well compare to p_GENI_types_US and p_CGENI_types_US.

Heuristic	$\bar{x}$	s^2	$\tilde{b}$	$\tilde{c}$	$500 - 100(1 - 0.05\tilde{b})^{-c}$	*Rank*
3_CGENI_ABC_US	0.6287	0.3717	0.5912	1.0634	396.7573	29
3_CGENI_AB_US	0.5377	0.3902	0.7257	0.7408	397.2234	25
3_CGENI_AC_US	0.6770	0.8090	1.1948	0.5666	396.4475	31
3_CGENI_BC_US	0.6632	0.3761	0.5671	1.1693	396.5787	30
5_CGENI_ABC_US	0.5059	0.3073	0.6074	0.8329	397.3977	21
5_CGENI_AB_US	0.4055	0.2709	0.6681	0.6069	397.9164	2
5_CGENI_AC_US	0.5213	0.2890	0.5545	0.9402	397.3211	24
5_CGENI_BC_US	0.5180	0.3267	0.6307	0.8213	397.3329	23
7_CGENI_ABC_US	0.4310	0.2730	0.6334	0.6805	397.7855	7
7_CGENI_AB_US	0.4682	0.2958	0.6318	0.7411	397.5923	12
7_CGENI_AC_US	0.4195	0.2686	0.6402	0.6553	397.8448	4
7_CGENI_BC_US	0.4877	0.2941	0.6030	0.8086	397.4929	17
9_CGENI_ABC_US	**0.4020**	**0.2465**	**0.6132**	**0.6555**	**397.9372**	1
9_CGENI_AB_US	0.4405	0.2522	0.5726	0.7693	397.7398	8
9_CGENI_AC_US	0.4094	0.2615	0.6387	0.6410	397.8974	3
9_CGENI_BC_US	0.4922	0.3039	0.6174	0.7972	397.4682	18
3_GENI_ABC_US	0.5717	0.3614	0.6322	0.9042	397.0527	26
3_GENI_AB_US	0.5788	0.4130	0.7136	0.8110	397.0010	28
3_GENI_AC_US	0.5799	0.2929	0.5051	1.1480	397.0196	27
3_GENI_BC_US	0.7255	0.4760	0.6560	1.1059	396.2423	32
5_GENI_ABC_US	0.4761	0.2817	0.5915	0.8049	397.5537	14
5_GENI_AB_US	0.4850	0.3334	0.6874	0.7056	397.5012	16
5_GENI_AC_US	0.5005	0.3621	0.7235	0.6917	397.4181	20
5_GENI_BC_US	0.5161	0.3176	0.6154	0.8386	397.3442	22
7_GENI_ABC_US	0.4689	0.2975	0.6344	0.7391	397.5884	13
7_GENI_AB_US	0.4220	0.2351	0.5572	0.7574	397.8366	5
7_GENI_AC_US	0.4678	0.2963	0.6335	0.7383	397.5946	11
7_GENI_BC_US	0.4983	0.3494	0.7012	0.7107	397.4310	19
9_GENI_ABC_US	0.4469	0.2862	0.6404	0.6978	397.7029	9
9_GENI_AB_US	0.4214	0.2601	0.6172	0.6827	397.8364	6
9_GENI_AC_US	0.4543	0.3449	0.7591	0.5985	397.6567	10
9_GENI_BC_US	0.4758	0.3519	0.7395	0.6434	397.5458	15
3_CCAUS	2.5762	2.7403	1.0636	2.4219	385.8476	36
5_CCAUS	2.0184	2.3661	1.1722	1.7218	389.0396	35
7_CCAUS	1.9587	1.9199	0.9802	1.9983	389.4366	34
9_CCAUS	1.7789	1.6005	0.8997	1.9772	390.4718	33

Table 6.3: Calculations for the expected utility function for the literature problems

6.2.3 Fitting of Weibull Distribution (Literature Problems)

In this section we will evaluate the heuristic solutions using the statistical extreme value theory, Weibull distribution, which was proved by Fisher and Tippet (1928) (cited in Golden, 1978) to obtain an estimate of the optimal objective function value and an estimate of the confidence interval for the problems. Given a probabilistic heuristic implemented several times on a given problem, let $z_{i1}, z_{i2}, ..., z_{iq}$

be the observed local minimum objective function values corresponding to q runs of ith batch of implementation. Thus the best known objective function value in the ith implementation can be denoted by $z_i^{'} = \min\{z_{i1}, z_{i2}, ..., z_{iq}\}$. Furthermore if the probabilistic heuristic is implemented r times then we have the best known objective function value of this problem by heuristic as $v = \min\{z_i^{'}, 1 \leq i \leq r\}$. By the extreme value theory, irrespective of the distribution of the objective function value, as q increases the random variable Z' (representing the minimum of a random sample of size q) follows Weibull distribution. We refer to Appendix A for further details on this subject.

We fitted the Weibull distribution to the solutions on the following problems: a280, bier127, ch130, ch150, d198, eil101, gil262, lin318, pr136, pr144, pr152, pr226, pr264, pr299, pr439, rat195, ts225, tsp225, u159, pcb442, rd400 and att532. Note that problems lin105, pr107, pr124, k24 and rd100 are left out because the heuristics obtained the best known solution as reported in the literature. For a given heuristic and a given problem, we take 10 independent samples, each of size 10. More specifically, in terms of implementing the heuristic, it is equivalent to running the heuristic 10 times and obtaining the best objective function value z_i, for ith run, $1 \leq i \leq 10$. Thus let $v = min\{z_i \mid 1 \leq i \leq 10\}$. We adapted the least square approach by Golden (1977, 1978) to solve the following equation:

$$c\ln(x_0 - a) - c\ln b = \ln(-\ln(1 - F(x_0))). \tag{6.2}$$

The equation (6.2) can be derived from the cumulative Weibull distribution by taking the logarithm twice. For different values of the parameter, 'a', say a_i, we use the least square method (6.2) to estimate b_i and c_i for the corresponding parameters 'b' and 'c'. The set of parameters that yields the smallest Kolmogorov Smirnov (K-S) statistic D is selected because it is much more sensitive to small changes in 'a' compared to selecting a set of parameter which yields the largest correlation coefficient.

The test statistic with modified form of D by Stephens (1974) (cited in Scheaffer and McClave, 1990) is used

$$T = D\left[\sqrt{n} + 0.12 + \frac{0.11}{\sqrt{n}}\right]. \tag{6.3}$$

The rejection starts at 1.358 for $\alpha = 0.05$ significance level.

The results show that final estimated solutions are within 8.2% of the best-known solutions for *p*_CGENI_*types*_US and 6.5% for *p*_GENI_*types*_US. The K-S statistic fall below the critical value 1.358 for 0.05 level of significance except for one case for d198 and four cases for u159. Table 6.4 shows the number of times that the estimated confidence interval included the best-known solution. Table 6.5 shows a list of various problems with the corresponding best heuristic and the estimated optimal objective. The heuristic and the estimate are chosen using the smallest $\frac{\tilde{b}}{v}$ among the various heuristics used. Note that $\frac{\tilde{b}}{v}$ is a measure of performance and the smaller it is the more powerful the corresponding heuristic is.

Heuristic	3_AB	3_ABC	3_AC	3_BC	5_AB	5_ABC	5_AC	5_BC
CGENIUS	19	19	20	20	18	20	19	19
GENIUS	17	18	17	16	19	17	17	19

Heuristic	7_AB	7_ABC	7_AC	7_BC	9_AB	9_ABC	9_AC	9_BC
CGENIUS	18	18	19	17	20	20	19	17
GENIUS	18	18	19	20	17	19	20	18

Table 6.4: Number of times the estimated confidence interval include the best known lower bound on the 22 literature problems with various heuristics

6.2.4 Conclusion

In conclusion, the *p*_CCAUS model does not perform well compared to the *p*_CGENI_*types*_US and the *p*_GENI_*types*_US. From the result of the expected utility function test, we learn that 9_CGENI_ABC_US appears to be the best heuristic among the 36 heuristics. Alternatively we can choose heuristic 5_CGENI_AB _US for quality of the solution and computational times.

Finally we include the number of distinct local optimal solutions $g_i^*, 1 \leq i \leq \gamma$ generated for each problem for each heuristic. Tables 6.6 and 6.7 show the number of distinct solutions obtained, i.e. γ, over the 100 runs for the *p*_CGENI_*types*_US and the *p*_GENI_*types*_US. These tables illustrate that the number of local min-

Problem	*Heuristic*	*Estimated optimal objective*	*Best known lower bound*	*Deviation*	$\frac{\tilde{b}}{v}$	*K-S statistic*
a280	5_CGENI_ABC_US	2604.00	2579	0.9%	0.0110	0.0837
att532	5_CGENI_AC_US	27916.00	27686	0.8%	0.0064	0.0830
bier127	9_CGENI_AB_US	118340.00	118282	0.04%	0.0066	0.2706
ch130	7_CGENI_BC_US	6112.00	6099	0.2%	0.0015	0.0893
ch150	7_CGENI_ABC_US	6552.00	6528	0.4%	0.0040	0.0840
d198	7_CGENI_BC_US	15787.00	15780	0.05%	0.0025	0.0840
eil101	7_CGENI_BC_US	635.00	629	0.95%	0.0124	0.0862
gil262	9_CGENI_BC_US	2393.00	2378	0.6%	0.0110	0.0830
lin318	3_CGENI_AB_US	42629.00	42029	1.4%	0.0052	0.0830
pcb442	5_CGENI_AB_US	51331.00	50778	1.1%	0.0065	0.0829
pr144	5_CGENI_AB_US	58565.00	58537	0.05%	0.0016	0.0861
pr152	9_CGENI_AB_US	73646.00	73682	-0.05%	0.0045	0.2488
pr226	9_CGENI_AC_US	80368.00	80369	-0.001%	0.0001	0.0972
pr264	5_CGENI_AC_US	49125.00	49135	-0.02%	0.0023	0.0857
pr299	3_CGENI_AB_US	48281.00	48191	0.18%	0.0035	0.0829
pr439	5_CGENI_BC_US	107836.00	107217	0.6%	0.0063	0.0830
rat195	9_CGENI_AC_US	2339.00	2323	0.7%	0.0072	0.2495
rd400	5_CGENI_AB_US	15412.00	15281	0.9%	0.0106	0.0835
ts225	7_CGENI_BC_US	126490.00	126643	-0.12%	0.0101	0.2488
tsp225	9_CGENI_ABC_US	3963.00	3919	1.1%	0.0067	0.0886
u159	5_CGENI_AC_US	41824.00	42080	-0.6%	0.0160	0.2555
a280	9_GENI_AC_US	2565.00	2579	-.5%	0.0235	0.1019
att532	7_GENI_ABC_US	27979.00	27686	1.1%	0.0052	0.0886
bier127	5_GENI_AC_US	118373.00	118282	0.07%	0.0018	0.3764
ch130	9_GENI_AB_US	6049.00	6099	-0.8%	0.0120	0.0895
ch150	7_GENI_AB_US	6505.00	6528	-0.35%	0.0120	0.1191
d198	9_GENI_AB_US	15737.00	15780	-0.3%	0.0041	0.1215
eil101	9_GENI_ABC_US	633.00	629	0.6%	0.0162	0.1351
gil262	3_GENI_BC_US	2370.00	2378	-0.33%	0.0260	0.1294
lin318	7_GENI_AB_US	42324.00	42029	0.7%	0.0047	0.3302
pcb442	3_GENI_BC_US	51678.00	50778	1.8%	0.0054	0.1389
pr136	5_GENI_AC_US	97107.00	96772	0.3%	0.0030	0.1484
pr144	9_GENI_AB_US	58529.00	58537	-0.01%	0.0013	0.1379
pr152	9_GENI_ABC_US	73632.00	73682	-0.07%	0.0030	0.3905
pr226	9_GENI_BC_US	80368.00	80369	-0.001%	0.0001	0.1320
pr264	9_GENI_AC_US	49085.00	49135	-0.1%	0.0023	0.1062
pr299	7_GENI_AC_US	48238.00	48191	0.09%	0.0036	0.1276
pr439	9_GENI_BC_US	108030.00	107217	0.7%	0.0037	0.0934
rat195	5_GENI_AB_US	2312.00	2323	-0.5%	0.0262	0.4000
rd400	5_GENI_AB_US	15448.00	15281	1.1%	0.0065	0.3271
ts225	5_GENI_AC_US	126593.00	126643	-0.03%	0.0073	0.1967
tsp225	9_GENI_AC_US	3936.00	3919	0.4%	0.0153	0.1047
u159	9_GENI_AB_US	42030.00	42080	-0.1%	0.0052	0.5741

Table 6.5: Computational results for the literature problems - Point estimation for p_CGENI_*types*_US and p_GENI_*types*_US chosen using the smallest performance measure

imum is highly dependent on neighbourhood structures. Some of the problems, e.g. bier127, seem to produce a large proportion of distinct local minimum solutions which indicate this may be hard problem with too many local minimum. So no algorithm will solve such problems in a reasonable time.

	3_*CGENI*_types_*US*				5_*CGENI*_types_*US*				7_*CGENI*_types_*US*				9_*CGENI*_types_*US*			
Problem	*AB*	*ABC*	*AC*	*BC*	*AB*	*ABC*	*AC*	*BC*	*AB*	*ABC*	*AC*	*BC*	*AB*	*ABC*	*AC*	*BC*
a280	72	66	71	66	63	72	69	66	67	66	64	67	71	65	65	63
bier127	97	97	100	94	96	94	96	93	93	92	90	87	93	90	93	84
ch130	82	84	75	76	70	73	73	68	57	67	63	65	60	60	67	72
ch150	75	80	83	78	77	77	83	79	76	82	72	70	71	76	78	74
d198	92	86	89	90	89	86	92	82	89	90	93	92	81	81	82	82
eil101	24	25	27	27	22	24	22	24	22	24	22	22	20	22	22	23
gil262	54	62	66	61	56	61	58	58	58	54	58	59	60	55	57	59
lin105	68	68	67	73	15	17	19	19	14	11	12	12	9	10	10	10
lin318	96	93	97	97	95	96	97	96	94	97	99	93	95	94	97	95
pr107	47	41	48	59	17	21	19	17	17	16	15	15	10	11	11	11
pr124	25	37	37	41	20	17	18	19	13	14	13	19	10	13	10	14
pr136	94	95	98	97	84	85	88	94	75	80	92	92	76	80	78	81
pr144	94	93	98	94	54	57	70	58	25	32	33	36	17	14	19	20
pr152	91	85	87	83	65	53	60	54	64	61	66	69	42	36	42	32
pr226	94	98	98	98	58	64	65	60	47	56	53	55	30	38	44	34
pr264	89	90	93	93	77	72	76	78	58	62	61	65	63	66	66	70
pr299	97	97	98	99	97	98	100	97	95	97	97	93	96	90	94	93
pr439	100	99	100	98	99	100	100	98	98	98	100	99	100	99	100	99
rat195	60	58	58	64	59	63	60	65	54	54	60	60	57	54	63	62
ts225	79	84	78	86	54	58	47	50	38	47	46	48	43	44	41	44
tsp225	70	75	75	76	68	70	72	70	63	65	63	59	66	63	60	61
u159	93	99	99	95	70	73	79	70	54	57	76	59	56	57	71	56
k24	41	43	49	34	11	13	20	17	10	8	16	9	6	7	18	10
pcb442	98	99	98	99	97	94	98	95	98	95	90	94	98	96	100	100
rd100	65	65	70	66	41	43	44	41	32	34	33	35	28	25	33	30
rd400	85	85	85	86	84	92	87	84	85	86	89	81	81	85	88	89
att532	92	92	91	92	91	97	94	91	93	95	90	93	86	100	92	91

Table 6.6: Number of distinct solutions, γ, using various p_CGENI_*types*_US heuristics

Problem	3_GENI_types_US				5_GENI_types_US				7_GENI_types_US				9_GENI_types_US			
	AB	*ABC*	*AC*	*BC*	*AB*	*ABC*	*AC*	*BC*	*AB*	*ABC*	*AC*	*BC*	*AB*	*ABC*	*AC*	*BC*
a280	67	68	69	73	66	48	48	46	71	73	68	71	71	72	72	74
bier127	99	98	99	98	91	59	57	58	97	97	99	98	95	97	95	95
ch130	80	82	81	81	75	45	50	48	66	73	70	63	61	63	66	67
ch150	87	83	86	86	59	72	76	76	75	74	78	72	68	72	76	83
d198	90	94	98	96	86	90	89	83	78	81	83	90	76	74	81	82
eil101	26	22	26	26	21	26	26	23	22	19	19	23	22	26	24	24
gil262	61	55	60	63	63	57	59	56	57	58	59	55	59	60	58	59
lin105	73	79	83	78	24	32	29	29	19	18	27	25	24	20	25	22
lin318	98	97	96	100	94	98	98	94	93	96	93	96	100	94	95	96
pr107	45	54	64	55	21	21	24	18	15	16	17	18	10	11	11	11
pr124	46	77	67	65	27	41	35	40	11	15	17	20	16	15	17	19
pr136	98	97	98	96	86	93	94	94	86	80	86	90	86	80	85	85
pr144	88	95	97	99	56	64	62	58	23	35	34	35	13	20	24	26
pr152	93	94	94	96	63	57	67	64	55	70	59	61	30	38	36	44
pr226	87	96	94	92	43	56	58	71	44	55	51	56	29	36	37	42
pr264	91	93	95	93	74	77	80	74	59	66	73	59	78	79	64	78
pr299	98	96	97	98	99	100	93	98	94	95	95	99	94	97	97	96
pr439	98	99	100	99	98	99	100	99	99	99	99	99	99	99	100	100
rat195	57	60	63	62	63	61	58	61	56	55	58	58	55	57	57	55
ts225	86	89	85	89	50	69	56	58	44	55	57	57	51	57	50	56
tsp225	68	79	77	70	68	65	66	63	64	62	65	59	54	61	60	62
u159	87	90	94	89	64	61	86	60	48	58	58	63	40	41	56	52
k24	33	50	33	36	13	19	24	17	13	12	15	13	12	9	14	9
pcb442	94	97	95	99	100	99	99	98	97	99	97	98	96	97	95	97
rd100	67	67	69	74	42	41	51	43	33	34	30	42	29	29	28	32
rd400	88	86	86	87	89	86	89	86	88	88	89	90	92	82	82	90
att532	95	96	98	92	100	92	94	93	95	92	92	89	100	99	100	99

Table 6.7: Number of distinct solutions, γ, using various p_GENI_*types*_US heuristics

6.3 Statistical Evaluation on the Randomly Generated Problems

We are interested in analysing the heuristics using the randomly generated problems since the literature problems may not be a typical real-life problem set. In this section we generated 20 Euclidean test problems with sizes ranging from 100 to 480 in increments of 20. Each of the 32 heuristics (i.e. p_CGENI_*types*_US and p_GENI_*types*_US) is executed on each problems 100 times and one time on p_CCAUS. Table 6.8 shows the best obtained solution of 100 runs from 36 heuristics for each problem. The best solution obtained for each heuristic over 100 runs is shown in Table B.9 in Appendix B.

Problem	*size n*	*Best obtained solution* $c(\tau*)$
p1	100	7758
p2	120	8541
p3	140	8866
p4	160	9418
p5	180	10111
p6	200	10595
p7	220	11696
p8	240	11969
p9	260	12229
p10	280	12612
p11	300	13203
p12	320	13215
p13	340	14019
p14	360	14516
p15	380	14865
p16	400	15042
p17	420	15133
p18	440	15896
p19	460	16172
p20	480	16137

Table 6.8: Randomly generated problems and best obtained solutions from 36 heuristics

Similar statistical tests, i.e. Friedman test, expected utility approach and fitting Weibull distribution to the obtained heuristic solutions are carried out for the randomly generated problems. We present the results in the following and refer the details of these tests to the previous section and Appendix A.

6.3.1 Friedman Test

We carried out the Friedman test on 3 cases:

Case 1. Compare all 36 heuristics.

Case 2. Compare the 16 *p*_CGENI_*types*_US heuristics.

Case 3. Compare the 16 *p*_GENI_*types*_US heuristics.

Our null hypothesis is that the heuristics are all equal in performance versus the alternate one. Tables B.10, B.12, and B.14 (see Appendix B) provide the ratio for various heuristics and cases. The ranking and calculations for Friedman test of the heuristics are shown on Tables B.11, B.13, B.15. The summary of the calculation for Friedman test for the 3 cases listed are given in Table 6.9 where T_F is the test statistic at $\alpha = 0.05$ level of significance. From the table we note that the null hypothesis is rejected for all 3 cases. That is we reject that the heuristics are all equal in their performance. Thus we need to locate the best heuristic and this can be attained through the expected utility approach.

	Case 1	*Case 2*	*Case 3*
Hypothesis:	All 36 heuristics are equally accurate.	All 16 *p*_CGENI_*types*_US heuristics are equally accurate.	All 16 *p*_GENI_*types*_US heuristics are equally accurate.
	$n = 20, k = 36$	$n = 20, k = 16$	$n = 20, k = 16$
$A_F = \sum_{i=1}^{n}\sum_{j=1}^{k}(R_{ij})^2$	320934	29582.5	29408
$B_F = \frac{1}{n}\sum_{j=1}^{k} R_j^2$	280802	24779.48	24815.65
$T_F = \frac{(n-1)(B_F-(nk(k+1)^2)/4)}{A_F-B_F}$	16.28	6.56	7.01
$F_\alpha((k-1),(n-1)(k-1))$	$F_{0.05}(35,646) = 1$	$F_{0.05}(15,285) = 1.67$	$F_{0.05}(15,285) = 1.67$
Conclusion:	**Reject** the null hypothesis that all 36 heuristics are equally accurate at $\alpha = 0.05$.	**Reject** the null hypothesis that all 16 *p*_CGENI_*types*_US heuristics are equally accurate at $\alpha = 0.05$.	**Reject** the null hypothesis that all 16 *p*_GENI_*types*_US heuristics are equally accurate at $\alpha = 0.05$.

Table 6.9: Computational result on Friedman test on three cases for the 20 random generated problems

6.3.2 Expected Utility Function

With the rejection of the null hypothesis from the Friedman test, we carried out the expected utility procedure to define the best heuristic. Let x be the percentage deviation from the heuristic solution and the best known solution of a particular heuristic on a given problem i.e.

$$x = \frac{\text{heuristic solution - best known solution}}{\text{best known solution}} \text{x} 100\%.$$

Empirically we fit a gamma function to x. The heuristic which yields the largest expected utility function is the most accurate. Golden and Assad (1984) suggested that the risk averse decreasing utility function is of the form:

$$u(x) = \gamma - \beta \exp^{tx}$$

where $\gamma, \beta, t > 0$. The expected utility function can be derived and is in the form:

$$\gamma - \beta(1 - \tilde{b}t)^{-\tilde{c}}$$

where $\tilde{b}$ and $\tilde{c}$ are the estimated parameters of the Gamma function.

We select the utility function where $\gamma = 500$, $\beta = 100$ and $t = 0.05$ where t measures the risk of aversion for the utility function and $t < \frac{1}{b}$. Note that γ and β are chosen arbitrarily. Table B.16 shows the percentage deviation, x (see Appendix B). Table 6.10 shows the summary of the calculation required for the expected utility function. The result shows that 9_CGENI_AB_US is the best heuristic among the 36 heuristics follow by 9_CGENI_ABC_US. The heuristics p_CCAUS do not perform well for the randomly generated problems. Note that p_CGENI_*types*_US and p_GENI_*types*_US are more sophisticated heuristics and require longer running time than p_CCAUS.

6.3.3 Fitting of Weibull Distribution (Randomly Generated Problems)

In this section, an estimate solution and a confidence interval for each estimate solution for each of the randomly generate problem is obtained. This is done

Heuristic	$\bar{x}$	s^2	$\tilde{b}$	$\tilde{c}$	$500 - 100(1 - 0.05\tilde{b})^{\tilde{c}}$	*Ranking*
3_CGENI_ABC_US	0.7886	0.1022	0.1887	2.8692	395.9466	29
3_CGENI_AB_US	0.5415	0.2374	0.3010	2.6193	397.2418	23
3_CGENI_AC_US	0.6966	0.1723	0.2473	2.8164	396.4333	27
3_CGENI_BC_US	0.7360	0.1714	0.2329	3.1601	396.2289	28
5_CGENI_ABC_US	0.5517	0.1336	0.2852	1.6417	397.1905	24
5_CGENI_AB_US	0.4683	0.0968	0.1754	3.1438	397.6132	15
5_CGENI_AC_US	0.5370	0.1989	0.3704	1.4497	397.2527	22
5_CGENI_BC_US	0.4924	0.1016	0.2064	2.3850	397.4941	16
7_CGENI_ABC_US	0.4433	0.1033	0.2606	1.5215	397.7423	11
7_CGENI_AB_US	0.3966	0.1264	0.2852	1.5539	397.9839	5
7_CGENI_AC_US	0.4160	0.0882	0.2119	1.9627	397.8864	8
7_CGENI_BC_US	0.3962	0.0684	0.1726	2.2945	397.9903	4
9_CGENI_ABC_US	0.4459	0.0608	0.2246	1.2059	397.7322	12
9_CGENI_AB_US	**0.2708**	**0.1012**	**0.2269**	**1.9649**	**398.6288**	**1**
9_CGENI_AC_US	0.3089	0.0686	0.2223	1.3892	398.4346	2
9_CGENI_BC_US	0.4670	0.1333	0.2855	1.6357	397.6199	14
3_GENI_ABC_US	0.8312	0.0882	0.1394	4.5404	395.7294	31
3_GENI_AB_US	0.6330	0.2036	0.2449	1.6794	396.7729	26
3_GENI_AC_US	0.7979	0.2255	0.2827	1.5006	395.9001	30
3_GENI_BC_US	0.8552	0.2108	0.2465	1.7223	395.6034	32
5_GENI_ABC_US	0.5230	0.2039	0.3937	1.3154	397.3348	21
5_GENI_AB_US	0.5179	0.1182	0.2260	1.1001	397.3498	19
5_GENI_AC_US	0.5205	0.1303	0.2503	1.0403	397.3362	20
5_GENI_BC_US	0.6167	0.1091	0.1769	1.4662	396.8539	25
7_GENI_ABC_US	0.4184	0.0850	0.2121	1.8879	397.8676	10
7_GENI_AB_US	0.4006	0.1386	0.3313	0.7269	397.9659	6
7_GENI_AC_US	0.5106	0.1146	0.2245	1.0777	397.399	17
7_GENI_BC_US	0.5178	0.1103	0.2130	1.1218	397.3628	18
9_GENI_ABC_US	0.4106	0.0718	0.2055	1.6998	397.9146	7
9_GENI_AB_US	0.3494	0.0862	0.2100	0.8958	398.2282	3
9_GENI_AC_US	0.4171	0.1382	0.3313	0.7247	397.8743	9
9_GENI_BC_US	0.4545	0.0955	0.2101	0.9915	397.6889	13
3_CCAUS	2.6818	0.7195	0.2682	5.1777	385.5461	36
5_CCAUS	2.1479	0.5463	0.2543	4.2589	388.5854	35
7_CCAUS	2.0928	0.7901	0.3775	3.4060	388.8577	34
9_CCAUS	1.9155	0.6013	0.3139	3.4187	389.865	33

Table 6.10: Expected utility function for the randomly generated problems

by fitting the Weibull distribution to the 32 heuristics (*p*_CGENI_*types*_US and *p*_GENI_*types*_US) for the 20 randomly generated problems. For a given heuristic and a given problem, we take 10 independent samples, each of size 10 and let $v = min\{z_i \mid 1 \leq i \leq 10\}$. We estimate the parameters $\tilde{a}$, $\tilde{b}$ and $\tilde{c}$ by solving equation (6.2) using the least square approach. The set of parameters that yields the smallest Kolmogorov Smirnov (K-S) statistic D is chosen. The modified form of D by Stephens (1974) (cited in Scheaffer and McClave, 1990) as shown in equation (6.3) is used and the rejection starts at 1.358 at $\alpha = 0.05$ significance level. We refer to the Appendix A for details of this subject.

The final estimated solutions are within 6.5% for both *p*_CGENI_*types*_US and *p*_GENI_*types*_US of the best solution. The K-S statistic falls below the critical value (1.358) at $\alpha = 0.05$ level of significance. Table 6.11 shows the number of times that the estimated confidence interval included the best-known solution. Table 6.12 shows the estimated optimal for the various problems chosen using the smallest $\frac{\tilde{b}}{v}$ among the various heuristics used. The value $\frac{\tilde{b}}{v}$ represents the measure of the interval width and the smaller it is, the powerful the heuristic is.

Heuristic	3_AB	3_ABC	3_AC	3_BC	5_AB	5_ABC	5_AC	5_BC
CGENIUS	20	20	20	20	19	20	20	20
GENIUS	20	20	20	20	20	20	20	20

Heuristic	7_AB	7_ABC	7_AC	7_BC	9_AB	9_ABC	9_AC	9_BC
CGENIUS	20	20	20	20	20	20	20	20
GENIUS	20	20	20	20	20	20	20	20

Table 6.11: Number of times the estimated confidence interval includes the best obtained lower bound on 20 randomly generated problems

We have successfully estimated a solution and the confidence interval for each of the 20 randomly generated problems.

6.3.4 Conclusion

In conclusion, the heuristic 9_CGENI_AB_US is the best heuristic followed by 9_CGENI_ABC_US for the randomly generated problems as indicated by the ex-

Problem	*Heuristic*	*Estimated optimal objective*	*Best known lower bound*	*Deviation*	$\frac{\tilde{b}}{v}$	*K-S statistic*
p1	9_CGENI_BC_US	7757.00	7758	-0.01%	0.0009	0.0886
p2	3_CGENI_ABC_US	8504.00	8541	-0.4%	0.0150	0.0832
p3	7_CGENI_AC_US	8846.00	8866	-0.2%	0.0102	0.0832
p4	9_CGENI_AB_US	9448.00	9418	0.3%	0.0021	0.0837
p5	5_CGENI_AB_US	10260.00	10111	1.5%	0.0075	0.0852
p6	7_CGENI_ABC_US	10671.00	10595	0.7%	0.0083	0.0832
p7	7_CGENI_ABC_US	11775.00	11696	0.6%	0.0036	0.0852
p8	7_CGENI_AB_US	12071.00	11969	0.8%	0.0029	0.0906
p9	7_CGENI_ABC_US	12209.00	12229	-0.2%	0.0157	0.0830
p10	3_CGENI_BC_US	12668.00	12612	0.4%	0.0082	0.0832
p11	5_CGENI_ABC_US	13150.00	13203	-0.4%	0.0206	0.0830
p12	9_CGENI_BC_US	13251.00	13215	0.3%	0.0112	0.0830
p13	9_CGENI_AB_US	14108.00	14019	0.6%	0.0061	0.0840
p14	3_CGENI_AC_US	14606.00	14516	0.6%	0.0089	0.0836
p15	3_CGENI_AC_US	14927.00	14865	0.4%	0.0064	0.0838
p16	5_CGENI_AC_US	15211.00	15042	1.1%	0.0077	0.0843
p17	5_CGENI_BC_US	15199.00	15133	0.4%	0.0104	0.0835
p18	9_CGENI_AC_US	16011.00	15896	-0.7%	0.0057	0.0836
p19	3_CGENI_BC_US	16292.00	16172	0.7%	0.0072	0.0831
p20	7_CGENI_BC_US	16149.00	16137	0.07%	0.0101	0.0830
p1	7_GENI_ABC_US	7756.00	7758	-0.02%	0.0007	0.0853
p2	9_GENI_AB_US	8535.00	8541	-0.07%	0.0050	0.0867
p3	5_GENI_AC_US	8855.00	8866	-0.1%	0.0056	0.0850
p4	7_GENI_AC_US	9436.00	9418	0.2%	0.0063	0.0887
p5	3_GENI_AB_US	10117.00	10111	0.06%	0.0172	0.0842
p6	9_GENI_AC_US	10642.00	10595	0.4%	0.0090	0.0838
p7	3_GENI_AB_US	11809.00	11696	1.0%	0.0060	0.0830
p8	7_GENI_AB_US	12084.00	11969	1.0%	0.0031	0.0832
p9	9_GENI_AC_US	12333.00	12229	0.85%	0.0053	0.0843
p10	7_GENI_AB_US	12620.00	12612	0.06%	0.0090	0.2416
p11	9_GENI_AB_US	13291.00	13203	0.7%	0.0076	0.0831
p12	7_GENI_AC_US	13325.00	13215	0.8%	0.0072	0.0832
p13	9_GENI_AC_US	14085.00	14019	0.5%	0.0074	0.0832
p14	3_GENI_AC_US	14699.00	14516	1.3%	0.0042	0.0843
p15	5_GENI_AC_US	14944.00	14865	0.5%	0.0047	0.0830
p16	3_GENI_AC_US	15202.00	15042	1.1%	0.0135	0.0831
p17	9_GENI_AC_US	15251.00	15133	0.8%	0.0044	0.0847
p18	5_GENI_ABC_US	15997.00	15896	0.6%	0.0042	0.0839
p19	9_GENI_BC_US	16163.00	16172	-0.06%	0.0093	0.0833
p20	9_GENI_AB_US	16171.00	16137	0.2%	0.0075	0.0832

Table 6.12: Computational results for the randomly generated problems - Point estimation for p_CGENI_*types*_US and p_GENI_*types*_US chosen using the smallest performance measure

pected utility approach. Note that the heuristic p_CCAUS does not perform well for the randomly generated problems.

6.4 Partial Enumeration Heuristic for the Travelling Salesman Problem

In this section we propose a hybrid heuristic algorithm for the TSP based on the Branch and Bound method and the Genius algorithm by Gendreau et al. (1992). Since Genius algorithm claimed to outperform known alternative heuristics to solve TSP in terms of solutions quality and computational time to large size problems (less than 532 cities). Therefore one is tempted to use the algorithm in every node of the search tree with the hope that it might provide an approximate solution closer to the lower bound of that node. With this motivation, we proposed an algorithm based on the following features:

- The partial enumeration scheme based on the branch and bound method.
- A modified Genius algorithm to solve the restricted relaxed problem at every node of the search tree.
- 1-tree relaxation method (Held and Karp (1970, 1971)) for constructing a lower bound at every node of the search tree.

We discuss the branch and bound method, 1-tree relaxation and the procedures required for the algorithm in the following.

Branch and Bound is a partial enumeration scheme which searches for an optimal solution by partitioning the set of all TSP tours throughout a search tree. Every node in the search tree corresponds to a subset of solutions and the node is represented by a set of included and excluded edges of the tours. The method is based on four important features viz:

1. The bounding method associates a lower bound to a node of the search tree

such that any feasible solution of that node has objective function value greater than or equal to the lower bound.

2. The branching method partitions the set of feasible solutions of a node into subsets associated with the children of this node.

3. A node may be fathomed if it is known that it will not produce any better solution than the current best known solution.

4. Updating the best-known solution in hand in the case of a better solution being obtained by locating the optimal solution of the restricted subset of feasible solutions of a node of the search tree.

Minimum cost 1-tree problem is used to find a lower bound (LB) for each subproblem. The problem can be solved using Prim's algorithm. First, find the minimum cost spanning tree in $V = \{1, 2, ..., n\} \setminus \{1\}$ and then find the two edges with least cost and join them to vertex 1. This method was first suggested by Held and Karp (1970, 1971). Johnson et al. (1996) demonstrated that the Held and Karp lower bound produces good quality near optimal solutions. For details of this approach we refer to Section 2.2. The problem $L(\pi^*) = \max_{\pi}\{L(\pi)\}$ is solved by using the subgradient method (Nemhauser and Wolsey (1988)). Note that $L(\pi)$ is defined as

$$L(\pi) = min \left\{ \sum_{i<j} (c_{ij} + \pi_i + \pi_j) x_{ij} - 2 \sum_{i} \pi_i : x_{ij} \text{ satisfies (2.12) – (2.15)} \right\} \quad (6.4)$$

and $\pi_i, 1 \leq i \leq n$ are the lagrangean multipliers associated to the degree constraints in (2.16).

Subgradient Method to find $L(\pi^*)$

Step 1. Let k be the iteration number and set $\pi = \pi_k$. Find $L(\pi_k)$ by solving the minimum cost 1-tree (6.4) for $\pi = \pi_k$.

Step 2. If the optimal tour T is found or the objective function value of T is greater than the upper bound, i.e. $z(T) \geq UB$, stop.

Otherwise set $\pi_{k+1}^i = \pi_k^i + t_k(d_k^i - 2), i \in V$ where t_k is the step length and is defined by $t_k = \dfrac{\alpha(UB - L(\pi_k))}{\sum_{i \in V}(d_k^i - 2)^2}$, $0 < \alpha \leq 2$. d^i is the degree of vertex i in T.

Set $k = k + 1$ and go back to Step 1.

Branching procedure: Consider the cost matrix $C = ((c_{ij}))$ associated to this node of the search tree. Note that in this matrix the excluded edges have the entries as '∞' and included edges have the least cost entry in that row or column. For convenience the cells with least cost entries in every row or column is called a zero cell. Thus every row and column has at least one zero cell. For every zero cell (i, j) not yet included, an extra cost e_{ij} of not using the cell (i, j) is computed as follows:

$$e_{ij} = \min_{k \neq j} c_{ik} + \min_{l \neq i} c_{lj}.$$

The edge (r, s), among the not yet included zero cells, is chosen such that $e_{rs} = \max\{e_{ij} : (i, j) \text{ is not yet included zero cell}\}$. The edge (r, s) is used for branching i.e. to create branches from the current node of the search tree.

Inclusion/Exclusion procedure: The edge (r, s) selected from the Branching procedure is used to branch and generate the next two child nodes. In the Inclusion procedure, the cost matrix is updated according to the following. Let c'_{ij} be the modified matrix of c_{ij} such that $c'_{rj} = \infty$ for all $j \neq s$, $c'_{is} = \infty$ for all $i \neq r$, $c'_{sr} = \infty$ and $c'_{ij} = c_{ij}$ for all other (i, j). In the Exclusion procedure, the cost matrix is updated according to the following: $c'_{rs} = \infty$ and $c'_{ij} = c_{ij}$ for all other (i, j).

We used the **best first search** technique in the algorithm. Two child nodes from the parent node are produced and the LB of each child nodes is calculated. If $LB \geq UB$ then the node is **fathomed**; Otherwise the child node is inserted into the search list such that the smallest LB is at the front of the search list.

We outline the proposed algorithm in the following.

Step 1. Initialisation of variables. Set A to be the set of active nodes in the

search tree.

Step 2. If A is empty, stop. Otherwise choose the first node from A and obtain the LB based on the 1-tree relaxation and subgradient method.

Step 3. Decide which edge to branch upon based on the Branching Procedure.

Step 4. Generate the new node. Update the cost matrix with the edge chosen from Step 3 for the Inclusion and Exclusion Procedure accordingly.

Step 5. Update the nodes in A.

Step 6. If A is empty, stop. Otherwise select the first node from the tree for branching, go to Step 3.

The procedure described is an exact algorithm. The flow chart of this algorithm is presented in Figure 6.8. But in the implementation we stop the execution when the number of nodes generated by the search tree exceeds a preset limit or CPU time exceeds a preset time.

The Inclusion and Exclusion Procedure is outlined in the following.

Step 1. Update the cost matrix with the edge selected from the Branching Procedure.

Step 2. Check for the change in condition of the 1-tree solution in the sense that addition of new edges inclusion or exclusion has changed the condition for 1-tree solution compared to the parent node. If the 1-tree solution has not changed then update the LB for the current node. Otherwise find the new lower bound using the 1-tree relaxation and the subgradient method.

Step 3. Check for condition if lower bound > best known objective function value ? If it is not true, perform the Genius algorithm (with the hope of getting a better solution) and return to main program. Otherwise return to the main program.

The flow chart of this routine is presented in Figure 6.9.

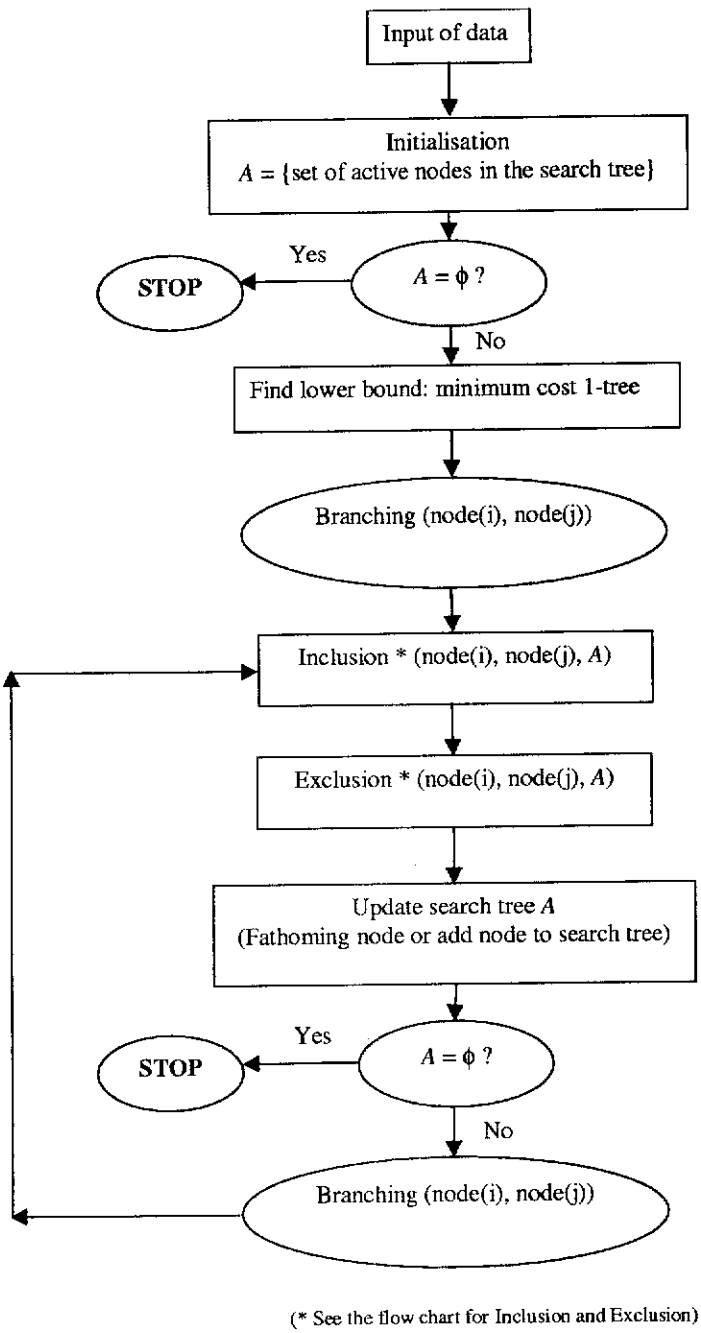


Figure 6.8: Branch and Bound and GENIUS Algorithms (*See the flow chart for Inclusion and Exclusion)

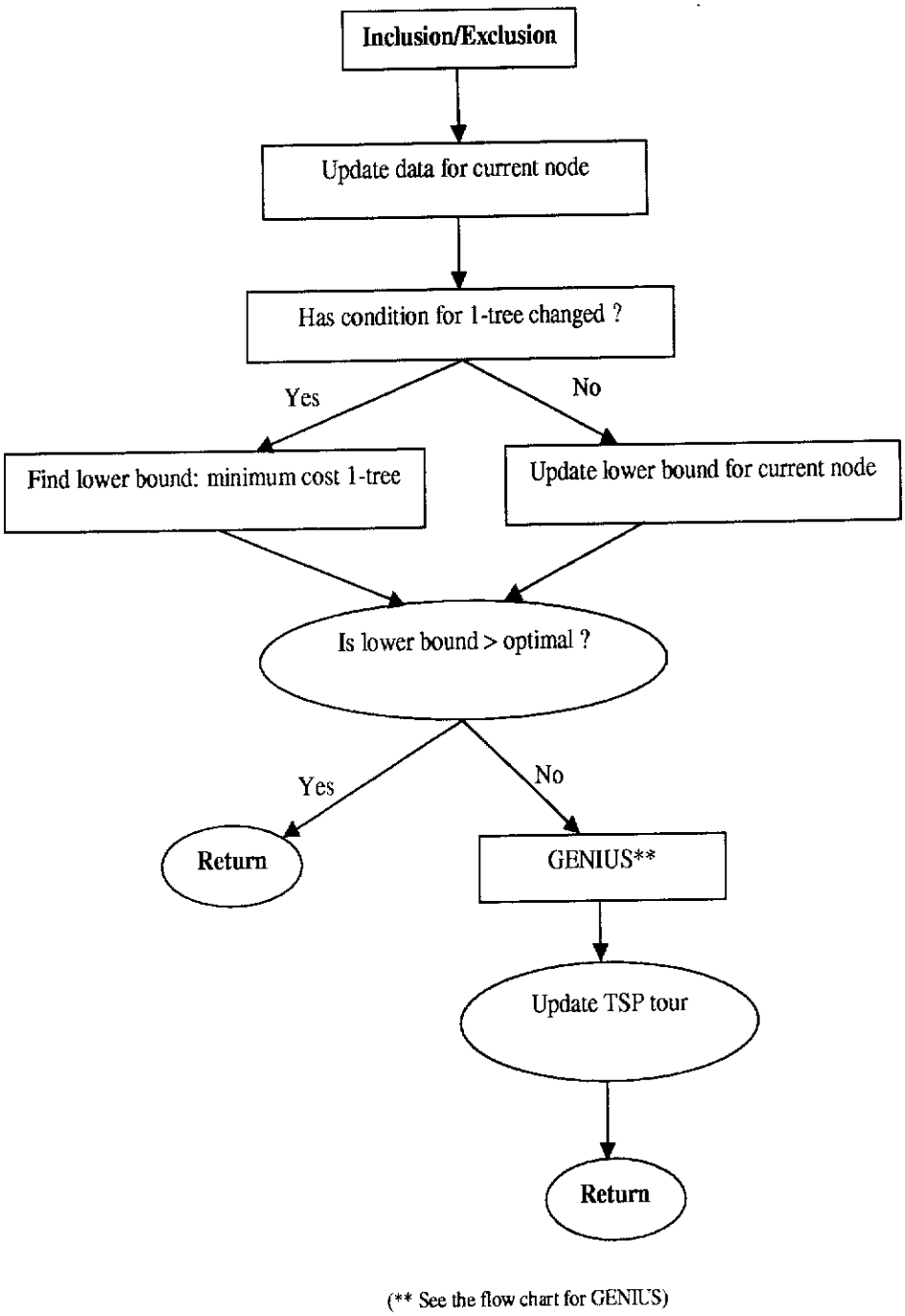


Figure 6.9: Inclusion/Exclusion Procedure (**See the flow chart for GENIUS algorithm)

We refer to Section 6.1 for the Genius algorithm. The modified Genius algorithm applied to the restricted problem at a node of the search tree can be briefly outlined as follows:

1. From the fixed edges obtain a finite number of disjoint paths denoted by say $P_1, P_2, ..., P_l$.

2. Suppose that the path $P_j = (v_j^1, ..., v_j^r)$. Replace path P_j by edge (v_j^1, v_j^r) with the cost of this edge equal to the cost of path P_j i.e. the sum of the costs of edges in P_j. For each path P_j and each vertex v not in any of the paths $P_i, 1 \leq i \leq l$, we will retain edges (v, v_j^1) and (v, v_j^r) provided they are not excluded edges at this node. Furthermore we eliminate the edges $(v, v_j^i), 2 \leq i \leq r-1$.

3. For any pair of vertices u and v not in paths $P_j, 1 \leq j \leq l$, we retain the edge (u, v) provided it is not excluded.

4. Apply the usual Genius algorithm to this reduced graph with the slight modification that the edge (v_j^1, v_j^r) will never be chosen for the unstring operation.

The flow chart of this modified Genius algorithm is presented in Figure 6.10 .

6.4.1 Computational Results

The proposed algorithm is tested on two sets of problems namely the literature problems from TSPLIB and the randomly generated problems. Note that the program is written in C and run on the Silicon Graphic machine at 195MHZ.

The Literature Problems

Six problems with size ranging between 500 - 800 cities from the TSPLIB are selected. In addition we include the following:

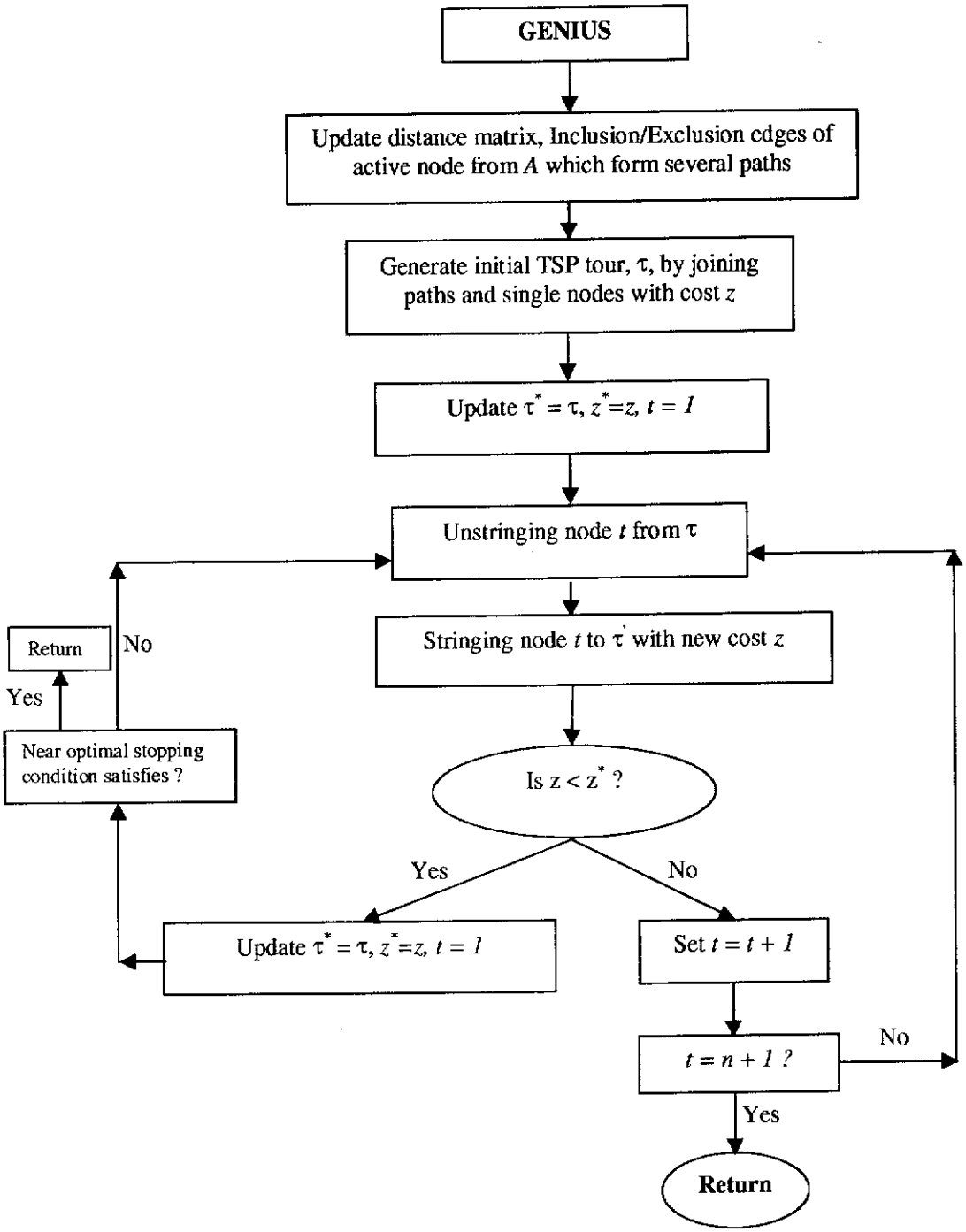


Figure 6.10: Modified GENIUS Algorithm

- Several tours are obtained using the modified Genius algorithm at the Root node and the best tour is chosen for upper bound (UB).

The result is shown in Table 6.13 with maximum running time set to 10 hours. The algorithm produced solutions to within 2.4% of the best-known solution.

Problem	*Solution at Root node*	*Improved Solution*	*Best known Solution*	*Percentage Deviation*	*Number of child node generated*
d657	50388	49596	48912	1.3	5
p654	37309	35167	34643	1.5	8
rat575	6999	6917	6773	2.1	7
rat783	9192	8952	8806	1.6	2
u574	37868	-	36905	1.6	16
u724	43093	42566	41910	2.4	8

Table 6.13: Computational results for the proposed algorithm on literature problems

Randomly Generated Problems

The proposed algorithm is also tested on the randomly generated problems. The problems are generated using the exponential distribution. The nodes are randomly generated such that the density of nodes is reduced as the distance from the depot increases. The depot is taken to be at the origin in the plane. The probability density function (p.d.f) of the exponential distribution is

$$f(x) = \lambda \exp^{-\lambda x}, 0 < x < \infty, \lambda > 0 \tag{6.5}$$

and the cumulative distribution function (c.d.f) can be obtained as

$$F(r) = \int_0^r \lambda \exp^{-\lambda x} dx = 1 - \exp^{-\lambda r} \tag{6.6}$$

Let

$$u = 1 - \exp^{-\lambda r}.$$

Then after rearranging, we have

$$r = -\tfrac{1}{\lambda} ln(1-u)$$

where $u \in (0,1)$ and r represents the radius for the location on the Cartesian coordinates. The problem is generated as follows:

Step 1. Randomly select a number between 0 and 1 for u and calculate the radius $r = -\frac{1}{\lambda} ln(1-u)$.

Step 2. Randomly select an angle $\theta \in (0, 2\pi)$.

Step 3. Convert (r, θ) into the x-y coordinate form where $x = rcos\theta$ and $y = rsin\theta$.

Note that we used $\lambda = 10, 20, 30$ and generated 9 problems with sizes from 500 - 700. Each of these problems is allowed to run for 10 hours. The following table shows the results of the randomly generated problems.

Problem	*Size of* λ	*Objective function value*		*Time (Seconds)*	*Max Iteration*	*Node* No.
		Best Solution using Genius	*Improve-ment*			
500	10	1207	1197(1)	1021.93	56	112
600	10	1408(0)	-	-	25	50
700	10	1545	1541(1)	3160.32	18	36
500	20	2068(0)	-	-	47	94
600	20	2483	2463 2461(2)	1941.16 3426.00	26	52
700	20	2504(0)	-	-	17	33
500	30	2919	2902 2890(14)	1172.52 10790.90	49	98
600	30	3603(0)	-	-	28	55
700	30	3490(0)	-	-	17	34

Table 6.14: Computational results for the proposed algorithm on randomly generated problems

Column 2 provides the value of λ. Column 3 refers to the best solution obtained using Genius algorithm at the root node. For every improvement of solution it is recorded in Column 4 and the time (in seconds) is recorded in Column 5 when the improvement is made. Columns 3 and 4 show the best solution obtained for the problem and in brackets the number of iterations taken to yield the best-known solution. The last column shows the number of child nodes generated by the algorithm for various problems.

In conclusion we note that some improvement is found for the randomly generated problems, but no new best solutions are found for the literature problems. This is perhaps an indication of the difficulty of these problems. We conclude the algorithm is unsuitable for finding optimal solutions.

In the next chapter, we continue the investigation of the ideas presented in Chapter 5 to TABUROUTE algorithm, a heuristic developed for the VRP.

Chapter 7

TABUROUTE Algorithm and The Values of Its Parameters

We have emphasised in Chapter 5 the need for statistical analysis in any probabilistic neighbourhood search method for a $\mathcal{NP}$-hard combinatorial optimisation problem. In this chapter we illustrate such analysis for a TABUROUTE algorithm proposed for VRP.

The probabilistic neighbourhood search algorithm, by its inherent nature, does not have the ability to reproduce the same solution for different runs of the algorithm on the same input of a problem. Thus while evaluating a probabilistic algorithm it is essential to seek answers to some of the following questions:

- For the given problem, how far is the solution produced by the algorithm from an optimal solution ?
- Can a confidence interval be developed using the solutions produced by the algorithm ?
- How does the algorithm perform on randomly generated problems of prescribed size ?

Many authors have used the following scheme to favour their algorithm. They demonstrate the closeness of solutions produced by the algorithm to the best known solution on a set of literature hard problems. Often these authors do not

talk about the reproducibility of their solution or the number of implementations used to generate their "good" solution.

For example, Xu and Kelly (1996) tested their algorithm on a set of benchmark problems and compared their solutions with the closeness of 5% and 1% of the best known solution and analysed the CPU time taken to construct such solutions. Rochat and Taillard (1995) tested their algorithm on a set of benchmark problems and compared the average of the five runs with the closeness of 5%, 2% and 1% of the best known solution.

We feel this method of comparison is unscientific and in fact may be fruitless. One may never face the so called hard problems of literature in real life. Furthermore, there is no guarantee that an algorithm which performed satisfactorily on a hard problem will perform at least equally satisfactorily on another randomly chosen problem.

Thus, as discussed in Chapter 5, we need to have a scheme of comparing algorithms in a more scientific way. We illustrate this through choosing optimal values of parameters for the TABUROUTE algorithm. We choose TABUROUTE algorithm for our demonstration since it is a probabilistic neighbourhood search method for VRP and also uses the GENIUS algorithm of Chapter 6 as a subroutine. Furthermore, extensive testing of the parameters for TABUROUTE is not done by Gendreau et al. (1994) and there is the need to verify this statistically. The statistical tools used in Chapter 6 on TSP are extended to CVRP and CDVRP and tested on both literature and randomly generated problems.

We describe the TABUROUTE algorithm by Gendreau et al. (1994) in Section 7.1. Comparisons of various heuristics of the TABUROUTE algorithm on literature and randomly generated problems using statistical tools are performed respectively in Section 7.2 and 7.3. Some of the results of this Chapter are published in Achuthan and Chong (1999a).

7.1 The TABUROUTE Algorithm

The TABUROUTE algorithm was developed by Gendreau et al. (1994) for the capacity and distance restricted VRP. The initial VRP solution is constructed from the TSP tour produced by the GENIUS algorithm (Gendreau et al., 1992). The algorithm incorporates the characteristics of the tabu search (TS) method and GENIUS algorithm to form a powerful heuristic. Relevant notation is discussed in the following.

Define S to be the VRP solution of m routes $R_1, ..., R_m$ where m ranges between 1 and an upper bound $\bar{m}$. Note that $R_r = (0, i_1^r, ..., i_{k_r}^r, 0)$ is a route that is a cycle passing through the depot '0' in G. The vertex i representing a customer belongs to exactly one route $R_r, 1 \leq r \leq m$. For a feasible solution S, a cost $F_1(S)$ can be defined as follows:

$$F_1(S) = \sum_{r=1}^{m} \sum_{(i,j) \in R_r} c_{ij} \tag{7.1}$$

where c_{ij} is the cost (or distance) matrix. For any feasible or infeasible solution, a cost $F_2(S)$ is associated and can be defined as follows:

$$F_2(S) = F_1(S) + \alpha \sum_{r=1}^{m} \left[\left(\sum_{i \in R_r} q_i \right) - Q \right]^+ + \beta \sum_{r=1}^{m} \left[\left(\sum_{(i,j) \in R_r} c_{ij} + \sum_{i \in R_r} \delta_i \right) - L \right]^+ \tag{7.2}$$

where α and β are two positive parameters and $[x]^+ = \max(0, x)$. The demand and service time of each customer i is denoted by q_i and δ_i respectively. Recall that Q and L are the prescribed vehicle capacity and prescribed upper limit on the length of each route.

If S is feasible then $F_1(S)$ and $F_2(S)$ are identical; Otherwise $F_2(S)$ consists of penalty terms for excessive vehicle capacity and/or route duration. Let S^* be the best known feasible solution and $\tilde{S}^*$ be the best known feasible or infeasible solution. Define $F_1^* = F_1(S^*)$ is the objective function value of the best known feasible solution and $F_2^* = F_2(\tilde{S}^*)$ is the objective function value of the best known solution, feasible or infeasible. In the following we describe the TABUROUTE algorithm developed by Gendreau et al. (1994).

TABUROUTE Algorithm

TABUROUTE algorithm starts by constructing a TSP tour using the GENIUS algorithm (Gendreau et al., 1992). The obtained TSP tour is then used to construct an initial solution for VRP solution. All the VRP routes start from the depot '0'. Typically the first route starts with the first city on the TSP tour and include cities up to where the capacity or distance restriction is not violated. The process is continued until all the cities are included in the routes (the solution is then feasible) or until $\bar{m} - 1$ vehicles are used and all the remaining cities are assigned to route $\bar{m}$ (in this case the solution may be infeasible). The obtained solution S is then passed on to procedure S1: SEARCH where better solution is searched for $\lambda = \frac{\sqrt{n}}{2}$ times (n refers to the number of cities). The best solutions S^* and $\tilde{S}^*$ are recorded and referred to as First Solution. The best solution S^* is then passed to an improvement step (S2: SEARCH). The vertices that are frequently moved and produced good solutions are again considered in an intensification step, S3: SEARCH. Figure 7.1 presents the flow chart for the TABUROUTE algorithm.

SEARCH Procedure

The SEARCH procedure is the heart of the TABUROUTE algorithm. The procedure incorporates the GENIUS algorithm along with the TS characteristics and forms a powerful heuristic for VRP. Figure 7.2 represents the flow chart of the SEARCH procedure.

In Step 1 of the procedure, a list of q vertices are randomly chosen from W where $W = V - \{0\}$. The size of q is fixed at $5m$ for S1 and S2 and at $\lfloor \frac{|V|}{2} \rfloor$ for S3. For each vertex v from the list of q vertices, the cost of removing v from its current route R_r and adding it to a new route R_s is calculated. The new route R_s has at least p closest neighbours of v. If no such new route exists then create a new route R_s with v. The new solution is defined to be S'. If this move is tabu and if the conditions in Step 2b are satisfied, then solution S' is retained. If the conditions

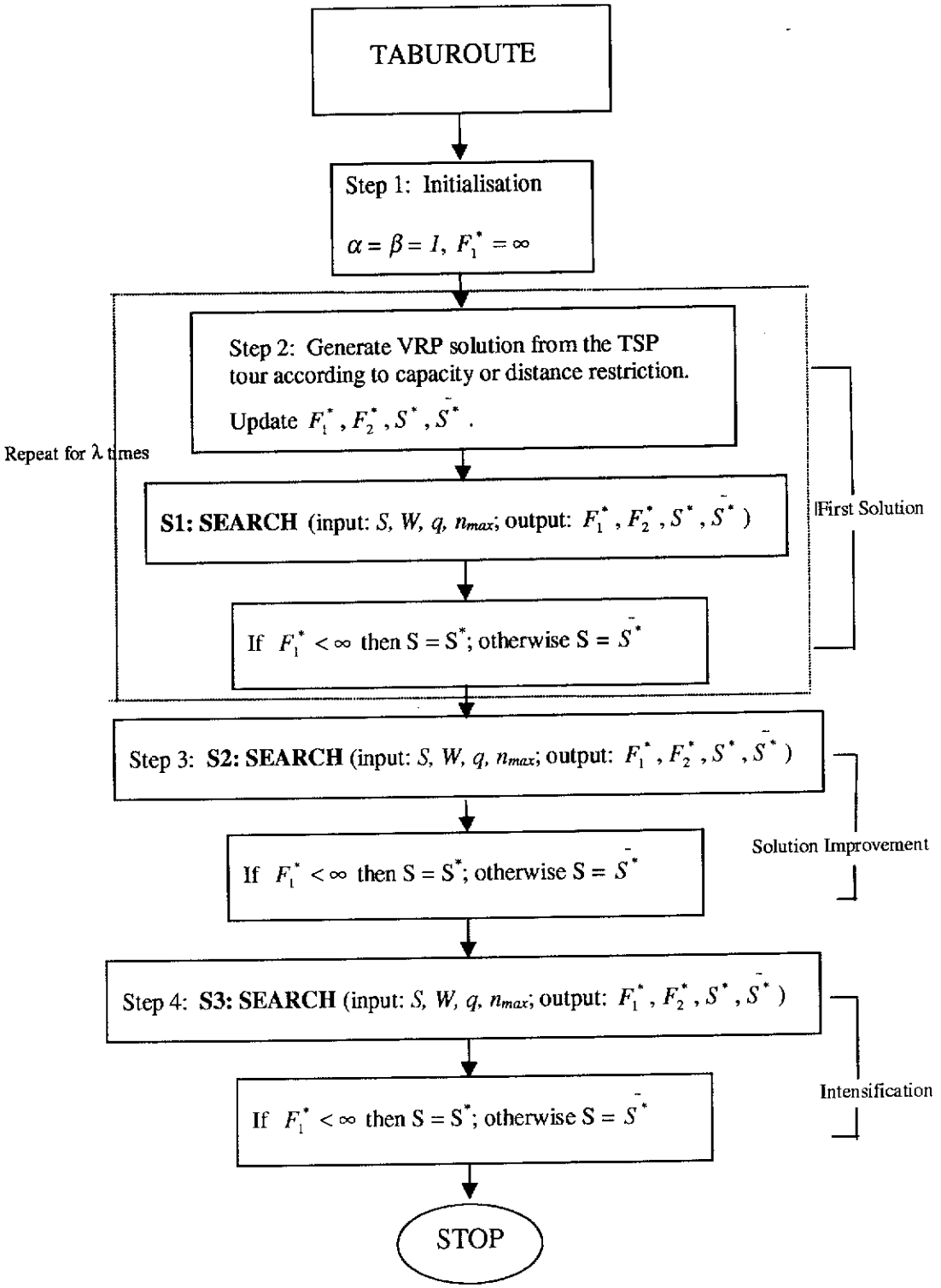


Figure 7.1: TABUROUTE Algorithm

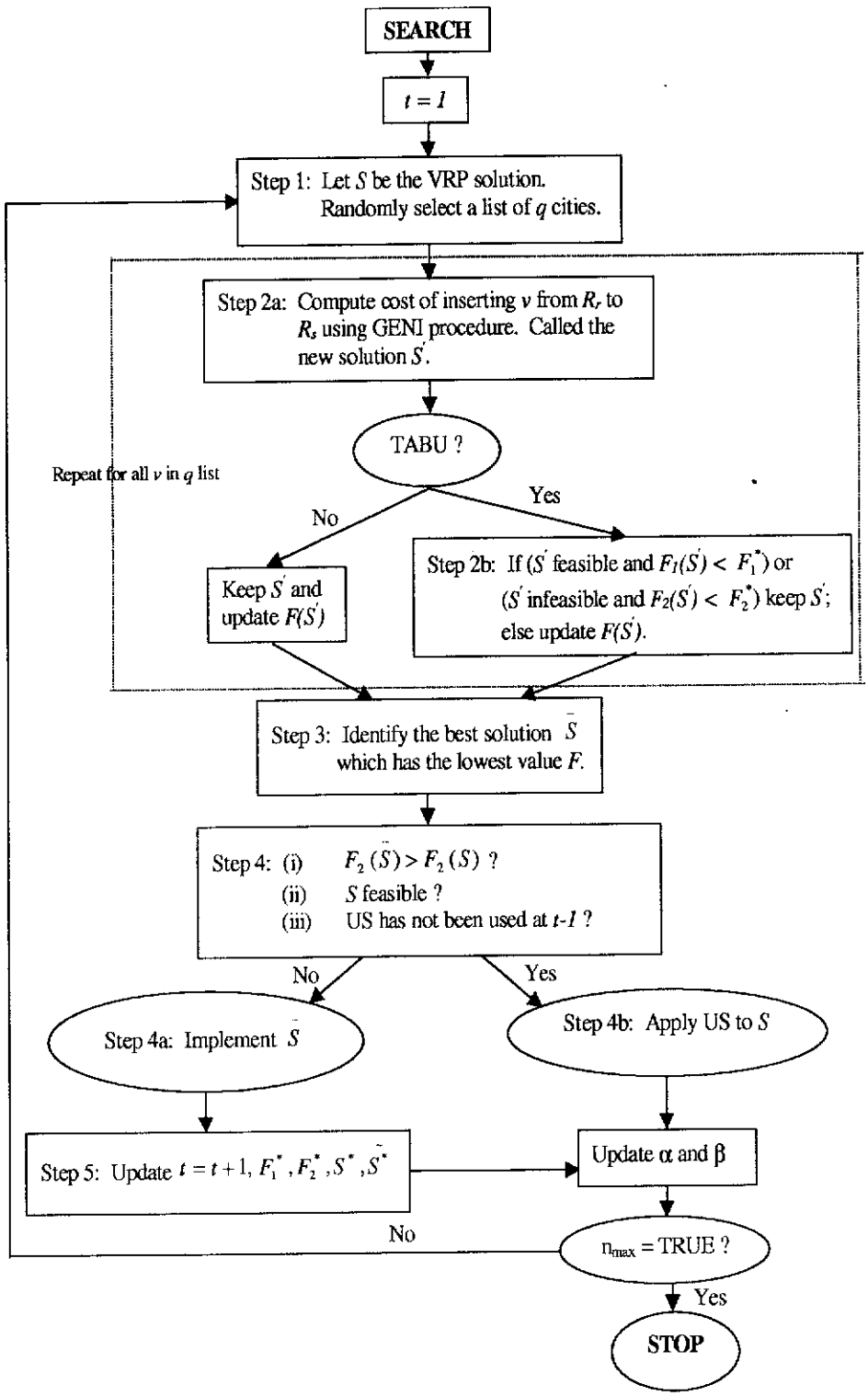


Figure 7.2: SEARCH Procedure

are not satisfied, $F(S')$ is updated according to the following:

$$F(S') = \begin{cases} F_2(S') & \text{if } F_2(S') < F_2(S) \\ F_2(S') + \delta_{max}\sqrt{m}gf_v & \text{otherwise} \end{cases} \tag{7.3}$$

where $\delta_{max} = \max \mid F_2^{t_1}(S) - F_2^{t_2}(S) \mid$ (where t_1 and t_2 are defined as the two successive iterations). g is a scaling factor and is set to 0.01. f_v is defined as the ratio of the number of times vertex v is removed to the iteration number t. This step is also called the diversification where the vertices which are moved frequently are penalised by adding some constant to the objective function value. This step ensures the algorithm explores different solution spaces.

Steps 2a and 2b are repeated for all v in the list q. The solution with the lowest F value is selected and is defined as $\bar{S}$. Note that $\bar{S}$ may not be implemented here because it is advantageous to improve the solution S by applying the Unstringing (US) procedure (Gendreau et al., 1992). If the following three conditions are satisfied

1. $F_2(\bar{S}) > F_2(S)$;
2. S is feasible; and
3. US has not been used at iteration $t-1$

then the post optimisation procedure US is applied to S; Otherwise $\bar{S}$ is implemented. Step 5 updates the parameters used in the procedure. Tabu list (that is the list of moves being prohibited for a certain number of iterations) is updated. The move which has been implemented in Step 4a is declared tabu for $t+\theta$ iterations where $\theta \in [\theta_{min}, \theta_{max}] = [5, 10]$ is chosen randomly at each iteration. The parameters and solutions $t = t+1, F_1^*, F_2^*, S^*$ and $\tilde{S}^*$ are updated.

The two penalty parameters, α and β, for capacity and distance are updated if t is a multiple of h ($h = 10$ in the algorithm) in Step 6. If all the previous h solutions were feasible for vehicle capacity, set $\alpha = \frac{\alpha}{2}$ and if they were all infeasible then set $\alpha = 2\alpha$. Similarly, if all the previous h solutions were feasible for route length, set $\beta = \frac{\beta}{2}$ and if they were all infeasible then $\beta = 2\beta$.

Steps 1 to Step 6 are repeated. The procedure is stopped if the value of F_1^* and F_2^* have not decreased for the last n_{max} iterations. The value n_{max} is chosen to be n for S1 and S3 and $50n$ for S2.

TABUROUTE algorithm uses the neighbourhood parameter p defined in the context of basic neighbourhood set $K_p(v, \tau')$ for any given complete tour τ', vertex v and integer p.

$K_p(v, \tau') = \{u$: city u is one of the p closest cities, in tour τ, to a given city v $\}$.

We compare the heuristics through the neighbourhood parameters used in the GENI algorithm within the TABUROUTE algorithm. The different values considered for the parameter p are 3, 5, 7, 9. we named the TABUROUTE algorithm with parameter p as p_TR. The algorithm is tested on 14 literature problems with sizes ranging from 50 to 199 cities and 28 randomly generated problems with sizes ranging from 60 to 120 cities. All problems are executed on Silicon Graphic INDY machine with a single 100 MHz IP22 processor with 32 Mbytes of memory size and run speed of 87.3 MIPS.

7.2 Statistical Evaluation on the Literature Problems

In this section the quality of solutions obtained by the corresponding four different TABUROUTE algorithms on the 14 literature problems of the CVRP and CDVRP is analysed. The problem size ranges from 50 to 199 cities. Table 7.1 provides the list of 14 literature problems and the corresponding best known solutions. These problems are selected because they are commonly used in the literature for comparing various heuristics.

For a given problem, the heuristic p_TR is implemented 100 times. Let z_i be the best objective function value of the ith run $1 \leq i \leq 100$. Table 7.2 provides the best solution obtained over 100 runs i.e. $z^* = \min_{1 \leq i \leq 100} z_i$ for the heuristic p_TR. For a given problem, let T_i be the CPU time of the ith run of the heuristic p_TR. Table

Problem	*size n*	*Types*	*Best known solution*	*References*
1	50	C	524.61	Taillard(1993)
2	75	C	835.26	Taillard(1993)
3	100a	C	826.14	Taillard(1993)
4	150	C	1028.42	Taillard(1993)
5	199	C	1291.45	Taillard(1993)
6	120	C	1042.11	Taillard(1993)
7	100	C	819.56	Taillard(1993)
8	50	C,D	1055.43	Taillard(1993)
9	75	C,D	1659.68	Taillard(1993)
10	100a	C,D	1865.94	Taillard(1993)
11	150	C,D	2662.55	Taillard(1993)
12	199	C,D	3385.85	Taillard(1993)
13	120	C,D	7541.14	Taillard(1993)
14	100	C,D	9866.37	Taillard(1993)

Table 7.1: Problems and best known solutions for CVRP and CDVRP in literature

7.3 provides average, maximum and minimum of the CPU time $T_i, 1 \leq i \leq 100$ for each heuristic and each problem. The various p_TR are compared using the Friedman test and fitting the Weibull distribution to the objective function values of the heuristic solutions. For each problem, an estimate of the optimal objective function value and a confidence interval are obtained.

Types	*Problems*	Heuristics			
		3_TR	*5_TR*	*7_TR*	*9_TR*
CVRP	100	819.56	819.56	825.65	823.91
	100a	830.47	828.67	829.44	829.16
	120	1051.29	1062.56	1064.78	1063.04
	150	1041.67	1037.91	1051.38	1037.97
	199	1323.33	1328.25	1327.44	1322.8
	50	524.61	524.61	524.61	524.61
	75	847.49	847.27	845.61	848.51
CDVRP	100	9887.41	9892.52	9904.71	9926.44
	100a	1866.87	1865.94	1866.74	1865.94
	120	7558.34	7551.96	7567.46	7559.88
	150	2685.89	2672.68	2684.13	2678.57
	199	3447.91	3454.52	3470.15	3467.3
	50	1055.74	1055.43	1055.71	1055.68
	75	1672.65	1688.46	1687.61	1685.54

Table 7.2: Best solutions obtained over 100 runs

Heuristics	*Problems*	CDVRP			CVRP		
		Avg	*Max*	*Min*	*Avg*	*Max*	*Min*
3_TR	100	340.3955	782.6575	196.8507	451.7983	3676.3335	143.6949
	100a	1149.0275	10122.9922	526.1581	943.9544	2800.6262	413.4459
	120	653.7323	4432.7461	357.7786	1593.423	10899.0752	710.5774
	150	2565.9272	7086.96	1206.1377	1966.539	4450.8276	1052.282
	199	5226.7062	13765.7686	2216.6616	3289.9219	8984.7559	1852.5862
	50	193.372	533.2775	90.3322	123.0666	259.8174	72.2992
	75	302.9238	1029.1932	162.6991	212.0408	525.5154	120.589
5_TR	100	365.0788	701.7106	196.7593	570.0944	2146.5833	164.9963
	100a	1174.8658	6746.4917	576.6296	1061.6984	4374.0269	346.5421
	120	991.3092	3213.5386	463.2013	1984.8854	24795.0391	692.5593
	150	2883.0921	10120.7607	1306.7087	1751.8382	6606.9277	708.0551
	199	6344.9444	24263.9551	2543.9893	2652.8088	13012.4111	1297.9062
	50	258.8524	3068.7844	127.7983	86.3125	266.5571	44.7964
	75	335.0851	927.53	197.318	148.7832	367.2972	74.3031
7_TR	100	469.8656	1044.2836	273.8224	545.5847	1899.7314	145.7821
	100a	1566.1907	7914.2822	698.923	1055.9017	6435.3081	363.1575
	120	952.8554	3622.0347	608.0027	2117.6162	7760.6211	855.1738
	150	3266.4523	9035.1904	1525.0378	1958.1425	4938.3833	1044.4165
	199	6866.6214	26574.375	2635.2129	3305.1717	10115.7949	1649.449
	50	234.9798	503.7396	132.316	119.335	416.822	4 58.8372
	75	383.571	2621.5396	230.6247	175.386	651.9861	89.1732
9_TR	100	552.2582	1226.1611	348.3135	690.4614	3239.4426	252.556
	100a	2095.7605	34781.3359	731.7193	1023.1036	3002.7605	522.8918
	120	1340.3151	3517.7886	850.2844	2991.5759	11581.917	1183.6176
	150	3394.6044	8983.999	1733.7213	2873.3756	8642.2158	1138.0166
	199	7309.5313	24273.5391	3412.3489	5569.4132	13296.7148	3014.4636
	50	303.8128	4933.3457	145.3665	217.6618	480.7551	110.1158
	75	570.2576	1802.6537	368.4459	375.6549	695.0828	245.2195

Table 7.3: Running Time (in seconds) for the literature problem on various heuristics

7.2.1 Friedman Test

We carried out the Friedman test for the CVRP and CDVRP, on four heuristics p_TR with the null hypothesis that there is no difference in performance between these heuristics against the alternate that there is difference. For details of the test refer to the Appendix A. Table 7.4 shows the

$$\text{ratio} = \frac{\text{objective function value of the heuristic solution}}{\text{objective function value of the best known solution}}$$

and its associated ranking for various heuristics on different problems for both CVRP and CDVRP. The summary of the Friedman test is given in Table 7.5 for both CVRP and CDVRP.

	Problems	CVRP				CDVRP			
		3_TR	*5_TR*	*7_TR*	*9_TR*	*3_TR*	*5_TR*	*7_TR*	*9_TR*
Ratio	100	1	1	1.0074	1.0053	1.0021	1.0026	1.0038	1.0060
	100a	1.0052	1.0030	1.0039	1.0036	1.0004	1	1.0004	1
	120	1.0088	1.0196	1.0217	1.0200	1.0022	1.0014	1.0034	1.0024
	150	1.0128	1.0092	1.0223	1.0092	1.0087	1.0038	1.0081	1.0060
	199	1.0246	1.0284	1.0278	1.0242	1.0183	1.0202	1.0248	1.0240
	50	1	1	1	1	1.0002	1	1.0002	1.0002
	75	1.0146	1.0143	1.0123	1.0158	1.0078	1.0173	1.0168	1.0155
Rank	100	1.5	1.5	4	3	1	2	3	4
	100a	4	1	3	2	4	1.5	3	1.5
	120	1	2	4	3	2	1	4	3
	150	3	1	4	2	4	1	3	2
	199	2	4	3	1	1	2	4	3
	50	2.5	2.5	2.5	2.5	4	1	3	2
	75	3	2	1	4	1	4	3	2
R_j		17	14	21.5	17.5	17	12.5	23	17.5
$\sum_i R_{ij}^2$		47.5	34.5	73.25	49.25	55	29.25	77	48.25

Table 7.4: Ratio and rank for the Friedman test for the CVRP and CDVRP

	CVRP	*CDVRP*
Hypothesis:	All 4 p_TR heuristics are equal in performance for CVRP.	All 4 p_TR heuristics are equal in performance for CDVRP.
$A_F = \sum_{i=1}^{n} \sum_{j=1}^{k} (R_{ij})^2$	204.5	209.5
$B_F = \frac{1}{n} \sum_{j=1}^{k} R_j^2$	179.07	182.93
$T_F = \frac{(n-1)(B_F - (nk(k+1)^2)/4)}{A_F - B_F}$	0.9606	1.79
$F_\alpha((k-1),(n-1)(k-1))$	$F_{0.05}(3,18) = 8.68$	$F_{0.05}(3,18) = 8.68$
Conclusion:	**Do not reject** the null hypothesis that all 4 p_TR heuristics are equal in performance for CVRP at $\alpha = 0.05$.	**Do not reject** the null hypothesis that all 4 p_TR heuristics are equal in performance for CDVRP at $\alpha = 0.05$.

Table 7.5: Computational result on Friedman test on CVRP and CDVRP

The null hypothesis that there is no difference between the four p_TR heuristics for both CVRP and CDVRP is accepted at 5% level of significance. This concludes that there is no significant difference between the quality of the solutions generated by heuristic p_TR with different neighbourhood parameters. Furthermore, these problems are hard in the sense that the set of feasible local optimum solutions of these problems may be very dense and hence most of the heuristics

will need enormous computational effort to find better solutions.

In the next section we attempt to estimate an optimal objective function value and its confidence interval for each problem.

7.2.2 Fitting of Weibull Distribution (Literature Problems)

In this section, we fit the Weibull distribution, an extreme value distribution, to the objective function values generated by Taburoute algorithm. Subsequently, an estimate of the optimal objective function value and its confidence interval is calculated.

Suppose a given probabilistic heuristic is implemented several times on a given problem. Let $z_{i1}, z_{i2}, ..., z_{iq}$ be the observed local minimum objective function values corresponding to the q runs of the ith batch of implementation. Thus the best known objective function value in the ith implementation can be denoted by $z_i^{'} = \min\{z_{i1}, z_{i2}, ..., z_{iq}\}$. Furthermore if the probabilistic heuristic is implemented r times then we have the best known objective function value of this problem by the heuristic as $v = \min\{z_i^{'}, 1 \leq i \leq r\}$. By the extreme value theory, irrespective of the distribution of the objective function values, as q increases the random variable Z' (representing the minimum of a random sample of size q) follows the Weibull distribution.

Hence for a given heuristic implemented on a given problem we fit the Weibull distribution to the set of random samples $z_i^{'}, 1 \leq i \leq r$ of size r. Weibull cumulative distribution function can be rewritten as the following equation of a straight line by taking the logarithm twice:

$$\ln(-\ln(1 - F(x_0))) = c\ln(x_0 - a) - c\ln b \tag{7.4}$$

where a, b and c are the location, scale and shape parameters of the Weibull distribution.

To fit the Weibull distribution to $z_i^{'}, 1 \leq i \leq r$, we follow the procedure suggested

by Golden (1977, 1978):

1. Fix a suitable value $\hat{a}$ (smaller than v) for 'a', the location parameter.

2. For the specific value $\hat{a}$, estimate $\hat{b}$ and $\hat{c}$ by the least square method fitting (7.4) to the observe values $z_i^{'}, 1 \leq i \leq r$ for x_0.

3. Consider the null hypothesis that Z' follows the Weibull distribution with location parameter $\hat{a}$, scale parameter $\hat{b}$ and shape parameter $\hat{c}$.

4. Compute the statistic

$$T = D\left[\sqrt{n} + 0.12 + \frac{0.11}{\sqrt{n}}\right] \tag{7.5}$$

   where D is the Kolmogorov-Smirnov (K-S) statistic (Scheaffer and McClave (1990)).

5. If $T > 1.358$, reject the null hypothesis and choose the next $\hat{a}$ as $\hat{a} = \hat{a} - 1$. Go to Step 2. Otherwise, accept the latest $\hat{a}$, $\hat{b}$ and $\hat{c}$ as the best fit parameter values for $\tilde{a}, \tilde{b}$ and $\tilde{c}$ respectively.

We fitted the Weibull Distribution to the solutions on all the 14 problems for CVRP and CDVRP. Tables 7.6 and 7.7 show the estimate parameters $\tilde{a}, \tilde{b}, \tilde{c}$, confidence interval $(v - \tilde{b}, v)$ for $\tilde{a}$, performance measure and the proportion deviation between the estimate solution and the best known objective function value.

In the case of CVRP we observe that

- The deviation of the estimated solution from the best known solution is below 6.1% (the 50-city problem was not included in the Weibull fitting because the problem was solved by the four heuristics).

- The best known solutions are located within the $100(1 - \exp^{-20})\% = 100\%$ confidence intervals.

- The average performance measure (or the relative interval width) for various heuristics is presented in Table 7.8. This is calculated by taking the sum

	Problems	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	Confidence Interval		$\frac{\tilde{b}}{v}$	Proportion
					Lower	Upper		Deviation
3_TR	100	769.56	53.9656	33.4867	765.59	819.55	0.0658	-0.0610
	100a	780.47	55.3891	23.7552	775.08	830.47	0.0667	-0.0552
	120	1001.29	77.8077	6.11	973.48	1051.29	0.074	-0.0391
	150	991.67	67.9173	10.3711	973.75	1041.67	0.0652	-0.0357
	199	1273.33	74.9491	7.7311	1248.38	1323.33	0.0566	-0.0140
	50	-	-	-	-	-	-	-
	75	797.49	55.3152	23.7619	792.17	847.48	0.0653	-0.0452
5_TR	100	769.56	56.5174	36.2545	763.04	819.55	0.069	-0.0610
	100a	778.67	56.0274	17.4871	772.64	828.66	0.0676	-0.0574
	120	1012.56	56.0756	19.6269	1006.47	1062.55	0.0528	-0.0283
	150	987.91	72.3899	11.0139	965.52	1037.91	0.0697	-0.0393
	199	1278.25	70.8641	7.2465	1257.38	1328.24	0.0534	-0.0102
	50	-	-	-	-	-	-	-
	75	797.27	58.438	19.7157	788.83	847.27	0.069	-0.0454
7_TR	100	775.65	50.696	66.592	774.95	825.64	0.0614	-0.0535
	100a	779.44	54.368	18.5033	775.07	829.43	0.0655	-0.0565
	120	1014.78	94.0792	2.1462	970.69	1064.77	0.0884	-0.0262
	150	1001.38	61.7625	13.4255	989.61	1051.37	0.0587	-0.0262
	199	1277.44	71.6685	8.9137	1255.76	1327.43	0.054	-0.0108
	50	-	-	-	-	-	-	-
	75	795.61	58.4145	18.8737	787.19	845.60	0.0691	-0.0474
9_TR	100	773.91	53.6362	36.7561	770.27	823.90	0.0651	-0.0557
	100a	779.16	56.3373	15.5829	772.82	829.16	0.0679	-0.0568
	120	1061.04	30.5667	0.868	1032.47	1063.04	0.0288	0.0181
	150	987.97	73.0187	10.5433	964.946	1037.96	0.0703	-0.0393
	199	1272.8	74.856	9.0176	1247.93	1322.79	0.0566	-0.0144
	50	-	-	-	-	-	-	-
	75	798.51	55.9607	19.729	792.54	848.50	0.066	-0.0439

Table 7.6: Estimates of parameters of Weibull distribution for the CVRP

of all the elements for each heuristic and divided by the total number of heuristics. From the table we can conclude empirically that 9_TR is the best among the four heuristics for CVRP on literature problems since it has the smallest performance measure.

In the case of CDVRP we observe that

- The deviation of the estimated solution from the best known solution is below 4.7%.
- The best known solutions are located within the $100(1-\exp^{-20})\% = 100\%$ confidence intervals except for one case (Problem 100 with 9_TR but the interval is very close to the best known solution).

	Problems	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{v}$	*Proportion Deviation*
					Lower	*Upper*		
3_TR	100	9837.41	122.3512	4.6538	9765.05	9887.40	0.0124	-0.0029
	100a	1816.87	60.1949	11.3813	1806.67	1866.86	0.0322	-0.0262
	120	7550.34	45.9487	1.2589	7512.39	7558.33	0.0061	0.0012
	150	2635.89	85.4018	6.1011	2600.48	2685.8894	0.0318	-0.0100
	199	3397.91	115.9567	5.258	3331.95	3447.9084	0.0336	0.0035
	50	1053.74	3.5912	3.6396	1052.14	1055.73	0.0034	-0.0016
	75	1622.65	92.3776	5.1122	1580.27	1672.64	0.0552	-0.0223
5_TR	100	9842.52	129.4352	3.8467	9763.08	9892,51	0.0131	-0.0024
	100a	1815.94	58.4773	12.6218	1807.46	1865.94	0.0313	-0.0267
	120	7537.96	40.0441	2.2156	7511.91	7551.9585	0.0053	-0.0004
	150	2622.68	106.5743	5.0097	2566.10	2672.6812	0.0399	-0.0149
	199	3404.52	132.856	3.8964	3321.66	3454.5247	0.0385	0.0055
	50	1005.43	52.5725	38.173	1002.85	1055.4302	0.0498	-0.0473
	75	1638.46	90.7552	4.0657	1597.70	1688.4648	0.0538	-0.0127
7_TR	100	9854.71	97.2426	5.7255	9807.47	9904.7148	0.0098	-0.0011
	100a	1816.74	56.1703	15.0452	1810.56	1866.7382	0.0301	-0.0263
	120	7517.46	56.7763	10.7768	7510.68	7567.4614	0.0075	-0.0031
	150	2634.13	74.3846	6.2243	2609.74	2684.1277	0.0277	-0.0106
	199	3420.15	96.2036	4.8282	3373.94	3470.1526	0.0277	0.0101
	50	1005.71	51.9401	50.3247	1003.76	1055.7061	0.0492	-0.0471
	75	1637.61	80.4566	5.0995	1607.15	1687.60	0.0477	-0.0132
9_TR	100	9895.44	59.7577	3.3116	9866.67	9926.43	0.006	0.0029
	100a	1815.94	56.5992	16.0497	1809.34	1865.94	0.0303	-0.0267
	120	7509.88	78.3164	4.0933	7481.56	7559.88	0.0104	-0.0041
	150	2628.57	86.693	5.4416	2591.87	2678.56	0.0324	-0.0127
	199	3417.3	105.2249	4.6902	3362.07	3467.30	0.0303	0.0092
	50	1006.68	51.5329	32.2946	1005.14	1056.67	0.0488	-0.0461
	75	1635.54	94.4606	4.1616	1591.08	1685.54	0.056	-0.0145

Table 7.7: Estimates of parameters of Weibull distribution for the CDVRP

3_TR	*5_TR*	*7_TR*	*9_TR*
0.0688	0.0545	0.0567	0.0507

Table 7.8: Average performance measure for different heuristics for CVRP

- The average performance measure for various heuristic is presented in Table 7.9. From the table, we can conclude empirically that 3_TR is the best among the four heuristic for CDVRP for literature problems.

3_TR	*5_TR*	*7_TR*	*9_TR*
0.0249	0.0331	0.0285	0.0306

Table 7.9: Average performance measure for different heuristics for CDVRP

7.2.3 Conclusion

From the above findings we can conclude from the Friedman test that there is no significant difference between the quality of the solutions generated by the algorithms based on different neighbourhood parameters. Empirically we can choose 9_TR for CVRP and 3_TR for CDVRP for good quality solution. However one may choose 3_TR since it uses less computational time.

We have illustrated the good quality of the solution for each problem by obtaining a point estimate and its confidence interval for each problem. The estimated solution for each problem is accepted here since the deviation from the best known solution is within 4.7% for CDVRP and 6.1% for CVRP. The results from the confidence interval are also encouraging.

In the following we present tables and graphs showing a few findings from the results obtained for the CVRP and CDVRP.

- Graphs 7.1(a) and 7.2(a) show the pattern of the average diversification steps for various heuristics on various problems for the CVRP and CDVRP. Recall the diversification strategy penalised those vertices which have been moved frequently by adding a penalty to the objective function value. For each heuristic applied to each problem, we collect the average number of diversification steps. Note that the pattern of the graphs is stable for different heuristics for both the CVRP and CDVRP. However the 150-city problem for the CDVRP has a larger number of steps than the CVRP. The 199-city problem has a higher number of steps for the lower parameter ($p = 3$) for both CVRP and CDVRP.

- The CPU time increases as the parameter number increases as shown in Graphs 7.1(b) and 7.2(b) for the CVRP and CDVRP. Note that as the size of the problem gets larger and as the parameter gets larger, the CPU time increases. This is especially noticeable for the 199-city problem.

- The pattern for the average number of times the Search procedure is used remains stable for different heuristics as shown in Graphs 7.1(c) and 7.2(c).

But the number of searches increases as the number of cities increase for both CVRP and CDVRP.

- In each run of a heuristic on a problem, we record the number of feasible neighbourhood moves and the total number of neighbourhood moves (feasible and infeasible). The proportion can be calculated as

$$p_i = \frac{\text{number of feasible neighbours}}{\text{total number of neighbours (feasible and infeasible)}}. \tag{7.6}$$

The average proportion is calculated by $\frac{\sum_{i=1}^{100} p_i}{100}$. The average proportion of feasible neighbourhood moves is rather small (less than 47% for CVRP and less than 25% for CDVRP respectively) as shown in Table 7.10.

- The average proportion of feasible solutions S implemented are small (less than 14% for CVRP and less than 2% for CDVRP respectively) as shown in Table 7.11. Note that the average proportion of feasible solutions S implemented can be calculated as follows. In each run of a heuristic on a problem, the number of solutions implemented is recorded and the proportion of feasible solutions S implemented can be calculated as

$$p_i^S = \frac{\text{feasible solutions implemented}}{\text{total solution searched}}. \tag{7.7}$$

So the average proportion over the 100 runs can be calculated as $\frac{\sum_i^{100} p_i^S}{100}$.

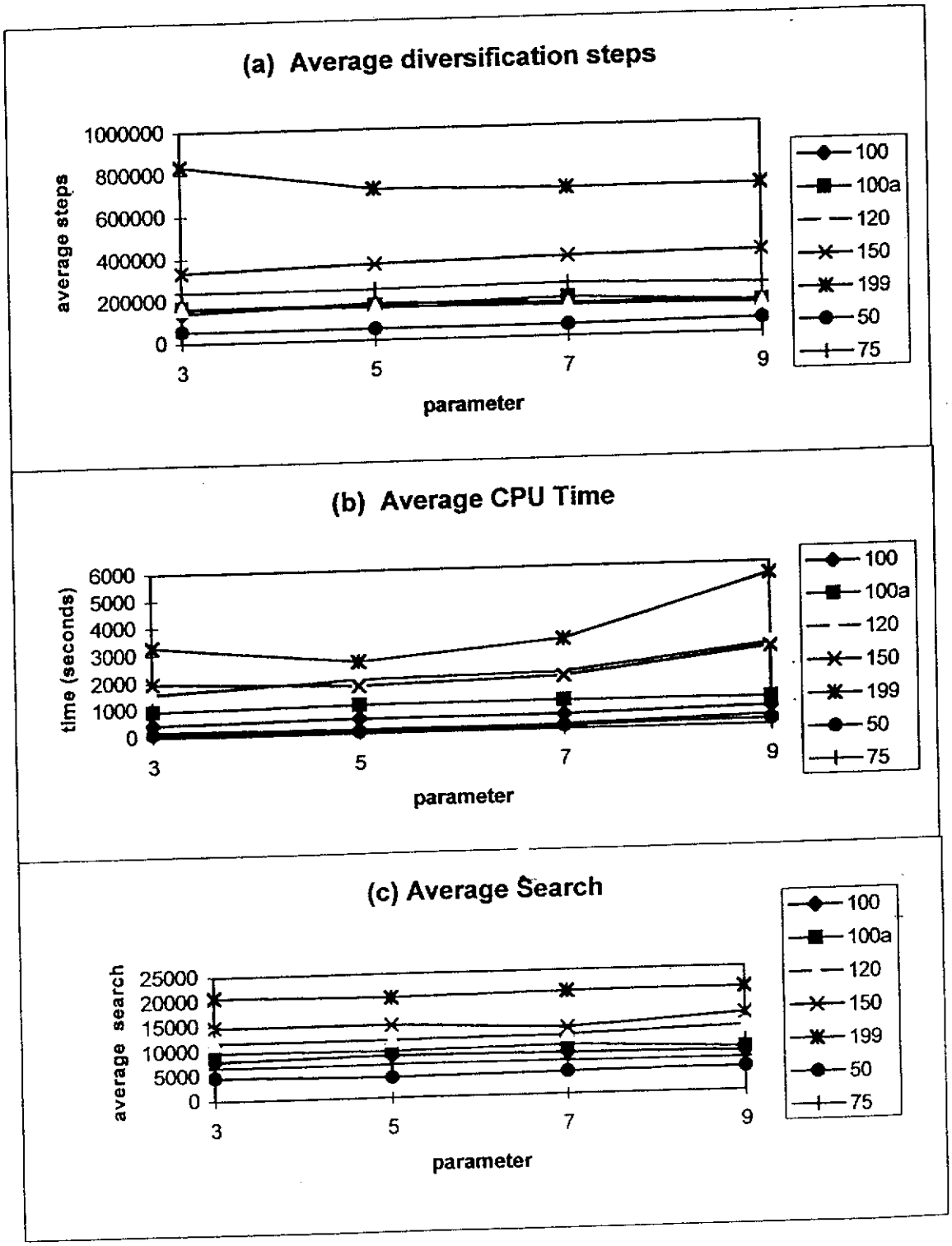
	CVRP				CDVRP			
Problems	*3_TR*	*5_TR*	*7_TR*	*9_TR*	*3_TR*	*5_TR*	*7_TR*	*9_TR*
100	0.4521	0.4735	0.4382	0.4099	0.2505	0.2466	0.2419	0.2264
100a	0.3895	0.3704	0.3632	0.3963	0.1049	0.1071	0.1028	0.1024
120	0.313	0.3212	0.3371	0.3279	0.1421	0.1321	0.1357	0.1138
150	0.227	0.2131	0.2363	0.2182	0.1131	0.1118	0.1142	0.1113
199	0.2106	0.2049	0.202	0.1848	0.1122	0.1102	0.107	0.1068
50	0.311	0.3042	0.3092	0.301	0.1493	0.1447	0.1465	0.1431
75	0.313	0.3013	0.2941	0.2493	0.2276	0.2202	0.2098	0.1538

Table 7.10: Proportion of feasible neighbourhood recorded while searching

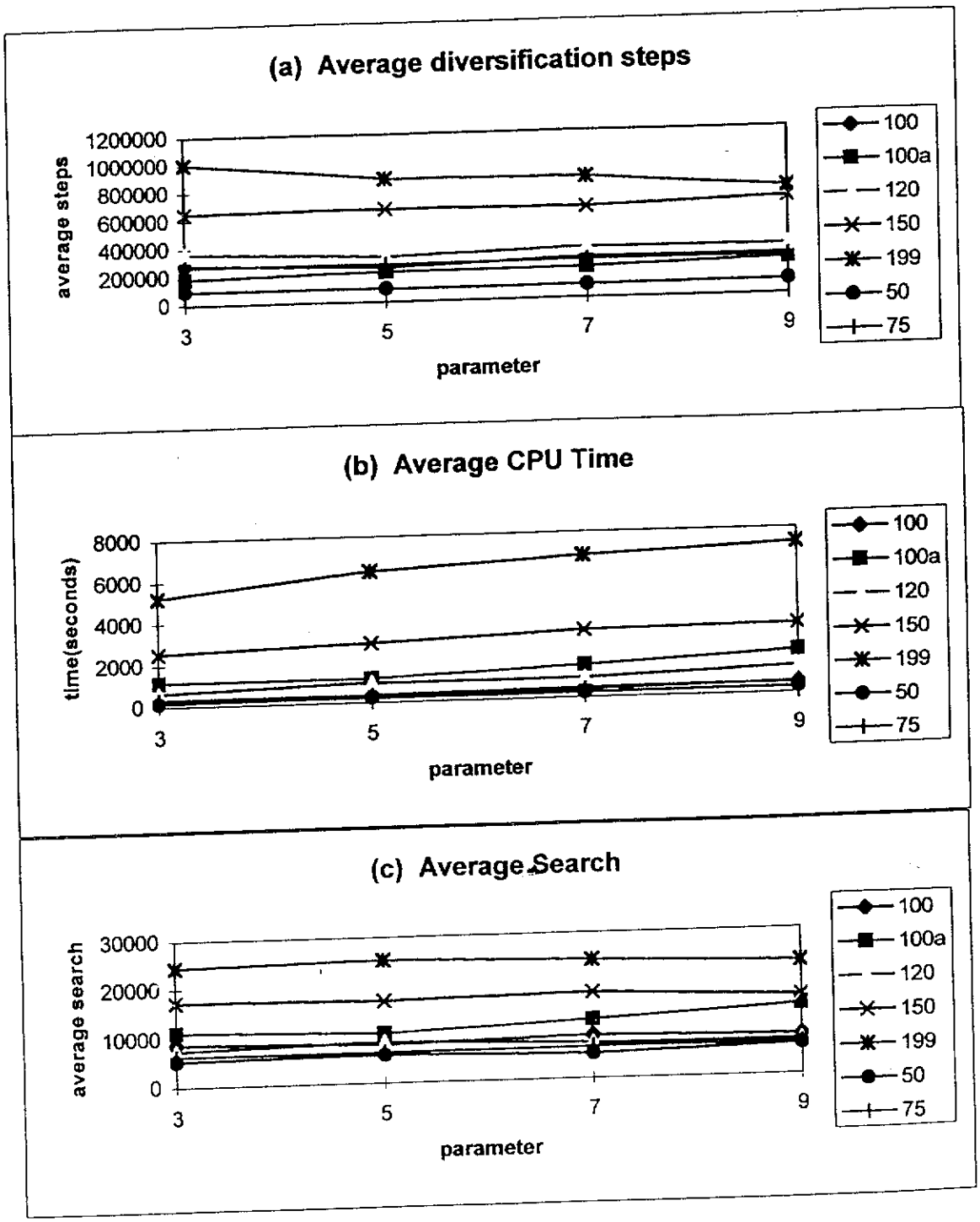
In the following section, the heuristics are compared through some randomly generated problems.

	CVRP				CDVRP			
Problems	*3_TR*	*5_TR*	*7_TR*	*9_TR*	*3_TR*	*5_TR*	*7_TR*	*9_TR*
100	0.0399	0.0283	0.0244	0.02	0.0012	0.0019	0.0015	0.0019
100a	0.0238	0.0267	0.0254	0.0283	0.0104	0.0114	0.0096	0.0074
120	0.0264	0.02	0.0265	0.0273	0.0039	0.0084	0.005	0.0051
150	0.0103	0.0105	0.0099	0.0088	0.003	0.0028	0.0036	0.0032
199	0.0052	0.0049	0.0052	0.0044	0.0011	0.0011	0.0012	0.0012
50	0.1102	0.1363	0.1278	0.1285	0.0207	0.0168	0.0192	0.0148
75	0.0168	0.0151	0.0181	0.0165	0.0035	0.0029	0.0029	0.0025

Table 7.11: Proportion of feasible solution S implemented during the search



Graph 7.1: Summary of results for the CVRP



Graph 7.2 Summary of results for the CDVRP

7.3 Statistical Evaluation on the Randomly Generated Problems

In this section we generate 28 Euclidean CDVRP test problems with sizes ranging from 60 to 120 cities in increments of 10. The randomly generated problems are based on the following parameters:

- The problems are classified into four groups with different θ, $0 \leq \theta \leq 1$, which measures the tightness of the vehicle capacity restrictions. The parameter θ is taken to be 0, 0.33, 0.67 and 1.0.
- The total demand Q is generated as in Laporte et al. (1985) $Q = (1-\theta)\max_{i \in V}\{q_i\} + \theta \sum_{i \in V} q_i$.
- The demand q_i is randomly generated between [1,100].
- The length of the problems is set to be $L = 1800$.

Note that similar test problems are used for the case of CVRP by omitting the length.

The purpose of running the algorithm with randomly generated problem is to validate the performance of the algorithm and to examine the algorithms with different problems with various tightness of the vehicle capacity θ. For the case of CDVRP, we have a total of 28 problems where the sizes of the problems range from 60 cities to 120 cities. For each size, say 60 cities problem, we have four problems with various vehicle capacity restrictions as indicated by θ. For the case of the CVRP, we used the same problems and the length of the problems is set to be infinity. The problems with $\theta = 1.0$ are omitted since for these problems, the capacity constraints are not binding. Each problem is executed for 30 runs for each p_TR where $p = \{3, 5, 7, 9\}$.

Tables 7.12 and 7.13 show the best obtained solution for various heuristics over the 30 runs for CVRP and CDVRP. Table 7.14 shows the best lower bound

solution obtained in 30 runs for CVRP and CDVRP. The average, maximum and minimum CPU time over the 30 runs for each problem for CVRP and CDVRP are displayed on Tables B.17-B.21 in Appendix B.

			Heuristic		
Size of Problems	θ	*3_TR*	*5_TR*	*7_TR*	*9_T R*
60	0	32882	32882	32990.64	32882
	0.33	7055.47	7055.47	7055.47	7055.47
	0.67	6554.05	6554.05	6554.05	6554.05
70	0	41310.95	41201.7	41272.44	41272.44
	0.33	7575.8	7565.48	7565.48	7565.48
	0.67	6530.05	6530.05	6530.05	6530.05
80	0	49237.42	48799.15	49136.33	49265.8
	0.33	8107.12	8107.12	8107.12	8107.12
	0.67	7204.35	7193.85	7193.85	7193.85
90	0	49275.49	48882.88	48529.66	48808.67
	0.33	8490.16	8490.16	8497.48	8490.16
	0.67	7640.16	7640.16	7640.16	7640.16
100	0	53151.21	52320.05	52330.26	52181.08
	0.33	8044.25	8044.25	8044.25	8044.25
	0.67	7884.73	7884.73	7943.18	7967.84
110	0	60173.6	60142.05	60876.18	60856.85
	0.33	8770.18	8770.18	8770.18	8770.18
	0.67	8367.18	8334.14	8318.87	8334.14
120	0	67243.66	67194.95	67038.76	66437.76
	0.33	9378.96	9384.57	9384.46	9384.46
	0.67	8751.35	8732.47	8707.93	8751.35

Table 7.12: Best solutions generated by 30 runs for randomly generated CVRP

In the following we carried out the statistical tests to compare the quality of the solutions generated.

7.3.1 Friedman Test

We carried out the Friedman test on the four VRP heuristics for the CVRP and CDVRP based on the null hypothesis that there is no significant difference in performance between the four heuristics. Tables 7.15 and 7.16 show the ratio and its associated ranking for CVRP and CDVRP. For details on this subject refer to Appendix A.

The summary of the Friedman test is given in Table 7.17 for both the CVRP

Size of Problems	θ	*3_TR*	*Heuristic* *5_TR*	*7_TR*	*9_T R*
60	0	32882	32882	32990.64	32882
	0.33	10316.01	10265.93	10280.75	12150.88
	0.67	10190.48	9875.03	9875.03	9876.83
	1	11803.82	11687.98	11897.62	11883.75
70	0	41310.95	41201.7	41272.44	41272.44
	0.33	9564.41	9547.81	9562.57	9555.19
	0.67	10549.62	10661.56	10569.63	10552.55
	1	11430.76	11426.52	11426.52	11426.52
80	0	49237.42	48799.15	49136.33	49136.33
	0.33	17605.76	18033.06	17533.73	17922.35
	0.67	13124.3	13702.25	12529.2	13554.15
	1	11880.61	11673.64	11871.4	11636.68
90	0	49316.42	48896.63	48570.59	49316.42
	0.33	12690.61	13815.28	13868.53	13242.18
	0.67	14690.79	14966.99	16703.74	16248.51
	1	11113.89	11968.12	11617.68	10960.87
100	0	53151.21	52181.08	52181.08	52181.08
	0.33	13641.3	14518.54	13535.99	13576
	0.67	9558.52	9568.17	9568.17	9568.17
	1	9513.19	9549.26	9549.26	9527.47
110	0	60173.6	60142.05	60810.58	60877.96
	0.33	12939.67	12660.21	12497.02	12517.48
	0.67	15709.97	14864.85	14457.82	14491.64
	1	12012.1	13485.23	13065.63	13697.44
120	0	67243.66	67194.95	67019.33	66437.76
	0.33	12152.37	12143.61	11984.98	12152.37
	0.67	16032.56	15776.17	16151.71	16542.77
	1	11772.77	11754.93	11662.22	12304.62

Table 7.13: Best solutions generated by 30 runs for randomly generated CDVRP

Sizes	θ	*CDVRP*	*CVRP*
60	0	32882.0	32882.0
	0.33	10265.93	7055.47
	0.67	9875.03	6554.05
	1.0	11687.98	-
70	0	41201.7	41201.7
	0.33	9547.81	7565.48
	0.67	10549.62	6530.05
	1.0	11426.52	-
80	0	48799.15	48799.15
	0.33	17533.73	8107.12
	0.67	12529.2	7193.85
	1.0	11636.68	-
90	0	48570.59	48529.66
	0.33	12690.61	8490.16
	0.67	14690.79	7640.16
	1.0	10960.87	-
100	0	52181.08	52181.08
	0.33	13535.99	8044.25
	0.67	9558.52	7884.73
	1.0	9513.19	-
110	0	60142.05	60142.05
	0.33	12497.02	8770.18
	0.67	14457.82	8318.87
	1.0	12012.12	-
120	0	66437.76	66437.76
	0.33	11984.98	9378.96
	0.67	15776.17	8707.93
	1.0	11662.22	-

Table 7.14: Best lower bound solution generated in 30 runs for CVRP and CD-VRP

Problems	θ	*Ratio*				*Rank*			
		3_TR	*5_TR*	*7_TR*	*9_TR*	*3_TR*	*5_TR*	*7_TR*	*9_TR*
	0	1	1	1.0033	1	2	2	4	2
60	0.33	1	1	1	1	2.5	2.5	2.5	2.5
	0.67	1	1	1	1	2.5	2.5	2.5	2.5
	0	1.0026	1	1.0017	1.0017	4	1	2.5	2.5
70	0.33	1.0013	1	1	1	4	2	2	2
	0.67	1	1	1	1	2.5	2.5	2.5	2.5
	0	1.0089	1	1.0069	1.0095	3	1	2	4
80	0.33	1	1	1	1	2.5	2.5	2.5	2.5
	0.67	1.0014	1	1	1	4	2	2	2
	0	1.0153	1.0072	1	1.0057	4	3	1	2
90	0.33	1	1	1.0008	1	2	2	4	2
	0.67	1	1	1	1	2.5	2.5	2.5	2.5
	0	1.0185	1.0026	1.0028	1	4	2	3	1
100	0.33	1	1	1	1	2.5	2.5	2.5	2.5
	0.67	1	1	1.0074	1.0105	1.5	1.5	3	4
	0	1.0005	1	1.0122	1.0118	2	1	4	3
110	0.33	1	1	1	1	2.5	2.5	2.5	2.5
	0.67	1.0058	1.0018	1	1.0018	4	2.5	1	2.5
	0	1.0121	1.0113	1.0090	1	4	3	2	1
120	0.33	1	1.0005	1.0005	1.0005	1	4	2.5	2.5
	0.67	1.0049	1.0028	1	1.0049	3.5	2	1	3.5
					R_j	60.5	46.5	51.5	51.5
					$\sum_i R_{ij}^2$	192.25	113.25	141.25	137.75

Table 7.15: Ratio and Rank for the Friedman test on randomly generated CVRP

Problems	θ	*Ratio*				*Rank*			
		3_TR	*5_TR*	*7_TR*	*9_TR*	*3_TR*	*5_TR*	*7_TR*	*9_TR*
60	0	1	1	1.0033	1	2	2	4	2
60	0.33	1.0048	1	1.0014	1.1836	3	1	2	4
60	0.67	1.0319	1	1	1.0001	4	1.5	1.5	3
60	1	1.0099	1	1.0179	1.0167	2	1	4	3
70	0	1.0026	1	1.0017	1.0017	4	1	2.5	2.5
70	0.33	1.0017	1	1.0015	1.0007	4	1	3	2
70	0.67	1	1.0106	1.0018	1.0002	1	4	3	2
70	1	1.0003	1	1	1	4	2	2	2
80	0	1.0089	1	1.0069	1.0069	4	1	2.5	2.5
80	0.33	1.0041	1.0284	1	1.0221	2	4	1	3
80	0.67	1.0474	1.0936	1	1.0818	2	4	1	3
80	1	1.0209	1.0031	1.0201	1	4	2	3	1
90	0	1.0153	1.0067	1	1.0153	3.5	2	1	3.5
90	0.33	1	1.0886	1.0928	1.0434	1	3	4	2
90	0.67	1	1.01880	1.1370	1.1060	1	2	4	3
90	1	1.0139	1.0918	1.0599	1	2	4	3	1
100	0	1.0185	1	1	1	4	2	2	2
100	0.33	1.0077	1.0725	1	1.0029	3	4	1	2
100	0.67	1	1.0010	1.0010	1.0010	1	3	3	3
100	1	1	1.0037	1.0037	1.0015	1	3.5	3.5	2
110	0	1.0005	1	1.0111	1.0122	2	1	3	4
110	0.33	1.0354	1.0130	1	1.0016	4	3	1	2
110	0.67	1.0866	1.0281	1	1.0023	4	3	1	2
110	1	1	1.1226	1.0877	1.1403	1	3	2	4
120	0	1.01213	1.0113	1.0087	1	4	3	2	1
120	0.33	1.0139	1.0132	1	1.0139	3.5	2	1	3.5
120	0.67	1.0162	1	1.0238	1.0485	2	1	3	4
120	1	1.0094	1.0079	1	1.0550	3	2	1	4
					R_j	76	66	65	73
					$\sum_i R_{ij}^2$	245.5	187.5	182	214

Table 7.16: Ratio and Rank for the Friedman test on randomly generated CDVRP

and CDVRP. From the table we conclude that the hypothesis is not rejected for CVRP and CDVRP. This suggests that there is no significant difference between the quality of the solutions generated by four algorithms based on different neighbourhood parameters. Since the null hypothesis is not rejected, the expected utility function test to select the best heuristic need not be carried out.

	CVRP	*CDVRP*
Hypothesis:	All 4 p_TR heuristics are equally accurate for CVRP.	All 4 p_TR heuristics are equally accurate for CDVRP.
$A_F = \sum_{i=1}^{n} \sum_{j=1}^{k} (R_{ij})^2$	644.5	829.0
$B_F = \frac{1}{n} \sum_{j=1}^{k} R_j^2$	529.86	703.07
$T_F = \frac{(n-1)(B_F - (nk(k+1)^2)/4)}{A_F - B_F}$	0.8474	0.6585
$F_\alpha((k-1),(n-1)(k-1))$	$F_{0.05}(3,60) = 8.53$	$F_{0.05}(3,60) = 8.53$
Conclusion:	**Do not reject** the null hypothesis that all 4 p_TR heuristics are equal in performance for CVRP at $\alpha = 0.05$.	**Do not reject** the null hypothesis that all 4 p_TR heuristics are equal in performance for CDVRP at $\alpha = 0.05$.

Table 7.17: Computational results for the Friedman

We further carry out the Friedman test on four groups according to various θ, i.e. the tightness of the vehicle capacity restrictions. The null hypothesis is H_0: There is no difference between the various heuristics for different θ (= 0, 0.33, 0.67, 1.0) on the CVRP/CDVRP. The final solutions are presented in Table

	CDVRP				CVRP			
θ	A_F	B_F	T_F	*Conclusion*	A_F	B_F	T_F	*Conclusion*
0.0	204.5	184.5	2.85	Do not reject H_0	207.5	182.93	1.94	Do not reject H_0
0.33	209.5	179.36	0.87	Do not reject H_0	185.5	175.35	0.21	Do not reject H_0
0.67	207.5	177.07	0.41	Do not reject H_0	191.5	178.71	1.74	Do not reject H_0
1.0	207.5	175.21	0.04	Do not reject H_0	-	-	-	-

Table 7.18: Computational results on Friedman for different θ

7.18 according to different θ for CVRP and CDVRP. The hypothesis is rejected if the test statistic $T_F > F_{0.05}(3,18) = 8.68$. The table shows that the hypothesis is not rejected for all cases. That is there is no difference between various p_TR in their performance on problems with various θ for both CVRP and CDVRP.

In the next section we fit the Weibull distribution to the objective function values of local optimum solutions generated by the heuristics.

7.3.2 Fitting of Weibull Distribution (Randomly Generated Problems)

The purpose of this section is to investigate for a point estimate and a confidence interval for the optimal objective function value. This provides us with a tool to measure the solutions and allows us to evaluate the performance of each heuristic. In particular, we fit the Weibull distribution to the local optimal objective function values.

We take $r = 6$ independent samples, each of size $q = 5$ and let $v = min\{z_i : 1 \leq i \leq 6\}$ be the smallest value. The Weibull cumulative distribution can be rewritten as an equation of a straight line as (7.4) which can be solved by the least square approach. Golden's (1977, 1978) least square approach was followed such that the parameter that yields the smallest Kolmogorov- Smirnov (K-S) statistic D is selected. The modified form of D by Stephens, (7.5) is used at $\alpha = 0.05$ level of significance and the rejection starts at 1.358. For details of this subject refer to Appendix A.

Tables 7.19 - 7.26 show the result obtained from fitting the Weibull distribution to the objective function values of the solutions generated by the heuristics for the CVRP and CDVRP. The results include the estimated lower bound, $\tilde{a}$, the estimated parameters for the Weibull distribution, $\tilde{b}$ and $\tilde{c}$, the $100(1 - \exp^{-r})\%$ confidence interval, the performance measure and the deviation of the estimated solution from the best known solutions.

We summarise the results in Tables 7.27 and 7.28 for CVRP and CDVRP. The second column shows the number of times the confidence interval includes the best known lower bound obtained. The deviation (Column 3) refers to the

$$\frac{\text{estimated solution - best known solution}}{\text{best known solution}}. \tag{7.8}$$

Size of Problems	θ	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{v}$	*Proportion Deviation*
					Lower	*Upper*		
60	0	-	-	-	-	-	-	-
60	0.33	7016.47	79.7529	1.5109	6975.71	7055.47	0.0113	-0.0055
60	0.67	6507.05	50.2491	28.3568	6503.79	6554.04	0.0077	-0.0071
70	0	41260.95	227.7451	1.3831	41083.20	41310.95	0.0055	0.0014
70	0.33	7525.8	86.0876	3.2097	7489.71	7575.80	0.0114	-0.0052
70	0.67	6480.05	94.4062	1.7002	6435.64	6530.04	0.0145	-0.0076
80	0	49187.42	148.9167	1.2871	49088.50	49237.41	0.003	0.0079
80	0.33	8057.12	62.3069	7.4898	8044.81	8107.11	0.0077	-0.0061
80	0.67	7175.35	54.212	2.7014	7150.14	7204.35	0.0075	-0.0025
90	0	49225.49	172.2354	0.8771	49103.25	49275.49	0.0035	0.0143
90	0.33	8440.16	100.8358	2.6762	8389.32	8490.16	0.0119	-0.0058
90	0.67	7590.16	135.6285	1.8007	7504.52	7640.15	0.0178	-0.0065
100	0	53101.21	211.4901	1.2168	52939.72	53151.21	0.004	0.0176
100	0.33	7994.25	97.7041	2.3825	7946.54	8044.25	0.0121	-0.0062
100	0.67	7834.73	154.5864	2.5388	7730.14	7884.72	0.0196	-0.0063
110	0	60123.6	863.1718	0.9099	59310.42	60173.59	0.0143	-0.0003
110	0.33	8753.18	51.2764	1.21	8718.90	8770.17	0.0058	-0.0019
110	0.67	8317.18	101.5639	3.0611	8265.61	8367.17	0.0121	-0.0002
120	0	67193.66	100.653	3.6412	67143.01	67243.66	0.0015	0.01137
120	0.33	9328.96	73.3284	5.0229	9305.63	9378.96	0.0078	-0.0053
120	0.67	8701.35	185.1985	1.7903	8566.15	8751.34	0.0212	-0.0007

Table 7.19: Estimates of parameters of the Weibull fitting for 3_TR on CVRP

Size of Problems	θ	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{v}$	*Proportion Deviation*
					Lower	*Upper*		
60	0	32832	259.6315	1.054	32622.36	32881.99	0.0079	-0.0015
60	0.33	7005.47	138.4675	1.462	6917.00	7055.47	0.0196	-0.0070
60	0.67	6508.05	47.7409	24.3611	6506.30	6554.04	0.0073	-0.0070
70	0	41155.7	96.7274	1.4542	41104.97	41201.70	0.0023	-0.0011
70	0.33	7515.48	98.1998	3.6163	7467.28	7565.48	0.013	-0.0066
70	0.67	6480.05	77.5151	2.0577	6452.53	6530.04	0.0119	-0.0076
80	0	48749.15	353.0742	0.9995	48446.07	48799.14	0.0072	-0.0010
80	0.33	8061.12	54.8991	4.2321	8052.22	8107.11	0.0068	-0.0056
80	0.67	7183.85	25.8015	1.1185	7168.04	7193.85	0.0036	-0.0013
90	0	48832.88	560.4262	0.8917	48322.45	48882.88	0.0115	0.0062
90	0.33	8440.16	90.9332	3.3029	8399.22	8490.16	0.0107	-0.0058
90	0.67	7590.16	110.2887	2.0693	7529.86	7640.15	0.0144	-0.0065
100	0	52270.05	1026.2986	0.9248	51293.75	52320.05	0.0196	0.0017
100	0.33	8000.25	61.6653	3.0865	7982.58	8044.25	0.0077	-0.0054
100	0.67	7834.73	135.5961	2.0571	7749.13	7884.72	0.0172	-0.0063
110	0	60092.05	593.4003	1.0071	59548.65	60142.05	0.0099	-0.0008
110	0.33	8720.18	77.2905	3.6867	8692.88	8770.17	0.0088	-0.0057
110	0.67	8284.14	126.8874	2.9076	8207.25	8334.14	0.0152	-0.0041
120	0	67144.95	107.6324	1.6489	67087.31	67194.94	0.0016	0.0106
120	0.33	9334.57	75.4683	5.0368	9309.09	9384.56	0.008	-0.0047
120	0.67	8682.47	177.9002	2.5597	8554.57	8732.47	0.0204	-0.0029

Table 7.20: Estimates of parameters of the Weibull fitting for 5_TR on CVRP

Size of Problems	θ	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{v}$	*Proportion Deviation*
					Lower	*Upper*		
60	0	-	-	-	-	-	-	-
60	0.33	7009.47	70.3299	1.7271	6985.14	7055.47	0.01	-0.0065
60	0.67	6508.05	47.7409	24.3611	6506.30	6554.04	0.0073	-0.0070
70	0	41226.44	66.7082	2.908	41205.72	41272.43	0.0016	0.0006
70	0.33	7515.48	101.0153	1.8811	7464.46	7565.48	0.0134	-0.0066
70	0.67	6510.05	32.7692	1.7152	6497.28	6530.04	0.005	-0.0030
80	0	49086.33	495.0295	1.1642	48641.30	49136.33	0.0101	0.0058
80	0.33	8060.12	48.3678	25.5599	8058.75	8107.11	0.006	-0.0057
80	0.67	7146.85	53.1637	8.7713	7140.68	7193.85	0.0074	-0.0065
90	0	48479.66	1049.3754	0.8518	47480.28	48529.65	0.0216	-0.0010
90	0.33	8447.48	89.9892	3.9561	8407.48	8497.47	0.0106	-0.0050
90	0.67	7606.16	67.7979	1.1916	7572.35	7640.15	0.0089	-0.0044
100	0	52280.26	932.3264	1.0123	51397.93	52330.25	0.0178	0.0019
100	0.33	7994.25	55.4293	12.8244	7988.82	8044.25	0.0069	-0.0062
100	0.67	7893.18	90.5022	3.6342	7852.68	7943.18	0.0114	0.0010
110	0	60828.18	49.6693	53.9354	60826.50	60876.17	0.0008	0.0114
110	0.33	8720.18	63.7785	4.4858	8706.40	8770.17	0.0073	-0.0057
110	0.67	8268.87	161.5308	2.1345	8157.33	8318.86	0.0194	-0.0060
120	0	66988.76	216.174	1.63	66822.58	67038.75	0.0032	0.0082
120	0.33	9364.46	48.9903	1.5993	9335.46	9384.45	0.0052	-0.0015
120	0.67	8657.93	239.2235	1.6713	8468.70	8707.93	0.0275	-0.0057

Table 7.21: Estimates of parameters of the Weibull fitting for 7_TR on CVRP

Size of Problems	θ	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{v}$	*Proportion Deviation*
					Lower	*Upper*		
60	0	-	-	-	-	-	-	-
60	0.33	7005.47	110.067	1.3682	6945.40	7055.47	0.0156	-0.0070
60	0.67	6508.05	48.4268	24.6854	6505.61	6554.04	0.0074	-0.0070
70	0	41232.44	68.4475	2.7512	41203.99	41272.43	0.0017	0.0007
70	0.33	7515.48	117.294	2.3728	7448.18	7565.48	0.0155	-0.0066
70	0.67	6480.05	77.1944	2.2663	6452.85	6530.04	0.0118	-0.0076
80	0	49215.8	334.9975	1.4265	48930.80	49265.80	0.0068	0.0085
80	0.33	8061.12	59.5116	6.6742	8047.60	8107.11	0.0073	-0.0056
80	0.67	7184.85	24.9415	1.0406	7168.90	7193.85	0.0035	-0.0012
90	0	48771.67	428.7371	0.7526	48379.93	48808.66	0.0088	0.0049
90	0.33	8440.16	90.5682	3.1897	8399.59	8490.16	0.0107	-0.0058
90	0.67	7590.16	151.6832	2.131	7488.47	7640.15	0.0199	-0.0065
100	0	52131.08	477.126	0.8467	51703.95	52181.07	0.0091	-0.0009
100	0.33	8039.25	9.7973	2.1028	8034.45	8044.25	0.0012	-0.0006
100	0.67	7917.84	64.4978	6.3772	7903.34	7967.84	0.0081	0.0041
110	0	60806.85	131.8363	2.4681	60725.01	60856.85	0.0022	0.0110
110	0.33	8720.18	105.9156	1.9973	8664.26	8770.17	0.0121	-0.0057
110	0.67	8284.14	123.2983	2.1834	8210.84	8334.14	0.0148	-0.0041
120	0	66387.76	897.3158	0.8743	65540.44	66437.75	0.0135	-0.0007
120	0.33	9334.46	60.9188	7.2489	9323.53	9384.45	0.0065	-0.0047
120	0.67	8701.35	164.393	2.0671	8586.95	8751.34	0.0188	-0.0007

Table 7.22: Estimates of parameters of the Weibull fitting for 9_TR on CVRP

Size of Problems	θ	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{v}$	*Proportion Deviation*
					Lower	*Upper*		
60	0	32836	74.7635	1.5097	32807.23	32881.99	0.0023	-0.0013
60	0.33	10266.01	421.754	0.9205	9894.25	10316.00	0.0409	7.7927-06
60	0.67	10140.48	653.3982	1.0456	9537.08	10190.48	0.0641	0.0268
60	1	11753.82	294.5452	1.2293	11509.27	11803.82	0.025	0.0056
70	0	41260.95	284.3769	1.9617	41026.57	41310.95	0.0069	0.0014
70	0.33	9514.41	86.2639	2.7202	9478.14	9564.41	0.009	-0.0034
70	0.67	10499.62	666.0506	0.8975	9883.56	10549.61	0.0631	-0.0047
70	1	11380.76	1026.7039	0.8266	10404.05	11430.75	0.0898	-0.0040
80	0	49187.42	93.0813	2.186	49144.33	49237.41	0.0019	0.0079
80	0.33	17555.76	2176.704	0.7618	15429.05	17605.75	0.1236	0.0012
80	0.67	13074.3	1121.3043	0.881	12002.99	13124.30	0.0854	0.0435
80	1	11830.61	247.1738	1.158	11633.43	11880.60	0.0208	0.0166
90	0	49266.42	123.4973	1.149	49192.92	49316.42	0.0025	0.0143
90	0.33	12640.61	1353.4794	0.7744	11337.13	12690.60	0.1067	-0.0039
90	0.67	14640.79	1389.1791	0.8634	13301.61	14690.79	0.0946	-0.0034
90	1	11063.89	1191.8186	0.8294	9922.07	11113.89	0.1072	0.0093
100	0	53101.21	285.78	2.3779	52865.43	53151.21	0.0054	0.0176
100	0.33	13591.3	1543.1797	1.0077	12098.12	13641.29	0.1131	0.0040
100	0.67	9510.52	53.5394	9.4833	9504.98	9558.52	0.0056	-0.0050
100	1	9463.19	229.2444	1.4886	9283.94	9513.19	0.0241	-0.0052
110	0	60123.6	739.6823	0.8311	59433.91	60173.59	0.0123	-0.0003
110	0.33	12889.67	793.627	0.7381	12146.04	12939.67	0.0613	0.0314
110	0.67	15659.97	830.676	1.1148	14879.29	15709.96	0.0529	0.0831
110	1	11962.1	2584.9096	0.8298	9427.19	12012.10	0.2152	-0.0041
120	0	67193.66	98.4574	4.6965	67145.20	67243.66	0.0015	0.0113
120	0.33	12102.37	157.2301	1.5256	11995.13	12152.36	0.0129	0.0097
120	0.67	15982.56	1519.9383	0.9651	14512.62	16032.56	0.0948	0.0130
120	1	11722.77	718.7304	0.9648	11054.03	11772.76	0.0611	0.0052

Table 7.23: Estimates of parameters of the Weibull fitting for 3_TR on CDVRP

Size of Problems	θ	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{v}$	*Proportion Deviation*
					Lower	*Upper*		
60	0	32832	344.0684	1.0454	32537.92	32881.99	0.0105	-0.0015
60	0.33	10215.93	248.8247	1.042	10017.10	10265.92	0.0242	-0.0048
60	0.67	9825.03	682.6261	0.8801	9192.39	9875.02	0.0691	-0.0050
60	1	11637.98	859.8125	0.7994	10828.16	11687.97	0.0736	-0.0042
70	0	41155.7	96.7274	1.4542	41104.97	41201.70	0.0023	-0.0011
70	0.33	9497.81	92.9457	3.9838	9454.86	9547.80	0.0097	-0.0052
70	0.67	10611.56	1130.1206	0.8864	9531.44	10661.56	0.106	0.0058
70	1	11423.52	91.6944	0.5039	11334.82	11426.52	0.008	-0.0002
80	0	48749.15	435.7469	1.2074	48363.40	48799.14	0.0089	-0.0010
80	0.33	17983.06	1423.7475	0.8563	16609.31	18033.05	0.079	0.0256
80	0.67	13652.25	218.7179	1.1814	13483.53	13702.24	0.016	0.0896
80	1	11623.64	665.2527	1.0237	11008.38	11673.63	0.057	-0.0011
90	0	48846.63	505.4161	1.2311	48391.21	48896.62	0.0103	0.0056
90	0.33	13765.28	140.2008	2.3457	13675.07	13815.27	0.0101	0.0846
90	0.67	14916.99	819.225	1.0255	14147.76	14966.98	0.0547	0.0153
90	1	11918.12	314.5135	0.9536	11653.61	11968.12	0.0263	0.0873
100	0	52131.08	964.4125	1.015	51216.66	52181.07	0.0185	-0.0009
100	0.33	14468.54	464.5027	1.0183	14054.03	14518.53	0.032	0.0688
100	0.67	-	-	-	-	-	-	-
100	1	9499.26	80.5015	3.1031	9468.75	9549.25	0.0084	-0.0014
110	0	60092.05	385.5444	0.8448	59756.51	60142.05	0.0064	-0.0008
110	0.33	12610.21	612.3281	0.9053	12047.87	12660.20	0.0484	0.0090
110	0.67	14814.85	1843.535	0.7982	13021.31	14864.84	0.124	0.0246
110	1	13435.23	484.6977	1.1852	13000.53	13485.23	0.0359	0.1184
120	0	67144.95	186.9259	1.6659	67008.01	67194.94	0.0028	0.0106
120	0.33	12111.61	214.7298	0.8194	11928.87	12143.60	0.0177	0.0105
120	0.67	15726.17	1099.077	1.0847	14677.09	15776.16	0.0697	-0.0031
120	1	11704.93	917.4092	0.9053	10837.51	11754.92	0.078	0.0036

Table 7.24: Estimates of parameters of the Weibull fitting for 5_TR on CDVRP

Size of Problems	θ	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{v}$	*Proportion Deviation*
					Lower	*Upper*		
60	0	-	-	-	-	-	-	-
60	0.33	10230.75	1025.3195	0.8753	9255.43	10280.75	0.0997	-0.0034
60	0.67	9866.03	204.8482	0.4886	9670.17	9875.02	0.0207	-0.0009
60	1	11847.62	1232.7283	0.7926	10664.89	11897.62	0.1036	0.0136
70	0	41222.44	83.6615	2.1124	41188.77	41272.43	0.002	0.0005
70	0.33	9533.57	58.8895	2.3297	9503.68	9562.57	0.0062	-0.0014
70	0.67	10519.63	1430.7605	0.5985	9138.86	10569.62	0.1354	-0.0028
70	1	11425.52	42.1803	0.3801	11384.34	11426.52	0.0037	-8.7515E-05
80	0	49086.33	212.4522	1.3025	48923.87	49136.33	0.0043	0.0058
80	0.33	17483.73	1210.8809	0.711	16322.84	17533.72	0.0691	-0.0028
80	0.67	12479.2	1607.7218	0.7508	10921.47	12529.20	0.1283	-0.0039
80	1	11821.4	172.638	2.0542	11698.76	11871.39	0.0145	0.0158
90	0	48520.59	1088.7469	1.0253	47481.83	48570.58	0.0224	-0.0010
90	0.33	13848.53	81.9703	1.1876	13786.55	13868.52	0.0059	0.0912
90,	0.67	16653.74	247.8057	1.4391	16455.93	16703.74	0.0148	0.1336
90	1	11567.68	607.1636	1.0612	11010.51	11617.68	0.0523	0.0553
100	0	52131.08	851.6394	0.8804	51329.43	52181.07	0.0163	-0.0009
100	0.33	13714.57	1069.0346	1.0074	12695.53	13764.57	0.0777	0.0131
100	0.67	9528.17	163.366	0.8513	9404.80	9568.17	0.0171	-0.0031
100	1	9499.26	74.985	2.7081	9474.27	9549.25	0.0079	-0.0014
110	0	60760.58	148.9994	1.9914	60661.58	60810.58	0.0025	0.0102
110	0.33	12447.02	1019.7807	0.6804	11477.24	12497.02	0.0816	-0.0040
110	0.67	14407.82	2393.3653	0.6798	12064.45	14457.82	0.1655	-0.0034
110	1	13015.63	985.0035	0.8683	12080.62	13065.62	0.0754	0.0835
120	0	66969.33	205.0229	1.6802	66814.30	67019.32	0.0031	0.0080
120	0.33	11934.98	291.0574	1.5076	11693.92	11984.97	0.0243	-0.0041
120	0.67	16101.71	734.1469	0.9998	15417.56	16151.71	0.0455	0.0206
120	1	11612.22	626.3576	1.0508	11035.86	11662.22	0.0537	-0.0042

Table 7.25: Estimates of parameters of the Weibull fitting for 7_TR on CDVRP

Size of Problems	θ	$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Confidence Interval*		$\frac{\tilde{b}}{\mu}$	*Proportion Deviation*
					Lower	*Upper*		
60	0	32835	82.9717	1.447	32799.02	32881.996	0.0025	-0.0014
60	0.33	12144.88	228.7438	0.5084	11922.13	12150.87	0.0188	0.1830
60	0.67	9826.83	86.3765	2.2564	9790.45	9876.83	0.0087	-0.0048
60	1	11833.75	899.7178	0.7901	10984.03	11883.75	0.0757	0.0124
70	0	41222.44	78.7776	3.1571	41193.65	41272.43	0.0019	0.0005
70	0.33	9505.19	75.2779	6.3591	9479.91	9555.18	0.0079	-0.0044
70	0.67	10502.55	1290.4041	0.6203	9262.14	10552.54	0.1223	-0.0044
70	1	11376.52	397.7214	0.5352	11028.80	11426.52	0.0348	-0.0043
80	0	49086.33	287.1177	0.9809	48849.21	49136.33	0.0058	0.0058
80	0.33	17872.35	580.2623	0.9853	17342.08	17922.34	0.0324	0.0193
80	0.67	13504.15	210.2248	1.3675	13343.92	13554.15	0.0155	0.0778
80	1	11586.68	439.1059	1.1399	11197.57	11636.68	0.0377	-0.0042
90	0	49266.42	189.828	1.004	49126.59	49316.42	0.0038	0.0143
90	0.33	13192.18	829.577	0.9052	12412.60	13242.17	0.0626	0.0395
90	0.67	16198.51	560.5336	0.9648	15687.97	16248.50	0.0345	0.1026
90	1	10910.87	1389.7355	0.7666	9571.13	10960.87	0.1268	-0.0045
100	0	52131.08	838.1142	0.7328	51342.96	52181.07	0.0161	-0.0009
100	0.33	13526	1003.8331	0.9531	12572.1649	13575.99	0.0739	-0.0007
100	0.67	-	-	-	-	-	-	-
100	1	9477.47	97.124	3.483	9430.3457	9527.46	0.0102	-0.0037
110	0	60827.96	83.1047	2.4947	60794.85	60877.95	0.0014	0.0114
110	0.33	12467.48	954.386	0.8794	11563.09	12517.47	0.0762	-0.0023
110	0.67	14441.64	1242.2467	0.8376	13249.39	14491.63	0.0857	-0.0011
110	1	13647.44	447.5608	0.8016	13249.87	13697.43	0.0327	0.1361
120	0	66387.76	660.3613	0.7293	65777.39	66437.75	0.0099	-0.0007
120	0.33	12102.37	101.809	2.6522	12050.55	12152.36	0.0084	0.0097
120	0.67	16492.77	624.8949	1.0679	15917.87	16542.76	0.0378	0.0454
120	1	12254.62	233.089	1.1986	12071.52	12304.61	0.0189	0.0507

Table 7.26: Estimates of parameters of the Weibull fitting for 9_TR on CDVRP

Heuristic	*Number of times the confidence interval includes the best known solution (Total)*	*Deviation*	*Performance Measure*	
			Min	*Max*
3_TR	16(20)	1.4%	0.03	0.021
5_TR	20(21)	1.06%	0.0016	0.02
7_TR	17(20)	1.1%	0.0008	0.0275
9_TR	16(20)	0.85%	0.0012	0.019

Table 7.27: Summary of the quality of the best known solution for the CVRP

and this is recorded using the largest value (in percentage) among all the problems. The maximum and minimum performance measure for each heuristics are recorded in Column 4.

In the case of CVRP

- 69 times, or about 97% (out of 71), the best known lower bounds are included in the $100(1 - \exp^{-6})\% = 99.75\%$ confidence interval.
- The deviation of the estimated solution and the best known lower bound is small (<1.4%) for all cases.
- Intervals of the performance measure are between 0.8% to 2.75%.
- Empirically one would prefer to use 5_TR for randomly generated problems for the CVRP since this heuristic produces good quality solutions. Note that it has the highest frequency for inclusion of the best known solution in the confidence interval.

Heuristic	*Number of times the confidence interval includes the best known solution (Total)*	*Deviation*	*Performance Measure*	
			Min	*Max*
3_TR	23(28)	8.3%	0.0015	0.2152
5_TR	21(27)	11.8%	0.0023	0.124
7_TR	19(26)	13.3%	0.1655	0.1656
9_TR	16(27)	18.3%	0.0014	0.1223

Table 7.28: Summary of the quality of the best known solution for the CDVRP

In the case of CDVRP:

- 79 times, or about 73% (out of 108), the best known solution is included in the $100(1 - \exp^{-6})\% = 99.75\%$ confidence interval. Note that in some cases the generated solutions are the same as the best known solutions and hence they are not counted in the analysis.

- The deviation of the best known lower bound and the estimated solution is within 18.3%.

- The performance measure is between 0.14% to 21.5%.

- Heuristic 3_TR has the highest frequency for inclusion of the best known solution in the confidence interval and the lowest measurement of the percentage deviation from the best known solution compared to other heuristics. Hence one would prefers to use 3_TR for randomly generated CDVRP problems.

7.3.3 Conclusion

Performing the Friedman test on randomly generated problems produced similar results to that given by the literature problems. That is there is no significant difference between the quality of the solutions generated by the algorithms based on different neighbourhood parameters. The result also suggests that the different neighbourhood parameters has no effect on the quality of the solution with different structure i.e. different capacity tightness.

By the fitting of Weibull distribution, we have successfully estimated the point estimate, $\tilde{a}$, and the confidence interval for the optimal objective function value for each CVRP problem. The deviation of the estimated solution from the best known solution is encouraging for the CVRP case.

In the next chapter we study the complexity of the solution space based on the 199-city problem and the running of the TABUROUTE algorithm.

Chapter 8

Complexity of the Solution Space of a Combinatorial Optimisation Problem

In the past 20 years an enormous number of algorithms have been developed to solve combinatorial optimisation problems. Many of these problems are proven to be $\mathcal{NP}$-hard (Lawler et al., 1985). All the exact algorithms for these hard problems cannot solve large size problems in a reasonable time. As a result, approximate or heuristic algorithms have been proposed to find good solutions to these problems. Recently, several probabilistic algorithms based on the principles of genetic algorithm, neural networks, simulated annealing and tabu search methods have been proposed. These algorithms aim to provide high quality solutions, compared to solutions derived using traditional heuristics, with a modest increase in computing time.

In order to test and compare the algorithms developed, benchmark problems are used. Yet we have little knowledge about the complexity of the solution space of these benchmark problems. This chapter uses the ideas developed in Chapter 5 and extends the work of Chapter 7. The objective of this chapter is to propose a method, using statistical analysis, to evaluate the complexity of the solution

space of a typical combinatorial optimisation problem in the context of a proposed algorithm. The analysis is illustrated using the 199-city VRP from the literature and TABUROUTE algorithm. The 199-city problem is studied because it is one of the common literature problems used to test many VRP algorithms.

The rest of this chapter is divided as follows. Section 8.1 gives the background of the complexity of the solution space of any combinatorial optimisation problem. A general neighbourhood search method is discussed. Section 8.2 discusses the 199-city capacity and distance restricted vehicle routing problem (CDVRP) and introduces the notation used. Section 8.3 examines the searching characteristics of the TABUROUTE and the results are shown in Section 8.4. Some of the results of this chapter have appeared in Achuthan and Chong (1999b).

8.1 Complexity of Solution Space

Formally any combinatorial optimisation problem can be stated as

$$\text{minimise } \{f(x) : x \in S\}$$

where S represents a finite, discrete set of feasible solutions of the problem, and f is a real valued function defined on S. Often it may be difficult to describe a typical element of S. It will be easy however to describe elements of a superset $\mathcal{S}$ of S where $\mathcal{S}$ is also finite and discrete. Thus, a typical element x of $\mathcal{S}$ can be represented using mathematical notation of finite size, such as $x \in B^n$ or a system of linear equations, etc. Furthermore, for a given $x \in \mathcal{S}$, it will be easy to check whether $x \in S$. The objective function $f(x)$ defined over S is extended to $f'(x)$ defined on $\mathcal{S}$ such that $f'(x) = f(x)$ for every $x \in S$. Although the size of $\mathcal{S}$ is finite, but for many combinatorial optimisation problems $\mathcal{S}$ grows exponentially in terms of some size parameters of the problem. For example, for a given set of n symbols $\{1, 2, ..., n\}$, the set $\mathcal{S}$ may be defined as the set of all permutations of the n symbols, so that $\mid \mathcal{S} \mid = n!$. The set of feasible solutions S can be described through some verifiable property for $x \in \mathcal{S}$. In other

words given any x it is easy to verify whether $x \in S$. For such combinatorial optimisation problems, complete enumeration of $\mathcal{S}$ is unrealistic due to the size of the set. Hence many approximate or heuristic algorithms developed for solving the combinatorial optimisation problem are based on local or neighbourhood search where the solution is improved from one solution to another until a certain termination condition is satisfied. Such improvement is accomplished through a well defined move operation from a solution to one of its neighbours.

We refer to the definition of the neighbourhood of x to Section 5.1.1. Recall that the neighbours of x, $N(x)$, is a subset of $\mathcal{S}$ where a suitable neighbourhood structure has been defined. We define x^* as a local minimum of $\mathcal{S}$ if $x^* \in \mathcal{S}$ and $f'(x^*) \leq f'(x)$ for all $x \in N(x^*)$. Let $f_i^*, 1 \leq i \leq \beta$ be all the distinct local minimum objective function values realised by the set of all local minimum points of $\mathcal{S}$. Similarly, $x^* \in S$ is a local minimum of S if $f(x^*) < f(x)$ for all $x \in N(x^*) \cap \mathcal{S}$. Let $g_i^*, 1 \leq i \leq \gamma$ be all the distinct local minimum objective function values realised by the set of all local minimum points of S. We present a general Neighbourhood Search Method in the following.

General Neighbourhood Search Method

Step 1. The given problem is min $\{f(x) : x \in S\}$. Define a superset $\mathcal{S}$ such that $S \subseteq \mathcal{S}$ and extend $f(x)$ to $f'(x)$ on $\mathcal{S}$ with the property $f'(x) = f(x)$ for every $x \in S$.

Step 2. Define the Neighbourhood Structure $N(x)$ for $x \in \mathcal{S}$.

Step 3. (Initialisation) Set $r = 1$ where r denotes the rth attempt to locate a minimum point. Set $R^* = R$ where R^* is the upper limit on the number of attempts to locate a set of local minimum points. Define a stopping rule, denoted by STR, for the search for a local minimum.

Step 4. (Search for a local minimum) Construct an initial random solution, x^r from $\mathcal{S}$. Set $t = 1$ and $x_1 = x^r$ where t denotes the tth iteration in the search for a local minimum.

Step 5. (Iteration t and move operation) Choose $x_{t+1} \in N(x_t)$ satisfying cer-

tain prespecified conditions regarding the objective function values of the neighbours.

Step 6. (Check stopping rule for a local minimum) If the current solution x_{t+1} satisfies the stopping rule STR, then go to **Step 7**. Otherwise set $t = t + 1$ and go to **Step 5**.

Step 7. (Local minimum of rth run) Set $x_r^* = x_{t+1}$ where x_r^* denotes a local minimum point obtained in the rth run of this search method. If $r \geq R^*$ go to **Step 8**. Otherwise set $r = r + 1$ and go to **Step 4**.

Step 8. Note that $x_1^*, ..., x_r^*, ..., x_{R^*}^*$ provide the set of local optimum solutions obtained by the heuristic. Define x^* as the best known solution where $f(x^*) = min\{f(x_r^*) : 1 \leq r \leq R^*\}$. Output x^* and $f(x^*)$. Stop.

For convenience, we have described a very simple version of the above General Neighbourhood Search Method. Many of the heuristics suggested in the literature involve some special features such as diversification, tabu search strategies, etc. as part of the search methods. We assume that all such features can be included as part of the selection criteria in Steps 4 and 5 and the stopping rule in Step 7. Several versions of the Neighbourhood Search Method can be obtained by varying the techniques of defining the superset $\mathcal{S}$, the neighbourhood structure $N(x)$, selection criteria in Steps 4 and 5 and the stopping rule.

The output of an implementation of the General Neighbourhood Search Method on a typical combinatorial optimisation problem provides a set of solution $x_r^*, 1 \leq r \leq R^*$ and the corresponding $f(x_r^*), 1 \leq r \leq R^*$. This realisation can be viewed as a random sample of observations taken from a set of all local minimum points and the corresponding distinct objective function values $f_1^*, ..., f_\beta^*$. Note that some of the observations will be part of the set of feasible solutions S and the corresponding distinct objective function values $g_1^*, ..., g_\gamma^*$. How 'good' is a neighbourhood search method in solving a 'typical instance' of a problem ? It is not easy to answer these questions in an objective way. In the following discussion we outline several important characteristics that may provide an assessment of

the search method on an instance of a problem. We first introduce some notation regarding the unknown population with reference to the given neighbourhood search method and an instance of the problem being solved.

Let the sets of local minimum points of $\mathcal{S}$ be denoted by level sets $\mathcal{S}_\theta^{LM} = \{x \in \mathcal{S} : f'(x) = f_\theta^*\}, 1 \leq \theta \leq \beta$. Similarly, the sets of local minimum points of S are denoted by level sets $S_\theta^{LM} = \{x \in S : f(x) = g_\theta^*\}, 1 \leq \theta \leq \gamma$. Note that at the outset, we have no prior knowledge of γ, β; $g_\theta^*, 1 \leq \theta \leq \gamma$; $f_\theta^*, 1 \leq \theta \leq \beta$; $\mathcal{S}_\theta^{LM}, 1 \leq \theta \leq \beta$ and $S_\theta^{LM}, 1 \leq \theta \leq \gamma$. Furthermore, note that g_θ^* and f_θ^* may not be related. The quality of the superset $\mathcal{S}$ can be assessed by estimating $\mid S \mid$ and $\mid \mathcal{S} \mid$; and $\frac{\mid S \mid}{\mid \mathcal{S} \mid}$, the proportion of the set of feasible solutions. The definition of superset $\mathcal{S}$ can be considered satisfactory when $\frac{\mid S \mid}{\mid \mathcal{S} \mid}$ is close to 1 and it is very unsatisfactory when $\frac{\mid S \mid}{\mid \mathcal{S} \mid}$ is very close to zero.

The complexity of the solution space of the given instance of the problem with reference to the search method is quantified by several characteristics such as

1. The number of distinct local minimum values, β and γ respectively for the superset $\mathcal{S}$ and the feasible set S.

2. The sizes of the level sets of local minimum values, $\mid \mathcal{S}_\theta^{LM} \mid, 1 \leq \theta \leq \beta$ and $\mid S_\theta^{LM} \mid, 1 \leq \theta \leq \gamma$ for the superset $\mathcal{S}$ and the feasible set S respectively.

3. The proportions of local minimum points $\frac{\sum_{\theta=1}^{\beta} \mid \mathcal{S}_\theta^{LM} \mid}{\mid \mathcal{S} \mid}$ and $\frac{\sum_{\theta=1}^{\gamma} \mid S_\theta^{LM} \mid}{\mid S \mid}$ for $\mathcal{S}$ and S respectively.

4. The proportion of feasible local minimum points $\frac{\sum_{\theta=1}^{\gamma} \mid S_\theta^{LM} \mid}{\mid \mathcal{S} \mid}$ in $\mathcal{S}$.

When β, γ, $\sum_{\theta=1}^{\beta} \mid \mathcal{S}_\theta^{LM} \mid$ and $\sum_{\theta=1}^{\gamma} \mid S_\theta^{LM} \mid$ are large integers, the neighbourhood search methods are unlikely to find the optimal solution to the original problem. Researchers use certain benchmark problems to assess any proposed new algorithm. But most of the benchmark problems are hard problems with unknown complexity of solution space. Their suitability to assess a proposed new algorithm

is questionable. In the next section we apply the definitions described above to one such benchmark problem, namely the 199-city CDVRP.

8.2 199-city Capacity and Distance Restricted Vehicle Routing Problem

In this section we study the 199-city literature problem in detail in relation to its structure and the search characteristics of TABUROUTE (Gendreau et al., 1994). Define the following notation:

$$\begin{aligned} \mathcal{S} &= \{x : x \text{ is a VRP route}\} \\ S_C &= \{x : x \in \mathcal{S} \text{ and } x \text{ satisfies capacity restriction}\} \\ S_D &= \{x : x \in \mathcal{S} \text{ and } x \text{ satisfies distance restriction}\} \\ S_{CD} &= S_C \cap S_D \end{aligned}$$

Note that S_{CD} is the set of feasible solutions denoted by S in the previous section. On $\mathcal{S}$ we define a neighbourhood structure based on the insertion methods of types A, B and C introduced by Gendreau et al. (1992) as discussed in Section 6.1. Furthermore the extension of the objective function f from S_{CD}, S_C or S_D to $\mathcal{S}$ can be the same i.e. $f'(\tau) = f(\tau), \forall \tau \in \mathcal{S}$.

Recall that for the general VRP, the problem is to find the minimum cost solution, that is to find $f(\tau^*) = min\{f'(\tau) : \tau \in \mathcal{S}\}$ where $\mathcal{S}$ is the set of all VRP solutions. More precisely the CDVRP can be stated as finding $f(\tau^{**}) = min\{f(\tau) : \tau \in S_{CD}\}$ where

$$S_{CD} = \left\{ \begin{array}{c} \tau = (\tau_1, ..., \tau_m) : \bigcup_{k=1}^{m} \tau_k = \{1, ..., n\}, \tau_k \cap \tau_l = \emptyset, k \neq l, \\ \sum_{i \in \tau_k} q_i \leq Q, d(\tau_k) \leq L, 1 \leq k \leq m \end{array} \right\} \quad (8.1)$$

Note that $\tau = (\tau_1, ..., \tau_m)$ denotes a set of m vehicle tours where $\tau_k = (0, v_{k1}, v_{k2}, ..., v_{kj_k}, 0)$ is the kth tour visiting customers $v_{k1}, v_{k2}, ..., v_{kj_k}$ from the depot 0. For notational convenience, the set of customers visited by tour τ_k is also denoted by τ_k, ignoring the depot. The total distance travelled by tour τ_k is denoted by $d(\tau_k)$.

Note that $\tau_k \cap \tau_l = \emptyset$ for $k \neq l$ and Q and L are the prescribed vehicle capacity and allowable distance for the vehicle tours, $\tau_k, 1 \leq k \leq m$ respectively. Hence we have $S_{CD} \subset \mathcal{S}$ and $f(\tau^{**}) \geq f(\tau^*)$. If $\tau^* \in S_{CD}$ then $f(\tau^{**}) = f'(\tau^*) = f(\tau^*)$.

For a given $x \in \mathcal{S}$, $N(x)$ is the neighbourhood of x and

$N(x) = \{x' \in \mathcal{S} : x'$ can be obtained from x by any one of the insertions type$\}$.

Let the feasible neighbours of x be denoted by $N_f(x) = N(x) \cap S_{CD}$.

Note that a run of the TABUROUTE algorithm can be visualised as starting from an arbitrary initial solution x_1 and travel through solutions $x_2, x_3, ..., x_k^*$ such that $x_i \in N_f(x_{i-1})$, $f(x_i) < f(x_{i-1})$ and $x_k^* \in S_{CD}^* = \cup_{\theta=1}^{\gamma} S_\theta^{LM}$. In an actual realisation of a run of TABUROUTE the sequence $x = x_1, x_2, ..., x_k = x_k^*$ will be a subsequence of the feasible and infeasible solutions travelled by the various iterations of the run. Furthermore, note that at the ith iteration of the rth run of TABUROUTE, we have the associated solution x_i^r and the corresponding neighbourhood sets $N(x_i^r)$ and $N_f(x_i^r)$. Thus the set of solutions scanned through as part of the neighbourhood sets by the rth run of TABUROUTE can be denoted by $N^r = \cup_{i=1}^{k_r} N(x_i^r)$ and $N_f^r = \cup_{i=1}^{k_r} N_f(x_i^r)$ where k_r is the total number of iterations used in the rth run.

Thus, we are particularly interested in answering the following questions for a given algorithm based on a given neighbourhood structure.

1. How many local minima are there in the problem ? That is, what is the size of S_{CD}^* ?

2. How much improvement has been made for a particular run ? That is, what is the magnitude of $f(x_1^r) - f(x_{k_r}^r)$ where x_{k_r} is the terminating solution of k_rth iteration of rth run?

For a large n, it is impossible to find $\mid S_{CD}^* \mid$ by a complete enumerations of S_{CD}. Furthermore each run of TABUROUTE stops at exactly one element of S_{CD}^*. Thus our effort will be to estimate $\mid S_{CD}^* \mid$ through implementing TABUROUTE for several runs, each run starting with a random solution.

We have gathered the solutions of 140 runs of the 199-city problem by TABUROUTE algorithm with neighbourhood parameter 5. We investigate the search characteristics of the TABUROUTE algorithm in the next section.

8.3 Search Characteristics of the TABUROUTE Algorithm for the 199-city Problem

In this section we are interested in finding out the characteristics of the TABU ROUTE algorithm. We run the experiment on the 199-city problem with the neighbourhood parameter $p = 5$ for 140 times with a maximum of 25 vehicles. In each run of the problem, we collect the following data:

- the total number of neighbourhood solutions encountered, $\mid N^r \mid$;
- the number of feasible neighbourhood solutions encountered, $\mid N_f^r \mid = \mid N^r \cap S_{CD} \mid$;
- CPU time (in seconds);
- the starting solutions and final solutions;
- the number of solutions being capacity feasible, $\mid N^r \cap S_C \mid$;
- the number of solutions that are distance feasible, $\mid N^r \cap S_D \mid$;
- the number of solutions with both capacity and distance infeasible, $\mid N^r \mid - \mid N_f^r \mid$;
- cumulating all the runs, the total number of solutions and both capacity and distance feasible solutions can be denoted by N and N_f respectively where $N = \cup_{r=1}^{140} N^r$ and $N_f = \cup_{r=1}^{140} N_f^r$;
- the total number of iterations; and
- the iteration number where the best solution x_k^* was obtained.

TABUROUTE algorithm has the following main random features:

- For each run the starting solution is generated through a random selection of customers into routes.
- At each iteration i of a run, the current solution x_i is improved by the move operation by locating $x_{i+1} \in N(x_i)$ such that $f(x_{i+1}) < f(x_i)$. Note that this operation is of deterministic nature since for a given x_i, the neighbourhood structure $N(x_i)$ and the objective function $f(x)$ are deterministic. However, the complexity of the neighbourhood structure, the enormousness of the number of solutions and the inherent features of the objective function will allow the move operation to be considered as a random movement in the desired direction.
- Thus the final solution x_k^* of each run k can be considered as a random sample from S_{CD}^*.

To understand the consistency and quality of the solutions provided by TABUROUTE, we study the sampling distribution of the objective function value of the starting solutions and the final solutions of various runs. The gap between the objective function values of the final and starting solutions is also examined. A natural question arises: Is the random sample generated by the 140 runs from a Normal population ? The result from the data is summarised in Table 8.1.

	Is the corresponding sample from a Normal distribution ?	*p-value*	*Objective function value*	
			mean	*standard deviation*
starting solution	*No*	0.046	3694.78	29.02
final solution	*No*	0.004	3572.74	45.06
gap	*Yes*	0.13	122.04	52.56

Table 8.1: The result of the Anderson-Darling normality test for various samples

Figure 8.1 provides the normal probability plot and shows the Anderson-Darling normality test results corresponding to Table 8.1. The starting solutions, final

solutions and the gap of the solutions are shown in Table B.22 in Appendix B. From Table 8.1 we can conclude that the improvement of the starting solutions to the final solutions i.e. the gap between the starting solutions to the final solutions follow a normal distribution with mean 122.04 and standard deviation 52.56.

In the following we study the distribution of the objective function values of the starting and final solutions.

Consider a set of r independent samples, each sample of size q, from a parent population bounded below by 'a'. If z_i is the smallest value from sample i, let $v = min\{z_i : 1 \leq i \leq r\}$ be the smallest value. As q gets larger, the distribution of z_i follows a Weibull distribution with 'a' as the location parameter (see Appendix A for details of this subject). The cumulative Weibull distribution function is given by

$$F(x_0) = P(x \leq x_0) = 1 - exp\left[-\left(\frac{x_0 - a}{b}\right)^c\right]$$

for $x_0 \geq a \geq 0, b > 0, c > 0$ where a is the location parameter, b is the scale parameter and c is the shape parameter. The null hypothesis is that the final solutions follow a Weibull distribution. Hence we fit the Weibull distribution to the final solutions obtained from the 140 runs with $r = 28$ and $q = 5$. Table 8.2 shows the result we obtained. We conclude that the final solutions follow a

$\tilde{a}$	$\tilde{b}$	$\tilde{c}$	*Best known solution*	*[confidence interval]*	*Deviation*
3371.79	156.0496	5.9664	3385.85	3315.7372, 3471.7869	-0.0042

Table 8.2: Results obtained from fitting the 140 solutions of the 199-city problem to Weibull distribution

Weibull distribution with estimated parameters $\tilde{a}, \tilde{b}$ and $\tilde{c}$ as shown in Table 8.2. The estimated solution $\tilde{a}$ is 0.4% from the best known solution in the literature.

We proposed that the objective function value of the starting solutions follow a log-normal distribution and fitted the log-normal distribution with μ=8.2146 and σ=0.0078 to the starting solution. The probability density function of the log-normal is given by

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp \frac{-(lnx - \mu)^2}{2\sigma^2}.$$

We use the test statistic, T, the modified form of D, by Stephens (1974) (cited in Scheaffer and McClave (1990))

$$T = D\left(\sqrt{(n)} + 0.12 + \frac{0.11}{\sqrt{(n)}}\right) \tag{8.2}$$

where D is based upon the maximum distance between the estimated distribution and the empirical distribution. At α=0.05 significant level, the null hypothesis is rejected if $T > 1.358$. From our sample, using (8.2) we have $T = 1.0736$. Thus we do not reject the null hypothesis that the starting solutions are a sample drawn from a log-normal distribution.

From the data gathered, we also observed the following:

- The average proportion of feasible neighbourhood solutions for 140 runs, $\frac{\mid N_f(x) \mid}{\mid N(x) \mid}$ is 0.1094 $\approx$ 11% and the standard deviation is 0.0112. This suggests the feasible neighbourhood is rather small.
- The proportion of the feasible solutions in each run, $\frac{\mid N_f^r \mid}{\mid N^r \mid}$, visited in each run is very small (with mean = 0.001473 and standard deviation = 0.000826) which suggested that the algorithm processes a lot of infeasible solutions.
- About 70% of the final solutions are obtained within first 30% of the search time which implies that 70% of the search time is wasted.

In the next section, we attempt to answer the first question posed in Section 8.2.

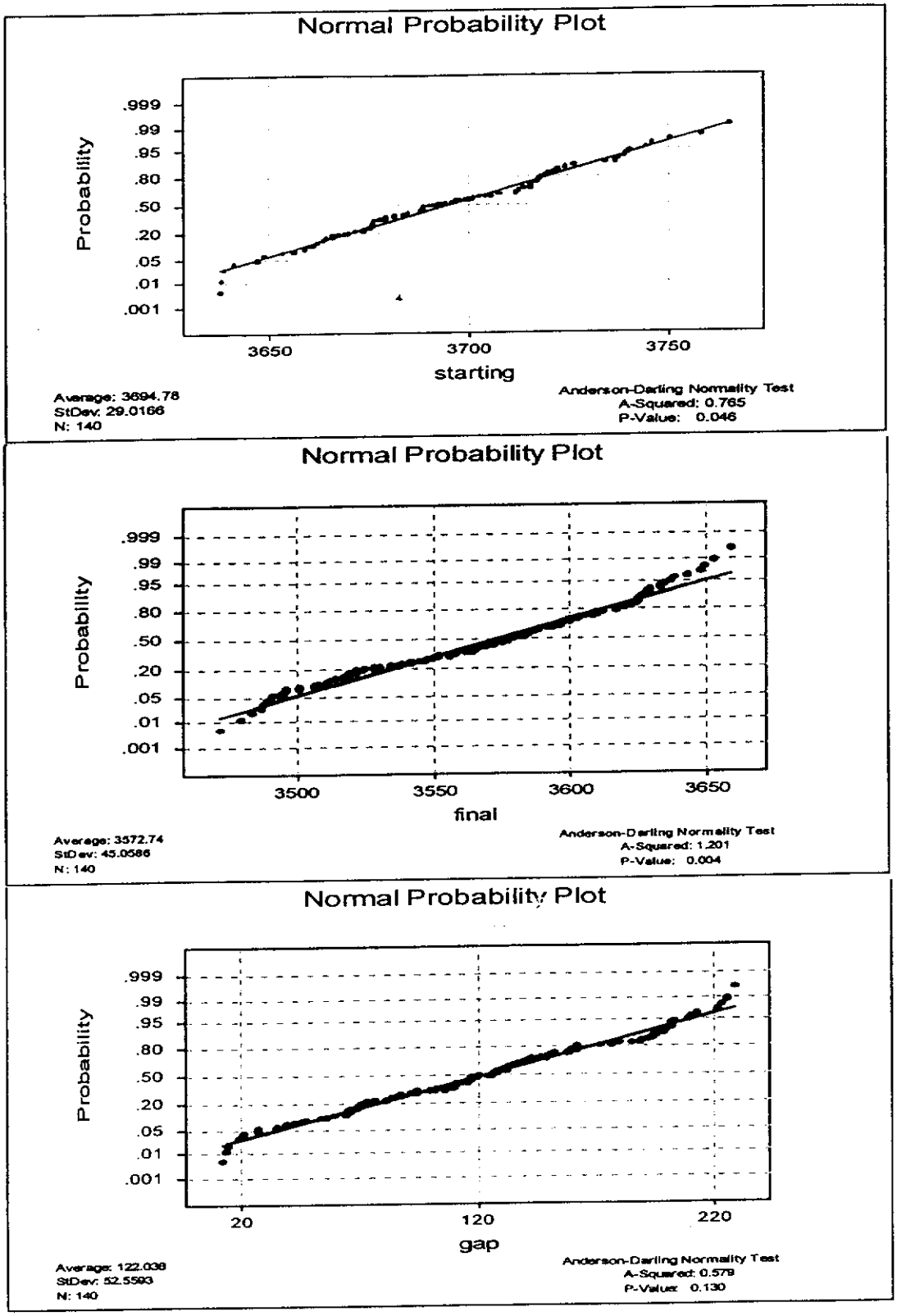


Figure 8.1: Normal probability plot and Anderson-Darling test for the starting, final solution and gap.

8.4 How many local minima are there for the 199-city problem ?

Let T = The total number of solutions for 199-city problem that can be formed using at most 25 vehicles. The number of local minima for the 199-city problem using at most 25 vehicles can be estimated as

$$\frac{140T}{\sum_{i=1}^{140} N^i}. \tag{8.3}$$

We can compute T as follows:

Let $S(k,n)$ = Number of solutions with n customers using k vehicles. There are two cases to be considered:

- Suppose we have allocated $n-1$ customers into $k-1$ vehicles, there is only one way to allocate the last nth customer into kth vehicle.
- Consider that $n-1$ customers are serviced using k vehicles. The last customer n can be inserted to any of the k vehicles and between any of the $n-1$ customers. Let's suppose that it is inserted to the right of i, $1 \leq i \leq n-1$ or to the right of the depot. Hence there are $n-1+k$ ways of inserting the last customer.

Thus the number of feasible solutions for n customers using k vehicles can be written recursively as :

$$S(k,n) = S(k-1,n-1) + (n-1+k)S(k,n-1) \tag{8.4}$$

The total number of solutions for n customers is given by

$$T = \sum_{k=1}^{n} S(k,n). \tag{8.5}$$

Note that T includes some solutions that are infeasible. Since the number of vehicles, k, required is at least $\left\lceil \frac{\sum_i q_i}{Q} \right\rceil$ we can range the values of k from $\left\lceil \frac{\sum_i q_i}{Q} \right\rceil$ to n.

The data obtained from 140 runs was with at most $k = 25$ vehicles, hence for this particular case we have $\left\lceil \frac{\sum_i q_i}{Q} \right\rceil = 14$ and from the calculation of Mathematica 3.0, T is approximated to be

$$T = \sum_{k=14}^{25} S(k, 199) = 1.1661\text{x}10^{382} \tag{8.6}$$

Thus from (8.3) we have the expected number of local minima for the 199-city problem based on at most 25 vehicles is

$$\frac{140T}{\sum_{i=1}^{140} N^i} = \frac{1.1661\text{x}10^{382}\text{x}140}{3.36\text{x}10^9} = 4.8587\text{x}10^{374}$$

where

$$\sum_{i=1}^{140} N^i = 3.36\text{x}10^9$$

is the total number of neighbourhood solutions encountered from the 140 runs.

In the following we propose to give an approximate estimate for the number of feasible local minima for the 199-city problem. Let $P(N_f)$ be the proportion of the neighbourhood feasible solutions and $P(S^*_{CD})$ be the proportion of the local minima in the set of feasible solutions. Then an approximate estimate of the number of feasible local minima for the 199-city problem is $P(N_f)P(S^*_{CD})T$. In order to calculate the number of feasible local minima for the 199-city problem for different k, we execute the algorithm 100 times for each k, $14 \leq k \leq 24$. Note that for $k = 14$ to 17 infeasible solutions were produced and we discarded these values of k. Hence our sampling includes for $k = 18$ to 24 only. Note that for $k = 18$ and 19, some runs are infeasible and so the total number of feasible runs obtained for $k = 18$ to 24 as 504. Hence

$$P(N_f) = \frac{\sum_{k=18}^{24} \sum_{r=1}^{100} N_f^r(k)}{\sum_{k=18}^{24} \sum_{r=1}^{100} N^r(k)} = \frac{1.24\text{x}10^9}{1.17\text{x}10^{10}} = 0.1059$$

and

$$P(S^*_{CD}) = \frac{504}{\sum_{k=18}^{24} \sum_{r=1}^{100} N_f^r(k)} = \frac{504}{1.24\text{x}10^9} = 0.4\text{x}10^{-6}.$$

The expected number of feasible solutions and feasible local minima can be calculated using the Binomial distribution approach since each solution is either 'feasible' or 'infeasible'. This is shown in Table 8.3. The total number of solutions, T, can be obtained using Mathematica 3.0 as follows:

$$T = \sum_{k=18}^{24} S(k, 199) = 1.7526\text{x}10^{381}.$$

Solution	*Proportion*	*Expected number*
Feasible	$P(N_f) = 0.1059$	$TP(N_f) = 1.8563\text{x}10^{380}$
Feasible local minimum	$P(S^*_{CD}) = 0.4\text{x}10^{-6}$	$TP(N_f)P(S^*_{CD}) = 7.4240\text{x}10^{373}$

Table 8.3: Expected number for $P(N_f)$ and $P(S^*_{CD})$

So the 95% upper limit for the number of local minimum S^*_{CD} can be obtained using the table above and $\mu + 4.47\sigma$ where $\mu = TP(N_f)P(S^*_{CD})$ and $\sigma = \sqrt{TP(N_f)P(S^*_{CD})(1 - P(N_f)P(S^*_{CD}))}$. So $\mu + 4.47\sigma \approx 7.4240\text{x}10^{373}$.

8.5 Conclusion

We have little knowledge about the complexity and the solution space of problems in the literature. In this chapter we have attempted to estimate the number of local optimal solutions for a particular problem using a statistical approach.

According to the study above, there are a lot of local minima for the 199-city problem and of all the 140 runs we obtained 140 different solutions. We estimated the expected number of local minima solutions for the 199-city problem with at most 25 vehicles is $4.8587\text{x}10^{374}$. The 95% upper limit for the number of local minima in S^*_{CD} for 199-city problem is $7.4240\text{x}10^{373}$. Furthermore the feasible region of the 199-city problem is rather small. This can be shown by that there is only approximately 11% of the neighbourhood solutions being feasible during the search by TABUROUTE.

Since the number of solutions found is huge, one has to run the algorithm many times to achieve the best minimum. This also suggests that this problem is hard

in the sense that there are too many local minima. We may not find this type of problem in the real world. Hence we feel that this problem should not be used as a benchmark problem to test different algorithms.

Chapter 9

Conclusion and Further Research

This thesis focuses on two classical routing problems, the Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). These problems are easy to describe in words but difficult to solve mathematically. Over the last 50 years, various algorithms both exact and heuristics, have been developed to solve these problems. In spite of the improvement and the use of the sophisticated computer technology and mathematical formulations, there are several instances of these problems that remain unsolved today. There is not one algorithm or method that can handle all instances and/or all sizes of these problems.

Statistical Evaluation for the GENIUS Algorithm

A statistical analysis is carried out to optimally choose the parameters of the GENIUS algorithm proposed by Gendreau, Hertz and Laporte (1992) for the TSP. The analysis is performed using 27 literature problems with sizes ranging from 100 to 532 cities and 20 randomly generated problems with sizes ranging from 100 to 480 cities. In the case of the literature problems, the result shows that 9_CGENI_ABC_US (with neighbourhood parameter $p = 9$, insertions methods type A, type B and type C and using the convex hull technique for constructing the starting solution) is the best heuristic among the 36 heuristics. In the case of randomly generated problems, 9_CGENI_AB_US (with neighbourhood param-

eter $p = 9$, insertions methods type A and type B and using the convex hull technique for constructing the starting solution) proves to be the best heuristic. Furthermore, an estimate of the optimal objective function value together with the corresponding confidence interval is developed using the Weibull distribution. These estimates of the optimal objective function values are within 8.2% of the best objective function value known for the literature problems and 6.5% for the randomly generated problems.

Since the GENIUS algorithm proved to be efficient for large dimension problems, a hybrid heuristic algorithm combining the GENIUS algorithm and the branch and bound method is proposed. This algorithm was tested on the literature problems with sizes ranging from 572 to 724 cities and randomly generated problems with sizes ranging from 500 to 700 cities. The solutions generated were found to be within 2.4% of the best known objective function value for the literature problems.

Statistical Evaluation for the TABUROUTE Algorithm

In the case of VRP, the TABUROUTE algorithm proposed by Gendreau, Hertz and Laporte (1994) was statistically analysed for choosing the best parameter values. The analysis was carried out based on different neighbourhood parameters. These are tested using 14 literature problems with sizes ranging from 50 cities to 199 cities and 49 randomly generated problems with sizes ranging from 60 cities to 120 cities. In both sets of problems, the statistical test accepted the hypothesis that there is no significant difference between the solutions generated by the TABUROUTE for various parameters used.

Complexity of the Solution Space

To assess the performance of a newly designed algorithm, benchmark problems from the literature are often used. However, there is limited documentation or study regarding the complexity of the benchmark problems. In this thesis, an approach to quantify the complexity of an instance of a problem in the context

of specific algorithm is proposed. In particular, the complexity of a benchmark problem is explored based on the concept of neighbourhood structures and the corresponding local and global optimum solutions. This is a main feature of every NP hard combinatorial optimisation problem. In the present context of CDVRP, using Taburoute algorithm, the number of local optimum solutions for the set of feasible solutions of a given problem is estimated. This is demonstrated through the 199-city literature problem.

Data regarding the set of feasible and infeasible solutions were collected based on 140 runs of the Taburoute algorithm on the 199-city problem. This sample includes the number of solutions with distance and capacity feasible, the number of feasible neighbourhood solutions encountered for one run, etc. to examine the feasible region and the number of solution for the problem. The results show that the expected number of local minima solutions for the 199-city problem with at most 25 vehicles is $4.8587 \text{x} 10^{374}$. Furthermore, the feasible region of the 199-city problem is rather small. There is only approximately 11% of the total size of S, the feasible region based on the TABUROUTE algorithm. Since the expected number of solutions found is huge, one has to run the algorithm many times to achieve the best minimum. This suggests that the problem is 'hard' in the sense that there are too many local minima. Therefore this problem should not be used as a benchmark to test any new algorithms.

Thus to summarise, the main findings of this thesis are

- Any heuristic suggested for a NP hard combinatorial optimisation problem of large dimension must be statistically explored to assess its closeness to the optimal objective function value.
- Whenever a heuristic does not provide a reasonable closeness to optimal objective function value, the solution space of the specific instances of the problem must be investigated.
- The benchmark literature problem may not be good indicator to assess the capability of new algorithms.

Further Research

For further research, one might like to characterise the statistical distribution followed by the objective function values of the local optimal solutions. Though the Weibull distribution is an extreme value distribution, it does assume that the underlying random variable has a continuous distribution. In the context of combinatorial optimisation problem the local optimal objective function value follows an extreme discrete distribution.

Assume specific properties on the problem instances and neighbourhood structures, then investigate into possible statistical distribution that can describe the random variables representing the objective function value of the local optimum. Such analysis on literature problem may not yield any universal statistical distribution.

Appendix A

A.1 Friedman Test

Comparison of k heuristics denoted by $H_1, ..., H_k$ can be made through a non-parametric test suggested by Friedman. Friedman test is an extension of the Wilcoxon matched pairs procedure. It is a non-parametric test of homogeneity, similar to the classical ANOVA. The null hypothese is that the k heuristics are all equal in their performance. This test can be described as follows: Let x_{ij} denote the objective function value of the solution obtained by heuristic i on problem j. In order to make the comparison of heuristics independent of the parameters of the problems, define the ratio $r_{ij} = \frac{x_{ij}}{L_j}$ where $L_j = \{$the best known solution of problem $j\}$. For a given problem j, the ratios r_{ij} are arranged in increasing order and ranked from 1 to k. Those ratios giving rise to a tie are given the average rank of the associated indices. Thus let R_{ij} be the rank given to heuristic i on problem j. Furthermore let

$$R_i = \sum_{j=1}^{n} R_{ij}$$

denote the total score associated to heuristic $i = 1, ..., k$. Under the null hypothesis: $H_0 : E[R_1] = E[R_2] = ... = E[R_k]$ (they all attain the same expected total score) where $E[R_i]$ is the expected total score of heuristic i. The test statistic

$$T_F = \frac{(n-1)(B_F - (nk(k+1)^2)/4)}{A_F - B_F}$$

follows the F-distribution with $(k-1)$ and $(n-1)(k-1)$ degrees of freedom, where

$$A_F = \sum_{i=1}^{k}\sum_{j=1}^{n}(R_{ij})^2$$

and

$$B_F = \frac{1}{n}\sum_{i=1}^{k} R_i^2.$$

The null hypothesis is rejected if $T_F > F((k-1),(n-1)(k-1))$ at α significance level. For further details of this test see Golden and Stewart (1985).

A.2 Expected Utility Approach

Friedman test is a test based on location (mean or median) and not the shape or dispersion of the distribution. Thus the results can be less satisfying. Hence Golden and Assad (1984) proposed the expected utility approach. The expected utility approach was proposed to compare two or more heuristics and provides answers to questions such as which heuristic is the most accurate? It is a test to compare two or more heuristics and is concerned with the downside risk and expected accuracy. The procedure searches for a heuristic which performs well on average and rarely performs poorly. The approach can be described in the following:

- Let x_{ij} be the percentage deviation of the obtained solution and the best known solution for heuristic i and problem j.
- Fit a gamma distribution to the histogram of frequency vs. percentage deviation from the solution.
- A risk averse decreasing utility function of the form $u(x) = \gamma - \beta exp^{tx}$ where $\gamma, \beta, t > 0$ is selected. Note that γ and β can be chosen arbitrarily and $t < 1/\tilde{b}$.

- Finally the expected utility for each heuristic is calculated and we select the one that yields the largest value.

The gamma function was used for several reasons: First, it has a simple density function with zero minimum value. In particular,

$$f(x) = \left(\frac{x}{b}\right)^{c-1} exp\left(\frac{-x}{b}\right) / b\Gamma(c)$$

where

$$\Gamma(c) = \int_0^\infty exp^{-u} u^{c-1} du.$$

Secondly, the parameters can be estimated by method of moments:

$$\tilde{b} = \frac{s^2}{\overline{x}} \text{ and } \tilde{c} = \left(\frac{\overline{x}}{s}\right)^2$$

where $\overline{x} = \frac{1}{n}\sum_i x_i$ and $s^2 = \frac{1}{n}\sum_i (x_i - \overline{x})^2$ are the sample mean and sample variance (unadjusted). Thirdly, the moment generating function is in a simple form:

$$E(exp^{tx}) = (1 - bt)^{-c}.$$

So the expected utility function can be easily computed as

$$\int_0^\infty u(x)f(x)dx = \int_0^\infty (\gamma - \beta exp^{tx}) f(x)dx = \gamma - \beta E(exp^{tx}) = \gamma - \beta(1 - bt)^{-c}.$$

The heuristic with the largest value of expected utility function is chosen to be the best heuristic.

A.3 Fitting of Weibull Distribution

Heuristic solutions are evaluated using statistical extreme value theory, Weibull distribution, which was proved by Fisher and Tippet (1928) (cited in Golden,

1978) to obtain the optimal point and confidence interval for the optimal solution. Suppose we have r independent samples, each of size q, from a parent population which is bounded from below by a. If z_i is the smallest value from sample i, let $v = min\{z_i : 1 \leq i \leq r\}$ be the smallest value. As q gets larger, the distribution of z_i follows a Weibull distribution with 'a' as the location parameter. The cumulative distribution of Weibull distribution is given by

$$F(x_0) = P(x \leq x_0) = 1 - exp\left[-\left(\frac{x_0 - a}{b}\right)^c\right]$$

for $x_0 \geq a \geq 0, b > 0, c > 0$ where a is the location parameter, b is the scale parameter and c is the shape parameter.

Estimate for the three parameters can be determined by using the principle of maximum likelihood which was suggested by Harter & Moore (1965), Mann et al. (1974) (cited in Golden, 1978), Zanakis (1977), Golden and Alt (1979) (cited in Golden and Stewart, 1985). Golden (1977, 1978) suggested the least square approach as outline in the following:

- The Weibull cumulative distribution can be written as

$$exp\left[-\left(\frac{x_0 - a}{b}\right)^c\right] = 1 - F(x_0)$$

- Taking the logarithms twice, we have

$$c\ln(x_0 - a) - c\ln b = \ln(-\ln(1 - F(X_0)))$$

  which is an equation of a straight line.

- The equation can also be written as

$$\ln(-\ln(1 - F(X_0))) = \beta_0 + \beta_1 \ln(x_0 - a)$$

  by substituting $\beta_0 = -c\ln b$ and $\beta_1 = c$.

- If 'a' is fixed, the least square method can be used to estimate β_0 and β_1. Then 'b' and 'c' can be estimated. The Kolmogorov Smirnov (K-S) statistic is used to identify the set of parameters.

The K-S test is outline in the following:

- Let $x_1, ..., x_n$ be a sample of observations and reorder them from smallest to largest, $x_{(1)} \leq ... \leq x_{(n)}$.
- Calculate

$$F_n(x) = \begin{cases} \frac{i-1}{n} & \text{if } x_{(i-1)} \leq i \leq x_{(i)}, i = 2, ..., n \\ 1 & \text{if } x \geq x_{(n)}. \end{cases}$$

- Under the null hypothesis, the random variable X follows the distribution funtion $F(x)$. So the K-S statistic D is based on the maximum distance between $F(x)$ and $F_n(x)$, i.e.

$$D = \max_x \mid F(x) - F_n(x) \mid$$

where $D = \max(D^+, D^-)$, $D^+ = \max_{1 \leq i \leq n}[\frac{i}{n} - F(x_i)]$, $D^- = \max_{1 \leq i \leq n}[F(x_i) - \frac{i-1}{n}]$.

- The null hypothesis is rejected if D is large. We used the modified form of D with specified $F(x)$ by Stephens (1974) (cited in Scheaffer and McClave (1990)) which can be given as

$$T = D\left[\sqrt{n} + 0.12 + \frac{0.11}{\sqrt{n}}\right]. \tag{A.1}$$

The rejection region starts at 1.358 for $\alpha = 0.05$ significant level.

The set of parameters which yields the smallest Kolmogorov Smirnov (K-S) statistic D is selected because it is much more sensitive to small changes in 'a' compared to selecting a set of parameter which yields the largest correlation coefficient. An approximate confidence interval of 'a' is given by $(v - \tilde{b}, v)$. Golden and Stewart (1985) pointed out that $\frac{\tilde{b}}{v}$ is a performance measure and the smaller it is the more powerful the heuristic.

Appendix B

B.1 Tables

Heuristic	*a280*	*bier127*	*ch130*	*ch150*	*d198*	*eil101*	*gil262*
3_CGENI_ABC_US	2593	118683	6099	6554	16032	641	2410
3_CGENI_AB_US	2612	118470	6099	6554	15836	641	2392
3_CGENI_AC_US	2605	118313	6118	6555	16457	640	2401
3_CGENI_BC_US	2630	119054	6107	6554	15900	638	2407
5_CGENI_ABC_US	2614	118616	6099	6564	15823	638	2410
5_CGENI_AB_US	2590	118580	6099	6555	15793	638	2404
5_CGENI_AC_US	2608	118856	6113	6570	15797	639	2407
5_CGENI_BC_US	2602	118616	6099	6555	15825	638	2405
7_CGENI_ABC_US	2615	118621	6099	6555	15797	638	2402
7_CGENI_AB_US	2600	118683	6113	6555	15780	638	2409
7_CGENI_AC_US	2608	118626	6113	6555	15780	638	2409
7_CGENI_BC_US	2599	118490	6113	6555	15795	638	2403
9_CGENI_ABC_US	2607	118461	6099	6554	15780	639	2406
9_CGENI_AB_US	2603	118616	6099	6555	15790	640	2398
9_CGENI_AC_US	2600	118616	6099	6554	15780	638	2397
9_CGENI_BC_US	2608	118616	6099	6555	15784	638	2407
3_GENI_ABC_US	2609	118454	6108	6566	15786	641	2420
3_GENI_AB_US	2603	118621	6111	6554	15780	639	2416
3_GENI_AC_US	2607	118501	6125	6555	15832	638	2406
3_GENI_BC_US	2606	118470	6118	6568	15804	638	2420
5_GENI_ABC_US	2603	118647	6113	6555	15787	638	2406
5_GENI_AB_US	2602	118490	6099	6555	15780	641	2400
5_GENI_AC_US	2614	118423	6099	6555	15793	638	2414
5_GENI_BC_US	2612	118647	6113	6545	15780	638	2411
7_GENI_ABC_US	2607	118615	6099	6550	15780	639	2403
7_GENI_AB_US	2598	118693	6113	6555	15780	638	2404
7_GENI_AC_US	2601	118647	6113	6554	15780	641	2411
7_GENI_BC_US	2619	118693	6099	6561	15780	638	2414
9_GENI_ABC_US	2601	118714	6099	6555	15780	638	2408
9_GENI_AB_US	2593	118423	6099	6555	15787	638	2399
9_GENI_AC_US	2615	118616	6113	6555	15780	639	2404
9_GENI_BC_US	2615	118313	6099	6555	15787	638	2407
3_CCAUS	2643	121062	6382	6657	15871	653	2440
5_CCAUS	2643	121062	6213	6657	15871	638	2417
7_CCAUS	2643	121062	6213	6657	15871	647	2415
9_CCAUS	2643	120988	6162	6700	15871	653	2418

Table B.1: The best objective function value obtained over 100 runs for various heuristics j on the given problem i i.e. $\min_{1 \leq r \leq 100} (f_{Hj}^r)_i$ for the TSP on the literature problems

Heuristic	*lin105*	*lin318*	*pr107*	*pr124*	*pr136*	*pr144*	*pr152*
3_CGENI_ABC_US	14380	42691	44303	59030	97477	58564	73931
3_CGENI_AB_US	14403	42743	44303	59030	97120	58537	73880
3_CGENI_AC_US	14403	42391	44303	59030	97129	58554	73880
3_CGENI_BC_US	14380	42587	44303	59076	97133	58607	73818
5_CGENI_ABC_US	14380	42345	44303	59030	97865	58571	73818
5_CGENI_AB_US	14380	42329	44303	59030	96994	58571	73682
5_CGENI_AC_US	14380	42336	44303	59030	97011	58554	73880
5_CGENI_BC_US	14380	42378	44303	59030	97420	58571	73682
7_CGENI_ABC_US	14380	42248	44303	59030	97388	58537	73682
7_CGENI_AB_US	14380	42327	44303	59030	97220	58537	73682
7_CGENI_AC_US	14380	42214	44303	59030	96994	58537	73682
7_CGENI_BC_US	14380	42371	44303	59030	97207	58537	73780
9_CGENI_ABC_US	14380	42357	44303	59030	97229	58537	73682
9_CGENI_AB_US	14380	42395	44303	59030	97179	58537	73786
9_CGENI_AC_US	14380	42265	44303	59030	97207	58537	73682
9_CGENI_BC_US	14380	42494	44303	59030	97388	58537	73682
3_GENI_ABC_US	14403	42541	44303	59076	97333	58537	73818
3_GENI_AB_US	14391	42630	44303	59030	96785	58607	73682
3_GENI_AC_US	14425	42791	44303	59164	96985	58746	73818
3_GENI_BC_US	14403	42741	44303	59030	97259	59110	73818
5_GENI_ABC_US	14380	42544	44303	59030	97207	58588	73682
5_GENI_AB_US	14380	42461	44303	59030	97007	58554	73682
5_GENI_AC_US	14380	42460	44303	59030	97157	58588	73682
5_GENI_BC_US	14380	42337	44303	59030	97401	58588	73682
7_GENI_ABC_US	14380	42201	44303	59030	97198	58537	73682
7_GENI_AB_US	14380	42374	44303	59030	96994	58537	73682
7_GENI_AC_US	14380	42314	44303	59030	97170	58537	73818
7_GENI_BC_US	14380	42306	44303	59030	97007	58537	73682
9_GENI_ABC_US	14380	42436	44303	59030	97007	58537	73682
9_GENI_AB_US	14380	42365	44303	59030	97192	58537	73682
9_GENI_AC_US	14380	42147	44303	59030	96994	58537	73682
9_GENI_BC_US	14380	42300	44303	59030	97170	58537	73682
3_CCAUS	14380	43386	44482	60626	100278	61213	76385
5_CCAUS	14380	43013	44482	60396	99756	60864	74876
7_CCAUS	14380	42842	44482	60347	99756	59434	75578
9_CCAUS	14380	42842	44482	60347	99756	59225	74285

Table B.1: (Cont.) The best objective function value obtained over 100 runs for various heuristics j on the given problem i i.e. $\min_{1 \leq r \leq 100} (f^r_{Hj})_i$ for the TSP on the literature problems

Heuristic	*pr226*	*pr264*	*pr299*	*pr439*	*rat195*	*ts225*	*tsp225*
3_CGENI_ABC_US	80414	49203	48364	108460	2359	126643	3967
3_CGENI_AB_US	80369	49135	48325	108186	2327	126962	3974
3_CGENI_AC_US	80459	49135	48321	108255	2350	126643	3972
3_CGENI_BC_US	80720	49203	48307	108109	2340	126643	3985
5_CGENI_ABC_US	80369	49203	48298	107592	2348	126643	3983
5_CGENI_AB_US	80369	49135	48199	107455	2346	126643	3962
5_CGENI_AC_US	80369	49135	48259	107843	2342	126643	3982
5_CGENI_BC_US	80369	49203	48304	107944	2347	126680	3979
7_CGENI_ABC_US	80369	49135	48197	107444	2348	126643	3959
7_CGENI_AB_US	80369	49135	48199	107929	2345	126643	3969
7_CGENI_AC_US	80369	49135	48193	107638	2345	126643	3977
7_CGENI_BC_US	80369	49135	48259	108075	2363	126962	3971
9_CGENI_ABC_US	80369	49135	48193	107732	2349	126643	3969
9_CGENI_AB_US	80369	49135	48193	107790	2345	126962	3960
9_CGENI_AC_US	80369	49135	48193	107696	2348	126643	3976
9_CGENI_BC_US	80369	49135	48244	107951	2344	126962	3977
3_GENI_ABC_US	80485	49135	48466	107950	2340	126713	3961
3_GENI_AB_US	80467	49203	48371	108340	2349	126643	3975
3_GENI_AC_US	80418	49135	48340	108075	2347	126962	3965
3_GENI_BC_US	80440	49203	48365	109144	2353	127043	3977
5_GENI_ABC_US	80369	49180	48224	107907	2341	126643	3976
5_GENI_AB_US	80369	49203	48244	107647	2362	127444	3969
5_GENI_AC_US	80369	49135	48267	107706	2352	126643	3970
5_GENI_BC_US	80369	49203	48338	107989	2350	126680	3980
7_GENI_ABC_US	80369	49135	48193	108103	2358	127444	3966
7_GENI_AB_US	80369	49135	48233	107967	2346	126643	3969
7_GENI_AC_US	80369	49135	48288	107831	2343	126643	3978
7_GENI_BC_US	80369	49135	48327	107949	2348	126643	3966
9_GENI_ABC_US	80369	49135	48224	107852	2351	126643	3978
9_GENI_AB_US	80369	49135	48254	107841	2347	126643	3968
9_GENI_AC_US	80369	49135	48278	107396	2349	126643	3986
9_GENI_BC_US	80369	49135	48244	108080	2345	126643	3977
3_CCAUS	81238	52326	48506	109430	2373	134748	3992
5_CCAUS	80518	52112	48199	109077	2372	134748	3989
7_CCAUS	80518	52092	48199	109077	2376	133412	3992
9_CCAUS	80518	51908	48199	108455	2376	130428	3992

Table B.1: (Cont.) The best objective function value obtained over 100 runs for various heuristics j on the given problem i i.e. $\min_{1 \leq r \leq 100} (f^r_{Hj})_i$ for the TSP on the literature problems

Heuristic	*u159*	*kroA100*	*pcb442*	*rd100*	*rd400*	*att532*
3_CGENI_ABC_US	42396	21282	51246	7911	15482	27873
3_CGENI_AB_US	42080	21282	51455	7911	15537	728053
3_CGENI_AC_US	42080	21282	51360	7911	15526	28022
3_CGENI_BC_US	42411	21282	51564	7911	15517	28049
5_CGENI_ABC_US	42080	21282	51392	7911	15468	27809
5_CGENI_AB_US	42080	21282	51514	7911	15431	28099
5_CGENI_AC_US	42324	21282	51483	7911	15456	28001
5_CGENI_BC_US	42080	21282	51620	7911	15529	27981
7_CGENI_ABC_US	42080	21282	51305	7911	15492	27990
7_CGENI_AB_US	42080	21282	51527	7911	15492	28013
7_CGENI_AC_US	42080	21282	51273	7911	15475	27907
7_CGENI_BC_US	42080	21282	51229	7911	15515	27946
9_CGENI_ABC_US	42080	21282	51197	7911	15403	27879
9_CGENI_AB_US	42080	21282	51395	7911	15458	27970
9_CGENI_AC_US	42080	21282	51171	7911	15515	27972
9_CGENI_BC_US	42080	21282	51529	7911	15465	27946
3_GENI_ABC_US	42088	21282	51400	7917	15520	28046
3_GENI_AB_US	42080	21282	51605	7917	15500	28135
3_GENI_AC_US	42080	21282	51448	7911	15468	28052
3_GENI_BC_US	42080	21282	51728	7911	15546	28105
5_GENI_ABC_US	42080	21282	51360	7911	15498	27952
5_GENI_AB_US	42080	21282	51215	7911	15501	27948
5_GENI_AC_US	42080	21282	51464	7911	15540	27929
5_GENI_BC_US	42080	21282	51527	7911	15435	28011
7_GENI_ABC_US	42080	21282	51516	7911	15428	27930
7_GENI_AB_US	42080	21282	51165	7911	15440	28029
7_GENI_AC_US	42080	21282	51322	7911	15444	27961
7_GENI_BC_US	42080	21282	51462	7911	15522	27974
9_GENI_ABC_US	42080	21282	51404	7911	15433	27944
9_GENI_AB_US	42080	21282	51422	7911	15501	27965
9_GENI_AC_US	42080	21282	51365	7911	15523	27918
9_GENI_BC_US	42080	21282	51614	7911	15517	27888
3_CCAUS	42929	21282	52018	8090	15842	28187
5_CCAUS	42548	21282	51885	8084	15734	28157
7_CCAUS	42548	21282	51952	8084	15693	28158
9_CCAUS	42548	21282	51849	8084	15682	28159

Table B.1: (Cont.) The best objective function value obtained over 100 runs for various heuristics j on the given problem i i.e. $\min_{1 \leq r \leq 100} (f^r_{Hj})_i$ for the TSP on the literature problems

Heuristic	*a280*	*bier127*	*ch130*	*ch150*	*d198*	*eil101*	*gil262*
3_CGENI_ABC_US	1.0054	1.0034	1	1.0039	1.0159	1.0191	1.0134
3_CGENI_AB_US	1.0128	1.0016	1	1.0039	1.0035	1.0191	1.0059
3_CGENI_AC_US	1.010	1.0003	1.003	1.004	1.0429	1.0175	1.0096
3_CGENI_BC_US	1.0197	1.0065	1.0013	1.0039	1.0076	1.0143	1.0122
5_CGENI_ABC_US	1.0135	1.0028	1	1.0055	1.0027	1.0143	1.0134
5_CGENI_AB_US	1.0042	1.0025	1	1.0041	1.0008	1.0143	1.0109
5_CGENI_AC_US	1.0112	1.0048	1.0022	1.0064	1.0011	1.0158	1.0122
5_CGENI_BC_US	1.0089	1.0028	1	1.0041	1.0028	1.0143	1.0113
7_CGENI_ABC_US	1.0139	1.0028	1	1.0041	1.0011	1.0143	1.0101
7_CGENI_AB_US	1.0081	1.0033	1.0022	1.0041	1	1.01430	1.0130
7_CGENI_AC_US	1.0112	1.0029	1.0022	1.0041	1	1.01430	1.0130
7_CGENI_BC_US	1.0077	1.0017	1.0022	1.0041	1.0009	1.01431	1.0105
9_CGENI_ABC_US	1.0108	1.0015	1	1.0039	1	1.0158	1.0117
9_CGENI_AB_US	1.0093	1.0028	1	1.0041	1.0006	1.0175	1.0084
9_CGENI_AC_US	1.0081	1.0028	1	1.0039	1	1.0143	1.0079
9_CGENI_BC_US	1.0112	1.0028	1	1.0041	1.0003	1.0143	1.0122
3_GENI_ABC_US	1.0116	1.0014	1.0014	1.0058	1.0003	1.0191	1.0176
3_GENI_AB_US	1.0093	1.0028	1.0019	1.0039	1	1.0158	1.0159
3_GENI_AC_US	1.0108	1.0018	1.0042	1.0041	1.0032	1.0143	1.0117
3_GENI_BC_US	1.0104	1.0015	1.0031	1.0061	1.0015	1.0143	1.0176
5_GENI_ABC_US	1.0093	1.0030	1.0022	1.0041	1.0004	1.0143	1.0117
5_GENI_AB_US	1.0089	1.0017	1	1.0041	1	1.0191	1.0092
5_GENI_AC_US	1.0135	1.0011	1	1.0041	1.0008	1.0143	1.0151
5_GENI_BC_US	1.0127	1.0030	1.0022	1.0026	1	1.0143	1.0138
7_GENI_ABC_US	1.0108	1.0028	1	1.0033	1	1.0158	1.0105
7_GENI_AB_US	1.0073	1.0034	1.0022	1.0041	1	1.0143	1.0109
7_GENI_AC_US	1.0085	1.0031	1.0022	1.0039	1	1.0191	1.0138
7_GENI_BC_US	1.0155	1.0034	1	1.0050	1	1.0143	1.0151
9_GENI_ABC_US	1.0085	1.0036	1	1.0041	1	1.0143	1.0126
9_GENI_AB_US	1.0054	1.0011	1	1.0041	1.0004	1.0143	1.0088
9_GENI_AC_US	1.0139	1.0028	1.0022	1.0041	1	1.0158	1.0109
9_GENI_BC_US	1.0139	1.0002	1	1.0041	1.0004	1.0143	1.0121
3_CCAUS	1.0248	1.0235	1.0464	1.0197	1.0057	1.0381	1.0260
5_CCAUS	1.0248	1.0235	1.0186	1.0197	1.0057	1.0143	1.0164
7_CCAUS	1.0248	1.0235	1.0186	1.0197	1.0057	1.02861	1.0155
9_CCAUS	1.0248	1.0228	1.0103	1.0263	1.0057	1.03815	1.0168

Table B.2: Ratio for the TSP on 27 literature problems for Case 1

Heuristic	*lin105*	*lin318*	*pr107*	*pr124*	*pr136*	*pr144*	*pr152*
3_CGENI_ABC_US	1	1.0158	1	1	1.0072	1.0004	1.00336
3_CGENI_AB_US	1.0015	1.01698	1	1	1.0035	1	1.0026
3_CGENI_AC_US	1.0016	1.0086	1	1	1.0036	1.0002	1.0026
3_CGENI_BC_US	1	1.0133	1	1.0007	1.0037	1.0011	1.0018
5_CGENI_ABC_US	1	1.0075	1	1	1.0112	1.0005	1.0018
5_CGENI_AB_US	1	1.0071	1	1	1.0022	1.0005	1
5_CGENI_AC_US	1	1.0073	1	1	1.0024	1.0002	1.0026
5_CGENI_BC_US	1	1.0083	1	1	1.0066	1.0005	1
7_CGENI_ABC_US	1	1.0052	1	1	1.0063	1	1
7_CGENI_AB_US	1	1.0071	1	1	1.0046	1	1
7_CGENI_AC_US	1	1.0081	1	1	1.0022	1	1
7_CGENI_BC_US	1	1.0078	1	1	1.0044	1	1.0013
9_CGENI_ABC_US	1	1.0087	1	1	1.0047	1	1
9_CGENI_AB_US	1	1.0056	1	1	1.0042	1	1.0014
9_CGENI_AC_US	1	1.0110	1	1	1.0044	1	1
9_CGENI_BC_US	1	1.0110	1	1	1.0063	1	1
3_GENI_ABC_US	1.0015	1.0122	1	1.0008	1.0057	1	1.0018
3_GENI_AB_US	1.0007	1.0143	1	1	1.0001	1.0011	1
3_GENI_AC_US	1.0031	1.0181	1	1.0022	1.0022	1.0035	1.0018
3_GENI_BC_US	1.0015	1.0169	1	1	1.0050	1.0097	1.0018
5_GENI_ABC_US	1	1.0122	1	1	1.0044	1.0008	1
5_GENI_AB_US	1	1.0103	1	1	1.0024	1.0002	1
5_GENI_AC_US	1	1.0102	1	1	1.0039	1.0008	1
5_GENI_BC_US	1	1.0073	1	1	1.0064	1.0008	1
7_GENI_ABC_US	1	1.0041	1	1	1.0044	1	1
7_GENI_AB_US	1	1.0082	1	1	1.0022	1	1
7_GENI_AC_US	1	1.0067	1	1	1.0041	1	1.0018
7_GENI_BC_US	1	1.0065	1	1	1.0024	1	1
9_GENI_ABC_US	1	1.0096	1	1	1.0024	1	1
9_GENI_AB_US	1	1.0079	1	1	1.0043	1	1
9_GENI_AC_US	1	1.0028	1	1	1.0022	1	1
9_GENI_BC_US	1	1.0064	1	1	1.0041	1	1
3_CCAUS	1	1.0322	1.0040	1.0270	1.0362	1.0457	1.0366
5_CCAUS	1	1.0234	1.0040	1.0231	1.0308	1.0397	1.0162
7_CCAUS	1	1.0193	1.0040	1.0223	1.0308	1.0153	1.0257
9_CCAUS	1	1.0193	1.0040	1.0223	1.0308	1.0117	1.0081

Table B.2: (Cont.) Ratio for the TSP on 27 literature problems for Case 1

Heuristic	*pr226*	*pr264*	*pr299*	*pr439*	*rat195*	*ts225*	*tsp225*
3_CGENI_ABC_US	1.0005	1.0013	1.0035	1.0116	1.0154	1	1.0122
3_CGENI_AB_US	1	1	1.0027	1.0090	1.0017	1.0025	1.0140
3_CGENI_AC_US	1.0011	1	1.0026	1.0096	1.0116	1	1.0135
3_CGENI_BC_US	1.0043	1.0013	1.0024	1.0083	1.0073	1	1.0168
5_CGENI_ABC_US	1	1.0013	1.0022	1.0034	1.0107	1	1.0163
5_CGENI_AB_US	1	1	1.0001	1.0022	1.0099	1	1.0109
5_CGENI_AC_US	1	1	1.0014	1.0058	1.0082	1	1.0160
5_CGENI_BC_US	1	1.0013	1.0023	1.0067	1.0103	1.0002	1.0153
7_CGENI_ABC_US	1	1	1.0001	1.0021	1.0107	1	1.0102
7_CGENI_AB_US	1	1	1.0001	1.0066	1.0094	1	1.0127
7_CGENI_AC_US	1	1	1.0000	1.0039	1.0094	1	1.0147
7_CGENI_BC_US	1	1	1.0014	1.0080	1.0172	1.0025	1.0132
9_CGENI_ABC_US	1	1	1.0000	1.0048	1.0111	1	1.0127
9_CGENI_AB_US	1	1	1.0000	1.0053	1.0094	1.0025	1.0104
9_CGENI_AC_US	1	1	1.0000	1.0044	1.0107	1	1.0145
9_CGENI_BC_US	1	1	1.0010	1.0068	1.0090	1.0025	1.0147
3_GENI_ABC_US	1.0014	1	1.0057	1.0068	1.0073	1.0005	1.0107
3_GENI_AB_US	1.0012	1.0013	1.0037	1.0104	1.0111	1	1.0142
3_GENI_AC_US	1.0006	1	1.0030	1.0080	1.0103	1.0025	1.0117
3_GENI_BC_US	1.0008	1.0013	1.0036	1.0179	1.0129	1.0031	1.0147
5_GENI_ABC_US	1	1.0009	1.0006	1.0064	1.0077	1	1.0145
5_GENI_AB_US	1	1.0013	1.0010	1.0040	1.0167	1.0063	1.0127
5_GENI_AC_US	1	1	1.0015	1.0045	1.0124	1	1.0130
5_GENI_BC_US	1	1.0013	1.0030	1.0072	1.0116	1.0002	1.0155
7_GENI_ABC_US	1	1	1.0008	1.0069	1.0150	1.0063	1.0119
7_GENI_AB_US	1	1	1.0008	1.0069	1.0099	1	1.0127
7_GENI_AC_US	1	1	1.0020	1.0057	1.0086	1	1.0150
7_GENI_BC_US	1	1	1.0028	1.0068	1.0107	1	1.0119
9_GENI_ABC_US	1	1	1.0006	1.0059	1.0120	1	1.0150
9_GENI_AB_US	1	1	1.0013	1.0058	1.0103	1	1.0125
9_GENI_AC_US	1	1	1.0018	1.0016	1.0111	1	1.0170
9_GENI_BC_US	1	1	1.0011	1.0080	1.0094	1	1.0147
3_CCAUS	1.0108	1.0649	1.0065	1.0206	1.0215	1.0639	1.0186
5_CCAUS	1.0018	1.0605	1.0001	1.0173	1.0210	1.0639	1.0178
7_CCAUS	1.0018	1.0601	1.0002	1.0173	1.0228	1.0534	1.0186
9_CCAUS	1.0018	1.0564	1.0001	1.0115	1.0228	1.0298	1.0186

Table B.2: (Cont.) Ratio for the TSP on 27 literature problems for Case 1

Heuristic	*u159*	*kroA100*	*pcb442*	*rd100*	*rd400*	*att532*
3_CGENI_ABC_US	1.0075	1	1.0092	1	1.0131	1.0067
3_CGENI_AB_US	1	1	1.0133	1	1.0167	1.0132
3_CGENI_AC_US	1	1	1.0114	1	1.0160	1.0121
3_CGENI_BC_US	1.0078	1	1.0154	1	1.0154	1.0131
5_CGENI_ABC_US	1	1	1.0121	1	1.0122	1.0044
5_CGENI_AB_US	1	1	1.0144	1	1.0098	1.0149
5_CGENI_AC_US	1.0057	1	1.0138	1	1.0114	1.0113
5_CGENI_BC_US	1	1	1.0165	1	1.0162	1.0106
7_CGENI_ABC_US	1	1	1.0103	1	1.0138	1.0109
7_CGENI_AB_US	1	1	1.0147	1	1.0138	1.0118
7_CGENI_AC_US	1	1	1.0097	1	1.0126	1.0079
7_CGENI_BC_US	1	1	1.0088	1	1.0153	1.0093
9_CGENI_ABC_US	1	1	1.0082	1	1.0079	1.0069
9_CGENI_AB_US	1	1	1.0121	1	1.0115	1.0102
9_CGENI_AC_US	1	1	1.0077	1	1.0153	1.0103
9_CGENI_BC_US	1	1	1.0147	1	1.0120	1.0093
3_GENI_ABC_US	1.0001	1	1.0122	1.0007	1.0156	1.0130
3_GENI_AB_US	1	1	1.0162	1.0007	1.0143	1.0162
3_GENI_AC_US	1	1	1.0131	1	1.0122	1.0132
3_GENI_BC_US	1	1	1.0187	1	1.0173	1.0151
5_GENI_ABC_US	1	1	1.0114	1	1.0142	1.0096
5_GENI_AB_US	1	1	1.0086	1	1.0143	1.0094
5_GENI_AC_US	1	1	1.0135	1	1.0169	1.0087
5_GENI_BC_US	1	1	1.0147	1	1.0100	1.0117
7_GENI_ABC_US	1	1	1.0145	1	1.0096	1.0088
7_GENI_AB_US	1	1	1.0076	1	1.0104	1.0123
7_GENI_AC_US	1	1	1.0107	1	1.0106	1.0099
7_GENI_BC_US	1	1	1.0134	1	1.0157	1.0104
9_GENI_ABC_US	1	1	1.0123	1	1.0099	1.0093
9_GENI_AB_US	1	1	1.0126	1	1.0143	1.0100
9_GENI_AC_US	1	1	1.0115	1	1.0158	1.0083
9_GENI_BC_US	1	1	1.0164	1	1.0154	1.0072
3_CCAUS	1.0201	1	1.0244	1.0226	1.0367	1.0180
5_CCAUS	1.0111	1	1.0218	1.0218	1.0296	1.0170
7_CCAUS	1.0111	1	1.0231	1.0218	1.0269	1.0170
9_CCAUS	1.0111	1	1.0210	1.0218	1.0262	1.0170

Table B.2: (Cont.) Ratio for the TSP on 27 literature problems for Case 1

Heuristic	*a280*	*bier127*	*ch130*	*ch150*	*d198*	*eil101*	*gil262*	*lin105*
3_CGENI_ABC_US	2.5	26.5	9	6	35	31	24.5	15.5
3_CGENI_AB_US	24.5	7.5	9	6	29	31	1	33.5
3_CGENI_AC_US	15	1.5	30.5	18.5	36	27.5	6	33.5
3_CGENI_BC_US	32	32	18	6	34	11	18.5	15.5
5_CGENI_ABC_US	26.5	16.5	9	29	26	11	24.5	15.5
5_CGENI_AB_US	1	12	9	18.5	20.5	11	11	15.5
5_CGENI_AC_US	21	31	25	32	23.5	24	18.5	15.5
5_CGENI_BC_US	10.5	16.5	9	18.5	27	11	13	15.5
7_CGENI_ABC_US	29	20.5	9	18.5	23.5	11	7	15.5
7_CGENI_AB_US	6.5	26.5	25	18.5	7	11	22.5	15.5
7_CGENI_AC_US	21	22	25	18.5	7	11	22.5	15.5
7_CGENI_BC_US	5	9.5	25	18.5	22	11	8.5	15.5
9_CGENI_ABC_US	18	6	9	6	7	24	15	15.5
9_CGENI_AB_US	13	16.5	9	18.5	19	27.5	3	15.5
9_CGENI_AC_US	6.5	16.5	9	6	7	11	2	15.5
9_CGENI_BC_US	21	16.5	9	18.5	14	11	18.5	15.5
3_GENI_ABC_US	23	5	19	30	15	31	34.5	33.5
3_GENI_AB_US	13	20.5	20	6	7	24	31	31
3_GENI_AC_US	18	11	32	18.5	28	11	15	36
3_GENI_BC_US	16	7.5	30.5	31	25	11	34.5	33.5
5_GENI_ABC_US	13	24	25	18.5	17	11	15	15.5
5_GENI_AB_US	10.5	9.5	9	18.5	7	31	5	15.5
5_GENI_AC_US	26.5	3.5	9	18.5	20.5	11	28.5	15.5
5_GENI_BC_US	24.5	24	25	1	7	11	26.5	15.5
7_GENI_ABC_US	18	13	9	2	7	24	8.5	15.5
7_GENI_AB_US	4	28.5	25	18.5	7	11	11	15.5
7_GENI_AC_US	8.5	24	25	6	7	31	26.5	15.5
7_GENI_BC_US	31	28.5	9	28	7	11	28.5	15.5
9_GENI_ABC_US	8.5	30	9	18.5	7	11	21	15.5
9_GENI_AB_US	2.5	3.5	9	18.5	17	11	4	15.5
9_GENI_AC_US	29	16.5	25	18.5	7	24	11	15.5
9_GENI_BC_US	29	1.5	9	18.5	17	11	18.5	15.5
3_CCAUS	34.5	35	36	34	31.5	35.5	36	15.5
5_CCAUS	34.5	35	33.5	34	31.5	11	32	15.5
7_CCAUS	34.5	35	33.5	34	31.5	34	30	15.5
9_CCAUS	34.5	33	35	36	31.5	35.5	33	15.5

Table B.3: Ranking for the Friedman test for the TSP on the literature problems for Case 1

Heuristic	*lin318*	*pr107*	*pr124*	*pr136*	*pr144*	*pr152*	*pr226*	*pr264*
3_CGENI_ABC_US	29	16.5	15	31	22	32	26	28.5
3_CGENI_AB_US	31	16.5	15	11	9.5	30	13	12
3_CGENI_AC_US	19	16.5	15	12	20	30	29	12
3_CGENI_BC_US	27	16.5	30.5	13	29.5	25.5	35	28.5
5_CGENI_ABC_US	13	16.5	15	32	24	25.5	13	28.5
5_CGENI_AB_US	10	16.5	15	4.5	24	10.5	13	12
5_CGENI_AC_US	11	16.5	15	10	20	30	13	12
5_CGENI_BC_US	18	16.5	15	30	24	10.5	13	28.5
7_CGENI_ABC_US	4	16.5	15	27.5	9.5	10.5	13	12
7_CGENI_AB_US	9	16.5	15	23	9.5	10.5	13	12
7_CGENI_AC_US	3	16.5	15	4.5	9.5	10.5	13	12
7_CGENI_BC_US	16	16.5	15	21	9.5	21	13	12
9_CGENI_ABC_US	14	16.5	15	24	9.5	10.5	13	12
9_CGENI_AB_US	20	16.5	15	17	9.5	22	13	12
9_CGENI_AC_US	5	16.5	15	21	9.5	10.5	13	12
9_CGENI_BC_US	24	16.5	15	27.5	9.5	10.5	13	12
3_GENI_ABC_US	26	16.5	30.5	26	9.5	25.5	31	12
3_GENI_AB_US	28	16.5	15	1	29.5	10.5	30	28.5
3_GENI_AC_US	32	16.5	32	2	31	25.5	27	12
3_GENI_BC_US	30	16.5	15	25	32	25.5	28	28.5
5_GENI_ABC_US	25	16.5	15	21	27	10.5	13	24
5_GENI_AB_US	23	16.5	15	8	20	10.5	13	28.5
5_GENI_AC_US	22	16.5	15	14	27	10.5	13	12
5_GENI_BC_US	12	16.5	15	29	27	10.5	13	28.5
7_GENI_ABC_US	2	16.5	15	19	9.5	10.5	13	12
7_GENI_AB_US	17	16.5	15	4.5	9.5	10.5	13	12
7_GENI_AC_US	8	16.5	15	15.5	9.5	25.5	13	12
7_GENI_BC_US	7	16.5	15	8	9.5	10.5	13	12
9_GENI_ABC_US	21	16.5	15	8	9.5	10.5	13	12
9_GENI_AB_US	15	16.5	15	18	9.5	10.5	13	12
9_GENI_AC_US	1	16.5	15	4.5	9.5	10.5	13	12
9_GENI_BC_US	6	16.5	15	15.5	9.5	10.5	13	12
3_CCAUS	36	34.5	36	36	36	36	36	36
5_CCAUS	35	34.5	35	34	35	34	33	35
7_CCAUS	33.5	34.5	33.5	34	34	35	33	34
9_CCAUS	33.5	34.5	33.5	34	33	33	33	33

Table B.3: (Cont.) Ranking for the Friedman test for the TSP on the literature problems for Case 1

Heuristic	*pr299*	*pr439*	*rat195*	*ts225*	*tsp225*	*u159*	*kroA100*
3_CGENI_ABC_US	32	32	30	11	8	31	18.5
3_CGENI_AB_US	28	28	1	27	17	14.5	18.5
3_CGENI_AC_US	27	29	24.5	11	16	14.5	18.5
3_CGENI_BC_US	26	27	2.5	11	31	32	18.5
5_CGENI_ABC_US	24	4	18.5	11	30	14.5	18.5
5_CGENI_AB_US	9	3	12.5	11	4	14.5	18.5
5_CGENI_AC_US	19.5	13	5	11	29	30	18.5
5_CGENI_BC_US	25	17	15	22.5	27	14.5	18.5
7_CGENI_ABC_US	6	2	18.5	11	1	14.5	18.5
7_CGENI_AB_US	9	16	9.5	11	11.5	14.5	18.5
7_CGENI_AC_US	3	5	9.5	11	22.5	14.5	18.5
7_CGENI_BC_US	19.5	23.5	32	27	15	14.5	18.5
9_CGENI_ABC_US	3	9	22	11	11.5	14.5	18.5
9_CGENI_AB_US	3	10	9.5	27	2	14.5	18.5
9_CGENI_AC_US	3	7	18.5	11	19.5	14.5	18.5
9_CGENI_BC_US	16	20	7	27	22.5	14.5	18.5
3_GENI_ABC_US	35	19	2.5	24	3	29	18.5
3_GENI_AB_US	34	30	22	11	18	14.5	18.5
3_GENI_AC_US	31	23.5	15	27	5	14.5	18.5
3_GENI_BC_US	33	35	28	30	22.5	14.5	18.5
5_GENI_ABC_US	12.5	15	4	11	19.5	14.5	18.5
5_GENI_AB_US	16	6	31	31.5	11.5	14.5	18.5
5_GENI_AC_US	21	8	27	11	14	14.5	18.5
5_GENI_BC_US	30	22	24.5	22.5	28	14.5	18.5
7_GENI_ABC_US	3	26	29	31.5	6.5	14.5	18.5
7_GENI_AB_US	14	21	12.5	11	11.5	14.5	18.5
7_GENI_AC_US	23	11	6	11	25.5	14.5	18.5
7_GENI_BC_US	29	18	18.5	11	6.5	14.5	18.5
9_GENI_ABC_US	12.5	14	26	11	25.5	14.5	18.5
9_GENI_AB_US	18	12	15	11	9	14.5	18.5
9_GENI_AC_US	22	1	22	11	32	14.5	18.5
9_GENI_BC_US	16	25	9.5	11	22.5	14.5	18.5
3_CCAUS	36	36	34	35.5	35	36	18.5
5_CCAUS	9	33.5	33	35.5	33	34	18.5
7_CCAUS	9	33.5	35.5	34	35	34	18.5
9_CCAUS	9	31	35.5	33	33	34	18.5

Table B.3: (Cont.) Ranking for the Friedman test for the TSP on the literature problems for Case 1

Heuristic	*pcb442*	*rd100*	*rd400*	*att532*	R_j	$\sum_i R_{ij}^2$	R_j^2
3_CGENI_ABC_US	6	15.5	14	2	550	14060	302500
3_CGENI_AB_US	19	15.5	30	29	507	12063	257049
3_CGENI_AC_US	10.5	15.5	28	24	540.5	12807.75	292140.25
3_CGENI_BC_US	28	15.5	23.5	27	614.5	16090.75	377610.25
5_CGENI_ABC_US	13	15.5	11.5	1	487	10482	237169
5_CGENI_AB_US	23	15.5	3	30	348	5701.5	121104
5_CGENI_AC_US	22	15.5	8	21	510.5	11100.75	260610.25
5_CGENI_BC_US	31	15.5	29	19	510.5	10783.75	260610.25
7_CGENI_ABC_US	8	15.5	15.5	20	372.5	6431.25	138756.25
7_CGENI_AB_US	25.5	15.5	15.5	23	410.5	7141.75	168510.25
7_CGENI_AC_US	7	15.5	13	5	351	5636	123201
7_CGENI_BC_US	5	15.5	21.5	10.5	441.5	8339.25	194922.25
9_CGENI_ABC_US	3	15.5	1	3	327	4989	106929
9_CGENI_AB_US	14	15.5	9	16	386	6562.5	148996
9_CGENI_AC_US	2	15.5	21.5	17	324	4772	104976
9_CGENI_BC_US	27	15.5	10	10.5	440.5	8051.25	194040.25
3_GENI_ABC_US	15	31.5	25	26	596.5	15592.25	355812.25
3_GENI_AB_US	29	31.5	18	32	570	14266	324900
3_GENI_AC_US	18	15.5	11.5	28	555	13571	308025
3_GENI_BC_US	32	15.5	32	31	681.5	18860.75	464442.25
5_GENI_ABC_US	10.5	15.5	17	13	442	7994.5	195364
5_GENI_AB_US	4	15.5	19.5	12	420.5	8191.25	176820.25
5_GENI_AC_US	21	15.5	31	7	451.5	8855.75	203852.25
5_GENI_BC_US	25.5	15.5	5	22	514	11444	264196
7_GENI_ABC_US	24	15.5	2	8	373	6888.5	139129
7_GENI_AB_US	1	15.5	6	25	369	6148	136161
7_GENI_AC_US	9	15.5	7	14	413.5	77692.75	170982.25
7_GENI_BC_US	20	15.5	26	18	445.5	8847.75	198470.25
9_GENI_ABC_US	16	15.5	4	9	392.5	6707.75	154056.25
9_GENI_AB_US	17	15.5	19.5	15	355.5	5249.75	126380.25
9_GENI_AC_US	12	15.5	27	6	410	7914.5	168100
9_GENI_BC_US	30	15.5	23.5	4	408	7405.5	166464
3_CCAUS	36	36	36	36	919.5	31973.75	845480.25
5_CCAUS	34	34	35	33	835	27328	697225
7_CCAUS	35	34	34	34	856	28234.5	732736
9_CCAUS	33	34	33	35	852.5	28002.25	726756.25
						$A_F = 416181$	$nB_F = 9844476.5$

Table B.3: (Cont.) Ranking for the Friedman test for the TSP on the literature problems for Case 1

Heuristic	*a280*	*bier127*	*ch130*	*ch150*	*d198*	*eil101*	*gil262*
3_CGENI_ABC_US	1.0054	1.0033	1	1.0039	1.0159	1.0190	1.0134
3_CGENI_AB_US	1.0127	1.0015	1	1.0039	1.0035	1.0190	1.0058
3_CGENI_AC_US	1.0100	1.0002	1.0031	1.0041	1.0429	1.0174	1.0096
3_CGENI_BC_US	1.0197	1.0065	1.0013	1.0039	1.0076	1.0143	1.0121
5_CGENI_ABC_US	1.0135	1.0028	1	1.0055	1.0027	1.0143	1.0134
5_CGENI_AB_US	1.0042	1.0025	1	1.0041	1.0008	1.0143	1.0109
5_CGENI_AC_US	1.0112	1.0048	1.0022	1.0064	1.0010	1.0158	1.0121
5_CGENI_BC_US	1.0089	1.0028	1	1.0041	1.0028	1.0143	1.0113
7_CGENI_ABC_US	1.0139	1.0028	1	1.0041	1.0010	1.0143	1.0100
7_CGENI_AB_US	1.0081	1.0033	1.0022	1.0041	1	1.0143	1.0130
7_CGENI_AC_US	1.0112	1.0029	1.0022	1.0041	1	1.0143	1.0130
7_CGENI_BC_US	1.0077	1.0017	1.0022	1.0041	1.0009	1.0143	1.0105
9_CGENI_ABC_US	1.0108	1.0015	1	1.0039	1	1.0158	1.0117
9_CGENI_AB_US	1.0093	1.0028	1	1.0041	1.0006	1.0174	1.0084
9_CGENI_AC_US	1.0081	1.0028	1	1.0039	1	1.0143	1.0079
9_CGENI_BC_US	1.0112	1.0028	1	1.0041	1.0002	1.0143	1.0121

Table B.4: Ratio for the TSP on the 27 literature problems for Case 2

Heuristic	*lin105*	*lin318*	*pr107*	*pr124*	*pr136*	*pr144*	*pr152*
3_CGENI_ABC_US	1	1.0157	1	1	1.0072	1.0004	1.0033
3_CGENI_AB_US	1.0015	1.0169	1	1	1.0035	1	1.0026
3_CGENI_AC_US	1.0015	1.0086	1	1	1.0036	1.0002	1.0026
3_CGENI_BC_US	1	1.0132	1	1.0007	1.0037	1.0011	1.0018
5_CGENI_ABC_US	1	1.0075	1	1	1.0112	1.0005	1.0018
5_CGENI_AB_US	1	1.0071	1	1	1.0022	1.0006	1
5_CGENI_AC_US	1	1.0073	1	1	1.0024	1.0003	1.0026
5_CGENI_BC_US	1	1.0083	1	1	1.0066	1.0005	1
7_CGENI_ABC_US	1	1.0052	1	1	1.0063	1	1
7_CGENI_AB_US	1	1.0071	1	1	1.0046	1	1
7_CGENI_AC_US	1	1.0044	1	1	1.0022	1	1
7_CGENI_BC_US	1	1.0081	1	1	1.0044	1	1.0013
9_CGENI_ABC_US	1	1.0078	1	1	1.0047	1	1
9_CGENI_AB_US	1	1.0087	1	1	1.0042	1	1.0014
9_CGENI_AC_US	1	1.0056	1	1	1.0044	1	1
9_CGENI_BC_US	1	1.0110	1	1	1.0063	1	1

Table B.4: (Cont.) Ratio for the TSP on the 27 literature problems for Case 2

Heuristic	*pr226*	*pr264*	*pr299*	*pr439*	*rat195*	*ts225*	*tsp225*
3_CGENI_ABC_US	1.0005	1.0013	1.0035	1.0116	1.0154	1	1.0122
3_CGENI_AB_US	1	1	1.0027	1.0090	1.0017	1.0025	1.0140
3_CGENI_AC_US	1.0011	1	1.0026	1.0096	1.0116	1	1.0135
3_CGENI_BC_US	1.0043	1.0013	1.0024	1.0083	1.0073	1	1.0168
5_CGENI_ABC_US	1	1.0013	1.0022	1.0034	1.0107	1	1.0163
5_CGENI_AB_US	1	1	1.0002	1.0022	1.0099	1	1.0109
5_CGENI_AC_US	1	1	1.0014	1.0058	1.0081	1	1.0160
5_CGENI_BC_US	1	1.0013	1.0023	1.0067	1.0103	1.0003	1.0153
7_CGENI_ABC_US	1	1	1.0001	1.0021	1.0107	1	1.0102
7_CGENI_AB_US	1	1	1.0002	1.0066	1.0094	1	1.0127
7_CGENI_AC_US	1	1	1.0000	1.0039	1.0094	1	1.0147
7_CGENI_BC_US	1	1	1.0014	1.0080	1.0172	1.0025	1.0132
9_CGENI_ABC_US	1	1	1.0000	1.0048	1.0111	1	1.0127
9_CGENI_AB_US	1	1	1.0000	1.0053	1.0094	1.0025	1.0104
9_CGENI_AC_US	1	1	1.0000	1.0044	1.0107	1	1.0145
9_CGENI_BC_US	1	1	1.0010	1.0068	1.0090	1.0025	1.0147

Table B.4: (Cont.) Ratio for the TSP on the 27 literature problems for Case 2

Heuristic	*u159*	*kroA100*	*pcb442*	*rd100*	*rd400*	*att532*
3_CGENI_ABC_US	1.0075	1	1.0092	1	1.0131	1.0067
3_CGENI_AB_US	1	1	1.0133	1	1.0167	1.0132
3_CGENI_AC_US	1	1	1.0114	1	1.0160	1.0121
3_CGENI_BC_US	1.0078	1	1.0154	1	1.0154	1.0131
5_CGENI_ABC_US	1	1	1.0120	1	1.0122	1.0044
5_CGENI_AB_US	1	1	1.0144	1	1.0098	1.0149
5_CGENI_AC_US	1.0057	1	1.0138	1	1.0114	1.0113
5_CGENI_BC_US	1	1	1.0165	1	1.0162	1.0106
7_CGENI_ABC_US	1	1	1.0103	1	1.0138	1.0109
7_CGENI_AB_US	1	1	1.0147	1	1.0138	1.0118
7_CGENI_AC_US	1	1	1.0097	1	1.0126	1.0079
7_CGENI_BC_US	1	1	1.0088	1	1.0153	1.0093
9_CGENI_ABC_US	1	1	1.0082	1	1.0079	1.0069
9_CGENI_AB_US	1	1	1.0121	1	1.0115	1.0102
9_CGENI_AC_US	1	1	1.0077	1	1.0153	1.0103
9_CGENI_BC_US	1	1	1.0147	1	1.0120	1.0093

Table B.4: (Cont.) Ratio for the TSP on the 27 literature problems for Case 2

Heuristic	*a280*	*bier127*	*ch130*	*ch150*	*d198*	*eil101*	*gil262*	*lin105*
3_CGENI_ABC_US	2	13.5	5.5	3	15	15.5	15.5	7.5
3_CGENI_AB_US	13	3	5.5	3	13	15.5	1	15.5
3_CGENI_AC_US	8	1	16	10	16	13.5	4	15.5
3_CGENI_BC_US	16	16	11	3	14	5.5	11	7.5
5_CGENI_ABC_US	14	8	5.5	15	11	5.5	15.5	7.5
5_CGENI_AB_US	1	5	5.5	10	7	5.5	7	7.5
5_CGENI_AC_US	11	15	13.5	16	9.5	11.5	11	7.5
5_CGENI_BC_US	6	8	5.5	10	12	5.5	8	7.5
7_CGENI_ABC_US	15	11	5.5	10	9.5	5.5	5	7.5
7_CGENI_AB_US	4.5	13.5	13.5	10	2.5	5.5	13.5	7.5
7_CGENI_AC_US	11	12	13.5	10	2.5	5.5	13.5	7.5
7_CGENI_BC_US	3	4	13.5	10	8	5.5	6	7.5
9_CGENI_ABC_US	9	2	5.5	3	2.5	11.5	9	7.5
9_CGENI_AB_US	7	8	5.5	10	6	13.5	3	7.5
9_CGENI_AC_US	4.5	8	5.5	3	2.5	5.5	2	7.5
9_CGENI_BC_US	11	8	5.5	10	5	5.5	11	7.5

Table B.5: Ranking for the Friedman test for TSP on the literature problems for Case 2

Heuristic	*lin318*	*pr107*	*pr124*	*pr136*	*pr144*	*pr152*	*pr226*	*pr264*
3_CGENI_ABC_US	15	8.5	8	15	12	16	14	14.5
3_CGENI_AB_US	16	8.5	8	4	5	14	7	6.5
3_CGENI_AC_US	11	8.5	8	5	10.5	14	15	6.5
3_CGENI_BC_US	14	8.5	16	6	16	11.5	16	14.5
5_CGENI_ABC_US	7	8.5	8	16	14	11.5	7	14.5
5_CGENI_AB_US	5	8.5	8	1.5	14	4.5	7	6.5
5_CGENI_AC_US	6	8.5	8	3	10.5	14	7	6.5
5_CGENI_BC_US	10	8.5	8	14	14	4.5	7	14.5
7_CGENI_ABC_US	2	8.5	8	12.5	5	4.5	7	6.5
7_CGENI_AB_US	4	8.5	8	10	5	4.5	7	6.5
7_CGENI_AC_US	1	8.5	8	1.5	5	4.5	7	6.5
7_CGENI_BC_US	9	8.5	8	8.5	5	9	7	6.5
9_CGENI_ABC_US	8	8.5	8	11	5	4.5	7	6.5
9_CGENI_AB_US	12	8.5	8	7	5	10	7	6.5
9_CGENI_AC_US	3	8.5	8	8.5	5	4.5	7	6.5
9_CGENI_BC_US	13	8.5	8	12.5	5	4.5	7	6.5

Table B.5: (Cont.) Ranking for the Friedman test for the TSP on the literature problems for Case 2

Heuristic	*pr299*	*pr439*	*rat195*	*ts225*	*tsp225*	*u159*	*kroA100*
3_CGENI_ABC_US	16	16	15	6	4	15	8.5
3_CGENI_AB_US	15	14	1	14.5	9	7	8.5
3_CGENI_AC_US	14	15	14	6	8	7	8.5
3_CGENI_BC_US	13	13	2	6	16	16	8.5
5_CGENI_ABC_US	11	3	11	6	15	7	8.5
5_CGENI_AB_US	6.5	2	8	6	3	7	8.5
5_CGENI_AC_US	9.5	8	3	6	14	14	8.5
5_CGENI_BC_US	12	10	9	12	13	7	8.5
7_CGENI_ABC_US	5	1	11	6	1	7	8.5
7_CGENI_AB_US	6.5	9	6	6	5.5	7	8.5
7_CGENI_AC_US	2.5	4	6	6	11.5	7	8.5
7_CGENI_BC_US	9.5	12	16	14.5	7	7	8.5
9_CGENI_ABC_US	2.5	6	13	6	5.5	7	8.5
9_CGENI_AB_US	2.5	7	6	14.5	2	7	8.5
9_CGENI_AC_US	2.5	5	11	6	10	7	8.5
9_CGENI_BC_US	8	11	4	14.5	11.5	7	8.5

Table B.5: (Cont.) Ranking for the Friedman test for the TSP on the literature problems for Case 2

Heuristic	*pcb442*	*rd100*	*rd400*	*att532*	R_j	$\sum_i R_j^2$	R_j^2
3_CGENI_ABC_US	4	8.5	8	2	283.5	3622.25	80372.25
3_CGENI_AB_US	10	8.5	16	15	257	3076	66049
3_CGENI_AC_US	7	8.5	14	13	277.5	3295.75	77006.25
3_CGENI_BC_US	15	8.5	13	14	311.5	4084.75	97032.25
5_CGENI_ABC_US	8	8.5	6	1	253.5	2798.25	64262.25
5_CGENI_AB_US	12	8.5	2	16	183	1564.5	33489
5_CGENI_AC_US	11	8.5	3	11	255	2749.5	65025
5_CGENI_BC_US	16	8.5	15	9	263	2826	69169
7_CGENI_ABC_US	6	8.5	9.5	10	196.5	1714.75	38612.25
7_CGENI_AB_US	13	8.5	9.5	12	215.5	1970.75	46440.25
7_CGENI_AC_US	5	8.5	7	4	187.5	1608.25	35156.25
7_CGENI_BC_US	3	8.5	11.5	5.5	222	2095	49284
9_CGENI_ABC_US	2	8.5	1	3	171.5	1342.75	29412.25
9_CGENI_AB_US	9	8.5	4	7	200.5	1717.25	40200.25
9_CGENI_AC_US	1	8.5	11.5	8	168.5	1253.25	28392.25
9_CGENI_BC_US	14	8.5	5	5.5	226	2134	51076
						$A_F = 37853$	$nB_F = 870978.5$

Table B.5: (Cont.) Ranking for the Friedman test for the TSP on the literature problems for Case 2

Heuristic	*a280*	*bier127*	*ch130*	*ch150*	*d198*	*eil101*	*gil262*
3_GENI_ABC_US	1.0116	1.0014	1.0014	1.0058	1.0004	1.0190	1.0176
3_GENI_AB_US	1.0093	1.0028	1.0019	1.0039	1	1.0158	1.0159
3_GENI_AC_US	1.0108	1.0018	1.0042	1.0041	1.0033	1.0143	1.0117
3_GENI_BC_US	1.0104	1.0015	1.0031	1.0061	1.0015	1.0143	1.0176
5_GENI_ABC_US	1.0093	1.0030	1.0022	1.0041	1.0004	1.0143	1.0117
5_GENI_AB_US	1.0089	1.0017	1	1.0041	1	1.0190	1.0092
5_GENI_AC_US	1.0135	1.0011	1	1.0041	1.0008	1.0143	1.0151
5_GENI_BC_US	1.0127	1.0030	1.0022	1.0026	1	1.0143	1.0138
7_GENI_ABC_US	1.0108	1.0028	1	1.0033	1	1.0158	1.0105
7_GENI_AB_US	1.0073	1.0034	1.0022	1.0041	1	1.0143	1.0109
7_GENI_AC_US	1.0085	1.0030	1.0022	1.0039	1	1.0190	1.0138
7_GENI_BC_US	1.0155	1.0034	1	1.0050	1	1.0143	1.0151
9_GENI_ABC_US	1.0085	1.0036	1	1.0041	1	1.0143	1.0126
9_GENI_AB_US	1.0054	1.0011	1	1.0041	1.0004	1.0143	1.0088
9_GENI_AC_US	1.0139	1.0028	1.0022	1.0041	1	1.0158	1.0109
9_GENI_BC_US	1.0139	1.0002	1	1.0041	1.0004	1.0143	1.0121

Table B.6: Ratio for the TSP on the 27 literature problems for Case 3

Heuristic	*lin105*	*lin318*	*pr107*	*pr124*	*pr136*	*pr144*
3_GENI_ABC_US	1.0015	1.0124	1	1.0007	1.0057	1
3_GENI_AB_US	1.0007	1.0142	1	1	1.0001	1.0011
3_GENI_AC_US	1.0031	1.0181	1	1.0022	1.0022	1.0035
3_GENI_BC_US	1.0015	1.0169	1	1	1.0050	1.0097
5_GENI_ABC_US	1	1.0122	1	1	1.0044	1.0008
5_GENI_AB_US	1	1.0102	1	1	1.0024	1.0003
5_GENI_AC_US	1	1.0102	1	1	1.0039	1.0008
5_GENI_BC_US	1	1.0073	1	1	1.0064	1.0008
7_GENI_ABC_US	1	1.0041	1	1	1.0044	1
7_GENI_AB_US	1	1.0082	1	1	1.0022	1
7_GENI_AC_US	1	1.0067	1	1	1.0041	1
7_GENI_BC_US	1	1.0065	1	1	1.0024	1
9_GENI_ABC_US	1	1.0096	1	1	1.0024	1
9_GENI_AB_US	1	1.0079	1	1	1.0043	1
9_GENI_AC_US	1	1.0028	1	1	1.0022	1
9_GENI_BC_US	1	1.0064	1	1	1.0041	1

Table B.6: (Cont.) Ratio for the TSP on the 27 literature problems for Case 3

Heuristic	*pr152*	*pr226*	*pr264*	*pr299*	*pr439*	*rat195*	*ts225*
3_GENI_ABC_US	1.0018	1.0014	1	1.0057	1.0068	1.0073	1.0005
3_GENI_AB_US	1	1.0012	1.0013	1.0037	1.0104	1.0111	1
3_GENI_AC_US	1.0018	1.0006	1	1.0030	1.0080	1.0103	1.0025
3_GENI_BC_US	1.0018	1.0008	1.0013	1.0036	1.0179	1.0129	1.0031
5_GENI_ABC_US	1	1	1.0009	1.0006	1.0064	1.0077	1
5_GENI_AB_US	1	1	1.0013	1.0011	1.0040	1.0167	1.0063
5_GENI_AC_US	1	1	1	1.0015	1.0045	1.0124	1
5_GENI_BC_US	1	1	1.0013	1.0030	1.0072	1.0116	1.0003
7_GENI_ABC_US	1	1	1	1.0000	1.0082	1.0150	1.0063
7_GENI_AB_US	1	1	1	1.0008	1.0069	1.0099	1
7_GENI_AC_US	1.0018	1	1	1.0020	1.0057	1.0086	1
7_GENI_BC_US	1	1	1	1.0028	1.0068	1.0107	1
9_GENI_ABC_US	1	1	1	1.0006	1.0059	1.0120	1
9_GENI_AB_US	1	1	1	1.0013	1.0058	1.0103	1
9_GENI_AC_US	1	1	1	1.0018	1.0016	1.0111	1
9_GENI_BC_US	1	1	1	1.0010	1.0080	1.0094	1

Table B.6: (Cont.) Ratio for the TSP on the 27 literature problems for Case 3

Heuristic	*tsp225*	*u159*	*kroA100*	*pcb442*	*rd100*	*rd400*	*att532*
3_GENI_ABC_US	1.0107	1.0001	1	1.0122	1.0007	1.0156	1.0130
3_GENI_AB_US	1.0142	1	1	1.0162	1.0007	1.0143	1.0162
3_GENI_AC_US	1.0117	1	1	1.0131	1	1.0122	1.0132
3_GENI_BC_US	1.0147	1	1	1.0187	1	1.0173	1.0151
5_GENI_ABC_US	1.0145	1	1	1.0114	1	1.0142	1.0096
5_GENI_AB_US	1.0127	1	1	1.0086	1	1.0143	1.0094
5_GENI_AC_US	1.0130	1	1	1.0135	1	1.0169	1.0087
5_GENI_BC_US	1.0155	1	1	1.0147	1	1.0100	1.0117
7_GENI_ABC_US	1.0119	1	1	1.0145	1	1.0096	1.0088
7_GENI_AB_US	1.0127	1	1	1.0076	1	1.0104	1.0123
7_GENI_AC_US	1.0150	1	1	1.0107	1	1.0106	1.0099
7_GENI_BC_US	1.0119	1	1	1.0134	1	1.0157	1.0104
9_GENI_ABC_US	1.0150	1	1	1.0123	1	1.0099	1.0093
9_GENI_AB_US	1.0125	1	1	1.0126	1	1.0143	1.01007
9_GENI_AC_US	1.0170	1	1	1.0115	1	1.0158	1.0083
9_GENI_BC_US	1.0147	1	1	1.0164	1	1.0154	1.0072

Table B.6: (Cont.) Ratio for the TSP on the 27 literature problems for Case 3

Heuristic	*a280*	*bier127*	*ch130*	*ch150*	*d198*	*eil101*	*gil262*	*lin105*
3_GENI_ABC_US	11	4	8	15	10	15	15.5	14.5
3_GENI_AB_US	6.5	10	9	3.5	5	12	14	13
3_GENI_AC_US	9.5	7	16	9	16	5.5	6.5	16
3_GENI_BC_US	8	5	15	16	15	5.5	15.5	14.5
5_GENI_ABC_US	6.5	12	12	9	12	5.5	6.5	6.5
5_GENI_AB_US	5	6	4	9	5	15	2	6.5
5_GENI_AC_US	13	2.5	4	9	14	5.5	12.5	6.5
5_GENI_BC_US	12	12	12	1	5	5.5	10.5	6.5
7_GENI_ABC_US	9.5	8	4	2	5	12	3	6.5
7_GENI_AB_US	2	14.5	12	9	5	5.5	4.5	6.5
7_GENI_AC_US	3.5	12	12	3.5	5	15	10.5	6.5
7_GENI_BC_US	16	14.5	4	14	5	5.5	12.5	6.5
9_GENI_ABC_US	3.5	16	4	9	5	5.5	9	6.5
9_GENI_AB_US	1	2.5	4	9	12	5.5	1	6.5
9_GENI_AC_US	14.5	9	12	9	5	12	4.5	6.5
9_GENI_BC_US	14.5	1	4	9	12	5.5	8	6.5

Table B.7: Ranking for the Friedman test for the TSP on the literature problems for Case 3

Heuristic	*lin318*	*pr107*	*pr124*	*pr136*	*pr144*	*pr152*	*pr226*	*pr264*
3_GENI_ABC_US	13	8.5	15	15	5	14.5	16	6
3_GENI_AB_US	14	8.5	7.5	1	14	6.5	15	14.5
3_GENI_AC_US	16	8.5	16	2	15	14.5	13	6
3_GENI_BC_US	15	8.5	7.5	14	16	14.5	14	14.5
5_GENI_ABC_US	12	8.5	7.5	13	12	6.5	6.5	12
5_GENI_AB_US	11	8.5	7.5	6	10	6.5	6.5	14.5
5_GENI_AC_US	10	8.5	7.5	8	12	6.5	6.5	6
5_GENI_BC_US	6	8.5	7.5	16	12	6.5	6.5	14.5
7_GENI_ABC_US	2	8.5	7.5	12	5	6.5	6.5	6
7_GENI_AB_US	8	8.5	7.5	3.5	5	6.5	6.5	6
7_GENI_AC_US	5	8.5	7.5	9.5	5	14.5	6.5	6
7_GENI_BC_US	4	8.5	7.5	6	5	6.5	6.5	6
9_GENI_ABC_US	9	8.5	7.5	6	5	6.5	6.5	6
9_GENI_AB_US	7	8.5	7.5	11	5	6.5	6.5	6
9_GENI_AC_US	1	8.5	7.5	3.5	5	6.5	6.5	6
9_GENI_BC_US	3	8.5	7.5	9.5	5	6.5	6.5	6

Table B.7: (Cont.) Ranking for the Friedman test for the TSP on the 27 literature problems for Case 3

Heuristic	*pr299*	*pr439*	*rat195*	*ts225*	*tsp225*	*u159*	*kroA100*
3_GENI_ABC_US	16	9	1	12	1	16	8.5
3_GENI_AB_US	15	15	9.5	5.5	9	8	8.5
3_GENI_AC_US	13	12	6.5	13	2	8	8.5
3_GENI_BC_US	14	16	14	14	11.5	8	8.5
5_GENI_ABC_US	2.5	7	2	5.5	10	8	8.5
5_GENI_AB_US	5.5	2	16	15.5	6.5	8	8.5
5_GENI_AC_US	8	3	13	5.5	8	8	8.5
5_GENI_BC_US	12	11	11	11	15	8	8.5
7_GENI_ABC_US	1	14	15	15.5	3.5	8	8.5
7_GENI_AB_US	4	10	5	5.5	6.5	8	8.5
7_GENI_AC_US	10	4	3	5.5	13.5	8	8.5
7_GENI_BC_US	11	8	8	5.5	3.5	8	8.5
9_GENI_ABC_US	2.5	6	12	5.5	13.5	8	8.5
9_GENI_AB_US	7	5	6.5	5.5	5	8	8.5
9_GENI_AC_US	9	1	9.5	5.5	16	8	8.5
9_GENI_BC_US	5.5	13	4	5.5	11.5	8	8.5

Table B.7: (Cont.) Ranking for the Friedman test for the TSP on the 27 literature problems for Case 3

Heuristic	*pcb442*	*rd100*	*rd400*	*att532*	R_j	$\sum_i R_j^2$	R_j^2
3_GENI_ABC_US	6	15.5	12	13	296	3820.5	87616
3_GENI_AB_US	14	15.5	8	16	278	3312.5	77284
3_GENI_AC_US	9	7.5	6	14	276	3323	76176
3_GENI_BC_US	16	7.5	16	15	339	4603.5	114921
5_GENI_ABC_US	4	7.5	7	7	217	1980	47089
5_GENI_AB_US	2	7.5	9.5	6	210	2030	44100
5_GENI_AC_US	11	7.5	15	3	222.5	2137.75	49506.25
5_GENI_BC_US	13	7.5	3	11	253	2723.5	64009
7_GENI_ABC_US	12	7.5	1	4	194	1844.5	37636
7_GENI_AB_US	1	7.5	4	12	182.5	1474.25	33306.25
7_GENI_AC_US	3	7.5	5	8	206.5	1900.25	42642.25
7_GENI_BC_US	10	7.5	13	10	221	2111	48841
9_GENI_ABC_US	7	7.5	2	5	191	1604	36481
9_GENI_AB_US	8	7.5	9.5	9	179	1365	32041
9_GENI_AC_US	5	7.5	14	2	203	1911	41209
9_GENI_BC_US	15	7.5	11	1	203.5	1879.25	41412.25
						A_F= 38020	nB_F=874270

Table B.7: (Cont.) Ranking for the Friedman test for the TSP on the 27 literature problems for Case 3

Heuristic	*a280*	*bier127*	*ch130*	*ch150*	*d198*	*eil101*	*gil262*
3_CGENI_ABC_US	0.5428	0.3390	0	0.3982	1.5969	1.9077	1.3456
3_CGENI_AB_US	1.2795	0.1589	0	0.3982	0.3548	1.9077	0.5887
3_CGENI_AC_US	1.0081	0.0262	0.3115	0.4136	4.2902	1.7488	0.9671
3_CGENI_BC_US	1.9775	0.6526	0.1311	0.3982	0.7604	1.4308	1.2195
5_CGENI_ABC_US	1.3571	0.2823	0	0.5514	0.2724	1.4308	1.3456
5_CGENI_AB_US	0.4265	0.2519	0	0.4136	0.0823	1.4308	1.0933
5_CGENI_AC_US	1.1244	0.4852	0.2295	0.6433	0.1077	1.5898	1.2195
5_CGENI_BC_US	0.8918	0.2823	0	0.4136	0.2851	1.4308	1.1354
7_CGENI_ABC_US	1.3958	0.2866	0	0.4136	0.1077	1.4308	1.0092
7_CGENI_AB_US	0.8142	0.3390	0.2295	0.4136	0	1.4308	1.3036
7_CGENI_AC_US	1.1244	0.2908	0.2295	0.4136	0	1.4308	1.3036
7_CGENI_BC_US	0.7754	0.1758	0.2295	0.4136	0.0950	1.4308	1.0513
9_CGENI_ABC_US	1.0856	0.1513	0	0.3982	0	1.5898	1.1774
9_CGENI_AB_US	0.9305	0.2823	0	0.4136	0.0633	1.7488	0.8410
9_CGENI_AC_US	0.8142	0.2823	0	0.3982	0	1.4308	0.7989
9_CGENI_BC_US	1.1244	0.2823	0	0.4136	0.0253	1.4308	1.2195
3_GENI_ABC_US	1.1632	0.1454	0.1475	0.5821	0.0380	1.9077	1.7661
3_GENI_AB_US	0.9305	0.2866	0.1967	0.3982	0	1.5898	1.5979
3_GENI_AC_US	1.0856	0.1851	0.4262	0.4136	0.3295	1.4308	1.1774
3_GENI_BC_US	1.0469	0.1589	0.3115	0.6127	0.1520	1.4308	1.7661
5_GENI_ABC_US	0.9305	0.3085	0.2295	0.4136	0.0443	1.4308	1.1774
5_GENI_AB_US	0.8918	0.1758	0	0.4136	0	1.9077	0.9251
5_GENI_AC_US	1.3571	0.1192	0	0.4136	0.0823	1.4308	1.5138
5_GENI_BC_US	1.2795	0.3085	0.2295	0.2604	0	1.4308	1.3877
7_GENI_ABC_US	1.0856	0.2815	0	0.3370	0	1.5898	1.0513
7_GENI_AB_US	0.7367	0.3474	0.2295	0.4136	0	1.4308	1.0933
7_GENI_AC_US	0.8530	0.3085	0.2295	0.3982	0	1.9077	1.3877
7_GENI_BC_US	1.5509	0.3474	0	0.5055	0	1.4308	1.5138
9_GENI_ABC_US	0.8530	0.3652	0	0.4136	0	1.4308	1.2615
9_GENI_AB_US	0.5428	0.1192	0	0.4136	0.0443	1.4308	0.8830
9_GENI_AC_US	1.3958	0.2823	0.2295	0.4136	0	1.5898	1.0933
9_GENI_BC_US	1.3958	0.0262	0	0.4136	0.0443	1.4308	1.2195
3_CCAUS	2.4815	2.3503	4.6401	1.9761	0.5766	3.8155	2.6072
5_CCAUS	2.4815	2.3503	1.8691	1.9761	0.5766	1.4308	1.6400
7_CCAUS	2.4815	2.3503	1.8691	1.9761	0.5766	2.8616	1.5559
9_CCAUS	2.4815	2.2877	1.0329	2.6348	0.5766	3.8155	1.6820

Table B.8: Percentage deviation of the heuristic solution and the best known solution for the TSP on the literature problems

Heuristic	*lin105*	*lin318*	*pr107*	*pr124*	*pr136*	*pr144*
3_CGENI_ABC_US	0	1.5751	0	0	0.7285	0.0461
3_CGENI_AB_US	0.1599	1.6988	0	0	0.3596	0
3_CGENI_AC_US	0.1599	0.8613	0	0	0.3689	0.0290
3_CGENI_BC_US	0	1.3276	0	0.0779	0.3730	0.1195
5_CGENI_ABC_US	0	0.7518	0	0	1.1294	0.0580
5_CGENI_AB_US	0	0.7137	0	0	0.2294	0.0580
5_CGENI_AC_US	0	0.7304	0	0	0.2469	0.0290
5_CGENI_BC_US	0	0.8303	0	0	0.6696	0.0580
7_CGENI_ABC_US	0	0.5210	0	0	0.6365	0
7_CGENI_AB_US	0	0.7090	0	0	0.4629	0
7_CGENI_AC_US	0	0.4401	0	0	0.2294	0
7_CGENI_BC_US	0	0.8137	0	0	0.4495	0
9_CGENI_ABC_US	0	0.7804	0	0	0.4722	0
9_CGENI_AB_US	0	0.8708	0	0	0.4205	0
9_CGENI_AC_US	0	0.5615	0	0	0.4495	0
9_CGENI_BC_US	0	1.1063	0	0	0.6365	0
3_GENI_ABC_US	0.1599	1.2419	0	0.0779	0.5797	0
3_GENI_AB_US	0.0764	1.4299	0	0	0.0134	0.1195
3_GENI_AC_US	0.3129	1.8130	0	0.2270	0.2201	0.3570
3_GENI_BC_US	0.1599	1.6940	0	0	0.5032	0.9788
5_GENI_ABC_US	0	1.2253	0	0	0.4495	0.0871
5_GENI_AB_US	0	1.0278	0	0	0.2428	0.0290
5_GENI_AC_US	0	1.0254	0	0	0.3978	0.0871
5_GENI_BC_US	0	0.7328	0	0	0.6499	0.0871
7_GENI_ABC_US	0	0.4116	0	0	0.4402	0
7_GENI_AB_US	0	0.8208	0	0	0.2294	0
7_GENI_AC_US	0	0.6781	0	0	0.4112	0
7_GENI_BC_US	0	0.6590	0	0	0.2428	0
9_GENI_ABC_US	0	0.9683	0	0	0.2428	0
9_GENI_AB_US	0	0.7994	0	0	0.4340	0
9_GENI_AC_US	0	0.2807	0	0	0.2294	0
9_GENI_BC_US	0	0.6447	0	0	0.4112	0
3_CCAUS	0	3.2287	0.4040	2.7037	3.6229	4.5714
5_CCAUS	0	2.3412	0.4040	2.3140	3.0835	3.9752
7_CCAUS	0	1.9343	0.4040	2.2310	3.0835	1.5323
9_CCAUS	0	1.9343	0.4040	2.2310	3.0835	1.1753

Table B.8: (Cont.) Percentage deviation of the heuristic solutions and the best known solution for the TSP on the literature problems

Heuristic	*pr152*	*pr226*	*pr264*	*pr299*	*pr439*	*rat195*	*ts225*
3_CGENI_ABC_US	0.3379	0.0559	0.1383	0.3589	1.1667	1.5497	0
3_CGENI_AB_US	0.2687	0	0	0.2780	0.9037	0.1721	0.2518
3_CGENI_AC_US	0.2687	0.1119	0	0.2697	0.9681	1.1622	0
3_CGENI_BC_US	0.1845	0.4367	0.1383	0.2407	0.8319	0.7318	0
5_CGENI_ABC_US	0.1845	0	0.1383	0.2220	0.3497	1.0761	0
5_CGENI_AB_US	0	0	0	0.0166	0.2219	0.9900	0
5_CGENI_AC_US	0.2687	0	0	0.1411	0.5838	0.8179	0
5_CGENI_BC_US	0	0	0.1383	0.2344	0.6780	1.0331	0.0292
7_CGENI_ABC_US	0	0	0	0.0124	0.2117	1.0761	0
7_CGENI_AB_US	0	0	0	0.0166	0.6640	0.9470	0
7_CGENI_AC_US	0	0	0	0.0041	0.3926	0.9470	0
7_CGENI_BC_US	0.1330	0	0	0.1411	0.8002	1.7219	0.2518
9_CGENI_ABC_US	0	0	0	0.0041	0.4803	1.1192	0
9_CGENI_AB_US	0.1411	0	0	0.0041	0.5344	0.9470	0.2518
9_CGENI_AC_US	0	0	0	0.0041	0.4467	1.0761	0
9_CGENI_BC_US	0	0	0	0.1099	0.6845	0.9040	0.2518
3_GENI_ABC_US	0.1845	0.1443	0	0.5706	0.6836	0.7318	0.0552
3_GENI_AB_US	0	0.1219	0.1383	0.3735	1.0474	1.1192	0
3_GENI_AC_US	0.1845	0.0609	0	0.3091	0.8002	1.0331	0.2518
3_GENI_BC_US	0.1845	0.0883	0.1383	0.3610	1.7972	1.2914	0.3158
5_GENI_ABC_US	0	0	0.0915	0.0684	0.6435	0.7748	0
5_GENI_AB_US	0	0	0.1383	0.1099	0.4010	1.6788	0.6324
5_GENI_AC_US	0	0	0	0.1577	0.4560	1.2483	0
5_GENI_BC_US	0	0	0.1383	0.3050	0.7200	1.1622	0.0292
7_GENI_ABC_US	0	0	0	0.0041	0.8263	1.5066	0.6324
7_GENI_AB_US	0	0	0	0.0871	0.6995	0.9900	0
7_GENI_AC_US	0.1845	0	0	0.2012	0.5726	0.8609	0
7_GENI_BC_US	0	0	0	0.2822	0.6827	1.0761	0
9_GENI_ABC_US	0	0	0	0.0684	0.5922	1.2053	0
9_GENI_AB_US	0	0	0	0.1307	0.5819	1.0331	0
9_GENI_AC_US	0	0	0	0.1805	0.1669	1.1192	0
9_GENI_BC_US	0	0	0	0.1099	0.8049	0.9470	0
3_CCAUS	3.6684	1.0812	6.4943	0.6536	2.0640	2.1523	6.3998
5_CCAUS	1.6204	0.1853	6.0588	0.0166	1.7347	2.1093	6.3998
7_CCAUS	2.5732	0.1853	6.0181	0.0166	1.7347	2.2815	5.3449
9_CCAUS	0.8183	0.1853	5.6436	0.0166	1.1546	2.2815	2.9887

Table B.8: (Cont.) Percentage deviation of the heuristic solutions and the best known solution for the TSP on the literature problems

Heuristic	*tsp225*	*u159*	*kroA100*	*pcb442*	*rd1 00*	*rd400*	*att532*
3_CGENI_ABC_US	1.2248	0.7509	0	0.9216	0	1.3153	0.6754
3_CGENI_AB_US	1.4034	0	0	1.3332	0	1.6752	1.3255
3_CGENI_AC_US	1.3523	0	0	1.1461	0	1.6032	1.2136
3_CGENI_BC_US	1.6841	0.7865	0	1.5479	0	1.5444	1.3111
5_CGENI_ABC_US	1.6330	0	0	1.2091	0	1.2237	0.4442
5_CGENI_AB_US	1.0972	0	0	1.4494	0	0.9816	1.4917
5_CGENI_AC_US	1.6075	0.5798	0	1.3883	0	1.1452	1.1377
5_CGENI_BC_US	1.5310	0	0	1.6581	0	1.6229	1.0655
7_CGENI_ABC_US	1.0206	0	0	1.0378	0	1.3807	1.0980
7_CGENI_AB_US	1.2758	0	0	1.4750	0	1.3807	1.1811
7_CGENI_AC_US	1.4799	0	0	0.9748	0	1.2695	0.7982
7_CGENI_BC_US	1.3268	0	0	0.8881	0	1.5313	0.9391
9_CGENI_ABC_US	1.2758	0	0	0.8251	0	0.7983	0.6971
9_CGENI_AB_US	1.0461	0	0	1.2150	0	1.1583	1.0257
9_CGENI_AC_US	1.4544	0	0	0.7739	0	1.5313	1.0330
9_CGENI_BC_US	1.4799	0	0	1.4789	0	1.2041	0.9391
3_GENI_ABC_US	1.0717	0.0190	0	1.2249	0.0758	1.5640	1.3002
3_GENI_AB_US	1.4289	0	0	1.6286	0.0758	1.4331	1.6217
3_GENI_AC_US	1.1737	0	0	1.3194	0	1.2237	1.3219
3_GENI_BC_US	1.4799	0	0	1.8708	0	1.7341	1.5134
5_GENI_ABC_US	1.4544	0	0	1.1461	0	1.4200	0.9607
5_GENI_AB_US	1.2758	0	0	0.8606	0	1.4396	0.9463
5_GENI_AC_US	1.3013	0	0	1.3509	0	1.6949	0.8776
5_GENI_BC_US	1.5565	0	0	1.4750	0	1.0077	1.1738
7_GENI_ABC_US	1.1992	0	0	1.4533	0	0.9619	0.8813
7_GENI_AB_US	1.2758	0	0	0.7621	0	1.0405	1.2388
7_GENI_AC_US	1.5054	0	0	1.0713	0	1.0666	0.9932
7_GENI_BC_US	1.1992	0	0	1.3470	0	1.5771	1.0402
9_GENI_ABC_US	1.5054	0	0	1.2328	0	0.9946	0.9318
9_GENI_AB_US	1.2503	0	0	1.2682	0	1.4396	1.0077
9_GENI_AC_US	1.7096	0	0	1.1560	0	1.5836	0.8379
9_GENI_BC_US	1.4799	0	0	1.6463	0	1.5444	0.7296
3_CCAUS	1.8627	2.0175	0	2.4420	2.2626	3.6712	1.8095
5_CCAUS	1.7861	1.1121	0	2.1800	2.1868	2.9644	1.7012
7_CCAUS	1.8627	1.1121	0	2.3120	2.1868	2.6961	1.7048
9_CCAUS	1.8627	1.1121	0	2.1091	2.1868	2.6241	1.7084

Table B.8: (Cont.) Percentage deviation of the heuristic solution and the best known solution for the TSP on the literature problems

Heuristic	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*	*p7*
3_CGENI_AB_US	7763	8541	8921	9422	10208	10690	11810
3_CGENI_ABC_US	7758	8562	8889	9443	10259	10763	11739
3_CGENI_AC_US	7763	8571	8935	9457	10126	10625	11763
3_CGENI_BC_US	7763	8552	8871	9455	10281	10694	11750
5_CGENI_AB_US	7758	8541	8866	9418	10273	10673	11749
5_CGENI_ABC_US	7758	8552	8901	9456	10146	10655	11809
5_CGENI_AC_US	7758	8541	8889	9418	10269	10598	11718
5_CGENI_BC_US	7758	8541	8866	9457	10136	10670	11704
7_CGENI_AB_US	7758	8541	8866	9426	10196	10625	11710
7_CGENI_ABC_US	7758	8541	8866	9418	10221	10697	11778
7_CGENI_AC_US	7758	8541	8886	9426	10218	10644	11745
7_CGENI_BC_US	7758	8541	8886	9435	10124	10610	11769
9_CGENI_AB_US	7758	8541	8889	9454	10169	10598	11745
9_CGENI_ABC_US	7758	8552	8889	9418	10222	10638	11696
9_CGENI_AC_US	7758	8541	8889	9418	10133	10625	11742
9_CGENI_BC_US	7758	8541	8866	9435	10111	10639	11762
3_GENI_AB_US	7758	8571	8868	9449	10159	10699	11815
3_GENI_ABC_US	7758	8562	8868	9442	10235	10688	11834
3_GENI_AC_US	7758	8541	8866	9442	10211	10761	11761
3_GENI_BC_US	7758	8571	8916	9426	10293	10655	11840
5_GENI_AB_US	7758	8552	8866	9455	10282	10598	11761
5_GENI_ABC_US	7758	8552	8889	9442	10243	10598	11754
5_GENI_AC_US	7758	8541	8866	9442	10139	10678	11746
5_GENI_BC_US	7758	8541	8892	9464	10213	10667	11780
7_GENI_AB_US	7758	8541	8886	9426	10203	10661	11712
7_GENI_ABC_US	7758	8541	8866	9435	10136	10595	11712
7_GENI_AC_US	7758	8552	8866	9442	10219	10674	11744
7_GENI_BC_US	7758	8541	8866	9457	10176	10595	11761
9_GENI_AB_US	7758	8541	8866	9418	10148	10595	11744
9_GENI_ABC_US	7758	8541	8866	9426	10204	10664	11781
9_GENI_AC_US	7758	8541	8866	9418	10132	10671	11758
9_GENI_BC_US	7758	8541	8889	9456	10124	10651	11751
3_CCAUS	7865	8769	9072	9682	10597	10995	12005
5_CCAUS	7843	8766	9071	9553	10449	10867	12039
7_CCAUS	7795	8766	9071	9733	10338	10969	12009
9_CCAUS	7795	8766	9071	9585	10333	10919	12009

Table B.9: The best solution obtained over 100 runs for various heuristics Hj on the given problem i, i.e. $\min_{1 \leq r \leq 100} (f^r_{Hj})_i$ for the TSP on the randomly generated problems

Heuristic	*p8*	*p9*	*p10*	*p11*	*p12*	*p13*	*p14*
3_CGENI_AB_US	12035	12300	12673	13289	13309	14019	14567
3_CGENI_ABC_US	12083	12242	12665	13281	13373	14155	14689
3_CGENI_AC_US	12094	12414	12722	13359	13322	14127	14639
3_CGENI_BC_US	12100	12314	12699	13299	13342	14177	14582
5_CGENI_AB_US	12026	12290	12674	13233	13307	14115	14566
5_CGENI_ABC_US	12051	12268	12612	13347	13288	14121	14606
5_CGENI_AC_US	12052	12236	12691	13283	13340	14073	14590
5_CGENI_BC_US	12063	12325	12644	13329	13320	14117	14578
7_CGENI_AB_US	12074	12322	12666	13312	13308	14049	14543
7_CGENI_ABC_US	12018	12324	12634	13326	13315	14122	14545
7_CGENI_AC_US	12050	12250	12669	13308	13318	14112	14528
7_CGENI_BC_US	12052	12291	12648	13288	13327	14088	14600
9_CGENI_AB_US	12049	12246	12624	13282	13286	14129	14571
9_CGENI_ABC_US	12051	12285	12632	13325	13296	14074	14542
9_CGENI_AC_US	12049	12259	12635	13279	13317	14085	14516
9_CGENI_BC_US	11991	12292	12635	13340	13301	14115	14539
3_GENI_AB_US	12084	12291	12697	13292	13347	14102	14618
3_GENI_ABC_US	12101	12335	12755	13426	13317	14109	14689
3_GENI_AC_US	12108	12298	12665	13368	13336	14193	14714
3_GENI_BC_US	12077	12386	12710	13344	13368	14066	14615
5_GENI_AB_US	12057	12365	12693	13341	13282	14138	14567
5_GENI_ABC_US	12053	12333	12663	13311	13290	14089	14632
5_GENI_AC_US	12042	12229	12664	13306	13348	14133	14567
5_GENI_BC_US	12057	12268	12675	13390	13302	14066	14633
7_GENI_AB_US	12087	12283	12649	13295	13215	14112	14600
7_GENI_ABC_US	12061	12240	12633	13372	13320	14075	14601
7_GENI_AC_US	12039	12317	12658	13203	13346	14128	14543
7_GENI_BC_US	12057	12340	12679	13365	13360	14079	14589
9_GENI_AB_US	12010	12271	12656	13340	13291	14081	14593
9_GENI_ABC_US	12045	12305	12685	13259	13296	14086	14551
9_GENI_AC_US	11969	12345	12656	13334	13318	14132	14541
9_GENI_BC_US	12055	12259	12654	13285	13323	14147	14645
3_CCAUS	12197	12585	12822	13581	13705	14486	14780
5_CCAUS	12208	12371	12806	13438	13684	14312	14777
7_CCAUS	12129	12371	12773	13497	13684	14323	14777
9_CCAUS	12129	12371	12773	13494	13663	14310	14756

Table B.9: (Cont.) The best solution obtained over 100 runs for various heuristics Hj on the given problem i, i.e. $\min_{1 \le r \le 100} (f^r_{Hj})_i$ for the TSP on the randomly generated problems

Heuristic	*p15*	*p16*	*p17*	*p18*	*p19*	*p20*
3_CGENI_AB_US	14942	15161	15254	15927	16305	16269
3_CGENI_ABC_US	14962	15287	15322	16053	16297	16277
3_CGENI_AC_US	14946	15261	15142	16070	16235	16256
3_CGENI_BC_US	14977	15096	15271	16100	16346	16277
5_CGENI_AB_US	14914	15161	15205	16020	16215	16220
5_CGENI_ABC_US	14981	15192	15269	16001	16278	16176
5_CGENI_AC_US	14957	15237	15322	16011	16267	16202
5_CGENI_BC_US	14903	15169	15267	16018	16267	16197
7_CGENI_AB_US	14891	15195	15196	15928	16221	16217
7_CGENI_ABC_US	14921	15042	15239	15975	16212	16187
7_CGENI_AC_US	14879	15126	15184	16014	16208	16217
7_CGENI_BC_US	14869	15110	15253	15998	16218	16198
9_CGENI_AB_US	14866	15063	15143	15896	16205	16159
9_CGENI_ABC_US	14935	15174	15269	15975	16245	16205
9_CGENI_AC_US	14888	15049	15210	16032	16184	16213
9_CGENI_BC_US	14934	15174	15240	16102	16282	16257
3_GENI_AB_US	14964	15166	15296	16008	16253	16234
3_GENI_ABC_US	14933	15240	15312	16018	16360	16223
3_GENI_AC_US	14974	15272	15269	16032	16301	16276
3_GENI_BC_US	15042	15254	15307	15960	16347	16309
5_GENI_AB_US	14865	15162	15256	15999	16172	16147
5_GENI_ABC_US	14884	15215	15201	16011	16257	16197
5_GENI_AC_US	14964	15244	15203	16034	16291	16237
5_GENI_BC_US	14965	15197	15251	15965	16288	16257
7_GENI_AB_US	14921	15140	15192	15961	16186	16211
7_GENI_ABC_US	14941	15210	15242	16000	16256	16169
7_GENI_AC_US	14904	15175	15268	16008	16250	16259
7_GENI_BC_US	14941	15138	15207	16003	16237	16234
9_GENI_AB_US	14913	15166	15171	15949	16232	16216
9_GENI_ABC_US	14941	15174	15133	15971	16203	16167
9_GENI_AC_US	14913	15186	15267	15900	16189	16225
9_GENI_BC_US	14906	15172	15274	15948	16231	16137
3_CCAUS	15237	15262	15548	16342	16486	16715
5_CCAUS	15184	15281	15500	16305	16310	16644
7_CCAUS	15201	15242	15500	16283	16206	16534
9_CCAUS	15142	15239	15460	16291	16208	16467

Table B.9: (Cont.) The best solution obtained over 100 runs for various heuristics Hj on the given problem i, i.e. $\min_{1 \le r \le 100} (f^r_{Hj})_i$ for the TSP on the randomly generated problems

Heuristic	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*	*p7*
3_CGENI_AB_US	1.0006	1	1.0062	1.0004	1.0095	1.0089	1.0097
3_CGENI_ABC_US	1	1.0024	1.0025	1.0026	1.0146	1.0158	1.0036
3_CGENI_AC_US	1.0006	1.0035	1.0077	1.0041	1.0014	1.0028	1.0057
3_CGENI_BC_US	1.0006	1.0012	1.0005	1.0039	1.0168	1.0093	1.0046
5_CGENI_AB_US	1	1	1	1	1.0160	1.0073	1.0045
5_CGENI_ABC_US	1	1.0012	1.0039	1.0040	1.0034	1.0056	1.0096
5_CGENI_AC_US	1	1	1.0025	1	1.0156	1.0002	1.0018
5_CGENI_BC_US	1	1	1	1.0041	1.0024	1.0070	1.0006
7_CGENI_AB_US	1	1	1	1.0008	1.0084	1.0028	1.0011
7_CGENI_ABC_US	1	1	1	1	1.0108	1.0096	1.0070
7_CGENI_AC_US	1	1	1.0022	1.0008	1.0105	1.0046	1.0041
7_CGENI_BC_US	1	1	1.0022	1.0018	1.0012	1.0014	1.0062
9_CGENI_AB_US	1	1	1.0025	1.0038	1.0057	1.0002	1.0041
9_CGENI_ABC_US	1	1.0012	1.0025	1	1.0109	1.0040	1
9_CGENI_AC_US	1	1	1.0025	1	1.0021	1.0028	1.0039
9_CGENI_BC_US	1	1	1	1.0018	1	1.0041	1.0056
3_GENI_AB_US	1	1.0035	1.0002	1.0032	1.0047	1.0098	1.0101
3_GENI_ABC_US	1	1.0024	1.0002	1.0025	1.0122	1.0087	1.0117
3_GENI_AC_US	1	1	1	1.0025	1.0098	1.0156	1.0055
3_GENI_BC_US	1	1.0035	1.0056	1.0008	1.0180	1.0056	1.0123
5_GENI_AB_US	1	1.0012	1	1.0039	1.0169	1.0002	1.0055
5_GENI_ABC_US	1	1.0012	1.0025	1.0025	1.0130	1.0002	1.0049
5_GENI_AC_US	1	1	1	1.0025	1.0027	1.0078	1.0042
5_GENI_BC_US	1	1	1.0029	1.0048	1.0100	1.0067	1.0071
7_GENI_AB_US	1	1	1.0022	1.0008	1.0090	1.0062	1.0013
7_GENI_ABC_US	1	1	1	1.0018	1.0024	1	1.0013
7_GENI_AC_US	1	1.0012	1	1.0025	1.0106	1.0074	1.0041
7_GENI_BC_US	1	1	1	1.0041	1.0064	1	1.0055
9_GENI_AB_US	1	1	1	1	1.0036	1	1.0041
9_GENI_ABC_US	1	1	1	1.0008	1.0091	1.0065	1.0072
9_GENI_AC_US	1	1	1	1	1.0020	1.0071	1.0053
9_GENI_BC_US	1	1	1.0025	1.0040	1.0012	1.0052	1.0047
3_CCAUS	1.0137	1.0266	1.0232	1.0280	1.0480	1.0377	1.0264
5_CCAUS	1.0109	1.0263	1.0231	1.0143	1.0334	1.0256	1.0293
7_CCAUS	1.0047	1.0263	1.0231	1.0334	1.0224	1.0352	1.0267
9_CCAUS	1.0047	1.0263	1.0231	1.0177	1.0219	1.0305	1.0267

Table B.10: Ratio for the TSP on the randomly generated problems for Case 1

Heuristic	*p8*	*p9*	*p10*	*p11*	*p12*	*p13*	*p14*
3_CGENI_AB_US	1.0055	1.0058	1.0048	1.0065	1.0071	1	1.0035
3_CGENI_ABC_US	1.0095	1.0010	1.0042	1.0059	1.0119	1.0097	1.0119
3_CGENI_AC_US	1.0104	1.0151	1.0087	1.0118	1.0080	1.0077	1.0084
3_CGENI_BC_US	1.0109	1.0069	1.0068	1.0072	1.0096	1.0112	1.0045
5_CGENI_AB_US	1.0047	1.0049	1.0049	1.0022	1.0069	1.0068	1.0034
5_CGENI_ABC_US	1.0068	1.0031	1	1.0109	1.0055	1.0072	1.0062
5_CGENI_AC_US	1.0069	1.0005	1.0062	1.0060	1.0094	1.0038	1.0050
5_CGENI_BC_US	1.0078	1.0078	1.0025	1.0095	1.0079	1.0069	1.0042
7_CGENI_AB_US	1.0087	1.0076	1.0042	1.0082	1.0070	1.0021	1.0018
7_CGENI_ABC_US	1.0040	1.0077	1.0017	1.0093	1.0075	1.0073	1.0019
7_CGENI_AC_US	1.0067	1.0017	1.0045	1.0079	1.0077	1.0066	1.0008
7_CGENI_BC_US	1.0069	1.0050	1.0028	1.0064	1.0084	1.0049	1.0057
9_CGENI_AB_US	1.0066	1.0013	1.0009	1.0059	1.0053	1.0078	1.0037
9_CGENI_ABC_US	1.0068	1.0045	1.0015	1.0092	1.0061	1.0039	1.0017
9_CGENI_AC_US	1.0066	1.0024	1.0018	1.0057	1.0077	1.0047	1
9_CGENI_BC_US	1.0018	1.0051	1.0018	1.0103	1.0065	1.0068	1.0015
3_GENI_AB_US	1.0096	1.0050	1.0067	1.0067	1.0099	1.0059	1.0070
3_GENI_ABC_US	1.0110	1.0086	1.0113	1.0168	1.0077	1.0064	1.0119
3_GENI_AC_US	1.0116	1.0056	1.0042	1.0124	1.0091	1.0124	1.0136
3_GENI_BC_US	1.0090	1.01283	1.0077	1.0106	1.0115	1.0033	1.0068
5_GENI_AB_US	1.0073	1.0111	1.0064	1.0104	1.0050	1.0084	1.0035
5_GENI_ABC_US	1.0070	1.0085	1.0040	1.0081	1.0056	1.0049	1.0079
5_GENI_AC_US	1.0060	1	1.0041	1.0078	1.0100	1.0081	1.0035
5_GENI_BC_US	1.0073	1.0031	1.0049	1.0141	1.0065	1.0033	1.0080
7_GENI_AB_US	1.0098	1.0044	1.0029	1.0069	1	1.0066	1.0057
7_GENI_ABC_US	1.0076	1.0008	1.0016	1.0128	1.0079	1.0039	1.0058
7_GENI_AC_US	1.0058	1.0071	1.0036	1	1.0099	1.0077	1.0018
7_GENI_BC_US	1.0073	1.0090	1.0053	1.0122	1.0109	1.0042	1.0050
9_GENI_AB_US	1.0034	1.0034	1.0034	1.0103	1.0057	1.0044	1.0053
9_GENI_ABC_US	1.0063	1.0062	1.0057	1.0042	1.0061	1.0047	1.00241
9_GENI_AC_US	1	1.0094	1.0034	1.0099	1.0077	1.0080	1.0017
9_GENI_BC_US	1.0071	1.0024	1.0033	1.0062	1.0081	1.0091	1.0088
3_CCAUS	1.0190	1.0291	1.0166	1.0286	1.0370	1.0333	1.0181
5_CCAUS	1.0199	1.0116	1.0153	1.0177	1.0354	1.0209	1.0179
7_CCAUS	1.0133	1.0116	1.0127	1.0222	1.0354	1.0216	1.0179
9_CCAUS	1.0133	1.0116	1.0127	1.0220	1.0339	1.0207	1.0165

Table B.10: (Cont.) Ratio for the TSP on the randomly generated problems for Case 1

Heuristic	*p15*	*p16*	*p17*	*p18*	*p19*	*p20*
3_CGENI_AB_US	1.0051	1.0079	1.0079	1.0019	1.0082	1.0081
3_CGENI_ABC_US	1.0065	1.0162	1.0124	1.0098	1.0077	1.0086
3_CGENI_AC_US	1.0054	1.0145	1.0005	1.0109	1.0038	1.0073
3_CGENI_BC_US	1.0075	1.0035	1.0091	1.0128	1.0107	1.0086
5_CGENI_AB_US	1.0032	1.0079	1.0047	1.0078	1.0026	1.0051
5_CGENI_ABC_US	1.0078	1.0099	1.0089	1.0066	1.0065	1.0024
5_CGENI_AC_US	1.0061	1.0129	1.0124	1.0072	1.0058	1.0040
5_CGENI_BC_US	1.0025	1.0084	1.0088	1.0076	1.0058	1.0037
7_CGENI_AB_US	1.0017	1.0101	1.0041	1.0020	1.0030	1.0049
7_CGENI_ABC_US	1.0037	1	1.0070	1.0049	1.0024	1.0030
7_CGENI_AC_US	1.0009	1.0055	1.0033	1.0074	1.0022	1.0049
7_CGENI_BC_US	1.0002	1.0045	1.0079	1.0064	1.0028	1.0037
9_CGENI_AB_US	1.0000	1.0013	1.0006	1	1.0020	1.0013
9_CGENI_ABC_US	1.0047	1.0087	1.0089	1.0049	1.0045	1.0042
9_CGENI_AC_US	1.0015	1.0004	1.0050	1.0085	1.0007	1.0047
9_CGENI_BC_US	1.0046	1.0087	1.0070	1.0129	1.0068	1.0074
3_GENI_AB_US	1.0066	1.0082	1.0107	1.0070	1.0050	1.0060
3_GENI_ABC_US	1.0045	1.0131	1.0118	1.0076	1.0116	1.0053
3_GENI_AC_US	1.0073	1.0152	1.0089	1.0085	1.0079	1.0086
3_GENI_BC_US	1.0119	1.0140	1.0114	1.0040	1.0108	1.0106
5_GENI_AB_US	1	1.0079	1.0081	1.0064	1	1.0006
5_GENI_ABC_US	1.0044	1.0072	1.0052	1.0012	1.0037	1.0025
5_GENI_AC_US	1.0046	1.0086	1.0073	1	1.0061	1.0025
5_GENI_BC_US	1.0067	1.0103	1.0077	1.0043	1.0071	1.0074
7_GENI_AB_US	1.0037	1.0065	1.0038	1.0040	1.0008	1.0045
7_GENI_ABC_US	1.0051	1.0111	1.0072	1.0065	1.0051	1.0019
7_GENI_AC_US	1.0026	1.0088	1.0089	1.0070	1.0048	1.0075
7_GENI_BC_US	1.0051	1.0063	1.0048	1.0067	1.0040	1.0060
9_GENI_AB_US	1.0032	1.0082	1.0025	1.0033	1.0037	1.0048
9_GENI_ABC_US	1.0051	1.0087	1	1.0047	1.0019	1.0018
9_GENI_AC_US	1.0032	1.0095	1.0088	1.0002	1.0010	1.0054
9_GENI_BC_US	1.0027	1.0086	1.0093	1.0032	1.0036	1
3_CCAUS	1.0250	1.0146	1.0274	1.0280	1.0194	1.0358
5_CCAUS	1.0214	1.0158	1.0242	1.0257	1.0085	1.0314
7_CCAUS	1.0226	1.0132	1.0242	1.0243	1.0021	1.0246
9_CCAUS	1.0186	1.0130	1.0216	1.0248	1.0022	1.0204

Table B.10: (Cont.) Ratio for the TSP on the randomly generated problems for Case 1

Heuristic	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*	*p7*	*p8*
3_CGENI_AB_US	31	11	31	8	18	27	29	6
3_CGENI_ABC_US	15	28.5	24	22	27	32	7	26
3_CGENI_AC_US	31	31	32	30	4	10	23	29
3_CGENI_BC_US	31	24.5	17	25.5	30	28	15	30
5_CGENI_AB_US	15	11	7.5	4	29	23	14	5
5_CGENI_ABC_US	15	24.5	29	27.5	10	16.5	28	13.5
5_CGENI_AC_US	15	11	24	4	28	5.5	6	15.5
5_CGENI_BC_US	15	11	7.5	30	7.5	21	2	23
7_CGENI_AB_US	15	11	7.5	11	15	10	3	24
7_CGENI_ABC_US	15	11	7.5	4	23	29	25	4
7_CGENI_AC_US	15	11	19	11	21	14	11.5	12
7_CGENI_BC_US	15	11	19	15	2.5	8	24	15.5
9_CGENI_AB_US	15	11	24	24	13	5.5	11.5	10.5
9_CGENI_ABC_US	15	24.5	24	4	24	12	1	13.5
9_CGENI_AC_US	15	11	24	4	6	10	8	10.5
9_CGENI_BC_US	15	11	7.5	15	1	13	22	2
3_GENI_AB_US	15	31	15.5	23	12	30	30	27
3_GENI_ABC_US	15	28.5	15.5	19	25	26	31	31
3_GENI_AC_US	15	11	7.5	19	19	31	20	32
3_GENI_BC_US	15	31	30	11	32	16.5	32	25
5_GENI_AB_US	15	24.5	7.5	25.5	31	5.5	20	20
5_GENI_ABC_US	15	24.5	24	19	26	5.5	17	17
5_GENI_AC_US	15	11	7.5	19	9	25	13	8
5_GENI_BC_US	15	11	28	32	20	20	26	20
7_GENI_AB_US	15	11	19	11	16	18	4.5	28
7_GENI_ABC_US	15	11	7.5	15	7.5	2	4.5	22
7_GENI_AC_US	15	24.5	7.5	19	22	24	9.5	7
7_GENI_BC_US	15	11	7.5	30	14	2	20	20
9_GENI_AB_US	15	11	7.5	4	11	2	9.5	3
9_GENI_ABC_US	15	11	7.5	11	17	19	27	9
9_GENI_AC_US	15	11	7.5	4	5	22	18	1
9_GENI_BC_US	15	11	24	27.5	2.5	15	16	18
3_CCAUS	36	36	36	35	36	36	33	35
5_CCAUS	35	34	34	33	35	33	36	36
7_CCAUS	33.5	34	34	36	34	35	34.5	33.5
9_CCAUS	33.5	34	34	34	33	34	34.5	33.5

Table B.11: Ranking for the Friedman test for the TSP on the randomly generated problems for Case 1

Heuristic	*p9*	*p10*	*p11*	*p12*	*p13*	*p14*	*p15*	*p16*
3_CGENI_AB_US	19	21	10	13	1	12	22	9.5
3_CGENI_ABC_US	4	17.5	5	32	30	30.5	25	36
3_CGENI_AC_US	35	31	27	21	23	28	23	32
3_CGENI_BC_US	21	29	13	26	31	16	30	4
5_CGENI_AB_US	14	22	2	11	18.5	10	13	9.5
5_CGENI_ABC_US	9.5	1	26	4	21	23	31	21
5_CGENI_AC_US	2	26	7	25	5	18	24	26
5_CGENI_BC_US	25	8	20	19.5	20	15	8	14
7_CGENI_AB_US	23	19	17	12	2	6.5	7	22
7_CGENI_ABC_US	24	5	19	14	22	8	14.5	1
7_CGENI_AC_US	6	20	15	17.5	16.5	2	4	6
7_CGENI_BC_US	15.5	9	9	23	12	20.5	3	5
9_CGENI_AB_US	5	2	6	3	25	14	2	3
9_CGENI_ABC_US	13	3	18	7.5	6	5	18	17
9_CGENI_AC_US	7.5	6.5	4	15.5	10	1	6	2
9_CGENI_BC_US	17	6.5	22.5	9	18.5	3	17	17
3_GENI_AB_US	15.5	28	11	28	14	25	26.5	12.5
3_GENI_ABC_US	27	32	32	15.5	15	30.5	16	28
3_GENI_AC_US	18	17.5	29	24	32	32	29	34
3_GENI_BC_US	34	30	25	31	3.5	24	32	31
5_GENI_AB_US	30	27	24	2	28	12	1	11
5_GENI_ABC_US	26	15	16	5	13	26	5	25
5_GENI_AC_US	1	16	14	29	27	12	26.5	30
5_GENI_BC_US	9.5	23	31	10	3.5	27	28	23
7_GENI_AB_US	12	10	12	1	16.5	20.5	14.5	8
7_GENI_ABC_US	3	4	30	19.5	7	22	20	24
7_GENI_AC_US	22	14	1	27	24	6.5	9	19
7_GENI_BC_US	28	24	28	30	8	17	20	7
9_GENI_AB_US	11	12.5	22.5	6	9	19	11.5	12.5
9_GENI_ABC_US	20	25	3	7.5	11	9	20	17
9_GENI_AC_US	29	12.5	21	17.5	26	4	11.5	20
9_GENI_BC_US	7.5	11	8	22	29	29	10	15
3_CCAUS	36	36	36	36	36	36	36	33
5_CCAUS	32	35	33	34.5	34	34.5	34	35
7_CCAUS	32	33.5	35	34.5	35	34.5	35	29
9_CCAUS	32	33.5	34	33	33	33	33	27

Table B.11: (Cont.) Ranking for the Friedman test for the TSP on the randomly generated problems for Case 1

Heuristic	*p17*	*p18*	*p19*	*p20*	R_j	$\sum_i R_{ij}^2$
3_CGENI_AB_US	18	3	31	28	348.5	7905.25
3_CGENI_ABC_US	31.5	29	29	30.5	481.5	13302.25
3_CGENI_AC_US	2	30	16	24	482	13290
3_CGENI_BC_US	26	31	33	30.5	491.5	13225.75
5_CGENI_AB_US	10	25	11	18	272.5	4685.75
5_CGENI_ABC_US	24	16	25	6	371.5	8409.25
5_CGENI_AC_US	31.5	20.5	23.5	11	328.5	7069.25
5_CGENI_BC_US	20.5	23.5	23.5	8.5	322.5	6283.75
7_CGENI_AB_US	7	4	13	16.5	245.5	3841.75
7_CGENI_ABC_US	13	11.5	10	7	267.5	4796.75
7_CGENI_AC_US	5	22	8.5	16.5	253.5	3890.25
7_CGENI_BC_US	17	13	12	10	259	4050
9_CGENI_AB_US	3	1	6	3	187.5	2902.75
9_CGENI_ABC_US	24	11.5	18	12	271	4729
9_CGENI_AC_US	12	26.5	2	14	195.5	2790.25
9_CGENI_BC_US	14	32	26	25.5	294.5	5679.25
3_GENI_AB_US	28	18.5	20	21.5	432	10225.5
3_GENI_ABC_US	30	23.5	35	19	494.5	13112.25
3_GENI_AC_US	24	26.5	30	29	479.5	12620.75
3_GENI_BC_US	29	7	34	32	505	14437.5
5_GENI_AB_US	19	14	1	2	320	7144
5_GENI_ABC_US	8	20.5	22	8.5	338	6764
5_GENI_AC_US	9	28	28	23	351	7649.5
5_GENI_BC_US	16	9	27	25.5	404.5	9480.75
7_GENI_AB_US	6	8	3	13	247	3846
7_GENI_ABC_US	15	15	21	5	270	4922
7_GENI_AC_US	22	18.5	19	27	337.5	6828.25
7_GENI_BC_US	11	17	17	21.5	348	7309.5
9_GENI_AB_US	4	6	15	15	207	2694.5
9_GENI_ABC_US	1	10	5	4	249	4106.5
9_GENI_AC_US	20.5	2	4	20	271.5	5061.25
9_GENI_BC_US	27	5	14	1	307.5	6172.75
3_CCAUS	36	36	36	36	712	25364
5_CCAUS	34.5	35	32	35	684.5	23451.75
7_CCAUS	34.5	33	7	34	651.5	21950.75
9_CCAUS	33	34	8.5	33	637.5	20941.25
					B_F= 280802.025	A_F= 320934

Table B.11: (Cont.) Ranking for the Friedman test for the TSP on the randomly generated problems for Case 1

Heuristic	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*	em p7
3_CGENI_AB_US	1.0006	1	1.0062	1.0004	1.0095	1.0086	1.0097
3_CGENI_ABC_US	1	1.0024	1.0025	1.0026	1.0146	1.0155	1.0036
3_CGENI_AC_US	1.0006	1.0035	1.0077	1.0041	1.0014	1.0025	1.0057
3_CGENI_BC_US	1.0006	1.0012	1.0005	1.0039	1.0168	1.0090	1.0046
5_CGENI_AB_US	1	1	1	1	1.0160	1.0070	1.0045
5_CGENI_ABC_US	1	1.0012	1.0039	1.0040	1.0034	1.0053	1.0096
5_CGENI_AC_US	1	1	1.0025	1	1.0156	1	1.0018
5_CGENI_BC_US	1	1	1	1.0041	1.0024	1.0067	1.0006
7_CGENI_AB_US	1	1	1	1.0008	1.0084	1.0025	1.0011
7_CGENI_ABC_US	1	1	1	1	1.0108	1.0093	1.0070
7_CGENI_AC_US	1	1	1.0022	1.0008	1.0105	1.0043	1.0041
7_CGENI_BC_US	1	1	1.0022	1.0018	1.0012	1.0011	1.0062
9_CGENI_AB_US	1	1	1.0025	1.0038	1.0057	1	1.0041
9_CGENI_ABC_US	1	1.0012	1.0025	1	1.0109	1.0037	1
9_CGENI_AC_US	1	1	1.0025	1	1.0021	1.0025	1.0039
9_CGENI_BC_US	1	1	1	1.0018	1	1.0038	1.0056

Table B.12: Ratio for the TSP on the randomly generated problems for Case 2

Heuristic	*p8*	*p9*	*p10*	*p11*	*p12*	*p13*	*p14*
3_CGENI_AB_US	1.0036	1.0052	1.0048	1.0042	1.0017	1	1.0035
3_CGENI_ABC_US	1.0076	1.0004	1.0042	1.0036	1.0065	1.0097	1.0119
3_CGENI_AC_US	1.0085	1.0145	1.0087	1.0095	1.0027	1.0077	1.0084
3_CGENI_BC_US	1.0090	1.00637	1.0068	1.0049	1.0042	1.0112	1.0045
5_CGENI_AB_US	1.0029	1.0044	1.0049	1	1.0015	1.0068	1.0034
5_CGENI_ABC_US	1.0050	1.0026	1	1.0086	1.0001	1.0072	1.0062
5_CGENI_AC_US	1.0018	1.0050	1.0062	1.0037	1.0040	1.0038	1.0050
5_CGENI_BC_US	1.0060	1.0072	1.0025	1.0072	1.0025	1.0069	1.0042
7_CGENI_AB_US	1.0069	1.0070	1.0042	1.0059	1.0016	1.0021	1.0018
7_CGENI_ABC_US	1.0022	1.0071	1.0017	1.0070	1.0021	1.0073	1.0019
7_CGENI_AC_US	1.0049	1.0011	1.0045	1.0056	1.0024	1.0066	1.0008
7_CGENI_BC_US	1.0050	1.0044	1.0028	1.0041	1.0030	1.0049	1.0057
9_CGENI_AB_US	1.0048	1.0008	1.0009	1.0037	1	1.0078	1.0037
9_CGENI_ABC_US	1.0050	1.0040	1.0015	1.0069	1.0007	1.0039	1.0017
9_CGENI_AC_US	1.0048	1.0018	1.0018	1.0034	1.0023	1.0047	1.0014
9_CGENI_BC_US	1	1.0045	1.0018	1.0080	1.0011	1.0068	1.0015

Table B.12: (Cont.) Ratio for the TSP on the randomly generated problems for Case 2

Heuristic	*p15*	*p16*	*p17*	*p18*	*p19*	*p20*
3_CGENI_AB_US	1.0051	1.0079	1.0073	1.0019	1.0074	1.0068
3_CGENI_ABC_US	1.0064	1.0162	1.0118	1.0098	1.0069	1.0073
3_CGENI_AC_US	1.0053	1.0145	1	1.0109	1.0031	1.0060
3_CGENI_BC_US	1.0074	1.0035	1.0085	1.0128	1.0100	1.0073
5_CGENI_AB_US	1.0032	1.0079	1.0041	1.0078	1.0019	1.0037
5_CGENI_ABC_US	1.0077	1.0099	1.0083	1.0066	1.0058	1.0010
5_CGENI_AC_US	1.0061	1.0129	1.0118	1.0072	1.0051	1.0026
5_CGENI_BC_US	1.0024	1.0084	1.0082	1.0076	1.0051	1.0023
7_CGENI_AB_US	1.0016	1.0101	1.0035	1.0020	1.0022	1.0035
7_CGENI_ABC_US	1.0036	1	1.0064	1.0049	1.0017	1.0017
7_CGENI_AC_US	1.0008	1.0055	1.0027	1.0074	1.0014	1.0035
7_CGENI_BC_US	1.0002	1.0045	1.0073	1.0064	1.0021	1.0024
9_CGENI_AB_US	1	1.0013	1.0000	1	1.0012	1
9_CGENI_ABC_US	1.0046	1.0087	1.0083	1.0049	1.0037	1.0028
9_CGENI_AC_US	1.0014	1.0004	1.0044	1.0085	1	1.0033
9_CGENI_BC_US	1.0045	1.0087	1.0064	1.0129	1.0060	1.0060

Table B.12: (Cont.) Ratio for the TSP on the randomly generated problems for Case 2

Heuristic	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*	*p7*	*p8*
3_CGENI_AB_US	15	6	15	6	9	13	16	4
3_CGENI_ABC_US	7	15	11	11	13	16	5	14
3_CGENI_AC_US	15	16	16	15.5	3	5	12	15
3_CGENI_BC_US	15	13	6	13	16	14	10	16
5_CGENI_AB_US	7	6	3	3	15	12	9	3
5_CGENI_ABC_US	7	13	14	14	6	10	15	8.5
5_CGENI_AC_US	7	6	11	3	14	1.5	4	10.5
5_CGENI_BC_US	7	6	3	15.5	5	11	2	12
7_CGENI_AB_US	7	6	3	7.5	8	5	3	13
7_CGENI_ABC_US	7	6	3	3	11	15	14	2
7_CGENI_AC_US	7	6	7.5	7.5	10	9	7.5	7
7_CGENI_BC_US	7	6	7.5	9.5	2	3	13	10.5
9_CGENI_AB_US	7	6	11	12	7	1.5	7.5	5.5
9_CGENI_ABC_US	7	13	11	3	12	7	1	8.5
9_CGENI_AC_US	7	6	11	3	4	5	6	5.5
9_CGENI_BC_US	7	6	3	9.5	1	8	11	1

Table B.13: Ranking for the Friedman test for the TSP on the randomly generated problems for Case 2

Heuristic	*p9*	*p10*	*p11*	*p12*	*p13*	*p14*	*p15*	*p16*
3_CGENI_AB_US	11	12	7	7	1	8	11	7.5
3_CGENI_ABC_US	2	9	3	16	15	16	14	16
3_CGENI_AC_US	16	16	16	12	13	15	12	15
3_CGENI_BC_US	12	15	8	15	16	11	15	4
5_CGENI_AB_US	8	13	1	5	8.5	7	7	7.5
5_CGENI_ABC_US	6	1	15	2	11	14	16	12
5_CGENI_AC_US	1	14	5	14	3	12	13	14
5_CGENI_BC_US	15	7	13	11	10	10	6	9
7_CGENI_AB_US	13	10	10	6	2	5	5	13
7_CGENI_ABC_US	14	4	12	8	12	6	8	1
7_CGENI_AC_US	4	11	9	10	7	2	3	6
7_CGENI_BC_US	9	8	6	13	6	13	2	5
9_CGENI_AB_US	3	2	4	1	14	9	1	3
9_CGENI_ABC_US	7	3	11	3	4	4	10	10.5
9_CGENI_AC_US	5	5.5	2	9	5	1	4	2
9_CGENI_BC_US	10	5.5	14	4	8.5	3	9	10.5

Table B.13: (Cont.) Ranking for the Friedman test for the TSP on the randomly generated problems for Case 2

Heuristic	*p17*	*p18*	*p19*	*p20*	R_j	$\sum_i R_{ij}^2$
3_CGENI_AB_US	10	2	15	14	189.5	2174.25
3_CGENI_ABC_US	15.5	13	14	15.5	241	3290.5
3_CGENI_AC_US	1	14	8	12	247.5	3460.25
3_CGENI_BC_US	14	15	16	15.5	259.5	3600.25
5_CGENI_AB_US	5	11	5	11	147	1339.5
5_CGENI_ABC_US	12.5	7	12	2	198	2379.5
5_CGENI_AC_US	15.5	8	10.5	6	173	1926
5_CGENI_BC_US	11	10	10.5	4	178	1856.5
7_CGENI_AB_US	4	3	7	9.5	140	1209.5
7_CGENI_ABC_US	7	4.5	4	3	144.5	1408.25
7_CGENI_AC_US	3	9	3	9.5	138	1089
7_CGENI_BC_US	9	6	6	5	146.5	1285.75
9_CGENI_AB_US	2	1	2	1	100.5	814.75
9_CGENI_ABC_US	12.5	4.5	9	7	148	1351
9_CGENI_AC_US	6	12	1	8	108	753.5
9_CGENI_BC_US	8	16	13	13	161	1644
					B_F =24779.475	A_F =29582.5

Table B.13: (Cont.) Ranking for the Friedman test for the TSP on the randomly generated problems for Case 2

Heuristic	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*	*p7*
3_GENI_AB_US	1	1.0035	1.0002	1.0032	1.0034	1.0098	1.0087
3_GENI_ABC_US	1	1.0024	1.0002	1.0025	1.0109	1.0087	1.0104
3_GENI_AC_US	1	1	1	1.0025	1.0085	1.0156	1.0041
3_GENI_BC_US	1	1.0035	1.0056	1.0008	1.0166	1.0056	1.0109
5_GENI_AB_US	1	1.0012	1	1.0039	1.0156	1.0002	1.0041
5_GENI_ABC_US	1	1.0012	1.0025	1.0025	1.0117	1.0002	1.0035
5_GENI_AC_US	1	1	1	1.0025	1.0014	1.0078	1.0029
5_GENI_BC_US	1	1	1.0029	1.0048	1.0087	1.0067	1.0058
7_GENI_AB_US	1	1	1.0022	1.0008	1.0078	1.0062	1
7_GENI_ABC_US	1	1	1	1.0018	1.0011	1	1
7_GENI_AC_US	1	1.0012	1	1.0025	1.0093	1.0074	1.0027
7_GENI_BC_US	1	1	1	1.0041	1.0051	1	1.0041
9_GENI_AB_US	1	1	1	1	1.0023	1	1.0027
9_GENI_ABC_US	1	1	1	1.0008	1.0079	1.0065	1.0058
9_GENI_AC_US	1	1	1	1	1.0007	1.0071	1.0039
9_GENI_BC_US	1	1	1.0025	1.0040	1	1.0052	1.0033

Table B.14: Ratio for the TSP on the randomly generated problems for Case 3

Heuristic	*p8*	*p9*	*p10*	*p11*	*p12*	*p13*	*p14*
3_GENI_AB_US	1.0096	1.0050	1.0050	1.0067	1.0099	1.0025	1.0052
3_GENI_ABC_US	1.0110	1.0086	1.0096	1.0168	1.0077	1.0030	1.0101
3_GENI_AC_US	1.0116	1.0056	1.0025	1.0124	1.0091	1.0090	1.0118
3_GENI_BC_US	1.0090	1.0128	1.0060	1.0106	1.0115	1	1.0050
5_GENI_AB_US	1.0073	1.0111	1.0047	1.0104	1.0050	1.0051	1.0017
5_GENI_ABC_US	1.0070	1.0085	1.0023	1.0081	1.0056	1.0016	1.0062
5_GENI_AC_US	1.0060	1	1.0024	1.0078	1.0100	1.0047	1.0017
5_GENI_BC_US	1.0073	1.0031	1.0033	1.0141	1.0065	1	1.0063
7_GENI_AB_US	1.0098	1.0044	1.0012	1.0069	1	1.0032	1.0040
7_GENI_ABC_US	1.0076	1.0008	1	1.0128	1.0079	1.0006	1.0041
7_GENI_AC_US	1.0058	1.0071	1.0019	1	1.0099	1.0044	1.0001
7_GENI_BC_US	1.0073	1.0090	1.0036	1.0122	1.0109	1.0009	1.0033
9_GENI_AB_US	1.0034	1.0034	1.0018	1.0103	1.0057	1.0010	1.0035
9_GENI_ABC_US	1.0063	1.0062	1.0041	1.0042	1.0061	1.0014	1.0006
9_GENI_AC_US	1	1.0094	1.0018	1.0099	1.0077	1.0046	1.0032
9_GENI_BC_US	1.0071	1.0024	1.0016	1.0062	1.0081	1.0057	1.0071

Table B.14 (Cont.) Ratio for the TSP on the randomly generated problems for Case 3

Heuristic	*p15*	*p16*	*p17*	*p18*	*p19*	*p20*
3_GENI_AB_US	1.0066	1.0018	1.0107	1.0067	1.0050	1.0060
3_GENI_ABC_US	1.0045	1.0067	1.0118	1.0074	1.0116	1.0053
3_GENI_AC_US	1.0073	1.0088	1.0089	1.0083	1.0079	1.0086
3_GENI_BC_US	1.0119	1.0076	1.0114	1.00377	1.0108	1.0106
5_GENI_AB_US	1	1.0015	1.0081	1.0062	1	1.0006
5_GENI_ABC_US	1.0012	1.0050	1.0044	1.0069	1.0052	1.0037
5_GENI_AC_US	1.0066	1.0070	1.0046	1.0084	1.0073	1.0061
5_GENI_BC_US	1.0067	1.0038	1.0077	1.0040	1.0071	1.0074
7_GENI_AB_US	1.0037	1.0001	1.0038	1.0038	1.0008	1.0045
7_GENI_ABC_US	1.0051	1.0047	1.0072	1.0062	1.0051	1.0019
7_GENI_AC_US	1.0026	1.0024	1.0089	1.0067	1.0048	1.0075
7_GENI_BC_US	1.0051	1	1.0048	1.0064	1.0040	1.0060
9_GENI_AB_US	1.0032	1.0018	1.0025	1.0030	1.0037	1.0048
9_GENI_ABC_US	1.0051	1.0023	1	1.0044	1.0019	1.0018
9_GENI_AC_US	1.0032	1.0031	1.0088	1	1.0010	1.0054
9_GENI_BC_US	1.0027	1.0022	1.0093	1.0030	1.0036	1

Table B.14 (Cont.) Ratio for the TSP on the randomly generated problems for Case 3

Heuristic	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*	*p7*	*p8*
3_GENI_AB_US	8.5	15.5	10.5	12	6	15	14	13
3_GENI_ABC_US	8.5	14	10.5	9	13	14	15	15
3_GENI_AC_US	8.5	5.5	5	9	10	16	10	16
3_GENI_BC_US	8.5	15.5	16	4	16	7	16	12
5_GENI_AB_US	8.5	12	5	13	15	4.5	10	9
5_GENI_ABC_US	8.5	12	13.5	9	14	4.5	7	6
5_GENI_AC_US	8.5	5.5	5	9	4	13	5	4
5_GENI_BC_US	8.5	5.5	15	16	11	10	12	9
7_GENI_AB_US	8.5	5.5	12	4	8	8	1.5	14
7_GENI_ABC_US	8.5	5.5	5	6	3	2	1.5	11
7_GENI_AC_US	8.5	12	5	9	12	12	3.5	3
7_GENI_BC_US	8.5	5.5	5	15	7	2	10	9
9_GENI_AB_US	8.5	5.5	5	1.5	5	2	3.5	2
9_GENI_ABC_US	8.5	5.5	5	4	9	9	13	5
9_GENI_AC_US	8.5	5.5	5	1.5	2	11	8	1
9_GENI_BC_US	8.5	5.5	13.5	14	1	6	6	7

Table B.15: Ranking for the Friedman test for the TSP on the randomly generated problems for Case 3

Heuristic	*p9*	*p10*	*p11*	*p12*	*p13*	*p14*	*p15*	*p16*
3_GENI_AB_US	7	14	4	13	8	11	12.5	4.5
3_GENI_ABC_US	12	16	16	7	9	15	8	13
3_GENI_AC_US	8	9	13	11	16	16	15	16
3_GENI_BC_US	16	15	11	16	1.5	10	16	15
5_GENI_AB_US	15	13	10	2	14	4.5	1	3
5_GENI_ABC_US	11	7	7	3	7	12	2	12
5_GENI_AC_US	1	8	6	14	13	4.5	12.5	14
5_GENI_BC_US	4	10	15	6	1.5	13	14	10
7_GENI_AB_US	6	2	5	1	10	8	7	2
7_GENI_ABC_US	2	1	14	9	3	9	10	11
7_GENI_AC_US	10	6	1	12	11	2	3	8
7_GENI_BC_US	13	11	12	15	4	6	10	1
9_GENI_AB_US	5	4.5	9	4	5	7	5.5	4.5
9_GENI_ABC_US	9	12	2	5	6	3	10	7
9_GENI_AC_US	14	4.5	8	8	12	1	5.5	9
9_GENI_BC_US	3	3	3	10	15	14	4	6

Table B.15: (Cont.) Ranking for the Friedman test for the TSP on the randomly generated problems for Case 3

Heuristic	*p17*	*p18*	*p19*	*p20*	R_j	$\sum_i R_{ij}^2$
3_GENI_AB_US	14	11.5	9	10.5	213.5	2503.75
3_GENI_ABC_US	16	14	16	8	249	3290.5
3_GENI_AC_US	12	15	14	15	240	3138.5
3_GENI_BC_US	15	4	15	16	245.5	3452.75
5_GENI_AB_US	9	8	1	2	159.5	1710.75
5_GENI_ABC_US	4	13	11	5	168.5	1680.75
5_GENI_AC_US	5	16	13	12	173	1867
5_GENI_BC_US	8	6	12	13	199.5	2286.75
7_GENI_AB_US	3	5	2	6	118.5	945.75
7_GENI_ABC_US	7	9	10	4	131.5	1138.75
7_GENI_AC_US	11	11.5	8	14	162.5	1623.75
7_GENI_BC_US	6	10	7	10.5	167.5	1693.75
9_GENI_AB_US	2	3	6	7	95.5	539.75
9_GENI_ABC_US	1	7	4	3	128	1022.5
9_GENI_AC_US	10	1	3	9	127.5	1111.25
9_GENI_BC_US	13	2	5	1	140.5	1401.75
					B_F =24815.65	A_F =29408

Table B.15: (Cont.) Ranking for the Friedman test for the TSP on the randomly generated problems for Case 3

Heuristic	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*	*p7*
3_CGENI_AB_US	0.0644	0	0.6203	0.0424	0.9593	0.8966	0.9746
3_CGENI_ABC_US	0	0.2458	0.2594	0.2654	1.4637	1.5856	0.3676
3_CGENI_AC_US	0.0644	0.3512	0.7782	0.4141	0.1483	0.2831	0.5728
3_CGENI_BC_US	0.0644	0.1287	0.0563	0.3928	1.6813	0.9344	0.4616
5_CGENI_AB_US	0	0	0	0	1.6022	0.7361	0.4531
5_CGENI_ABC_US	0	0.1287	0.3947	0.4034	0.3461	0.5663	0.9661
5_CGENI_AC_US	0	0	0.2594	0	1.5626	0.0283	0.1880
5_CGENI_BC_US	0	0	0	0.4141	0.2472	0.7078	0.0683
7_CGENI_AB_US	0	0	0	0.0849	0.8406	0.2831	0.1196
7_CGENI_ABC_US	0	0	0	0	1.0879	0.9627	0.7010
7_CGENI_AC_US	0	0	0.2255	0.0849	1.0582	0.4624	0.4189
7_CGENI_BC_US	0	0	0.2255	0.1805	0.1285	0.1415	0.6241
9_CGENI_AB_US	0	0	0.2594	0.3822	0.5736	0.0283	0.4189
9_CGENI_ABC_US	0	0.1287	0.2594	0	1.0978	0.4058	0
9_CGENI_AC_US	0	0	0.2594	0	0.2175	0.2831	0.3932
9_CGENI_BC_US	0	0	0	0.1805	0	0.4152	0.5642
3_GENI_AB_US	0	0.3512	0.0225	0.3291	0.4747	0.9815	1.0174
3_GENI_ABC_US	0	0.2458	0.0225	0.2548	1.2263	0.8777	1.1798
3_GENI_AC_US	0	0	0	0.2548	0.9890	1.5667	0.5557
3_GENI_BC_US	0	0.3512	0.5639	0.0849	1.8000	0.5660	1.2311
5_GENI_AB_US	0	0.1287	0	0.3928	1.6912	0.0283	0.5557
5_GENI_ABC_US	0	0.1287	0.2594	0.2548	1.3055	0.0283	0.4958
5_GENI_AC_US	0	0	0	0.2548	0.2769	0.7833	0.4274
5_GENI_BC_US	0	0	0.2932	0.4884	1.0088	0.6795	0.7181
7_GENI_AB_US	0	0	0.2255	0.0849	0.9099	0.6229	0.1367
7_GENI_ABC_US	0	0	0	0.1805	0.2472	0	0.1367
7_GENI_AC_US	0	0.1287	0	0.2548	1.0681	0.7456	0.4103
7_GENI_BC_US	0	0	0	0.4141	0.6428	0	0.5557
9_GENI_AB_US	0	0	0	0	0.3659	0	0.4103
9_GENI_ABC_US	0	0	0	0.0849	0.9197	0.6512	0.7267
9_GENI_AC_US	0	0	0.5453	0	0	0.2076	0.7173
9_GENI_BC_US	0	0	0.2594	0.4034	0.1285	0.5285	0.4702
3_CCAUS	1.3792	2.6694	2.3234	2.8031	4.8066	3.7753	2.6419
5_CCAUS	1.0956	2.6343	2.3122	1.4334	3.3428	2.5672	2.9326
7_CCAUS	0.4769	2.6343	2.3122	3.3446	2.2450	3.5299	2.6761
9_CCAUS	0.4769	2.6343	2.3122	1.7732	2.1956	3.0580	2.6761

Table B.16: Percentage deviation of the heuristic solutions and the best known lower bound, x, on randomly generated problems for the TSP

Heuristic	*p8*	*p9*	*p10*	*p11*	*p12*	*p13*	*p14*
3_CGENI_AB_US	0.5514	0.5805	0.4836	0.6513	0.7113	0	0.3513
3_CGENI_ABC_US	0.9524	0.1063	0.4202	0.5907	1.1956	0.9701	1.1917
3_CGENI_AC_US	1.0443	1.5127	0.8721	1.1815	0.8096	0.7703	0.8473
3_CGENI_BC_US	1.0944	0.6950	0.6898	0.7271	0.9610	1.1270	0.4546
5_CGENI_AB_US	0.4762	0.4988	0.4915	0.2272	0.6961	0.6847	0.3444
5_CGENI_ABC_US	0.6851	0.3189	0	1.0906	0.5524	0.7275	0.6200
5_CGENI_AC_US	0.6934	0.0572	0.6263	0.6059	0.9458	0.3851	0.5097
5_CGENI_BC_US	0.7853	0.7850	0.2537	0.9543	0.7945	0.6990	0.4271
7_CGENI_AB_US	0.8772	0.7604	0.4281	0.8255	0.7037	0.2139	0.1860
7_CGENI_ABC_US	0.4093	0.7768	0.1744	0.9316	0.7567	0.7347	0.1997
7_CGENI_AC_US	0.6767	0.1717	0.4519	0.7952	0.7794	0.6633	0.0826
7_CGENI_BC_US	0.6934	0.5069	0.2854	0.6437	0.8475	0.4921	0.5786
9_CGENI_AB_US	0.6683	0.1390	0.0951	0.5983	0.5372	0.7846	0.3788
9_CGENI_ABC_US	0.6851	0.4579	0.1585	0.9240	0.6129	0.3923	0.1791
9_CGENI_AC_US	0.6683	0.2453	0.1823	0.5756	0.7718	0.4707	0
9_CGENI_BC_US	0.1838	0.5151	0.1823	1.0376	0.6507	0.6847	0.1584
3_GENI_AB_US	0.9608	0.5069	0.6739	0.6740	0.9988	0.5920	0.7026
3_GENI_ABC_US	1.1028	0.8667	1.1338	1.6890	0.7718	0.6419	1.1917
3_GENI_AC_US	1.1613	0.5642	0.4202	1.2497	0.9156	1.2411	1.3640
3_GENI_BC_US	0.9023	1.2838	0.7770	1.0679	1.1577	0.3352	0.6820
5_GENI_AB_US	0.7352	1.1121	0.6422	1.0452	0.5069	0.8488	0.3513
5_GENI_ABC_US	0.7018	0.8504	0.4043	0.8179	0.5675	0.4993	0.7991
5_GENI_AC_US	0.6099	0	0.4123	0.7801	1.0064	0.8131	0.3513
5_GENI_BC_US	0.7352	0.3189	0.4995	1.4163	0.6583	0.3352	0.8060
7_GENI_AB_US	0.9858	0.4415	0.2933	0.6968	0	0.6633	0.5786
7_GENI_ABC_US	0.7686	0.0899	0.1665	1.2800	0.7945	0.3994	0.5855
7_GENI_AC_US	0.5848	0.7196	0.3647	0	0.9912	0.7775	0.1860
7_GENI_BC_US	0.7352	0.9076	0.5312	1.2269	1.0972	0.4279	0.5028
9_GENI_AB_US	0.3425	0.3434	0.3488	1.0376	0.5751	0.4422	0.5304
9_GENI_ABC_US	0.6349	0.6214	0.5788	0.4241	0.6129	0.4779	0.2411
9_GENI_AC_US	0	0.9485	0.3488	0.9921	0.7794	0.8060	0.1722
9_GENI_BC_US	0.7185	0.2453	0.3330	0.6210	0.8172	0.9130	0.8886
3_CCAUS	1.9049	2.9111	1.6650	2.8629	3.7079	3.3311	1.8186
5_CCAUS	1.9968	1.1611	1.5382	1.7798	3.5489	2.0900	1.7980
7_CCAUS	1.3367	1.1611	1.2765	2.2267	3.5489	2.1684	1.7980
9_CCAUS	1.3367	1.1611	1.2765	2.2040	3.3900	2.0757	1.6533

Table B.16: (Cont.) Percentage deviation of the heuristic solutions and the best known lower bound, x, on randomly generated problems for the TSP

Heuristic	*p15*	*p16*	*p17*	*p18*	*p19*	*p20*
3_CGENI_AB_US	0.5179	0.7911	0.7995	0.1950	0.8224	0.8179
3_CGENI_ABC_US	0.6525	1.6287	1.2489	0.9876	0.7729	0.8675
3_CGENI_AC_US	0.5449	1.4559	0.0594	1.0946	0.3895	0.7374
3_CGENI_BC_US	0.7534	0.3589	0.9119	1.2833	1.0759	0.8675
5_CGENI_AB_US	0.3296	0.7911	0.4757	0.7800	0.2658	0.5143
5_CGENI_ABC_US	0.7803	0.9972	0.8986	0.6605	0.6554	0.1287
5_CGENI_AC_US	0.6189	1.2963	1.2489	0.7234	0.5874	0.4028
5_CGENI_BC_US	0.2556	0.8443	0.8854	0.7674	0.5874	0.3718
7_CGENI_AB_US	0.1749	1.0171	0.4163	0.2013	0.3029	0.4957
7_CGENI_ABC_US	0.3767	0	0.7004	0.49698	0.2473	0.3098
7_CGENI_AC_US	0.0941	0.5584	0.3370	0.7423	0.2226	0.4957
7_CGENI_BC_US	0.0269	0.4520	0.7929	0.6416	0.2844	0.3780
9_CGENI_AB_US	0.0067	0.1396	0.0660	0	0.2040	0.1363
9_CGENI_ABC_US	0.4709	0.8775	0.8986	0.4969	0.4513	0.4213
9_CGENI_AC_US	0.1547	0.0465	0.5088	0.8555	0.0742	0.4709
9_CGENI_BC_US	0.4641	0.8775	0.7070	1.2959	0.6801	0.7436
3_GENI_AB_US	0.6659	0.8243	1.0771	0.7045	0.5008	0.6011
3_GENI_ABC_US	0.4574	1.3163	1.1828	0.7674	1.1625	0.5329
3_GENI_AC_US	0.7332	1.5290	0.8986	0.8555	0.7976	0.8613
3_GENI_BC_US	1.1907	1.4093	1.1498	0.4026	1.0821	1.0658
5_GENI_AB_US	0	0.7977	0.8127	0.6479	0	0.0619
5_GENI_ABC_US	0.1278	1.1501	0.4493	0.7234	0.5255	0.3718
5_GENI_AC_US	0.6659	1.3429	0.4625	0.8681	0.7358	0.6196
5_GENI_BC_US	0.6727	1.0304	0.7797	0.4340	0.7172	0.7436
7_GENI_AB_US	0.3767	0.6515	0.3898	0.4089	0.0865	0.4585
7_GENI_ABC_US	0.5112	1.1168	0.7202	0.6542	0.5194	0.1983
7_GENI_AC_US	0.2623	0.8841	0.8920	0.7045	0.4823	0.7560
7_GENI_BC_US	0.5112	0.6382	0.4889	0.6731	0.4019	0.6011
9_GENI_AB_US	0.3229	0.8243	0.2511	0.3334	0.3710	0.4895
9_GENI_ABC_US	0.5112	0.8775	0	0.4718	0.1916	0.1859
9_GENI_AC_US	0.3229	0.9573	0.8854	0.0251	0.1051	0.5453
9_GENI_BC_US	0.2758	0.8642	0.9317	0.3271	0.3648	0
3_CCAUS	2.5025	1.4625	2.7423	2.8057	1.9416	3.5818
5_CCAUS	2.1459	1.5888	2.4251	2.5729	0.8533	3.1418
7_CCAUS	2.2603	1.3296	2.4251	2.4345	0.2102	2.4601
9_CCAUS	1.8634	1.3096	2.1608	2.4849	0.2226	2.0449

Table B.16: (Cont.) Percentage deviation of the heuristic solutions and the best known lower bound, x, on randomly generated problems for the TSP

Heuristics	*Problems*	θ	*Avg*	*Max*	*Min*
3_TR	60	0	237.5564	1768.9958	96.9684
	60	0.33	2234.9847	6346.4771	1013.11
	60	0.67	3428.6286	5743.9868	1928.985
	70	0	453.1362	2136.5752	148.0993
	70	0.33	3521.3148	6464.9248	916.3478
	70	0.67	6710.9231	12710.8945	2704.053
	80	0	1395.9621	3566.5703	298.0847
	80	0.33	76278.7704	15860.5566	2808.843
	80	0.67	6720.2301	12976.5332	3864.989
	90	0	815.4179	4368.6094	288.361
	90	0.33	7472.8743	14750.7168	4281.2036
	90	0.67	9244.5631	23234.2422	6265.6646
	100	0	1766.5985	7150.2915	532.3311
	100	0.33	12531.7682	29862.3555	4732.7114
	100	0.67	14944.2568	24566.9062	9984.7119
	110	0	1442.7178	4045.1799	286.078
	110	0.33	6125.1091	22020.2812	2288.6021
	110	0.67	25793.7119	45487.6211	15282.1133
	120	0	849.3315	2766.9734	198.3229
	120	0.33	9308.9945	35635.875	3808.7434
	120	0.67	15376.9596	37326.8203	8180.697
5_TR	60	0	479.8037	1377.3898	121.2088
	60	0.33	2745.5551	6595.647	1113.4293
	60	0.67	4344.4562	10065.0439	2064.3862
	70	0	361.347	958.7052	181.918
	70	0.33	3988.0489	9106.1523	1660.2761
	70	0.67	8040.6092	13978.7666	4256.0186
	80	0	1429.088	3962.5876	463.8429
	80	0.33	7551.6667	16430.3594	3663.6875
	80	0.67	8995.1722	30182.668	5943.1289
	90	0	1295.0065	3978.9236	403.8918
	90	0.33	9954.8268	24040.9863	5325.3857
	90	0.67	12981.145	21756.4082	7489.313
	100	0	1761.6704	5439.48	599.4321
	100	0.33	16963.0449	39832.5977	6930.9741
	100	0.67	20201.672	26961.127	13730.6738
	110	0	2216.943	9569.0225	652.5438
	110	0.33	22914.9227	52694.0078	9386.1982
	110	0.67	36885.0431	72151.1406	23263.5762
	120	0	716.6344	4758.8276	292.2454
	120	0.33	21003.719	59908.4883	7746.4707
	120	0.67	40961.4724	81183.3125	20920.4727

Table B.17: CPU time for various heuristics and problems for CVRP

Heuristics	*Problems*	θ	*Avg*	*Max*	*Min*
	60	0	264.859	888.5994	182.7571
	60	0.33	3263.5624	8490.7041	1442.662
	60	0.67	6432.8156	13313.3525	3119.7461
	70	0	587.6851	4880.5459	307.6363
	70	0.33	4885.654	10212.8906	2238.6794
	70	0.67	12139.3609	23991.2383	3652.2393
	80	0	1896.7009	4057.1406	552.9423
	80	0.33	10518.0645	31177.9395	5050.8687
7_TR	80	0.67	12424.4113	35029.5977	5369.4414
	90	0	552.3931	1590.3988	266.8705
	90	0.33	5597.5221	14076.7148	2655.5669
	90	0.67	7844.2219	15161.2559	4561.6328
	100	0	2807.0371	8553.5732	830.3708
	100	0.33	20190.1297	58946.0273	9077.5928
	100	0.67	14624.5785	28992.0098	9548.501
	110	0	879.8393	4235.1992	292.1474
	110	0.33	10828.6287	27325.7383	3801.0698
	110	0.67	23292.5882	46511.4844	14246.0029
	120	0	983.3389	2891.4153	452.817
	120	0.33	19141.0622	34118.5586	8409.598
	120	0.67	31470.7941	63322.6797	12548.1064
	60	0	492.0439	1951.0975	251.7275
	60	0.33	3831.6523	9953.5742	1828.7433
	60	0.67	7736.4583	20456.7871	3696.8167
	70	0	732.5354	2181.6436	398.9037
	70	0.33	6469.3581	11981.5586	2582.2588
	70	0.67	14576.3382	26389.082	4034.8188
	80	0	2158.8872	4550.9917	844.3342
	80	0.33	12765.0469	28906.4746	5574.0562
	80	0.67	14679.1878	25725.1816	5008.5381
	90	0	628.8134	1687.6681	315.7257
	90	0.33	7409.6235	24504.4336	3091.4841
9_TR	90	0.67	9538.8135	15826.8066	6346.9829
	100	0	1229.13	3249.0081	451.7488
	100	0.33	10284.3587	26413.4004	4820.9292
	100	0.67	21094.6431	43981.0703	13450.1045
	110	0	2159.0415	6916.6689	576.6367
	110	0.33	16558.5615	38720.9883	7308.646
	110	0.67	31254.8077	63677.4375	17650.7188
	120	0	889.747	1451.1758	610.9728
	120	0.33	33490.6921	68498.4297	11553.8145
	120	0.67	36204.3628	64688.4922	23563.5

Table B.17: (Cont.) CPU time for various heuristics and problems for CVRP

Heuristics	*Problems*	θ	*Avg*	*Max*	*Min*
	60	0	91.4658	403.7842	37.0375
	60	0.33	299.6909	882.53	56.4964
	60	0.67	417.8179	2899.628	69.1873
	60	1	221.4221	550.4393	109.1583
	70	0	143.3803	324.678	68.7589
	70	0.33	353.1786	1039.097	108.5089
	70	0.67	310.7105	662.1725	134.6801
	70	1	550.5395	1451.16	263.961
	80	0	605.6641	1520.528	192.4874
	80	0.33	3453.189	92959.18	346.2704
	80	0.67	1024.137	5955.112	322.8848
3_TR	80	1	518.6251	1141.502	247.2048
	90	0	580.2012	2547.663	167.8304
	90	0.33	1122.983	3653.372	192.2978
	90	0.67	650.3455	2095.03	304.7519
	90	1	596.4514	2074.287	146.8414
	100	0	795.9616	2444.602	269.3813
	100	0.33	1577.632	4504.363	645.0719
	100	0.67	1042.804	2819.647	665.6364
	100	1	1226.758	4105.629	555.1238
	110	0	1317.354	5092.291	321.8372
	110	0.33	1552.556	6241.788	692.4044
	110	0.67	2389.266	8607.697	784.1398
	110	1	2029.404	6253.177	827.9911
	120	0	1064.751	3392.222	332.8735
	120	0.33	2124.932	5957.85	324.2401
	120	0.67	2192.492	8235.302	1072.92
	120	1	1900.938	8482.218	440.6808

Table B.18: CPU time for various problems on 3_TR for CDVRP

Heuristics	*Problems*	θ	*Avg*	*Max*	*Min*
	60	0	317.3374	1533.117	63.2211
	60	0.33	327.7784	1261.082	59.1955
	60	0.67	1001.227	5607.92	181.7493
	60	1	379.4076	969.2667	122.3564
	70	0	333.147	2208.991	88.8582
	70	0.33	637.7458	2529.564	145.0143
	70	0.67	677.0948	1693.696	206.5935
	70	1	843.581	1920.55	234.4286
	80	0	1585.653	3845.671	362.5106
	80	0.33	2199.765	13419.03	518.6433
	80	0.67	2216.417	6349.86	322.7588
	80	1	1063.535	3245.304	172.7024
	90	0	601.3566	1770.835	231.7165
	90	0.33	1393.088	3583.186	543.5041
	90	0.67	969.1379	4780.03	449.14
5_TR	90	1	1429.524	7237.5	532.4551
	100	0	938.6132	3284.673	383.5909
	100	0.33	1849.693	4224.337	700.4343
	100	0.67	1047.254	2629.211	363.4647
	100	1	1070.346	2700.708	441.3165
	110	0	1095.18	4749.581	353.7647
	110	0.33	1758.117	5897.814	633.9471
	110	0.67	2476.075	8512.382	775.8676
	110	1	2665.932	6610.774	967.6436
	120	0	997.2349	6266.61	475.5369
	120	0.33	2766.291	8526.02	456.7755
	120	0.67	2286.557	7266.052	1246.639
	120	1	2797.643	7666.105	537.3464

Table B.19: CPU time for various problems on 5_TR for CDVRP

Heuristics	*Problems*	θ	*Avg*	*Max*	*Min*
7_TR	60	0	140.8014	784.6545	70.444
	60	0.33	370.5199	838.8128	122.9931
	60	0.67	1023.614	2455.84	155.1746
	60	1	231.0896	412.3763	136.9723
	70	0	240.0102	805.3698	134.1521
	70	0.33	505.6273	2064.36	184.1071
	70	0.67	567.506	1506.8	251.3545
	70	1	1232.064	4458.91	388.6183
	80	0	857.2558	1899.872	331.2632
	80	0.33	1356.708	3769.536	315.7648
	80	0.67	1728.605	6736.96	248.9617
	80	1	847.1529	5780.78	197.5079
	90	0	673.1002	1922.811	346.0992
	90	0.33	2417.261	8846.94	591.2238
	90	0.67	983.8583	2279.877	539.7195
	90	1	820.5672	3275.003	187.7541
	100	0	1164.535	4218.234	438.8352
	100	0.33	2231.273	15165.76	951.1664
	100	0.67	1176.113	3357.347	413.7453
	100	1	1580.283	4126.207	846.7077
	110	0	1576.383	5494.737	512.952
	110	0.33	1708.168	5426.039	310.9808
	110	0.67	3169.725	22220.18	871.7354
	110	1	2957.037	8119.226	1002.574
	120	0	1209.197	9903.661	636.6119
	120	0.33	2844.387	8096.103	687.4197
	120	0.67	2437.627	11145.57	1525.277
	120	1	3165.13	16914.8	613.295

Table B.20: CPU time for various problems on 7_TR for CDVRP

Heuristics	*Problems*	θ	*Avg*	*Max*	*Min*
	60	0	241.3917	924.9945	112.097
	60	0.33	402.009	897.47	91.6341
	60	0.67	797.9925	2621	86.8334
	60	1	327.7953	789.0801	202.4427
	70	0	402.9915	1190.1	202.1509
	70	0.33	533.0048	2496.13	251.9802
	70	0.67	821.5097	2233.17	271.7017
	70	1	1693.862	7291.72	524.56
	80	0	898.6741	1969.629	390.029
	80	0.33	1471.691	4666.595	431.6797
	80	0.67	1605.325	3691.7	327.0862
	80	1	1007.208	3396.796	232.2562
	90	0	762.5736	3146.992	498.2704
	90	0.33	2595.932	13617.24	514.278
	90	0.67	920.6093	1945.916	273.3677
9_TR	90	1	1885.471	32311.82	266.1679
	100	0	1746.467	4554.93	621.6412
	100	0.33	4214.518	76788.17	936.819
	100	0.67	1090.289	2137.654	552.1661
	100	1	1737.685	4077.154	550.3028
	110	0	2577.257	7584.188	730.4549
	110	0.33	3483.789	33508.02	415.3878
	110	0.67	3236.889	9615.815	756.2133
	110	1	3305.496	14221.09	479.6924
	120	0	1384.698	4086.99	911.1996
	120	0.33	2961.918	12495.92	587.5276
	120	0.67	2743.446	8743.357	544.7787
	120	1	3154.182	17528.71	673.4438

Table B.21: CPU time for various problems on 9_TR for CDVRP

Starting solution	*Final solution*	*gap*
3738.205383	3549.784094	188.421289
3737.526927	3627.390183	110.136744
3687.929709	3628.496363	59.433346
3719.947139	3576.905808	143.041331
3658.621466	3524.428616	134.19285
3715.450563	3608.344914	107.105649
3712.641891	3611.019286	101.622605
3673.357062	3652.704283	20.652779
3646.893872	3633.506052	13.38782
3673.357062	3658.878542	14.47852
3696.413103	3549.251196	147.161907
3699.845979	3588.123885	111.722094
3700.967377	3637.076382	63.890995
3719.947139	3593.77145	126.175689
3750.652521	3624.338941	126.31358
3713.645241	3490.591405	223.053836
3683.299999	3577.372264	105.927735
3688.470069	3612.28323	76.186839
3648.731982	3519.912752	128.81923
3638.421303	3619.561186	18.860117
3712.641891	3595.9108	116.731091
3702.377403	3585.014945	117.362458
3674.743245	3489.16507	185.578175
3689.864192	3598.411542	91.45265
3707.591448	3557.512752	150.078696
3678.916092	3587.211036	91.705056
3653.358999	3518.856932	134.502067
3694.21328	3624.967554	69.245726
3661.824091	3511.205533	150.618558
3712.831126	3541.200354	171.630772
3705.415754	3594.77108	110.644674
3739.366723	3537.250444	202.116279
3717.811747	3528.137373	189.674374
3695.832358	3495.513226	200.319132
3719.947139	3626.303611	93.643528
3638.421303	3619.561186	18.860117
3684.778161	3649.808845	34.969316
3669.692478	3517.11258	152.579898
3734.290087	3624.89726	109.392827
3692.145714	3533.892457	158.253257
3715.59696	3577.118772	138.478188
3700.967377	3617.059051	83.908326
3661.824091	3593.271264	68.552827
3711.728202	3573.092396	138.635806
3677.303364	3487.066539	190.236825
3688.524822	3529.832627	158.692195
3716.061927	3628.496363	87.565564

Table B.22: Starting solution, final solution and their gaps for 140 runs for the 199-city problem

Starting solution	*Final solution*	*gap*
3717.109689	3599.274745	117.834944
3715.59696	3564.32157	151.27539
3713.704176	3586.117912	127.586264
3717.811747	3496.140035	221.671712
3665.833025	3548.803984	117.029041
3718.114469	3605.150745	112.963724
3715.59696	3566.211032	149.385928
3678.714448	3608.213314	70.501134
3744.541072	3582.461342	162.07973
3678.964856	3518.973645	159.991211
3699.058487	3583.735308	115.323179
3693.300026	3566.334927	126.965099
3758.500236	3596.549216	161.95102
3665.696717	3600.1815	65.515217
3681.298085	3567.809079	113.489006
3660.028448	3648.064046	11.964402
3641.128852	3572.176126	68.952726
3676.207335	3565.523636	110.683699
3721.684416	3521.986306	199.69811
3684.778161	3487.819204	196.958957
3684.267898	3603.390863	80.877035
3684.778161	3575.292377	109.485784
3675.915015	3633.434003	42.481012
3674.557132	3599.421359	75.135773
3667.01398	3602.469582	64.544398
3671.29311	3584.306476	86.986634
3722.242282	3518.750774	203.491508
3717.811747	3576.937235	140.874512
3669.01892	3629.587138	39.431782
3719.947139	3517.738002	202.209137
3663.651964	3580.57306	83.078904
3684.267898	3612.192444	72.075454
3676.010944	3541.916023	134.094921
3664.141736	3597.676547	66.465189
3648.731982	3521.811193	126.920789
3675.647931	3581.705932	93.941999
3707.473599	3591.917517	115.556082
3646.893872	3514.221977	132.671895
3675.647931	3619.514114	56.133817
3739.522101	3610.2544	129.267701
3724.496857	3585.903956	138.592901
3744.541072	3564.932251	179.608821
3715.59696	3523.759387	191.837573
3689.864192	3479.608153	210.256039
3674.065177	3471.786906	202.278271
3675.647931	3555.549966	120.097965
3717.811747	3576.213641	141.598106

Table B.22: (Cont.) Starting solution, final solution and their gaps for 140 runs for the 199-city problem

Starting solution	*Final solution*	*gap*
3658.621466	3496.229903	162.391563
3724.496857	3587.667817	136.82904
3697.7452	3626.86886	70.87634
3740.050526	3623.53902	116.51150
3726.36547	3609.617353	116.74811
3678.916092	3546.530439	132.385653
3716.506524	3556.799785	159.706739
3688.470069	3642.953997	45.516072
3675.915015	3621.979154	53.935861
3737.526927	3596.624357	140.90257
3704.343727	3585.130735	119.212992
3765.323145	3571.554762	193.768383
3717.109689	3491.351517	225.758172
3719.024647	3557.537813	161.486834
3663.550157	3570.801069	92.749088
3734.290087	3594.160923	140.129164
3711.728202	3624.89726	86.830942
3736.55232	3508.084159	228.468161
3660.763277	3587.770281	72.992996
3675.647931	3544.466143	131.181788
3688.470069	3538.191484	150.278585
3687.844925	3511.021893	176.823032
3675.647931	3551.543151	124.10478
3678.066974	3561.727417	116.339557
3696.635791	3483.788748	212.847043
3705.313811	3506.32439	198.989421
3637.844587	3573.136848	64.707739
3746.040879	3551.84666	194.194219
3765.323145	3638.421303	126.901842
3722.242282	3512.904212	209.33807
3711.728202	3575.445698	136.282504
3661.824091	3634.648347	27.175744
3691.019086	3558.217951	132.801135
3638.421303	3568.93002	69.491283
3637.404192	3551.048797	86.355395
3696.413103	3500.959051	195.454052
3715.59696	3569.241615	146.355345
3712.641891	3579.506175	133.135716
3665.696717	3582.070112	83.626605
3689.864192	3493.983446	195.880746
3713.704176	3602.616765	111.087411
3655.857368	3628.991065	26.866303
3687.929709	3622.650626	65.279083
3669.692478	3622.420976	47.271502
3687.929709	3588.879147	99.050562
3676.024635	3547.553311	128.471324
μ= 3694.77892	3572.7409	122.0379
σ = 29.0166	45.0586	52.5593

Table B.22: (Cont.) Starting solution, final solution and their gaps for 140 runs for the 199-city problem

Bibliography

[1] Aarts, E. and Korst, Ji. (1989), "Simulated annealing and boltzman machines", John Wiley & Sons Ltd., Chichester.

[2] Aarts, E. and Lenstra, J. K. (ed.) (1997), "Local search in combinatorial Optimization", John Wiley & Sons Ltd., Chichester.

[3] Achuthan, N. R. and Caccetta, L. (1991), "Integer linear programming formulation for a vehicle routing problem", *European Journal of Operational Research*, **52**, 86 - 89.

[4] Achuthan, N. R., Caccetta, L. and Hill, S. P. (1996a), "An improved branch and cut algorithm for the capacitated vehicle routing problem", *Technical Report 2/96, School of Mathematics and Statistics, Curtin University of Technology, Perth, Western Australia.*

[5] Achuthan, N. R., Caccetta, L. and Hill, S. P. (1996b), "The vehicle routing problem with capacity and distance restrictions", *Technical Report 6/96, School of Mathematics and Statistics, Curtin University of Technology, Perth, Western Australia.*

[6] Achuthan, N. R., Caccetta, L. and Hill, S. P. (1996c), "A new subtour elimination constraint for the vehicle routing problem", *European Journal of Operational Research*, **91**, 573 - 586.

[7] Achuthan, N. R., Caccetta, L. and Hill, S. P. (1998), "Capacitated Vehicle Routing Problem: Some new Cutting Planes", *Asia-Pacific Journal of Operational Research, 15*, 109 - 123.

[8] Achuthan, N. R. and Chong, Y. N. (1998), "Statistical Evaluation of Genius Algorithm for the Symmetric Travelling Salesman Problem", in *The Fourth International Conference of Optimization: Techniques and Applications*, eds. Caccetta, L., Teo, K. L., Siew, P. F., Leung, Y. H., Jennings, L. S. and Rehbock, V. , Curtin University of Technology, Western Australia, 618 - 625.

[9] Achuthan, N. R. and Chong, Y. N. (1999a), "Algorithm Taburoute for Vehicle Routing Problems: Choice of Parameter Values and Quality of Soltuion Generated", in *The 15th National Conference of The Australian Society for Operations Research Inc. ASOR Queensland*, ed. Kozan, E., ASOR, Gold Coast, Australia, 79 - 88.

[10] Achuthan, N. R. and Chong, Y. N. (1999b), "Complexity of the Solution Space of a Combinatorial Optimization Problem", Presented in *The 15th Triennial Conference, the International Federation of Operational Research Societies (IFORS)*, Beijing, China.

[11] Applegate, D., Bixby, R., Chvátal, V. and Cook, W. (1995), "Finding cuts in the TSP (a preliminary report)", *DIMACS Technical Report 95-05.*

[12] Applegate, D., Bixby, R., Chvátal, V. and Cook, W. (1998), "On the solution of Traveling Salesman Problems", *Documenta Mathematica, Extra Volume ICM III*, 645 - 656.

[13] Araque, J. R., Kudva, G., Morin, T. L. and Pekny, J. F. (1994), "A Branch-and-Cut Algorithm for Vehicle Routing Problems", *Annals of Operations Research*, **50**, 37 - 59.

[14] Augerat, P., Belenguer, J. M., Benavent, E., Corberan, A., Naddef, D. and Rinaldi, G. (1995), "Computational results with a branch and cut code for the capacitated vehicle routing problem", *Research Report 949-M, Universite Joseph Fourier, Grenoble, France.*

[15] Bachem, A., Hochstättler, W. and Malich, M. (1996), "The Simulted Trading Heuristic for Solving Vehicle Routing Problems", *Discrete Applied Mathematics*, **65**, 47 - 72.

[16] Balas, E. and Christofides, N. (1981), "A restricted lagrangean approach to the traveling salesman problem", *Mathematical Programming*, **21**, 19 - 46.

[17] Balas, E. and Toth, P. (1985), "Branch and bound methods", in [104], 361 - 401.

[18] Bazaraa, M. S. and Goode, J. J. (1977), "The traveling salesman problem: a duality approach", *Mathematical Programming*, **13**, 221 - 237.

[19] Bellmore, M. and Malone, J. C. (1971), "Pathology of traveling salesman subtour elimination algorithms", *Operations Research*, **19**, 278 - 307.

[20] Bodin, L. D., Golden, B. L., Assad, A. A. and Ball, M. O. (1983), "Routing and Scheduling of Vehicles and Crews, the State of the Art", *Computers and Operations Research*, **10**, 69 - 211.

[21] Bramel, J. and Simchi-Levi, D. (1995), "A location based heuristic for general routing problems", *Operations Research*, **43(4)**, 649 - 660.

[22] Bramel, J. and Simchi-Levi, D. (1997), "The Logic of Logistics: Theory, Algorithms and Applications", Springer-Verlag, New York.

[23] Brandt, R.D., Wang, Y., Laub, A.J. and Mitra, S.K. (1988), "Alternative networks for solving the traveling salesman problem and the list- matching problem", *IEEE International Conference on Neural Networks*, IEEE, New York II 333 - 340.

[24] Breedam, A. V. (1995), "Improvement heuristics for the vehicle routing problem based on simulated annealing", *European Journal of Operational Research*, **86**, 480 - 490.

[25] Budinich, M. (1996), "A self-organizing Neural Network for the Traveling Salesman Problem that is competitive with Simulated Annealing", *Neural Computation*, **8**, 416 - 424.

[26] Carpaneto, G., Dell'amico, M. and Toth, P. (1995), "Exact solution of large scale, asymmetric traveling salesman problems", *ACM Transaction on Mathematical Software*, **21(4)**, 394 - 409.

[27] Carpaneto, G., Fischetti, M. and Toth, P. (1989), "New lower bounds for the symmetric travelling salesman problem", *Mathematical Programming*, **45**, 233 - 254.

[28] Carpaneto, G. and Toth, P. (1980), "Some new branching and bounding criteria for the asymmetric traveling salesman problem", *Management Science*, **26(7)**, 736 - 743.

[29] Chardaire, P., Lutton J. L. and Sutter, A. (1995), "Thermostatistical persistency: A powerful improving concept for simulated annealing algorithms", *European Journal of Operational Research*, **86**, 565 - 579.

[30] Chatterjee, S., Carrera, C. and Lynch L. A. (1996), "Genetic algorithms and traveling salesman problems", *European Journal of Operational Research*, **93**, 490 - 510.

[31] Christofides, N. (1970), "The shortest Hamiltonian chain of a graph", *SIAM J. Appl. Math*, **19**, 689 - 696.

[32] Christofides, N. (1985), "Vehicle routing", In [104], 431 - 448.

[33] Christofides, N. and Eilon, S. (1969), "An algorithm for the vehicle- dispatching problem", *Operational Research Quarterly*, **20**, 309 - 318.

[34] Christofides, N., Mingozzi, A. and Toth, P. (1979), "The vehicle routing problem" In *Combinatorial Optimization*, Christofides, N., Mingozzi, A., Toth, P. and Sandi, C. (ed.), Wiley, Chichester, 315 - 338.

[35] Christofides, N., Mingozzi, A. and Toth, P. (1981a), "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxation", *Mathematical Programming*, **20**, 255 - 282.

[36] Christofides, N., Mingozzi, A. and Toth, P. (1981b), "State-space relaxation procedures for the computation of bounds to routing problems", *Networks*, **11**, 145 - 164.

[37] Clarke, G. and Wright, J. W. (1964), "Scheduling of vehicles from a central depot to a number of delivery points", *Operations Research*, **12(4)**, 568 - 581.

[38] Conway, R. W., Maxwell, W. L. and Miller, L. W. (1967), "Theory of Scheduling", *Addison-Wesley Publishing Company, Massachusetts.*

[39] Cornuejols, G. and Harche, F. (1993), "Polyhedral study of the capacitated vehicle routing problem", *Mathematical Programming*, **60**, 21 - 52.

[40] Croes, G. A. (1958), "A method for solving traveling salesman problems", *Oper. Res.*, **6**, 791 - 812.

[41] Crowder, H. and Padberg, M. (1980), "Soving large scale symmetric traveling salesman problems to optimality", *Management Science*, **26(5)**, 495 - 509.

[42] Dantzig, G.B., Fulkerson, D.R., and Johnson, S.M. (1954), "Solution of a large-scale traveling-salesman problem", *Operations Reseach*, **2**, 393 - 410.

[43] Davis, L. (1985), "Job Shop Schedulling with Genetic Algorithms", in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, ed. Grefenstette, J. J., Lawrence Erlbaum Associates, Hillsdale, NJ, 136 - 140.

[44] Desrochers, M. and Laporte, G. (1991), "Improvements and extensions to the Miller-Tucker-Zemin subtour elimination constraints", *Operations Research Letters*, **10**, 27 - 36.

[45] Durbin, R. and Willshaw, D. (1987), "An analogue approach to the traveling salesman problem using an elastic net method", *Nature*, **326** , 689 - 691.

[46] Dowsland, K. A. (1993), "Simulated Annealing" in [145].

[47] Eastman, W. L. (1958), "Linear programming with pattern constraints", PhD thesis Havard University, Cambridge, MA.

[48] El Ghaziri, H. (1991), "Solving routing problems by a self-organizing map", In *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula and J. Kangas (Eds.), North-Holland, Amsterdam, 829 - 834.

[49] Fiechter, C. N. (1994), "A parallel tabu search algorithm for large traveling salesman problems", *Discrete Applied Mathematics*, **51**, 243 - 267.

[50] Fischetti, M. and Toth, P. (1989), "An additive bounding procedure for combinatorial optimization problems", *Operations research*, **37(2)**, 319 - 328.

[51] Fisher, M. L. (1994), "Optimal solution of vehicle routing problems using minimum K-trees", *Operations Research*, **42(4)**, 626 - 642.

[52] Gavin, W. M., Crandall, H. W., John, J. B. and Spellman, R. A. (1957), "Applications of Linear Programming in the Oil Industry", *Management Science*, **3**, 407 - 430.

[53] Gavish, B. and Srikanth, K. (1986), "An optimal solution method for large-scale multiple traveling salesmen problems", *Operations Research*, **34(5)**, 698 - 717.

[54] Garcia, B. L., Potvin, J. Y. and Rousseau, J. M. (1994), "A parallel implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints", *Computers Ops. Res.*, **21(9)**, 1025 - 1033.

[55] Gaskell, T. J. (1967), "Bases for vehicle fleet scheduling", *Operational Research Quaterly*, **18**, 281 - 295.

[56] Gendreau, M., Hertz, A. and Laporte, G. (1992), "New insertion and postoptimization procedures for the traveling salesman problem", *Operations Research*, **40(6)**, 1086 - 1094.

[57] Gendreau, M., Hertz, A. and Laporte, G. (1994), "A Tabu Search Heuristic for the Vehicle Routing Problem", *Management Science*, **40**, 1276 - 1290.

[58] Gendreau, M., Laporte, G. and Potvin, J. (1997), "Vehicle routing: modern heuristics", in [2], 311 - 336.

[59] Gillett, B. E. and Miller, L. R. (1974), "A heuristic algorithm for vehicle-dispatch problem", *Operations Research*, **22**, 340 - 349.

[60] Glover, F. (1986), "Future paths of integer programming and links to artificial intelligence", *Computers and Operations Research*, **13**, 533 - 549.

[61] Glover, F. (1990), "Tabu search: A tutorial", *Interfaces*, **20(4)**, 74 - 94.

[62] Glover, F. (1995), "Tabu search fundamentals and uses", *Working Paper, University of Colorado, Boulder.*

[63] Glover, F. (1996), "Tabu search and adaptive memory programming - advances, applications and challenges", *Interfaces in Computer Science and Operations Research* (To appear).

[64] Glover, F. and Laguna, M. (1997), "Tabu Search", Klumer Academic Publishers, Boston.

[65] Goldberg, C. R. and Lingle, R. (1985), "Alleles, loci and the traveling salesman problems", in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, ed. Grefenstette, J. J., Lawrence Erlbaum Associates, Hillsdale, NJ, 154 - 159.

[66] Golden, B. L. (1977), "A Statistical Approach to the TSP", *Networks*, **7**, 209 - 225.

[67] Golden, B. L. (1978), "Point estimation of a global optimum for large combinatorial problems", *Commun. Statist. Simula. Computa.*, **b7(4)**, 361 - 367.

[68] Golden, B. L. and Assad, A. A. (1984), "A decision-theoretic framework for comparing heuristics", *European Journal of Operational Research*, **18**, 167 - 171.

[69] Golden, B. L. and Assad, A. A. (1984), "Vehicle Routing: Methods and Studies", *North-Holland, Amsterdam.*

[70] Golden, B. L., Bodin, L, D., Doyle, T. and Stewart, Jr. W., (1980), "Approximate travelling salesman algorithm", *Oper. Res.*, **28**, 694 - 711.

[71] Golden, B. L. and Stewart, W. R. (1985), "Empirical analysis of heuri stics", in [104], 207 - 250.

[72] Gomory, R.E. and Hu, T.C. (1961), "Multi-terminal network flows", *SIAM J. Appl. Math.*, **9**, 551 - 556.

[73] Grefenstette, J.J., Gopal, R., Rosmaita, B. and Van Gucht, D. (1985), "Genetic algorithms for traveling salesman problem", *Proc. of the first International conference on Genetic Algorithm.*

[74] Grötschel, M. and Holland, O. (1991), "Solution of large scale symmetric traveling salesman problems", *Mathematical Programming*, **51**, 141 - 202.

[75] Grötschel, M. and Padberg, M. (1979a), "On the symmetric traveling salesman problem I: inequalities", *Mathematical Programming*, **16**, 265 - 280.

[76] Grötschel, M. and Padberg, M. (1979b), "On the symmetric traveling salesman problem II: lifting theorems and facets", *Mathematical Programming*, **16**, 281 - 302.

[77] Grötschel, M. and Padberg, M. (1985), "Polyhedral theory", in [104], 251 - 305.

[78] Grötschel, M. and Pulleyblank, W. R. (1985), "Clique tree inequalities and the symmetric traveling salesman problem", *Math. Oper. Res.*, **11**, 537 - 569.

[79] Hansen, P. (1986), "The Steepest Ascent, Mildest Descent Heuristic for Combinatorial Programming", Presented at the Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

[80] Helbig Hansen, K. and Krarup, J. (1974), "Improvements of the Held-Karp algorithm for the symmetric traveling salesman problem", *Mathematical Programming*, **7**, 87 - 96.

[81] Held, M. and Karp, R. M. (1970), "The traveling salesman problem and minimum spanning trees", *Oper. Res.*, **18**, 1138 - 1162.

[82] Held, M. and Karp, R. M. (1971), "The traveling salesman problem and minimum spanning trees: Part II", *Mathematical Programming*, **1**, 6 - 25.

[83] Hill, S. P. (1995), "Branch and Cut Methods for the Symmetric Capacitated Vehicle Routing Problems", PhD Thesis, Curtin University of Technology, Perth, Western Australia.

[84] Homaifar, A., Guan, S. and Liepins, G. E. (1993), "A New Approach on the Traveling Salesman Problems by Genetic Algorithms", *Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA*, 460 - 466.

[85] Houck Jr, D. J., Picard, J. C., Queyranne, M. and Vemuganti, R. R. (1980), "The traveling salesman problem as a constrained shortest parh problem: theory and computational experience", *Opsearch*, **17**, 93 - 109.

[86] Johnson, D. S. and McGeoch, L. A. (1997), "The traveling salesman problem: a case study", in [2], 215 - 310.

[87] Johnson, D. S., McGeoch, L. A. and Rothberg, E. E. (1996), "Asymptotic experimental analysis for the Held-Karp traveling salesman bound", *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM, New York and SIAM, Philadelphia, PA, 341 - 350.

[88] Johnson, D. S. and Papadimitriou, C. H. (1985), "Performance guarantees for heuristics", in [104], 145 - 180.

[89] Jünger, M., Reinelt, G. and Rinaldi, G. (1995), "Travelling Salesman Problem", in *Networks Models*, Ball, M., Magnanti, T., Monma, C. and Nemhauser, G. (ed.), Handbook in Operations Research and Management Science, Vol 7, 225 - 330.

[90] Jünger, M., Reinelt, G. and Thienel, S. (1994), "Provably good solutions for Traveling Salesman Problem", *Zeitschrift für Operations Research*, **40**, 183 - 217.

[91] Jünger, M. and Störmer, P. (1995), "Solving large scale traveling salesman problems with parallel branch-and-cut", *Technical Report 95.191, Universität zu Köln, Germany.*

[92] Karp, R. M. and Steele, J. M. (1985), "Probabilistic analysis of heuristic", in [104], 181 - 206.

[93] Kindervater, G.A.P. and Savelsbergh, M.W.P. (1997), "Vehicle routing: handling edge exchanges", in [2], 337 - 360.

[94] Kirkpatrick, S., Gellat, C. D. and Vecchi, M. P. (1983), "Optimization by simulated annealing", *Science*, **220**, 671 - 680.

[95] Knox, J. (1994), "Tabu Search Performance on The Symmetric Traveling Salesman Problem", *Computers and Operations Research*, **21(8)**, 867 - 876.

[96] Kohonen, T. (1988), "Self-organization and associative memory", Springer, Berlin.

[97] Krolak, P., Felts, W. and Marble, G. (1971), "A man-machine approach toward solving the traveling salesman problem", *Communications of the ACM*, **14**, 327 - 334.

[98] Laporte, G. (1992), "The Traveling Salesman Problem: An overview of exact and approximate algorithms", *European Journal of Operational Research*, **59**, 231 - 247.

[99] Laporte, G. (1992), "The vehicle routing problem: An overview of exact and approximate algorithms", *European Journal of Operational Research* , **59**, 345 - 358.

[100] Laporte, G., Gendreau, M., Potvin, J. and Semet, F. (1999), "Classical and modern heuristics for the vehicle routing problem", in *The 15th National*

Conference of The Australian Society for Operations Research Inc. ASOR Queensland, ed. Kozan, E., ASOR, Gold Coast, Australia, 1-16.

[101] Laporte, G. and Nobert, Y. (1987), "Exact algorithms for the vehicle routing problem", *Annals of Discrete Mathematics*, **31**, 147 - 184.

[102] Laporte, G., Nobert, Y. and Desrochers, M. (1984), "Two Exact Algorithms for the Distance-Constrained Vehicle Routing Problem", *Networks*, **14**, 161 - 172.

[103] Laporte, G., Nobert, Y. and Desrochers, M. (1985), "Optimal routing under capacity and distance restrictions", *Operations Research*, **33(5)**, 1050 - 1073.

[104] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. and Shmoys, D. B. (ed.) (1985), "The Traveling Salesman Problem", John Wiley &Sons Ltd., Chichester.

[105] Lenstra, J. K. and Rinnooy Kan, A. H. G. (1981), "Complexity of vehicle routing and scheduling problems", *Networks*, **11**, 221 - 227.

[106] Lin, S. (1965), "Computer solutions of the traveling salesman problem", *Bell System Tech. J.*, **44**, 2245 - 2269.

[107] Lin, F. T., Kao, C. Y and Hsu, C. C. (1993), "Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Problems", *IEEE Transactions on Systems, Man and Cybernetics*, **23**, 1752 - 1767.

[108] Lin, S., and Kernighan, B. W. (1973), "An effective heuristic algorithm for the Traveling Salesman Problem", *Operations Research*, **21**, 498 - 516.

[109] Lin, S. and Hsueh J. (1994), "A new methodology of simulated annealing for the optimisation problems", *Physica A*, **205**, 367 - 374.

[110] Little, J., Murty, K., Sweeney, D. and Karel, C. (1963), "An algorithm for the traveling salesman problem", *Operations Research*, **11**, 972 - 989.

[111] Malek, M., Guruswamy, M., Pandya, M. and Owens, H. (1989), "Serial and Parallel Simulated Annealing and Tabu Search Algorithm for The Traveling Salesman Problem", *Annals of Operations Research*, **21**, 59 - 84.

[112] Malik, K. and Fisher, M. L. (1990), "A dual ascent algorithm for the 1-tree relaxation of the symmetric traveling salesman problem", *Operations research Letters*, **9**, 1 - 7.

[113] Martello, S. and Toth, P. (1990), "Knapsack Problems: Algorithms and Computer Implementations", John Wiley.

[114] Matsuyama, Y. (1991), "Self-organizaion via competition, cooperation and categorization applied to extended vehicle routing problems", *IJCNN-91 Seattle:International Joint Conference on Neural Networks*, IEEE, New York, I 385 - 390.

[115] Meeran, S. and Shafie, A. (1997), "Optimum Path Planning Using Convex Hull and Local Search Heuristic Algorithms", *Mechatronics*, **7**, 737 - 756.

[116] Miliotis, P. (1976), "Integer programming approaches to the traveling salesman problem", *Mathematical Programming*, **10**, 367 - 378.

[117] Miliotis, P. (1978), "Using cutting planes to solve the symmetric traveling salesman problem", *Mathematical Programming*, **15**, 177 - 188.

[118] Miller, C.E., Tucker, A.W. and Zemlin, R.A. (1960), "Integer programming formulations and traveling salesman problems", *J. Assoc. Comput. Mach.*, **7**, 326 - 329.

[119] Miller, D. L. and Pekny, J. F. (1991), "Exact solution of large asymmetric traveling salesman problems", *Science*, **251**, 754 - 761.

[120] Mingozzi, A., Christofides, N. and Hadjiconstantinou, E. (1994), "An exact algorithm for the vehicle routing problem based on the set partitioning formulation", *Working Paper*, University of Bologna, Italy.

[121] Naddef, D. and Rinaldi, G. (1991), "The symmetric Traveling Salesman Polytope and its Graphical Relaxation: Composition of Valid Inequalities", *Mathematical Programming*, **51**, 359 - 400.

[122] Naddef, D. and Rinaldi, G. (1993), "The Graphical Relaxation: A New Framework for the Symmetric Travelling Salesman Polytope", *Mathematical Programming*, **58**, 53 - 88.

[123] Nagata, Y. and Kobayashi, S. (1997), "Edge assembly crossover: A high-power Genetic Algorithm for the Traveling Salesman Problem", in *Proceedings of the 7th International Conference on Genetic Algorithms*, ed. Bäck, Morgan Kaufmann, 450 - 457.

[124] Nemhauser, G. L. and Wolsey, L. A. (1988), "Integer and Combinatorial Optimization", John Wiley and Sons.

[125] Norback, J.P. and Love, R.F. (1977), "Geometric approaches to solving the traveling salesman problem", *Management Sci.*, **23**, 1208 - 1223.

[126] Norback, J.P. and Love, R.F. (1979), "Heuristic for the Hamiltonian path problem in Euclidean two space", *Journal of Operations Research Society*, **30**, 363 - 368.

[127] Oliver, I. M., Smith, D. J. and Holland, J. R. C. (1987), "A study of permutation crossover operators on the traveling salesman problem", in *Proceedings of the 2nd International Conference on Genetic Algorithms*, ed. Grefenstette, J. J., Lawrence Erlbaum Associates, Hillsdale, NJ, 224 - 230.

[128] Or, I. (1976), "Traveling salesman type combinatorial problems and their relation to the logistics of regional blood banking", Doctoral thesis, Northwestern University, Evanston.

[129] Osman, I.H. (1993), "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", *Annals of Operations Research*, **41**, 421 - 451.

[130] Otten, R.H.J.M. and van Ginneken, L.P.P.P. (1989), "The annealing algorithm", Kluwer Academic Publishers, Boston.

[131] Padberg, M. and Hong, S. (1980), "On the symmetric travelling salesman problem: a computational study", *Mathematical Programming Study*, **12**, 78 - 107.

[132] Padberg, M. and Rao, M.R. (1982), "Odd minimum cut-sets and b- matching", *Math. Oper. Res.*, **7**, 67 - 80.

[133] Padberg, M. and Rinaldi, G. (1987), "Optimization of a 532-city symmetric traveling salesman problem by branch and cut", *Operations Reseach Letters*, **6**, 1 - 7.

[134] Padberg, M. and Rinaldi, G. (1990a), "Facet identification for the symmetric traveling salesman polytope", *Mathematical Programming*, **47**, 219 - 257.

[135] Padberg, M. and Rinaldi, G. (1990b), "An efficient algorithm for the minimum capacity cut problem", *Mathematical Programming*, **47**, 19 - 36.

[136] Padberg, M. and Rinaldi, G. (1991), "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems", *SIAM Review*, **33(1)**, 60 - 100.

[137] Paessens, H. (1988), "The savings algorithm for the vehicle routing problem", *European Journal of Operational Research*, **34**, 336 - 344.

[138] Papadimitriou, C. H. (1992), "The complexity of the Lin-Kernighan heuristic for the traveling salesman problem", *SIAM J. Comput.*, **22(3)**, 450 - 465.

[139] Pekny, J. F. and Miller, D. L. (1992), "A parallel branch and bound algorithm for solving large asymmetric traveling salesman problems", *Mathematical Programming*, **55**, 17 - 33.

[140] Pekny, J. F., Miller, D. L. and Stodolsky, D. (1991), "A note on exploiting the Hamiltonian cycle problem substructure of the Asymmetric Traveling Salesman Problem", *Operations Research Letters*, **10**, 173 - 176.

[141] Perttunen, J. (1994), "On the Significance of the Initial Solution in Travelling Salesman Heuristics", *J. Opl Res. Soc*, **45**, 1131 - 1140.

[142] Peterson, C. and Söderberg, B. (1993), "Artificial Neural Networks", in [145].

[143] Poon, P. W. and Carter, J. N. (1995), "Genetic Algorithm Crossover Operators for Ordering Applications", *Computers Ops. Res.*, **22(1)**, 135 - 147.

[144] Potvin, J. Y. and Robillard, C. (1995), "Clustering for vehicle routing with a competitive neural network", *Neurocomputing*, **8**, 125 - 139.

[145] Reeves, C. R. (ed) (1993), "Modern Heuristic techniques for Combinatorial Problems", Blackwell, Oxford.

[146] Reinelt, G. (1994), "The Traveling Salesman Problem Computational solutions for TSP applications", *Springer-Verlag Berlin Heidelberg*

[147] Reinelt, G. (1995), "TSPLIB95", *Universitat Heidelberg, Institut fur Angewandte Mathematik.* (can be obtained from ftp://ftp.zib-berlin.de/pub/mp-testdata/tsp/index.html).

[148] Renaud, J., Boctor, F. F. and Laporte, G. (1996), "An improved petal heuristic for the vehicle routing problem", *Journal of the Operational Research Society*, **47**, 329 - 336.

[149] Rochat, Y. and Semet, F. (1994), "A Tabu Search Approach for Delivering Pet Food and Flour in Switzerland", *Journal of Operational Research*, **45**, 1233 - 1246.

[150] Rochat, Y. and Taillard, E.D. (1995), "Probabilistic Diversification and Intersification in Local Search for Vehicle Routing", *CRT-95-13, ORWP 95/03.*

[151] Rosenkrantz, D.J., Stearns, R.E. and Lewis II, P.M. (1977), "An analysis of several heuristics for the traveling salesman problem", *SIAM J. Comput.*, **6**, 563 - 581.

[152] Ryan, D. M., Hjorring, C. and Glover, F. (1993), "Extension of the petal method for vehicle routing", *J. Opl. Res. Soc.*, *44*, 289 - 296.

[153] Scheaffer, R. L. and McClave, J. T. (1990), "Probability and Statistics for Engineers", PWS-Kent Publishing Company, Boston.

[154] Semet, F. and Taillard, E. (1993), "Solving real-life vehicle routing problems efficiently using tabu search", *Annals of Operations Research*, **41**, 469 - 488.

[155] Smith, W. E. (1956), "Various Optimizers for single state production", *Nav. Res. Log. Quart.*, **3(1)**, 59 - 66.

[156] Smith, T.H.C. (1978), "A LIFO implicit enumeration algorithm for the asymmetric travelling salesman problem using a one-arborescence relaxation", *Annals of Discrete Mathematics 1*, 495 - 506.

[157] Smith, T.H.C., Srinivasan, V. and Thompson, G.L. (1977), "Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems", *Ann. Discrete Math*, **1**, 495 - 506.

[158] Smith, T.H.C. and Thompson, G.L. (1977), "A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation", *Annals of Discrete Mathematics*, **1**, 479 - 493.

[159] Somhon, S., Modares, A. and Enkawa, T. (1997), "A self-organising model for the travelling salesman problem", *Journal of the Operational Research Society*, **48**, 919 - 928.

[160] Sun, T., Meakin, P. and Jossang, T. (1993), "A fast optimization method based on a hierarchical strategy for the travelling salesman problem", *Physica A*, **199**, 232 - 242.

[161] Taillard, E. (1993), "Parallel Iterative Search Methods for Vehicle Routing Problems", *Networks*, **23**, 661 - 673.

[162] Tillman, F. A. and Cochran, H. (1968) "A heuristic approach for solving the delivery problem", *J. Ind. Engineering*, **19**, 354.

[163] Tsubakitani, S. and Evans, J. R. (1998a), "An empirical study of a new metaheuristic for the traveling salesman problem", *European Journal of Operational Research*, **104**, 113 - 128.

[164] Tsubakitani, S. and Evans, J.R. (1998b), "Optimizing Tabu List for The Traveling Salesman Problem", *Computers and Operations Research*, **25**, 91 - 97.

[165] Whitley, D., Starkweather, T. and Fuquay, D. (1989), "Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator", in *Proceedings of the 3rd International Conference on Genetic Algorithms*, ed. Schaffer, J. D., Morgan Kaufmann, San Mateo, CA, 133 - 140.

[166] Wong, R. T. (1980), "Integer programming formulations of the travelling salesman problem", *Proceedings of the IEEE International Conference on Circuits and Computers*, 149 - 152.

[167] Xu, J. and Kelly, J.P. (1996), "A network flow-based tabu search heuristic for the Vehicle Routing Problem", Report - University of Colorado, Boulder.

[168] Yagiura, M. and Ibaraki, T. (1996), "The use of dynamic programming in genetic algorithms for permutation problems", *European Journal of Operational Research*, **92**, 387 - 401.

[169] Yellow, P. (1970), "A computational modification to the savings method of vehicle scheduling", *Operational Research Quarterly*, **21**, 281 - 283.