2007

Efficient routing of snow removal vehicles

Masoud Omer
West Virginia University

EFFICIENT ROUTING OF SNOW REMOVAL VEHICLES

Masoud Omer

Thesis submitted to the College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Industrial Engineering

Approved by

Dr. Wafik Iskander, Committee Chairperson
Dr. Majid Jaraiedi
Dr. Bhaskaran Gopalakrishnan

Industrial and Management Systems Engineering

Morgantown, West Virginia
2007

Keywords: Vehicle Routing, Arc Routing, Heuristics, Simulated Annealing, Optimization

Abstract

EFFICIENT ROUTING OF SNOW REMOVAL VEHICLES

Masoud Omer

This research addresses the problem of finding a minimum cost set of routes for vehicles in a road network subject to some constraints. Extensions, such as multiple service requirements, and mixed networks have been considered. Variations of this problem exist in many practical applications such as snow removal, refuse collection, mail delivery, etc. An exact algorithm was developed using integer programming to solve small size problems. Since the problem is NP- hard, a heuristic algorithm needs to be developed. An algorithm was developed based on the Greedy Randomized Adaptive Search Procedure (GRASP) heuristic, in which each replication consists of applying a construction heuristic to find feasible and good quality solutions, followed by a local search heuristic. A simulated annealing heuristic was developed to improve the solutions obtained from the construction heuristic. The best overall solution was selected from the results of several replications. The heuristic was tested on four sets of problem instances (total of 115 instances) obtained from the literature. The simulated annealing heuristic was able to achieve average improvements of up to 26.36% over the construction results on these problem instances. The results obtained with the developed heuristic were compared to the results obtained with recent heuristics developed by other authors. The developed heuristic improved the best-known solution found by other authors on 18 of the 115 instances and matched the results on 89 of those instances. It worked specially better with larger problems. The average deviations to known lower bounds for all four datasets were found to range between 0.21 and 2.61%.

Acknowledgement

I would like to thank Dr. Iskander for his guidance, support and patience while working on this thesis. Throughout my thesis-writing period, he provided encouragement, sound advice, and lots of good ideas. I am grateful for his countless hours of sincere help and for his detailed and constructive comments.

I would also like to gratefully acknowledge the support and help of my committee members, Dr. Majid Jaraiedi, and Dr. Bhaskaran Gopalakrishnan.

I am thankful to Dr. Christian Prins and Dr. Wahiba Ramdane-Cherif for their suggestions in this research and for providing me access to datasets and their solutions. I would also like to thank my friend Stanimir Mihai for his help and suggestions in programming the heuristic as part of this research.

Finally, I am forever indebted to my family for their understanding, endless patience and encouragement when it was most required.

Table of Contents

Abstract ii
Acknowledgement iii
1 Introduction 1
1.1 Background 1
1.2 Importance of the Vehicle Routing Problem (VRP) 1
1.3 Classification of Vehicle Routing Problems 2
1.4 Classification of Solution Approaches 4
1.4.1 Complexity of Routing Problems 6
1.5 Public Sector Service and Snow Removal Operation 7
1.5.1 Snow Removal Operation 7
1.6 Graph Theory 10
1.7 Statement of the Problem and Research Objectives 11
1.8 Restrictions 12
1.8.1 Distance Limit 12
1.8.2 Time Limit 12
1.8.3 Capacity Restrictions 13
1.8.4 Road Directions and Frequency of Service 13
1.8.5 Prevention of Sub tours: 13
1.9 Assumptions 13
1.10 Research Objectives 14
2 Literature Review 15
2.1 Introduction to Arc Routing Problems 15
2.2 Literature Review 18
2.2.1 Rural Postman Problem 18
2.2.1.1 Undirected Rural Postman Problem (URPP) 19
2.2.1.2 Directed Rural Postman Problem (DRPP) 19
2.2.2 Capacitated Arc Routing Problem (CARP) 20
2.2.2.1 Meta-heuristics for the CARP 22
2.2.3 Arc Routing for Snow Removal 27
3 Exact Algorithm 33
3.1 Introduction 33
3.2 Mathematical Model 33
3.2.1 Objective Function 34
3.2.2 Constraints 34
4 Heuristic Algorithm 40
4.1 Introduction 40
4.2 Phase 1: Route Construction Algorithm (RCA) 40
4.2.1 Route Construction Algorithm (RCA) Example 43
4.3 Phase 2: Simulated Annealing Heuristic 47
4.3.1 Background 47
4.3.2 SA Heuristic Algorithm 49
4.3.2.1 Route Improvement Algorithm (RI) 50
4.3.2.2 Types of Moves 57
5 Experimentation 65

5.1 Introduction 65
5.2 Problem Instances 65
5.3 Experimentation Overview 66
5.3.1 Selection of the threshold parameter α in the RCA 66
5.3.2 Selection of the SA parameters 66
5.3.2.1 Values of parameters used 67
5.4 Comparisons of the results 68
5.4.1 Comparison of results with the Mathematical Programming (MP) model 68
5.4.2 Comparison of the results with problem instances in literature 69
6 Conclusion 76
6.1 Overview of Research 76
6.2 Contribution to Literature 77
6.3 Recommendations for Future Research 77
REFERENCES 79
APPENDIX A 92
APPENDIX B 94
APPENDIX C 95

List of Tables

Table 2-1 Summary of Meta-heuristics for the CARP 25
Table 3-1 Mathematical Model Nomenclature 33
Table 3-2 Example Problem Parameters (i) 39
Table 3-3 Example Problem Parameters (ii) 39
Table 4-1 RCA Nomenclature 41
Table 4-2 Example Problem Parameters 44
Table 4-3 RCA Example Distance Matrix 45
Table 4-4 Possible Splitting Points and Rearrangements 51
Table 4-5 List of Arcs Including the Inserted Arc 53
Table 4-6 Matrix for Creating a Route in the RI Example 54
Table 4-7 Simulated Annealing Nomenclature 59
Table 5-1 Problem Instances 65
Table 5-2 Simulated Annealing Parameters 68
Table 5-3 Comparison of Results with the MP model 69
Table 5-4 Computational Results for the gdb Instances 72
Table 5-5 Computational Results for the val Instances 73
Table 5-6 Computational Results for the egl Instances 74
Table 5-7 Computational Results for the mval Instances 75
Table 6-1 Summary of Results 77

List of Figures

Figure 2.1 The seven bridges of Konigsberg (from Eiselt et al. 1995a) 15
Figure 3.1 Example to Illustrate the Sub-tour Breaking Constraint 37
Figure 3.2 Example to Illustrate the Mathematical Formulation 38
Figure 4.1 Example to Illustrate the RCA 44
Figure 4.2 SA Flow Diagram 62

List of Acronyms

ADT = Average Daily Traffic

ARP = Arc Routing Problem

CARP = Capacitated Arc Routing Problem

CASPER = Computer Assisted System for Planning Efficient Routes

CCPP = Capacitated Chinese Postman Problem

CMST= Capacitated Minimum Spanning Tree

CPP = Chinese Postman Problem

DCPP = Directed Chinese Postman Problem

DRPP = Direct Rural Postman Problem

ECARP = Extended Capacitated Arc Routing Problem

GLS = Guided Local Search

GRASP = Greedy Randomized Adaptive Search Procedure

HCPP = Hierarchical Chinese Postman Problem

IP = Integer Programming

LB = Lower Bound

MA = Memetic Algorithm

MCARP = Mixed Capacitated Arc Routing Problem

MCARP = Mixed Capacitated Arc Routing Problem

MCPP = Mixec Chinese Postman Problem

MP = Mathematical Programming

NP = Non-polynomial

RCA = Route Construction Algorithm

RCL = Restricted Candidate List

RI = Route Improvement

RPP = Rural Postman Problem

SA = Simulated Annealing

SCP = Stacker Crane Problem

TSP = Traveling Salesman Problem

UCARP = Undirected Capacitated Arc Routing Problem

URPP = Undirected Rural Postman Problem

VND = Variable Neighborhood Descent

VRP = Vehicle Routing Problem

WPP = Windy Postman Problem

CHAPTER 1

1 Introduction

1.1 Background

The routing and scheduling of vehicles is an important area for both transportation planners and operations researchers. The recent advances in computing technology has led to breakthroughs in problem formulations and solution approaches in this area. It has been well documented that efficient scheduling and routing of vehicles can save industry and government millions of dollars every year (Bodin and Golden, 1981). The field of vehicle routing and scheduling covers diverse activities such as snow removal, postman delivery, meter reading, school bus routing, refuse collection, street maintenance, etc.

The basic routing problem is: Given a set of nodes and/or arcs to be serviced by a fleet of vehicles, determine low cost and feasible routes for each vehicle starting and ending at a depot. A vehicle route is a sequence of points or nodes, which the vehicle must traverse in order, starting and ending at a depot. When the arrival and departure times at the nodes are specified, the problem is called a vehicle-scheduling problem. When the arrival times at the nodes are not specified, the problem is the simple routing problem (Bodin and Golden, 1981).

1.2 Importance of the Vehicle Routing Problem (VRP)

Effective distribution management presents different types of decision-making problems at all three levels of strategic, tactical, and operational planning. Decisions related to the location of facilities (depots, plants, etc.) may be categorized as strategic, while the problems related to fleet size and mix determination could be termed tactical. Decisions associated with the routing and scheduling of vehicles and the staffing decision for these vehicles may be termed as operational planning decisions. There is a significant relationship among these three types of decisions. For example, for routing of vehicles, the location of the depot is the input. Conversely, location decisions also depend on transportation costs between various geographic locations (Bodin et al. 1983). This research is primarily concerned with vehicle routing decisions.

The issues related to vehicle routing have a significant effect on decision making in both private and public sectors. In the case of snow removal, the objective is to find efficient

routes of snow removal trucks so that roads are cleared as soon as possible in order to have safe traffic conditions.

The significance of routing/distribution problems is evident from the magnitude of the associated distribution costs. The costs associated with operating vehicles for delivery and service purposes form an important part of the total distribution costs and is one of the largest cost components for many businesses.

An efficient distribution network can significantly reduce costs and improve profitability. Small percentage savings in these costs could result in substantial total savings over a number of years. Due to escalating fuel costs, growing salaries for crew, and high capital costs of replacing the vehicles, the significance of these potential savings has become increasingly apparent. The use of analytical routing and scheduling models and techniques coupled with an effective management information system can play a crucial role in the operational planning of distribution activities (Bodin et al. 1983).

Although the main objective of most routing problems is cost minimization, other objectives may assume primary importance, especially in the context of public sector. Some public sector problems may focus on safety and convenience. For example, in routing and scheduling of street sweepers and vehicles for household refuse collection (Bodin and Kursh 1978), minimizing the number of U-turns or left hand turns is usually important since such turns are dangerous to make on major arteries. These types of turns are also not preferred in the context of the snow removal problem. In certain type of routing and scheduling problems, the choice of an appropriate objective function constitutes an important modeling question (Bodin et al. 1983).

1.3 Classification of Vehicle Routing Problems

There are a number of characteristics that describe any vehicle routing or scheduling problem. A specific vehicle routing or scheduling problem can be classified on the basis of these characteristics. The following taxonomy from Bodin and Golden (1981) outlines the complexity and wide range of problem characteristics that can exist for a vehicle routing problem.

A. Time to serve a particular node or arc:
    1. time specified and fixed in advance (pure vehicle scheduling problem)
    2. time windows (combined vehicle routing and scheduling problem)

3. time unspecified (pure vehicle routing problem, unless there are precedence relationships)

B. Number of domiciles
    1. one domicile
    2. more than one domicile

C. Size of vehicle fleet available
    1. one vehicle
    2. more than one vehicle

D. Type of fleet available
    1. all vehicles are the same (homogeneous case)
    2. not all vehicles are the same (heterogeneous case)

E. Nature of demands
    1. deterministic
    2. probabilistic

F. Location of demands
    1. at nodes (not necessarily all)
    2. on arcs (not necessarily all)
    3. mixed

G. Underlying network
    1. undirected
    2. directed
    3. mixed

H. Vehicle capacity constraints
    1. imposed - all the same
    2. imposed - not all the same
    3. not imposed

I. Maximum vehicle route times
    1. imposed - all the same
    2. imposed - not all the same
    3. not imposed

J. Costs
    1. variable or routing costs

2. fixed operating or capital costs

K. Operations
   1. pickups only
   2. drop-offs only
   3. mixed
   4. split deliveries (allowed or disallowed)

L. Objective
   1. minimize routing costs incurred
   2. minimize sum of fixed and variable costs
   3. minimize number of vehicles required
   4. maximize utility functions based on convenience or service
   5. maximize utility function based on priorities

M. Other problem dependent constraints.

A large number of different problem scenarios requiring a unique modeling assumption will emerge considering the possible combinations of the characteristics. The focus of this research is the snow removal problem, in which the demands are located on the arcs (Classification F).

1.4 Classification of Solution Approaches

Dantzig and Ramser (1959) first introduced the vehicle routing problem and proposed a linear programming based algorithm for its solution. Since then, the majority of attempts for solving the problems have focused on heuristic approaches (Magnanti, 1981). Marshal Fisher (1995) categorized the solution approaches to vehicle routing into three categories based on the time they were generated. The first category is of simple heuristics developed in the 1960s and 1970s, which were mainly based on sweep or local search. The second category (1980s) consists of mathematical programming based heuristics such as generalized assignment problem and set partitioning problem to approximate the VRP. The third category is currently undergoing heavy research and consists of artificial intelligence methods and exact optimization algorithms. Examples of optimization algorithms are K-tree, branch and bounds, etc. Some examples of artificial intelligence based methods are simulated annealing and Tabu search (Tan et al., 2001).

The Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP) are both members of the class of NP-complete problems. All known exact algorithms for these problems require a number of computational steps that grows as an exponential function of the number of nodes/arcs that must be covered. Christofides (1976) stated that the largest vehicle routing problem of any complexity solved till then by exact methods and reported in literature contained only 31 demand points (Magnanti, 1981).

Solution strategies for vehicle routing problems can be classified as follows (Bodin and Golden, 1981):

1. Cluster first - route second
2. Route first - cluster second
3. Improvement/exchange
4. Savings/insertion
5. Mathematical programming based
6. Interactive optimization
7. Exact procedures

Cluster first - route second procedures group demand nodes and/or arcs first and then determine routes over each cluster. For examples, refer to Gillett and Miller (1974) and Gillett and Johnson (1976).

Route first - cluster second procedures are the antithesis of cluster first-route second procedures. First, a large route is constructed including all of the demand nodes and/or arcs. The large route is then partitioned into a number of smaller, feasible routes. Newton and Thomas (1974) and Bodin and Berman (1979) used this approach for routing of school buses.

Improvement or exchange procedures such as the branch exchange heuristic developed by Lin (1965) and Lin and Kernighan (1973) involve altering the current solution to yield another feasible solution with a reduced cost. This process continues till no further reductions in cost are possible. Bodin and Sexton (1979) modified this approach to schedule minibuses for the subscriber dial-a-ride problem.

Savings or insertion procedures involve building a solution in such a way that at each step of the procedure, a current configuration of routes that is possibly infeasible is compared to a possible alternative configuration (that may also be infeasible). The current configuration is replaced with the configuration that gives the highest savings in terms of

some criterion such as total cost, or the one that inserts a demand arc or node into the existing routes least expensively. Examples of savings/insertion procedures for node and arc routing problems are presented by Clarke and Wright (1964), Golden et al. (1980), and Golden and Wong (1981).

Mathematical programming procedures include algorithms based on a mathematical programming formulation of the underlying routing problem. Christofides et al. (1981) and Stewart and Golden (1979) discuss Langrangean relaxation procedures for the routing of vehicles. Fisher and Jaikumar (1981) provided an example of mathematical programming based procedures.

Interactive optimization based approaches involve human input or interaction into the solution process. The underlying basis is that experienced decision makers have the capability of setting and revising parameters and providing assessments based on experience into the optimization model. Some adaptations of this approach to vehicle routing problems are presented by Krolak et al. (1971), (1972).

Exact procedures include specialized branch and bound and cutting plane algorithms for solving vehicle routing problems. Refer to Held and Karp (1970), (1971) and Christofides et al. (1981) for examples of exact procedures for VRP.

1.4.1 Complexity of Routing Problems

An important consideration in the formulation and solution of routing and scheduling problems is the computational complexity associated with different solution approaches for these problems. As most routing problems are NP-hard, known approaches for solving these problems optimally suffer from an exponential growth in computational time with problem size. If a problem is NP-hard, one frequently resorts to heuristic procedures to obtain near optimal solutions. A heuristic algorithm is a procedure that uses the problem structure in a mathematical and usually intuitive way to arrive at a feasible near-optimal solution (Bodin et al., 1983).

Most public sector operations are on preset routes. The effectiveness of such operations depend on the efficiency of those routes. The next section discusses the background of the public sector route design and the snow removal process.

1.5 Public Sector Service and Snow Removal Operation

An important factor in the development of strategies for the delivery of public services is the design of service routes. Effective route design in the public sector not only provides benefits in terms of reduced costs, but also offers a variety of intangible benefits such as quality and equity of service. Public sector route design is generally more difficult than private sector route design because the decision-making environment within which routing and delivery of service takes place is more complex. The conditions can vary dramatically over time and space. As a result, there may exist uncertainties about the level of demand for services. Intangible objectives, such as the public's safety and satisfaction with the quality of service are difficult to measure. In addition, public sector routing is generally characterized by multiple and often conflicting objectives (Wright, 1993).

There are other significant differences between public sector and private sector routing problems. Whereas private sector routing problems are generally evaluated in economic terms, public sector vehicle routing problems are usually concerned more with improving service and public welfare (Marks and Stricker, 1971). Haslam (1988) stated that snow removal in winter season is perhaps the most complex service for which careful route planning is important.

The primary focus of this research is the design of service routes for snow removal operation. Next section gives a brief overview of the snow removal operation.

1.5.1 Snow Removal Operation

Refer to Minsk (1970) and Gray and Male (1981) for some historical perspectives on solving the snow removal problem. Cortina and Low (2001), Keseling (1994), and Lindsey and Seely (1999) provided background and information on snow removal procedures in Brighton, NY, Maryland and Utah, respectively.

There are four major steps in snow removal and disposal operations, all of which require solving the underlying arc routing problem to find efficient routes for vehicles through road networks. They are (Campbell and Langevin, 1995):

1. spreading chemicals and abrasives,
2. snow plowing,
3. snow loading, and
4. snow disposal.

According to the Wisconsin Department of Transportation report in Better Roads Magazine (October 2003), spreading chemicals (also called deicing chemicals) prevents ice forming. Deicing chemicals work by reducing the freezing point of water. A dry deicing chemical must dissolve into a brine solution before it can act. The necessary moisture can come from snow on the road surface or from humidity. The heat required to change ice or snow into water comes from the air, sun, pavement, or traffic friction. The resulting brine creates a film over the road surface. This film prevents compacted snow from bonding to the road surface, which enables easy removal of snow by plows. Abrasives such as sand and other agents improve vehicle traction on snow and ice-covered roads. They are especially useful when it is too cold for chemical deicers to work, as they work efficiently at all temperatures. Sand is the most common abrasive, but slag and cinders are also used commonly. Abrasives must be treated with salt to keep them usable.

In areas that experience heavy snowfall and persistent low temperatures, snow plowed to the side of streets and sidewalks impedes traffic. Therefore, it must first be windrowed in the middle of the street by plows before it can be physically loaded into trucks by snowblowers and transported to disposal sites, where it remains until melting.

The Snow Removal Problem is considered to be one of the most complex public service operations because of the dynamic environment and complexities of equipment, infrastructure and operations. There is also a significant variation in roadway snow removal operations themselves due to the vast differences in climate, network complexity, and size and level of service (Campbell and Langevin, 2000). The problem of snow and ice removal is complex as each city has its own unique conditions, which makes the problem increasingly complex (Cook and Alprin, 1976). For example (Cook and Alprin, 1976), New York City, with a population (at that time) of eight million people, and an average snowfall of 30 inches, had snow removal problems very different from the city of Tulsa, Oklahoma, with an approximate population of 500,000 and an average annual snowfall of 10 inches. A major problem in New York City is the plowing and removal of snow whereas the city of Tulsa relies only on spreading of salt and abrasives to keep the major arteries clear in an ice or snow emergency.

In developing a strategy for winter season snow removal, the goal of authorities is to provide efficient service within the constraints on available resources (salt and sand supplies, plowing and abrasive spreading equipment, and manpower) without excessively interfering

with public transportation (Haslam, 1988). In a Chicago study, Cook and Alprin (1976) concluded that "The vehicular accident rate is highest when a light snow or freezing rain has not been given treatment… Therefore, early salt application can significantly reduce the number and severity of accidents, which create congestion and involve the cost of lost time, property damage and personal injury… The accident rate on a wet or slippery pavement, a condition produced by a light untreated snow, is 330 percent that of a dry pavement… Light snow or freezing rain creates the most severe accident problems. If the rate can be cut in half by faster and better handling, the potential annual saving (to Chicago) is $1,500,000 or more".

Other characteristics of the problem include (Campbell and Langevin, 2000):

1. Weather conditions such as temperature, wind, accumulation rate, etc. can vary significantly even over small distances due to topographic features and water sources and can vary dramatically over large distances. Weather conditions also vary temporally, and as conditions change, the appropriate snow control actions will change. For example, light dry snow, heavy wet snow, and freezing rain all require different actions, yet they may all occur in a matter of few hours.
2. There are different environments in which the problems occur in terms of location, equipment, infrastructure, policies, and operation. Rural problems are often simpler due to sparser road networks; and for many storms, plowing snow off of the roadways or spreading chemicals and abrasives is sufficient. Also, in case of heavier storms, the availability of open areas adjacent to rural roadways means that snow can be piled beside the road and left over to accumulate. In urban areas, the problem is more complex due to the need for not only clearing the roadways but also to clear sidewalks, crosswalks, fire hydrants, public transit stops, and intersections.
3. There are also problems due to turn restrictions for U-turns and turns across traffic lanes. A snowplow driving on the right side of a roadway and designed to move snow towards the right edge of the roadway will cause a row of snow to accumulate in the middle of an intersection if the snowplow truck takes a left turn, which would be objectionable to the general public.
4. Snow removal operation also becomes more complex due to the wide range of equipment available. This equipment includes plows, spreader vehicles, and snow loaders and each one of them is available in multiple variations. A single authority

may include a diverse mix of such equipment. For example, spreader trucks have different carrying capacities for materials and can distribute a variety of different materials, at different rates. Also, trucks can spread one or two lanes in a single pass.

5. The wide range of roadways available in a jurisdiction lead to different arc routing problems. A typical agency may treat a mix of different types of roadways, such as one way roads, single lane roads, multiple lane roads, gravel roads, etc. Some roads under another authority's jurisdiction may be available for travel, but need not be cleared.
6. The location of depots for chemicals and abrasives, and vehicle garages also provides restrictions on snow removal operations. Several silos or depots may be located across an area so that the spreader truck can be refilled with chemical and/or abrasives without returning to the main depot.
7. Many agencies assign priorities for treatment to certain roadways to ensure that travel on high traffic roads remains safe. Assignment of these priorities may be based on Average Daily Traffic (ADT) or some other parameter. Higher priority roads may be served first and with greater frequency than relatively lesser priority roads.

1.6 Graph Theory

Since snow route design can be classified into the category of arc routing problems, it would be helpful to review some important terms in graph theory that are relevant to this research.

- Graph: A graph is a collection of nodes and lines joining all or only some of these nodes. Some authors use the term edge for a line with no direction and the term arc for a line with direction. In this research, the term arc will be used for a line connecting two nodes irrespective of the direction of the line connecting two nodes.
- Undirected Graph: If all the arcs in the graph have no direction, the graph is called an undirected graph.
- Directed Graph: If all the arcs in the graph have directions, the graph is called a directed graph.
- Mixed Graph: If both directed and undirected arcs exist in a graph, the graph is called a mixed graph.
- Path: A path is a sequence of consecutive edges in a graph.

- Connected Graph: An undirected graph is connected if there is a path connecting all nodes or vertices.
- Strongly Connected Graph: A directed graph is strongly connected if there is a path connecting all nodes or vertices.
- Degree: The degree of a node is the number of arcs connected at that node. For a directed graph, the degree of a node is the sum of in-degree and out-degree (explained below) of that node.
- In-degree (directed graph): In a directed graph, the in-degree of a node is the number of arcs entering into that node.
- Out-degree (directed graph): In a directed graph, the out-degree of a node is the number of arcs emanating from that node.
- Even-degree node or vertex: A node or vertex for which the degree is an even number.
- Odd-degree node or vertex: A node or vertex for which the degree is an odd number.
- Eulerian or Unicursal Graph: A connected graph is said to be Eulerian or unicursal if there exists a closed tour in the graph containing each arc exactly once and each vertex at least once (Eiselt et al. 1995a). The closed tour is called Eulerian circuit or Eulerian tour.

1.7 Statement of the Problem and Research Objectives

Arc Routing problems play an important role in logistics and distribution management and have been investigated by many researchers. Arc routing problems (ARPs) arise naturally in several applications where streets require maintenance, or customers located along road must be serviced. Efficient route planning can have a significant impact on the overall cost of the service (snow removal, refuse collection, mail delivery, etc.). Designing routes for snow removal involves solving an arc routing problem subject to several constraints such as capacity limit on vehicles, time and/or distance limits on each route, direction of roads, etc. The objective in these kinds of problems is usually to minimize the total distance, time or cost. The Capacitated Arc Routing Problem (CARP), introduced by Golden and Wong (1981), is a class of arc routing problems that is closely related to this problem. The classical Capacitated Arc Routing Problem is defined as: Given an undirected network with non negative demand on the arcs and a depot having a homogeneous fleet of vehicles, the objective is to find a minimal cost set of routes for vehicles, each route starting and ending at

the depot, such that each edge with positive demand is serviced exactly once by one vehicle, and the total demand serviced by each vehicle does not exceed its capacity.

This problem is NP-hard, and it has been shown by Golden and Wong (1981) that even finding a solution whose cost is 1.5 times the optimal cost is NP-hard. In contrast with the extensive literature for the node routing problem, very few local search heuristics are available for the CARP. In the last few years, there has been an increase in research on arc routing problems, evident by a few meta-heuristics developed for the CARP (Lacomme et al. 2004, Beullens et al. 2003).

Recently, many extensions of the basic CARP have been investigated. Examples include the multi-depot CARP (Amberg et al. 2000), and CARP with intermediate facilities for replenishing (Ghiani et al. 2001). Very few of the authors have considered a mixed network. A very recent paper by Lacomme et al. (2004) is the first paper found, that addresses the problem of parallel arcs between nodes. This case may occur when a road segment is too wide to be serviced in one traversal and may need multiple traversals. This research addresses the problem of capacitated arc routing in the context of snow removal. This problem can be stated as:

Given a graph having both directed and undirected arcs, some of which may not be treated but can be used for traveling, the problem is to find a total least distance set of routes starting and ending at the depot and satisfying the vehicle capacity, distance, and time constraints, and also providing the service as many times as needed on the arcs.
The following section describes the restrictions or constraints considered in this problem.

1.8 Restrictions

1.8.1 Distance Limit

Restrictions on distance traveled on each route: In this research, a maximum mileage that each route can cover is considered. This is a practical consideration since a vehicle may need to be refueled at the depot, and maintain a reasonable work load on the driver.

1.8.2 Time Limit

A closely related restriction considered in this research is the time taken to cover each route. This is particularly important in the context of snow removal due to sometimes widely varying speed limits on a road network, traffic conditions, and different speeds while spreading

chemicals and when not spreading chemicals (deadheading). During snow removal, for example, it may be important to clear some important roads within two hours and others within four hours of a snowstorm. Also, having a time restriction on each route will allow a reasonable workload on the driver and the crew.

1.8.3 Capacity Restrictions

Snow removal trucks have a fixed capacity for carrying chemicals. Each time a vehicle services a road or satisfies demand for a road segment, it spreads a specific amount of chemical. Generally, the amount of chemical that is needed to service a road is proportional to the length of the road and is specified in terms of pound of chemicals per lane mile. A higher spreading rate may be employed if the road is a high priority road or in case of a severe snowstorm. When the chemical capacity is exhausted, the vehicle needs to visit the depot to refill with the chemical and begin a new route.

1.8.4 Road Directions and Frequency of Service

Real road networks are a combination of one-way and bi-directional roads. Bi-directional roads could be such that a vehicle traveling in one direction can spread both directions of the road. This is a common situation in snow removal. The bi-directional road could be wide enough such that each direction of the road needs to be traversed at least once to service it. Some roads may require to be serviced multiple times in one or both directions if the roads are too wide to be serviced just once. This may be considered to be similar to a situation where some high priority roads need to be serviced multiple times (even if they can be serviced in one traversal). All of these considerations need to be incorporated in an arc routing problem for snow removal, so as to model real world problems as closely as possible.

1.8.5 Prevention of Sub tours:

It is important to prevent independent tours that do not start and end at the depot node. Each route should begin and end at the depot node. This restriction may be enforced, even though it eliminates some feasible routes that start and end at the depot, but still contain sub-tours that do not include the depot.

1.9 Assumptions

The following assumptions are made in formulating the problem:

- Data on road network such as distance between nodes, and service requirement of each arc are available.
- Vehicle capacities are fixed and same for all vehicles.
- If a vehicle starts servicing or traveling without service on an arc or road segment, it must complete that arc or road segment. In other words, breaking a route in the middle of an arc is not allowed.
- The number of routes is not fixed in advance.
- Priorities for treatment of roads are not considered, but can be incorporated by first running the model on high priority roads, and then repeat with the other roads. However, quality of the results may suffer.
- Speed while servicing the roads may be less than the speed for deadheading (traveling without service).
- Time needed to service an arc depends on the distance and the speed of traveling on that arc (equal to deadheading speed or servicing speed depending on whether the arc is being serviced or not).
- Traffic stoppages due to traffic signals or bad road conditions are not considered.

1.10 Research Objectives

This research attempts to meet the following objectives:

1. Develop an Integer Programming (IP) Model to minimize the total distance traveled, subject to the restrictions described above. Small size problems will be developed and solved using the IP model.
2. Since the problem is NP-hard, the second objective is to develop an efficient meta-heuristic. A software will be developed for this algorithm and used to find good solutions for larger size problems.
3. Compare the results obtained with the meta-heuristic to results provided by other heuristics in the literature on a set of publicly available problem instances.

The next chapter presents a literature review of various types of arc routing problems.

CHAPTER 2

2 Literature Review

2.1 Introduction to Arc Routing Problems

Finding routes for snow removal involves finding paths or cycles that traverse a set of arcs in a graph. Snow route design can be classified into the category of arc routing problems. In arc routing problems (ARPs), the aim is to determine least cost cycles or paths on a specified arc subset of a graph, with or without constraints.

Eiselt et al. (1995a, 1995b) provided an exhaustive survey of different types of arc routing problems. According to Eiselt et al. (1995a), the earliest documented reference to ARPs is the famous Konigsberg bridge problem. In this problem, the objective was to determine whether one could traverse each of the seven bridges on the Pregel river in Konigsberg (Figure 1) only once, and return to the origin point. Swiss mathematician Leonhard Euler (1736) showed that this was impossible and found conditions for the existence of a closed circuit. The problem of determining such a closed circuit was solved by Hierholzer (1873). Fleischner (1990) provided English translation of the original articles of Euler and Hierholzer (Eiselt et al., 1995a)

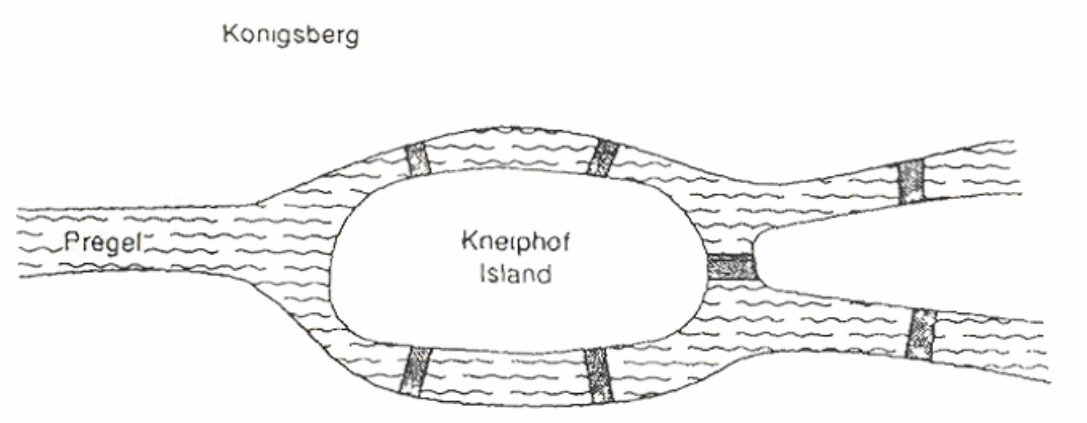

Figure 2.1 The seven bridges of Konigsberg (from Eiselt et al. 1995a)

ARPs can be defined as a special case of the class of general routing problems studied by Orloff (1974) and by Male et al. (1977) (Eiselt et al., 1995a).

Let G= (X, E) be a connected graph, where X= $\{x_1,x_2,\ldots, x_n\}$ is the node set and E=$\{(x_i,x_j):x_i, x_j$ belong to X, $i \neq j\}$ is the arc set. With every arc (x_i, x_j) is associated a

nonnegative cost d_{ij}. d_{ij} can be assumed to be ∞ if the are (x_i, x_j) does not exist. This could also be the case when (x_i, x_j) is a one-way arc. Depending on whether E is a set of directed arcs or edges (undirected arcs), the associated ARP is referred to as directed or undirected ARP. Graphs that include both directed arcs and undirected arcs are called mixed graphs (Eiselt et al., 1995a).

ARP can be classified into many categories depending on various characteristics of the problem such as whether all arcs in the graph, or only a subset of the arcs, need to be traversed, whether the graph is directed, undirected, or mixed, whether there are capacity restrictions or not, etc. Broadly, ARPs can be classified into two categories: the Chinese Postman Problem (CPP) and the Rural Postman Problem (RPP) depending on whether all the arcs in the graph, or only a subset of the arcs, need to be traversed. As described below, these two categories of problems can be further categorized into many sub-categories depending on various characteristics.

1. The Chinese Postman Problem (CPP)

The Chinese Postman Problem is named after the Chinese mathematician, Mei-Ko Kwan (1962) who defined this problem. The CPP is concerned with finding a minimum cost covering tour that visits every arc in a given network (or graph) at least once. Some of the real world applications of this problem are routing of police cars, robot exploration, design of Very Large-Scale Integration (VLSI) circuits, and automated guided vehicles. The classical Chinese Postman Problem assumes that all arcs in the network are undirected. The problem involving graphs in which all arcs are directed is called the Directed Chinese Postman Problem (DCPP); and the one in which some of the arcs are directed is called Mixed Chinese Postman Problem (MCPP). The Capacitated Chinese Postman Problem (CCPP) arises when each arc has a positive demand associated with it and the vehicles that cover them have a finite capacity. Due to vehicle capacity restrictions, the vehicle needs to return back to the depot to either replenish its capacity or end the route. Windy Postman Problem (WPP) is a problem similar to the mixed Chinese Postman problem (MCPP). In the WPP, the cost of traversing an edge of an undirected graph depends on the direction of travel. Hierarchical CPP (HCPP) is an extension of the CPP in which there is a precedence

relationship between arcs in the network, and the order of servicing the arcs must follow this relationship (Eiselt et al., 1995a).

2. The Rural Postman Problem (RPP)

The Rural Postman Problem (RPP) is the problem of finding a minimum cost tour that must visit a subset of arcs in a given network (or graph) at least once. The arcs in the graph that do not need to be covered can be used for traveling. Few practical problems require the service of all arcs in the network. Therefore, most practical arc routing applications are of the RPP type (Eiselt et al., 1995b). Some examples of real world problems that can be modeled as the RPP are routing for urban waste collection, snow removal problem, electric meter reading, etc. The RPP involving graphs in which all arcs are directed is called Directed Rural Postman Problem (DRPP), the one in which all arcs are undirected is called Undirected Rural Postman Problem (URPP), and the one in which only some of the arcs need to be traversed at least once in a given direction but can be traversed as often as needed in the reverse direction is called Stacker Crane Problem (SCP). The Capacitated Arc Routing Problem (CARP), introduced by Golden and Wong (1981), is closely related to the RPP. In this problem, each arc in the network has a non-negative demand and all arcs with positive demand must be covered by a fleet of vehicles subject to vehicle capacity constraints. The classical CARP considers all arcs to be undirected, with at most one edge between two nodes and vehicle capacities to be homogenous. Various extensions of the classical CARP have been studied over the past few years (Lacomme et al. 2004, Amberg et al. 2000). The CARP is probably the most common problem in arc routing, since most real world problems have capacity restrictions (Eiselt et al. 1995b). The chemical and/or abrasive spreading problem for snow removal can be most naturally seen as a CARP. In case of the snow removal problem, demands on some arcs of the graph could be zero. This is a common situation in road salting applications. For example, a county authority is responsible for only county roads but could traverse through roads in city jurisdiction and state highways. The demand on each arc can be calculated as the product of distance of the road segment and the chemical and/or abrasive spreading rate (typically 100-300 lb/lane mile). Due to the capacity restriction, one route may not be able to service all the arcs, and trucks would need to return to depot or storage facility to refill with the chemical and/or abrasive.

There is an extensive body of literature addressing the topic of arc routing. Dror (2000) provided a detailed review of arc routing methods, solution approaches and applications. The purpose of the next section is to review that portion of the literature, which is most relevant to the problem considered in this research.

2.2 Literature Review

In this section, a summary of the literature for RPP, and CARP is presented and is followed by a discussion of literature on the snow removal operation. Since this research is concerned mainly with a problem closely related to CARP, CARP and snow removal literature will be covered in more detail.

2.2.1 Rural Postman Problem

The Rural Postman Problem was introduced by Orloff (1974). Both the directed and undirected cases of RPP are NP-hard (Lenstra and Rinooy Kan, 1976). However, when the subset of arcs to be serviced or traversed is equal to all the arcs in the graph, the problem becomes the simple CPP and can be solved in polynomial time (Eiselt et al., 1995b). Fredrickson et al. (1978) proved that the Stacker Crane Problem (SCP) is NP-hard. The Capacitated Arc Routing Problem (CARP) was introduced by Golden and Wong (1981) and proven to be NP-hard. The RPP is a special case of CARP where the arcs to be serviced have a unit demand, the arcs that need not be serviced have a zero demand, and the vehicle capacity is equal to the total number of required arcs (arcs to be serviced) (Eiselt et al., 1995b). Most practical problems belong to the category of RPP because usually not all arcs or street segments in a network require service.

A standard algorithmic approach for the RPP is to first determine a least cost augmentation of the graph to make it unicursal, and then obtain a Eulerian circuit on the augmented graph. Heuristics are used for the augmentation phase since RPPs are NP-hard. These heuristics often use variations of matching algorithms or shortest spanning tree algorithms to create an augmented graph satisfying the unicursality condition. Exact algorithms for the generation of the unicursal graphs use techniques commonly used for the traveling salesman problem (TSP) (Eiselt et al., 1995b).

Most applications of RPP involve multiple vehicles, single or multiple depots, and a number of limits due to vehicle capacity, distance covered, and time taken. A common

strategy for solving RPPs by heuristic means is to first decompose the problem into smaller problems by breaking the original network into smaller graphs (For example, see Chapleau et al. 1985, and Levy and Bodin 1989) (Eiselt et al., 1995b). Often, to model real world problems as closely as possible, traffic restrictions such as restrictions on U-turn, left-turns, and right -turns have been considered (For example, see Bodin and Kursh 1979, and McBride 1982). The following sections detail the literature found in the areas of undirected RPP, directed RPP, CARP, and specific literature in the area of arc routing for snow removal.

2.2.1.1 Undirected Rural Postman Problem (URPP)

In the URPP, all arcs are undirected, i.e. all the arcs can be traveled in either direction. When the subset of the graph that needs to be serviced is connected, the problem can be solved by computing shortest chains (in the complete graph) between odd-degree vertices and then applying the methods used for the Undirected Chinese Postman Problem (UCPP) (Eiselt et al. 1995b).

Frederickson (1979) suggested a method of modifying the graph and solving the modified graph using a heuristic based on Christofides (1976) heuristic for symmetrical TSP. Christofides (1981), Sanchis (1990) and Corberan and Sanchis (1994) proposed integer programming formulations for the RPP (Eiselt et al. 1995b). Hertz, Laporte, and Nanchen-Hugo (1999) discussed post optimization heuristics for the undirected RPP.

2.2.1.2 Directed Rural Postman Problem (DRPP)

In a DRPP, all the required arcs in the graph are directed. The directed CPP (DCPP) is a special case of the DRPP when the subset of the graph that needs to be serviced is connected (Eiselt et al. 1995b).

Christofides et al. (1986) proposed a heuristic for the DRPP, which involves constructing a shortest spanning arborescence at an arbitrary vertex (refer to Edmonds 1967) and solving a transportation problem to find a Eulerian graph from the modified graph. Finally, a Eulerian circuit is determined on the augmented graph. A mathematical programming formulation and an exact algorithm were also presented by Christofides et al. (1986).

2.2.2 Capacitated Arc Routing Problem (CARP)

In the CARP, each arc has a non-negative demand that needs to be satisfied by a fleet of vehicles having a fixed capacity. In the CARP, one needs to determine the least cost traversal of all arcs having a positive demand so that the total demand of all arcs serviced by a vehicle does not exceed its capacity. Some authors have solved the CARP by first transforming it to the more common node routing problem (refer to Pearn et al., 1987).

Golden and Wong (1981) proved that finding a solution for the CARP for which the cost is guaranteed to be less than 1.5 times the optimal value is NP-hard. Golden and Wong (1981) proposed an integer programming formulation for the undirected CARP. Belenguer and Benevant (1991) also developed an integer programming formulation for the undirected CARP.

Lower bounds for the undirected CARP were developed by Golden and Wong (1981), Assad et al. (1987), Win (1987), Pearn (1988) and Benavent et al. (1992). More recently, tighter bounds were developed by Belenguer and Benavent (1998) by mathematical formulation and linear relaxation of the CARP and generating valid inequalities.
Heuristic algorithms for the CARP can be broadly classified into three categories (Eiselt et al. 1995b)

1. Simple construction methods,
2. two-phase construction methods, and
3. improvement methods.

1. Simple construction methods

Most simple construction methods for the CARP were developed in the context of undirected Capacitated Chinese Postman Problem (CCPP), but can be easily adapted for the general CARPs (Hertz and Mittaz, 2001). The Capacitated Chinese Postman Problem (CCPP) is a special case of the CARP where all the arcs in the graph have a positive and non-zero demand.

Christofides (1973) developed the Construct-Strike Algorithm for the solution of the undirected CCPP. This algorithm gradually constructs feasible tours and removes them from the graph. The removal of the tours is such that it does not separate the graph into disconnected components. When a cycle is constructed, all required arcs on the tour are removed from the original graph. The Construct-Strike procedure is repeated until no more

feasible vehicle routes can be determined. Pearn (1989) extended this idea and modified the algorithm by eliminating the restriction that that the graph obtained after removing a tour be connected.

Golden et al. (1983) proposed a heuristic algorithm, called the Path Scanning Algorithm in which feasible cycles are constructed one at a time using a set of five rules for arc selection. The procedure is repeated using each of the five rules and the best solution is selected. Pearn (1989) proposed using one of the five rules at random every time an edge is selected. The problem is solved several times using different random numbers and the best solution is selected.

Golden et al. (1983) also proposed a heuristic algorithm called the Augment-Merge Algorithm, which is based on Clarke and Wright (1964) algorithm for the node routing problem. In this algorithm, incomplete cycles are created, each containing a different arc. Then, the incomplete cycles are combined based on a savings criterion.

Chapleau et al. (1984) proposed an algorithm called the Parallel-Insert Algorithm in which several routes are constructed in parallel using the same method as in the Path Scanning Algorithm. The objective is to obtain a balance between costs of different routes. This algorithm was applied to school bus routing.

Pearn (1991) developed an algorithm based on the steps used in the Augment-Merge and Parallel-Insert algorithms. In the first step, cycles connected to the depot are gradually constructed with all arcs in the network that can be included in the cycles while maintaining feasibility. For constructing the feasible cycles, criteria based on cost and demand are suggested. In the second step, the arcs remaining are merged in the existing cycles (created in first step) using a savings criterion (Eiselt et al. 1995b).

2. Two-Phase Construction Algorithms

Two-Phase Construction algorithms for the CARP can be categorized as (Eiselt et al. 1995b):

a) Cluster First, Route Second Heuristics

Heuristics in this class cluster all arcs in the graph first and then construct routes over each cluster while satisfying the capacity constraint. Win (1987) proposed the application of a greedy criterion and Benavent et al. (1990) proposed a generalized

assignment algorithm for partitioning the arcs into clusters. By using a simple modification of a CPP algorithm, vehicle routes can be determined.

Route First, Cluster Second Heuristics

According to Win (1987), these heuristics involve first constructing a giant Euler tour covering all edges having positive demands. If these edges create a connected graph, then a simple modification of a CPP algorithm can be applied to obtain the tour in polynomial time. Otherwise, Frederickson's ½-approximation algorithm (Frederickson 1979) can be applied for the resulting RPP. A ½-approximate algorithm is one that gives results that are at most 1.5 times the optimal value. As a next step, Win (1987) suggested the use of a heuristic, which he termed fit bin packing heuristic, for partitioning the giant tour into feasible clusters. Ulusoy (1985) described an algorithm for the clustering phase, which involves transforming the original graph into another one on which a shortest path problem is solved. Hertz et al. (2000) presented an algorithm called CUT for partitioning of a route into smaller and feasible routes.

2. Improvement Methods

Improvement methods can be applied for enhancing the solutions obtained by construction algorithms. The following section presents the meta-heuristics that have been applied for the CARP.

2.2.2.1 Meta-heuristics for the CARP

Many recent meta-heuristics have been proven to be highly effective for the solution of the CARP. Meta-heuristics such as simulated annealing and Tabu search have been applied quite successfully to a variety of practical problems. Li (1992) applied simulated annealing and Tabu search procedures to a road gritting problem. Eglese (1994) proposed a simulated annealing algorithm for the CARP, which can also be used for problems with multiple depot locations and many side constraints. The papers by Li and Eglese (1996), and Eglese (1994) will be discussed in more detail in the literature pertaining to routing for snow removal.

A Tabu search based algorithm called CARPET was presented by Hertz et al. (2000) for solving the undirected CARP. This algorithm uses a neighborhood structure similar to that

of the TABUROUTE procedure used for the node routing problem (Gendreau et al., 1994). A number of procedures used in the search process are described: SHORTEN, DROP, ADD, PASTE, CUT, SWITCH, and POSTOPT. An initial solution is constructed by solving the RPP using Frederickson's algorithm (Frederickson, 1979). An objective function was defined based on total length of solution and a penalty function that depends on the total demand exceeding the vehicle capacity. Computational results show that CARPET outperformed all known heuristics (up to the time of the paper) on benchmark problems and often produced an optimal solution.

Hertz and Mittaz (2001) considered the undirected CARP (UCARP) and proposed an adaptation of the variable neighborhood descent (VND) algorithm for its solution. The initial solution was constructed using Frederickson's Algorithm (Frederickson, 1979) by relaxing the capacity constraints. Several procedures, that were used in the CARPET meta-heuristic such as SHORTEN, CUT, SWITCH, ADD and DROP, as described in Hertz (2000), were used in this paper. Extensive computational experience was reported using three of the datasets found in the literature, and showed that for larger instances, an adaptation of the variable neighborhood descent performed better than CARPET both in terms of solution quality and time.

Belenguer and Benavent (1998) studied the polyhedron associated with the CARP and generated valid inequalities to develop a linear programming based cutting plane algorithm for the problem. The lower bounds produced by this procedure outperformed all previously known lower bounds for a set of 34 available instances in the literature. Belenguer and Benavent (2003) further extended this research, proposed new valid inequalities, and developed a cutting plane algorithm method based on the new inequalities. Computational results on three data sets revealed that this procedure produced lower bounds better than any other procedure in the literature; and for 47 out of 87 instances, the lower bounds proved that the previously known heuristic solutions were optimal.

Amberg et al. (2000) considered a capacitated arc routing problem with multiple depot locations (M-CARP). A route first cluster second approach was applied in which routes are first created by transforming the M-CARP into a multiple center capacitated minimum spanning tree (CMST). An algorithm was developed to create intial solutions on the modified CMST and simulated annealing was used to improve the initial solutions. The algorithm developed was applied to real world problems in the areas of Konigstein and

Wennigsen. The authors concluded that heuristics can produce good solutions, and suggested ways to improve the running time of the algorithm.

A Tabu scatter search procedure to solve the CCPP was described by Greistorfer (2003). The Capacitated Chinese Postman Problem (CCPP) is a special case of the CARP. For constructing the initial routes, the graph is made Eulerian by using a matching algorithm (Lawler. 1976). Initial routes are combined into bigger ones using a savings based approach. A hybrid Tabu scatter search algorithm based on Tabu search and population based strategies was developed and the computational results showed that the algorithm was competitive with benchmark instances in the literature.

Mourao and Amado (2005) discussed a heuristic method for a mixed capacitated arc routing problem (MCARP) for the refuse collection in Lisbon. An algorithm, which they termed as service direction algorithm was used to transform the mixed network into a directed one. A lower bound was produced and a heuristic method based on Eularian and directed graph was applied. Comparison with heuristics developed and extended by Lacomme et al. (2002) showed a good performance of the developed heuristic.

A local search technique, using methods derived from a node routing contexts, such as moving arcs between routes and within the same route, was embedded into a meta-heuristic called guided local search (GLS), by Beullens et al. (2003) to solve the capacitated arc routing problem. The mechanisms of neighbor lists and edge marking were used in order to check some moves in the local search procedure rather than checking complete neighborhoods. Experiments on benchmark problems showed that this heuristic finds all the known upper bounds and improves the bounds for a few instances.

An efficient hybrid genetic algorithm (HGA) was used to solve the basic undirected CARP by Lacomme et al. (2001a). It outperformed the CARPET meta-heuristic developed by Hertz (2000). In continued research, Lacomme et al. (2001b) developed a cutting plane algorithm to generate tight lower bounds or an optimum value for an extended CARP (ECARP) with a mixed network, turn restrictions, and different costs for servicing and traveling without servicing an arc. Further, a bi-objective function considering the total cost of trips and total overload was defined and an enhancement of the HGA was applied. The results produced were comparable to those of the cutting plane algorithm.

A powerful memetic algorithm (MA) for solving the extended CARP as described above was presented by Lacomme et al. (2004). Memetic algorithms are heuristic-based

search procedures that are closely related to genetic algorithms. Compared to an earlier paper by Lacomme et al. (2001a), this paper addressed other objectives such as makespan and number of vehicles used, and covered new extensions such as mixed graphs, two distinct costs per link, parallel arcs between two nodes, limit on trip length, limited fleet, and turn penalties. Three classical heuristics (Path Scanning by Golden et al (1983), Augment-Merge by Golden and Wong (1981) and Ulusoy's heuristic by Ulusoy (1985)) were enhanced to handle the ECARP and provide initial solutions. Computational evaluation was performed on three classical instances in the literature (DeArmon (1981), Belenguer and Benevant (2003), and Eglese instances built by Belenguer and Benevant (2003)) and the best memetic algorithm for the CARP was found to outperform all known heuristics.

A mixed capacitated arc routing problem (MCARP) with different costs for treatment and deadheading was considered by Belenguer et al. (2006). According to the authors, no published paper addressed this kind of problem in which a mixed graph (directed and undirected arcs) and a subset of arcs needing service are considered. Three heuristics (Path Scanning by Golden et al. (1983), Augment-Merge by Golden and Wong (1981) and Ulusoy's heuristic by Ulusoy (1985)) were extended for the MCARP with additional features such as a mixed network, parallel links, different costs for servicing and deadheading, prohibited turns and turn penalties, trip cost limit, dumping sites, and windy edges. A lower bound for the MCARP was developed using a cutting plane algorithm without the extensions mentioned. A memetic algorithm was developed and the results on various instances were compared to the lower bound generated and were promising.

Table 2-1 presents a summary of recent meta-heuristics applied to the CARP.

Table 2-1 Summary of Meta-heuristics for the CARP

Author	Problem and solution method
Belenguer et al. (2006)	Problem: A mixed capacitated arc routing problem (MCARP) with extensions. A lower bound was developed using a cutting plane algorithm. Three construction heuristics for constructing initial solutions and a memetic algorithm for local search were applied.
Mourao and Amado (2005)	Problem: Mixed capacitated arc routing problem (MCARP) for refuse collection in Lisbon. A service direction algorithm was used to transform the mixed network into a directed one and a heuristic method based on Eularian and directed graph was applied. A lower bound was also produced.

Author	Problem and solution method
Lacomme et al. (2004)	Problem: CARP with extensions. Three construction heuristics were extended to create initial solutions, and a memetic algorithm was applied. Computational evaluation was performed on three classical instances in the literature.
Belenguer and Benavent (2003)	Problem: Basic CARP. New valid inequalities were developed and a cutting plane algorithm method based on the new inequalities was proposed. This procedure obtained best known lower bounds for three datasets in literature. For 47 out of 87 instances, the lower bounds proved that the previously known heuristic solutions were optimal.
Beullens et al. (2003)	Problem: Basic CARP. Moves based on methods derived from a node routing context were embedded in a guided local search (GLS) meta-heuristic.
Greistorfer (2003)	Problem: A special case of CARP- Capacitated Chinese Postman Problem. The graph is made Eulerian by using a matching algorithm and initial routes are constructed (Lawler, 1976). Initial routes are combined into bigger ones using a savings based approach. A hybrid Tabu scatter search algorithm with population based strategies was developed.
Hertz and Mittaz (2001)	Problem: Basic CARP. An adaptation of the variable neighborhood descent (VND) algorithm was applied. The initial solution was constructed using Frederickson's Algorithm (Frederickson, 1979) by relaxing the capacity constraints. Several procedures that were used in the CARPET meta-heuristic as described in Hertz (2000) were used in this paper.
Amberg et al. (2000)	Problem: CARP with multiple depot locations (M-CARP). A route first cluster second approach was applied in which routes are first created by transforming the M-CARP into a multiple center capacitated minimum spanning tree (CMST). An algorithm was developed to create intial solutions on the modified CMST and simulated annealing was used to improve the initial solutions. The algorithm developed was applied to real world problems.
Hertz et al. (2000)	Problem: Basic CARP. A Tabu search based algorithm called CARPET was applied. This algorithm uses a neighborhood structure similar to that of the TABUROUTE procedure used for the node routing problem (Gendreau et al., 1994). A number of procedures were used in the search process. An objective function was defined based on total length of solution and a penalty function that depends on the total demand exceeding the vehicle capacity.
Belenguer and Benavent (1998)	Problem: Basic CARP. Valid inequalities were generated to develop a linear programming based cutting plane algorithm for the problem. The lower bounds produced by this procedure outperformed all previously known lower bounds for a set of 34 available instances in the literature.
Other meta-heuristics	Li (1992) applied simulated annealing and Tabu search procedures to a road gritting problem. Eglese (1994) proposed a simulated annealing algorithm. Li and Eglese (1996) described a time constraint two-phase heuristic algorithm and tested it on three administrative regions in U.K.

The following section presents the literature pertinent to the problem of snow route generation.

2.2.3 Arc Routing for Snow Removal

Campbell and Langevin (2000) provided an excellent review of arc routing for roadway snow and ice control and presented literature with recent application of two snow route design software.

Scientific and analytical research work on roadway snow and ice control began to attract significant attention in the 1960s. By 1979, many authors had begun to address the arc routing problems for snow removal (Campbell and Langevin 2000).

A systems study of snow removal by Minsk (1979) and a value engineering study by Russel and Sorenson (1979) were conducted. A simulation model for urban snow removal was developed by Tucker and Clohan (1979) which allows decision makers to vary parameters of the snow removal process related to truck and snow storm characteristics and observe the results of the model before implementing them. Their model applies to the problem of snow plowing, but they stated that it can be easily applied to the problem of spreading chemicals and/or abrasives. This model was validated using truck routes and storm data from Newington, Connecticut. Their guidelines for routing included minimizing U-turns at intersections, left turns, and deadheading. Five storms were simulated and a sensitivity analysis was performed to analyze the role of snowfall rate and initial depth of snow at the start of plowing.

According to Marks and Stricker (1971), parked cars, cars stuck in the middle of a road, and priorities in plowing are critical factors in modeling snow removal. In their analysis, they revealed the shortcomings of the CPP model with respect to real world constraints. A few ways were described to consider priorities for plowing roads. They developed a decomposition heuristic to obtain a solution for the CPP. Finally, they presented an application for trash collection in Cambridge, Massachusetts.

A graph theory formulation was used by Liebling (1970, 1973) to divide the city of Zurich, Switzerland into sectors for snow removal, and a CPP was solved in each sector. This study was presented to assist municipal authorities in Zurich in the choice of equipment for street cleaning, determination of sites for depot location and planning of routes for snow removal.

A dynamic routing heuristic for the problem of routing of salt spreader trucks was developed by Cook and Alprin (1976). The objective considered was the minimization of the time required to spread salt over a network of streets. Street segments were defined in such a way that they could be serviced with exactly one truck load of salt. After each truck is refilled with salt at the salt storage facility, it is assigned to the nearest untreated street segment. A simulation model for spreading salt was developed for the city of Tulsa, Oklohoma, and was demonstrated to the city officials who felt that it would be useful to apply the developed heuristic in the following snow season.

An algorithm that works by tracing a Eulerian circuit on a directed graph and that considers street priorities by selecting higher priority roads whenever possible was developed by Lemieux and Campagna (1984). It was assumed that street plowing must be done on both sides, once in each direction. The authors developed an interactive computer program and tested it on a small network.

Routing of snow blowers for loading snow into trucks was considered by Gilbert (1989), who modeled the problem as a node routing problem, where the nodes correspond to the street segments to be cleared. The author considered a fixed depot and a number of workdays to complete the snow removal operation. A large-scale non-linear programming model with several precedence constraints and restrictions on snow loading operation in the city was presented. An algorithm using an insertion method that adds nodes iteratively to a given work shift was developed. In the first step of the insertion method, higher priority nodes are added in the first shift of the workday. As the last days of work for completing the snow removal operation approach, the algorithm adds nodes that balance the work schedule. The heuristic was tested using data from one district in Montreal, Canada, and produced good results.

A solution approach for the single vehicle arc routing problem for snow removal that involves constructing Eulerian subgraphs, determining lower and upper bounds by dynamic programming, and solving a traveling salesman problem by branch and bound was developed by Gelinas (1992). Precedence constraints on the road network were also considered. The developed heuristic was tested using data from one district in Montreal, Canada.

A decision support system for snow and ice control called the SnowMaster system was developed by Evans (1990) and Evans and Weant (1990). The system was designed to aid

local agencies to route snow and ice control vehicles for improved service, higher equipment utilization, and lower capital and operating costs. The route design component of the system includes five different arc selection rules. The results were generated for each of the five rules and the user can select the best route. Capital costs of equipment were considered in order to aid in the identification of the appropriate mix of equipment required. The authors presented an application of the SnowMaster system for Butler County, Ohio. Simulation models for Butler county, OH showed that savings of up to $250,000 could be achieved over three years by reduction in equipment purchase costs due to improved routes generated by the SnowMaster system.

Recent decision support systems combine arc routing heuristic algorithms, user-friendly interfaces, interactive route design capability, and ability to work with digital maps. In a joint project involving researchers at Purdue University and personnel of the Indiana Department of transportation (INDOT), a computerized system called Computer Aided System for Planning Efficient Routes (CASPER) was developed for rural snow and ice control. CASPER has evolved over many years incorporating a number of improvements, and is currently used to design service routes for snow removal in Indiana. Several papers were written during various phases of the development of CASPER. Reference may be made to Haslam and Wright (1991), Wang (1992), Wright (1993), Wang and Wright (1994), Wang et al. (1995), and Goode and Nantung (1995) for more details on CASPER. The objective in CASPER is to minimize cost by reducing the number of routes and reducing the number of deadhead miles traveled, while satisfying specified service levels and maintaining class continuity (priorities). CASPER combines spatial network data, multiobjective heuristic optimization procedures, and a user controlled interactive graphical interface. In CASPER, first a route generation heuristic is used that is guided by a penalty function that incorporates conflicting objectives of satisfying time limits for routes, minimizing deadhead travel, and maintaining class continuity. The next step involves a local improvement heuristic based on Tabu search. In this procedure, one or two arcs are swapped and penalty values are used to evaluate the improvement as a result of the move if the move results in infeasible routes. The route improvement heuristic is a modification of the algorithm presented by Haslam and Wright (1991). Reference is made to Wang (1992) for discussion of the application of Tabu search to the problem of snow route design. INDOT estimated savings of $2.2 million

in the first year and a total of $4.8 million over ten years with the use of CASPER. It was observed that the depot location plays an important role in creating good service routes.

An adaptation of the GENIUS algorithm developed earlier for the TSP by Gendreau et al. (1992) was used for construction of routes in GeoRoute software for route optimization (Campbell and Langevin, 2000). Giro Enterprises Inc. of Montreal, Canada developed GeoRoute software for snow and ice control, which has been implemented in several regions by PSR Group Ltd., a consulting company based in Canada. GeoRoute is an interactive decision support system consisting of four modules: network manager, route manager, site editor, and map generator. The optimization module in the software is the Route manager, which allows for each type of operation: spreading, plowing, and snow blowing. GeoRoute gives the user the option to evaluate a variety of different snow removal environments, such as turn restrictions, urban and rural conditions, repetitive treatment, number of passes, time windows, etc. GeoRoute has been used extensively in several locations, including the cities of Laval, Charlesbourg and Ottawa in Canada, and in the United Kingdom. For the city of Ottawa, Canada, testing showed that optimized routes generated by GeoRoute increased productivity by 60-90 percent.

Eglese (1994) described rural snow and ice control research, based on previous research carried out by Eglese (1988), in which the objective was to find the most cost efficient way of carrying out winter gritting within some practical constraints. An important element of the study was design of a heuristic algorithm to route gritters. Extensions to the winter gritting problem such as road treatment time constraints, multiple depot locations, and limited vehicle capacities were considered. It was assumed that trucks can spread both directions of the road in one pass. Eglese proposed decomposing the even connected graph (the underlying road network) into cycles according to a checkerboard pattern. In this pattern, a cycle has common boundary with adjacent cycles, but it does not overlap with another cycle. A graph is then formed in such a way that a vertex in the graph corresponds to a cycle obtained by the previous decomposition. An edge is included between two vertices in the graph if the corresponding cycles have a common node in the original graph. Each tree containing the depot node in this graph corresponds to a vehicle tour in the original graph. The construction algorithm was based on a procedure described by Male and Liebman (1978) and involved using a heuristic similar to the Clarke-Wright (1964) savings heuristic on a graph derived from solving the CPP on the network. A simulated annealing

based heuristic was used to improve the results obtained from the construction heuristic. The objective considered was to minimize the number of routes, a penalty due to the additional distance over the maximum allowed, and a penalty due to additional time taken over the maximum gritting time allowed. It was found that simulated annealing improved the initial solutions. The number of routes generated by the heuristic was equal to or less than the current number of routes. This algorithm was tested in three areas having 111, 303, and 280 roads respectively. The most significant result of the study was that the number of required depots could be reduced by more than 50% without increasing the number of gritters.

A time constraint two-phase heuristic algorithm was described by Li and Eglese (1996) in which the farthest untreated arc is chosen to begin a route. Phase one of the algorithm extends the route back from the near end of the selected arc to the depot, and phase two extends the route from the far end of the selected arc to the depot. As each arc is added, capacity and time constraints are checked to maintain feasibility. The user can either run the algorithm automatically or interact with the system in selecting the next road to be inserted into the route. The authors considered multiple salt storage facilities for reloading trucks with salt without going back to the depot, and mixed networks (directed and undirected arcs). They found that this algorithm produced better results than the one described in Eglese (1994). The authors tested this algorithm using data from three administrative regions in Lancashire, U.K. Results showed that allowing user intervention produced fewer routes and reduced the total distance when compared to the automatic version.

A combined location and routing problem dealing with location of main depots as well as supplementary salt storage depots (silos) was considered by Lotan et al. (1996) for the problem of winter gritting in the province of Antwerp, Belgium. The Province has 1074 kilometers of road, 747 roads, and 257 nodes. Equal capacity trucks and priorities of roads were considered. In stage 1 of their approach, the authors consider the problem of partitioning the region into districts and determining the location of main depots such that there exists a feasible route with minimum deadheading distance. In stage 2, they locate supplementary depots and solve the associated routing problem. Some findings include the advantages of locating silos close to borders between districts and the differences between spreading both directions of a road in one pass and making two passes to spread in both directions.

Many real world road characteristics that cause problems in snow removal such as right and left turns, street crossings, U-turns, and street changes were modeled by Gendreau et al. (1997). Vertices were replicated for each one of the mentioned characteristics and artificial arcs were introduced. The authors were asked by LogiRoute company to conduct numerical experiments in order to determine the appropriate penalties for: deadheading, left turns, U-turns, street crossings, and street changes. Experiments were performed on 30-street networks in three suburbs of Montreal, Canada. Refuse collection and snow plowing problems were considered in order to determine the penalties.

A decision support system for aiding the Maryland State Highway Administration Office of Maintenance staff in developing snow emergency routes for Calvert County, Maryland was developed by Haghani and Qiao (2001). A mathematical programming formulation that takes into account time windows for network hierarchy was presented and a combination of currently existing heuristic procedures with slight adaptations was proposed to solve the arc routing problem for snow removal. Calvert County was divided into four sections for salting. The application of this algorithm resulted in savings in terms of reduced deadhead miles and reduced number of trucks used. It was projected that the algorithm would save between 10 and 38 % of total route length and between 15 and 54 % of deadhead mileage.

In an extension to their earlier research, Haghani and Qiao (2002) considered more realistic constraints in order to model the snow removal problem. They presented two problems, one to minimize the total number of trucks, and the other to minimize the total deadhead distance given a fixed number of trucks. Mathematical programming formulations for these two problems were presented. It was assumed that all salting roads are bidirectional. For solving these problems, a Capacitated Minimum Spanning Tree (CMST) was constructed. Based on test results conducted for Calvert county, Maryland, it was found that the number of trucks currently used could be reduced by two (or 15 %) and the total deadhead distance could be reduced by 4 %.

CHAPTER 3

3 Exact Algorithm

3.1 Introduction

The CARP is NP-Hard (Golden and Wong, 1981) and only very small problems could be solved optimally. For larger problems, the computation time is great and it is impractical to apply exact algorithms to them. Even though only small problems can be solved by exact methods, developing mathematical models provides an insight into the complexity involved in solving the problem and provide better understanding of its details.

3.2 Mathematical Model

In this section, a mathematical formulation, based on the model originally provided by Golden and Wong (1981) and later modified by Haghani and Qiao (2001), is presented. In their model, Haghani and Qiao considered time windows for servicing of arcs and variable service requirements of arcs (number of times an arc needs to be serviced). The indexes, parameters and decision variable definitions for the mathematical formulation are given below:

Table 3-1 Mathematical Model Nomenclature

Type of Variable	Notation	Definition
Parameters	A	Set of arcs in the network
	i, j	Node index
	p	Vehicle index
	k	Maximum number of vehicles available for servicing the network
	n	Number of nodes in the network
	W	Vehicle capacity
	N_{ij}	Number of times arc (i,j) needs to be serviced in the direction i to j
	Q_{ij}	Demand of arc (i,j) in the direction i to j
	T_{ij}	Time required to travel without servicing arc (i,j) in the direction i to j (deadheading time)

	G_{ij}	Difference between servicing time and deadheading time for an arc (*i,j*) in the direction *i* to *j*
	D	Maximum distance a vehicle can cover in a route
	T	Maximum time a vehicle can take to cover a route
	F_{ij}^{p}	Flow variable for arc (*i,j*) and vehicle *p* in the direction *i* to *j*. This variable is used in the formulation to ensure that no subtour is allowed
	C_{ij}	Distance between nodes *i* and *j*
Decision Variables	X_{ij}^{p}	X_{ij}^{p} =1 if an arc (*i,j*) is used by vehicle p for either servicing or traveling without servicing (deadheading) in the direction *i* to *j*
	L_{ij}^{p}	L_{ij}^{p} =1 if an arc (*i,j*) is serviced by vehicle p in the direction *i* to *j*

This formulation assumes that each vehicle can service each arc at most one time. A required arc is an arc that needs to be serviced, i.e. it has a positive demand.

3.2.1 Objective Function

$$Minimize \sum_{p=1}^{k} \sum_{i,j \in A} C_{ij}.X_{ij}^{p} \qquad (1)$$

The objective is to minimize the total distance traveled by all the available vehicles to service all the required arcs in the network. This includes arcs that are required to be serviced and arcs that are not required to be serviced but can be used for traveling to reach a required arc or the depot.

3.2.2 Constraints

- Route continuity constraint:

To make sure that a vehicle entering a node exits that node.

$$\sum_{\substack{r=1 \\ r \neq i}}^{n} X_{ri}^{p} - \sum_{\substack{r=1 \\ r \neq i}}^{n} X_{ir}^{p} = 0 \quad \begin{matrix} \forall i = 1..n \\ p = 1..k \end{matrix} \qquad (2)$$

- Vehicle capacity constraint:

This constraint ensures that in a route, the total amount of chemicals required during servicing of arcs does not exceed the vehicle capacity.

$$\sum_{i,j \in A} L_{ij}^{p} . Q_{ij} \leq W \quad \forall\, p = 1..k \tag{3}$$

- Service frequency constraint:

This ensures that each arc with a positive demand is serviced the number of times needed.

For arcs that can be serviced by traveling once in either directions (from *i* to *j* or *j* to *i*):

$$\sum_{p=1}^{k} L_{ij}^{p} + L_{ji}^{p} = 1 \quad \forall (i, j) \in A \tag{4}$$

For arcs that have to be serviced a finite number of times (N_{ij}) in one direction (from *i* to *j*):

$$\sum_{p=1}^{k} L_{ij}^{p} = N_{ij} \quad \forall (i, j) \in A \text{ where } N_{ij} >= 1 \tag{5}$$

- Travel time constraint

This constraint limits the travel time on a route to a finite value. It also includes the deadheading (traveling without servicing) and considers that the time for servicing may be higher than the time for traveling.

$$\sum_{i,j \in A} T_{ij} . X_{ij}^{p} + \sum_{i,j \in A} G_{ij} . L_{ij}^{p} \leq T \quad \forall\, p = 1..k \tag{6}$$

- Service/Deadheading constraint

This constraint ensures that an arc can be serviced only if the vehicle travels on that arc.

$$X_{ij}^{p} \geq L_{ij}^{p} \qquad \begin{matrix} \forall (i,j) \in A \\ p = 1..k \end{matrix} \tag{7}$$

- Total distance constraint

This constraint limits the total distance traveled on a route and includes the deadhead distance and the service distance.

$$\sum_{i,j \in A} X_{ij}^{p} . C_{ij} \leq D \;\; \forall \; p = 1..k \tag{8}$$

- Sub-tour eliminating constraint

This constraint is adapted from a formulation of a different problem by Golden and Wong (1981). It prevents the formation of sub-tours in the route. A sub-tour is a loop that does not include the depot node and is not connected to the depot node through other arcs in the network.

The flow variable F_{ij}^{p} can take positive values only if X_{ij}^{p}=1, i.e. if an arc (*i,j*) is used by vehicle *p* for either servicing or traveling without servicing.

$$\sum_{\substack{r=1 \\ r \neq i}}^{n} F_{ir}^{\;\;p} - \sum_{\substack{r=1 \\ r \neq i}}^{n} F_{ri}^{\;\;p} = \sum_{j=1}^{n} L_{ij}^{\;\;p} \qquad \begin{matrix} \forall i = 2..n \\ p = 1..k \end{matrix} \tag{9}$$

$$F_{ij}^{p} \leq n^{2} . X_{ij}^{p} \;\; \forall (i,j) \in A \tag{10}$$

$$F_{ij}^{p} \geq 0 \tag{11}$$

In order to explain the sub-tour breaking constraint, an example is presented below (See Figure 3.1). Node 1 is the depot note and all the arcs can be used for traveling in either direction. One or more arcs in the network may have a positive demand and need to be serviced. One vehicle is used to service the network.

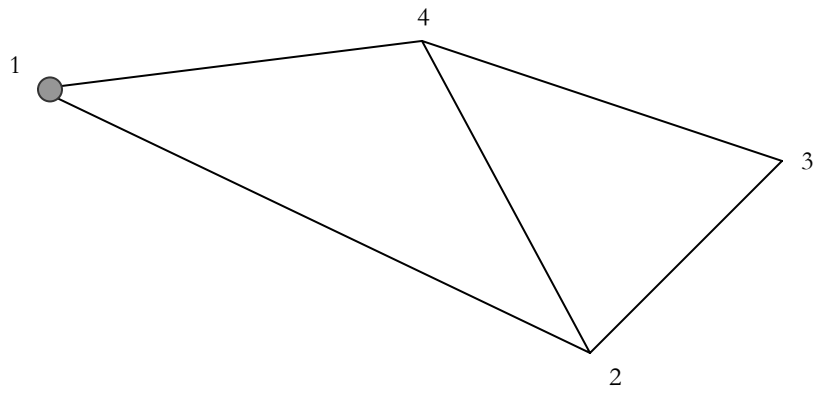

Figure 3.1 Example to Illustrate the Sub-tour Breaking Constraint

Applying the first sub-tour breaking constraint (Equation 9):

$$i = 2, p = 1: \; F_{21}^{1} + F_{23}^{1} + F_{24}^{1} - F_{12}^{1} - F_{32}^{1} - F_{42}^{1} = L_{21}^{1} + L_{23}^{1} + L_{24}^{1} \quad (12)$$

$$i = 3, p = 1: \; F_{31}^{1} + F_{32}^{1} + F_{34}^{1} - F_{13}^{1} - F_{23}^{1} - F_{43}^{1} = L_{31}^{1} + L_{32}^{1} + L_{34}^{1} \quad (13)$$

$$i = 4, p = 1: \; F_{41}^{1} + F_{42}^{1} + F_{43}^{1} - F_{14}^{1} - F_{24}^{1} - F_{34}^{1} = L_{41}^{1} + L_{42}^{1} + L_{43}^{1} \quad (14)$$

Adding the above three equations,

$$F_{21}^{1} - F_{12}^{1} + F_{31}^{1} - F_{13}^{1} + F_{41}^{1} - F_{14}^{1} = L_{21}^{1} + L_{23}^{1} + L_{24}^{1} + L_{31}^{1} + L_{32}^{1} + L_{34}^{1} + L_{41}^{1} + L_{42}^{1} + L_{43}^{1} \quad (15)$$

The right hand side of equation 15 has a value greater than 0 as the network has at least one arc with a positive demand. All the terms in the left hand side of equation 15 are flow variables for arcs connected to the depot node 1. In order to satisfy equation 15, the left hand side of equation 15 must be a positive number. Since the flow variable for an arc can be a positive value only if a vehicle travels on that arc (equation 10), equation 15 ensures that at least one arc connected to the depot node is used for traveling, thus preventing the formation of sub-tours.

To illustrate the mathematical formulation, a simple problem having 9 nodes and 12 arcs, as shown in Figure 3.2, is considered. The distance, demand, and service requirement of each arc are presented in Table 3-2. A requirement of 0.50 for an arc means that that arc can be serviced by traveling in either direction of the arc. All other parameters are shown in Table 3-3.

A deadheading speed of 25 distance units per hour and a servicing speed of 15 distance units per hour are assumed in this model. Therefore the time to deadhead (T_{ij}) on arc (i,j) is $1/25 * C_{ij}$. The difference between servicing time and deadheading time (G_{ij}) is $(1/15-1/25)*C_{ij} = 1/37.5*C_{ij}$

A complete formulation of the problem is listed in appendix A. The problem was first coded and solved using MPL/CPLEX to find the minimum route distance. The model has 403 constraints, 132 integer variables, and 1095 continuous variables. MPL/CPLEX solved the problem in approximately 8 seconds. Adding 1 arc to this problem caused the computation time to increase to more than 34 minutes.

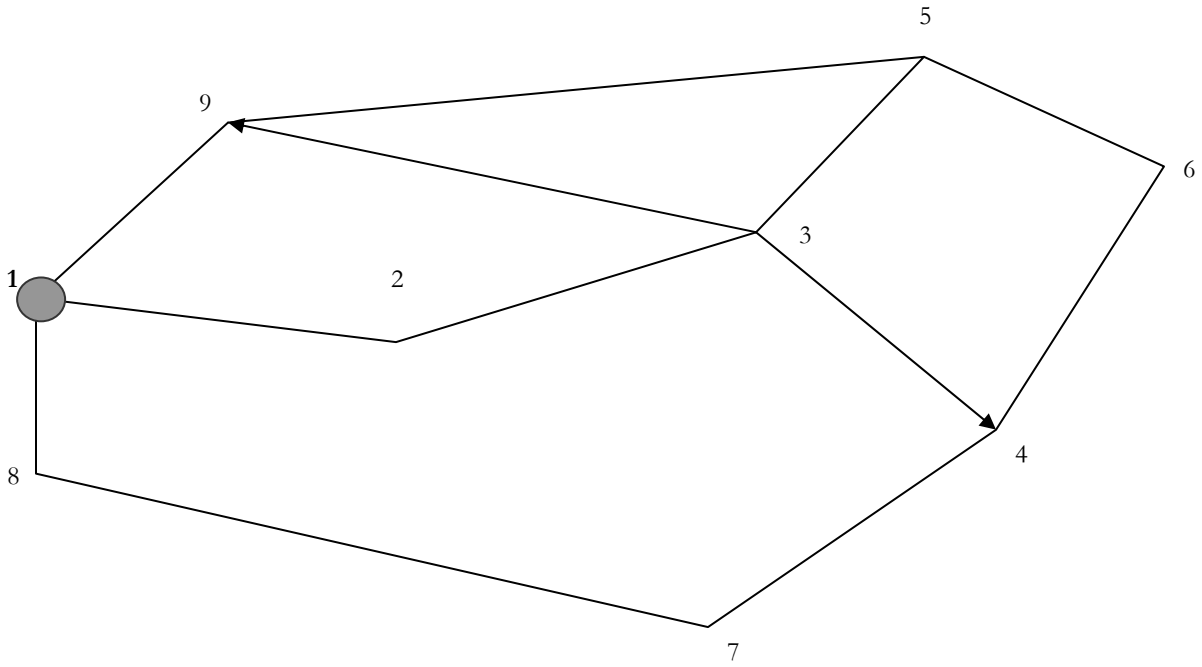

Figure 3.2 Example to Illustrate the Mathematical Formulation

Table 3-2 Example Problem Parameters (i)

Beginning Node (i)	Ending Node (j)	Distance (C_{ij})	Demand (Q_{ij})	Requirement (N_{ij})
1	2	2	2	0.50
1	8	2	2	0.50
1	9	3	3	0.50
2	3	3	3	0.50
3	4	2	2	1.00
3	5	3	3	0.50
3	9	6	6	1.00
4	6	5	5	0.50
4	7	4	4	0.50
5	6	4	4	1.00
5	9	6	6	0.50
6	5	4	4	1.00
7	8	7	7	0.50

Table 3-3 Example Problem Parameters (ii)

Parameter	Value
k	3 vehicles
n	9 nodes
W	18 units
T	2 hours

Larger problems could not be solved by this method as they required too much computation time, and the computation time increases exponentially with the size of the problem. Therefore, a heuristic algorithm needed to be developed to obtain a near optimal solution for large problems.

CHAPTER 4

4 Heuristic Algorithm

4.1 Introduction

Golden and Wong (1981) proved that even finding a solution for the CARP that lies within 1.5 times the optimal cost is NP-hard. Therefore, to solve this problem, a heuristic method needs to be developed. Heuristics are common-sense rules drawn from experience to solve combinatorial problems. They do not always lead to the optimum solution but usually produce good results in a reasonable computation time.

In this research, a heuristic algorithm based on a Greedy Randomized Adaptive Search Procedure (GRASP) (Resende and Ribeiro, 2003) was developed to solve the problem. GRASP is a multi-start meta-heuristic for combinatorial problems, in which each iteration consists of two parts: construction and local search. The construction phase builds a feasible solution, whose neighborhood is investigated during the local search phase until a local minimum is found. The best overall solution is kept as the final result. A Route Construction Algorithm (RCA) is used to create initial solutions for the problem, and then a Simulated Annealing meta-heuristic is applied to perform the local search on the results obtained by RCA in each iteration of the GRASP.

This chapter explains the methodology and detailed description of the heuristic algorithm used along with a numerical illustration.

4.2 Phase 1: Route Construction Algorithm (RCA)

RCA works by tracing a route beginning at the depot node and incrementally adding an arc based on constraints and certain criteria at each iteration, until a route is completed. At each iteration, a list of candidates, Restricted Candidate List (RCL), is created by considering all possible candidates that satisfy a greedy evaluation function and that can be added to the current partial solution without destroying the feasibility of the solution. The greedy evaluation function calculates the incremental increase in total cost due to addition of the candidate element to the partial solution and considers only the candidates whose incremental cost lie below a threshold value. A candidate is selected from the RCL randomly and added to the current partial solution. This step is repeated until the final solution is obtained.

In the context of the problem on hand, the term cost is used to refer to the distance. The term "required arc" in this discussion is used for an arc that has a positive demand and needs to be serviced by a vehicle. The nomenclature used in the construction heuristic is given in Table 4-1.

Table 4-1 RCA Nomenclature

Variable	Definition
RCL	Restricted Candidate List
α	Threshold parameter. Value between 0 and 1
C_{min}	Lowest incremental cost of a candidate element to be added to the partial solution
C_{max}	Highest incremental cost of a candidate element to be added to the partial solution
E	List of possible candidates that can be added to the partial solution
No_Routes	Number of routes developed in the network

The steps of the construction algorithm (RCA) as applied to the CARP are outlined below:

1. Calculate the shortest distance between each pair of nodes in the network using Floyd's algorithm. For details on Floyd's Algorithm, please refer to Appendix B. Set No_Routes = 0.
2. Set current node = depot node, Partial Route as no route, Partial Cost=0, and Remaining Capacity = Capacity of Vehicle. Each route starts from the depot node. If no required arcs exist in the network (i.e. all arc requirement = 0), go to step 7. Otherwise, go to step 3.
3. If one or more required arc is connected to the current node then go to step 4, otherwise go to Step 6.
4. Create a restricted candidate list (RCL) consisting of all arcs that can be added to the current route without violating the vehicle capacity constraint, and having a cost in the range $R = [C_{min}, C_{min} + \alpha * (C_{max}-C_{min})]$

   The value of α is selected between 0 and 1. A value of α close to 0 implies that the next arc for insertion would always be a low cost arc and a value of α close to 1 implies that an arc would be selected at random from all the candidates. The selection of the value of α is discussed in detail in chapter 7 (Experimentation).
5. If RCL has at least one arc element, select an arc at random from the RCL and add that arc to the partial route. Update Partial route as the current Partial route with the

added arc, Partial cost = Partial cost + cost of the added arc, Remaining capacity = Remaining Capacity - capacity requirement of the added arc. If the requirement of the added arc is an integer (>=1), subtract 1 from the requirement of the added arc, otherwise set the requirement of the added arc, in both directions, to zero. Set current node as the ending node of the recently added arc and go to step 3. Otherwise, go to step 6.

6. If no required arc in the network can be added to the route without violating the capacity constraint of the route, find the shortest route back to the depot; update the Partial route with the shortest path to the depot, Route cost = Partial cost + shortest distance to the depot. Set No_Routes = No_Routes + 1, and go to step 2. Otherwise, find the shortest path to the closest node that is connected to a required arc whose service requirement does not exceed the remaining vehicle capacity. Update the Partial route with the shortest path to the closest node, Route cost = Partial cost + shortest distance (cost) to the closest node. Set the current node as that node, and go to step 4.
7. The solution is the set of routes covering all the required arcs in the network. The total number of routes in the network is No_Routes.

Another variation of the above algorithm was tried. Instead of setting an upper limit on the cost of arcs that can be part of RCL (as in the version described above), a lower limit was set on the cost of arcs that can be part of RCL. This ensures that arcs having the longest distances are given preference. The range R in step 4 of the above algorithm was modified to $[C_{max} - \alpha * (C_{max}-C_{min}), C_{max}]$ for this purpose. In this case, a value of α close to 0 implies that the next arc for insertion would always be the high cost arc and a value of α close to 1 implies that an arc would be selected at random from all the candidate arcs. Results obtained with this approach did not show significant difference from those obtained with the above version of RCA.

One of the rules from Golden et al.'s Path Scanning Algorithm (1983) was also adopted to generate the initial solution. The steps vary from the RCA in the selection of the candidate to be added to the partial solution at each step. If the remaining vehicle capacity after the partial route is constructed is less than half of the total vehicle capacity, an arc that minimizes the distance from end of the arc to the depot is selected for inserting in the route.

Otherwise, an arc that maximizes the distance from end of the arc to the depot is selected. The value of α in step 4 of RCA can also be modified for this rule as shown below:

$\alpha = 0$, if vehicle capacity is less than half-full
$= 1$, otherwise

The solution obtained from RCA is used as the starting solution in a simulated annealing algorithm. In the next section, an example of the Route Construction Algorithm (RCA) is presented.

4.2.1 Route Construction Algorithm (RCA) Example

A simple illustration of the RCA is given below.

Consider a set of arcs as shown in Figure 4.1 with the depot at node 1. Each arc has beginning and ending nodes, length (distance), arc demand, and capacity requirement as shown in Table 4-2. A capacity requirement of 0.5 for an arc means that the arc can be serviced by traveling in either direction on the arc (from beginning node to the ending node or the ending node to the beginning node). A requirement expressed as an integer implies that the arc needs to be serviced that many times in the direction from the beginning node to the ending node. All arcs can be used for traveling in either direction, except arcs 3-4 and 3-9, which are one way arcs.

A vehicle capacity of 18 is used in this example.

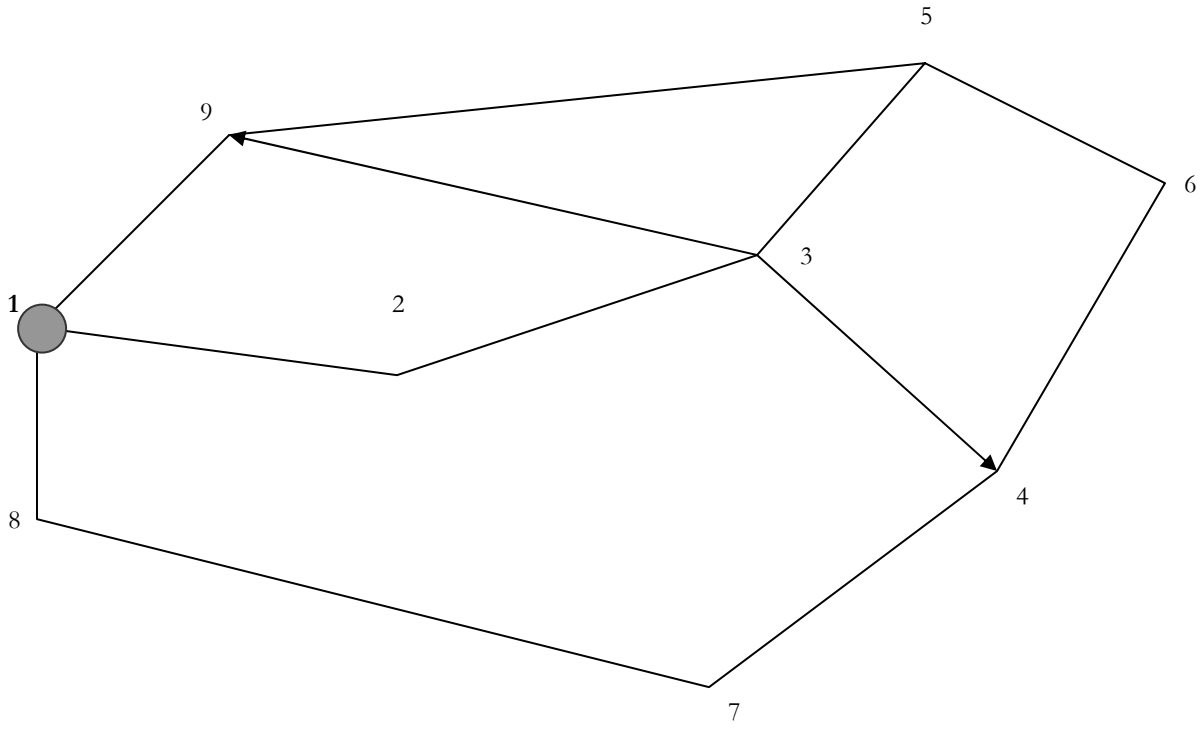

Figure 4.1 Example to Illustrate the RCA

Table 4-2 Example Problem Parameters

Beginning Node	Ending Node	Distance	Demand	Requirement
1	2	2	2	0.50
1	8	2	2	0.50
1	9	3	3	0.50
2	3	3	3	0.50
3	4	2	2	1.00
3	5	3	3	0.50
3	9	6	6	1.00
4	6	5	5	0.50
4	7	4	4	0.50
5	6	4	4	1.00
5	9	6	6	0.50
6	5	4	4	1.00
7	8	7	7	0.50

The RCA algorithm as applied to the above example is illustrated in the following steps:

Step 1: Shortest distance between each pair of nodes in the network is calculated using Floyd's algorithm. A table showing the distances between all pairs of nodes is shown below.

Number of routes = 0

Table 4-3 RCA Example Distance Matrix

From/To	1	2	3	4	5	6	7	8	9
1	-	2	-	-	-	-	-	2	3
2	2	-	3	-	-	-	-	-	-
3	-	3	-	2	3	-	-	-	6
4	-	-	-	-	-	5	4	-	-
5	-	-	3	-	-	4	-	-	6
6	-	-	-	5	4	-	-	-	-
7	-	-	-	4	-	-	-	7	-
8	2	-	-	-	-	-	7	-	-
9	3	-	-	-	6	-	-	-	-

Step 2: Set current node = 1 (depot node)

Partial route: None, Partial cost: 0, Remaining capacity: 18

Steps 3 and 4: Since three required arcs (1-2, 1-8, 1-9) are connected to the depot node, a Restricted Candidate List (RCL) is created. The lowest cost arc connected to the depot node is arc 1-2 with cost 2 and the highest cost arc connected to the depot node is arc 1-9 with cost 3 and all these arcs can be added to the route without violating the route capacity constraints. Therefore, C_{min}= 2 and C_{max}= 3

Value of α was selected as 0.4.

Range of costs for inclusion in RCL = [2, 2+0.4*(3-2)] = [2, 2.4]

Two arcs 1-2 and 1-8 connected to node 1 have cost in the range [2, 2.4] and are therefore included in the RCL.

Step 5: An arc is selected at random from RCL and the selected arc is added to the partial route. Selected arc: 1-2

Partial route: 1-2, Partial cost: 0+2 = 2, Capacity remaining: 18-2 = 16, Requirement for arc 1-2 is 0.

Set current node = 2 and go to step 3.

Steps 3 and 4: Since one required arc (2-3) is connected to the node 2 and arc (2,3) can be added to the partial route without violating route capacity constraint, RCL consists of arc (2,3) and this arc is added to the partial route.

Partial route: 1-2-3, Partial cost: 2+3 = 5, Remaining capacity: 16-3 = 13, Requirement for arc 2-3 is 0.

Set current node = 3 and go to step 3.

Steps 3 and 4: Three required arcs (3-4, 3-5, 3-9) are connected to the current node 3. All these three arcs can be added to the current route without violating route capacity constraint.

C_{min}= 2 (for arc 3-4) and C_{max}= 6 (for arc 3-9)

Value of α = 0.4

Range of costs for inclusion in RCL = [2, 2+0.4*(6-2)] = [2, 3.6]

Two arcs 3-4 and 3-5 connected to node 3 have cost in the range [2, 3.6] and are therefore included in the RCL.

Step 5: An arc is selected at random from RCL and the selected arc is added to the partial route. Selected arc: 3-4

Partial route: 1-2-3-4, Partial cost: 5+2 = 7, Remaining capacity: 13-2 = 11, Requirement for arc 3-4 is 0.

Set current node = 4 and go to step 3.

Steps 3 and 4: Two required arcs (4-6, 4-7) are connected to the current node 4. Both these arcs can be added to the current route without violating route capacity constraint. C_{min} = 4 (for arc 4-7) and C_{max}= 5 (for arc 4-6)

Value of α = 0.4

Range of costs for inclusion in RCL = [4, 4+0.4*(5-4)] = [4, 4.4]

Arc 4-7 has cost in the range [4, 4.4] and is the only arc in the RCL.

Step 5: Since arc 4-7 is the only arc in RCL, it is added to the partial route.

Partial route: 1-2-3-4-7, Partial cost: 7+4 = 11, Remaining capacity: 11-4 = 7, Requirement for arc 4-7 is 0.

Set current node = 7 and go to step 3.

Steps 3 and 4: One required arc (7-8) is connected to the current node 7 and can be added to the current route without violating route capacity constraint. Therefore, RCL consists of arc 7-8.

Step 5: Since RCL consists of one arc (7-8), this arc is added to the partial route
Partial route: 1-2-3-4-7-8, Partial cost: 11+7 = 18, Remaining capacity: 7-7 = 0, Requirement for arc 7-8 is 0.
Set current node = 8 and go to step 3.
Steps 3 and 4: One required arc (8-2) is connected to the current node 8, but this arc cannot be added to the partial route without violating the route capacity constraint. Therefore, RCL doesn't have any arc elements.
Step 5: Since RCL doesn't consist of any arc elements, go to step 6.
Step 6: No required arc in the network can be added to the partial route without violating the route capacity constraint. Therefore, trace the shortest path back to the depot node 1 (which is 8-1).
Route: 1-2-3-4-7-8-1, Route cost: 18+2 = 20.
Increment the number of routes by 1 i.e. Number of routes = 1, and go to Step 2.
Step 2: Set current node = 1 (depot node) and continue the steps in the algorithm to cover all the required arcs in the network.

In the following section, the simulated annealing algorithm is discussed in detail.

4.3 Phase 2: Simulated Annealing Heuristic

4.3.1 Background

Simulated annealing is a search technique used to find good solutions to combinatorial problems. Metropolis et al. (1953) first published the ideas that form the basis of simulated annealing. The idea behind simulated annealing is that if molten material is cooled back into a solid state, the structural properties of the cooled solid depend on the rate of cooling. If sufficiently slow cooling is used, large and orderly arranged crystals are formed and this state is stable and has the lowest energy. On the other hand, if the material is cooled quickly, imperfections are formed in the crystal structure. In this state, the material is in a state of instability and has a higher energy. Simulated Annealing algorithm provides an algorithmic way of exploiting the connection between this type of thermodynamic behavior and the search for global minima for a combinatorial optimization problem (Henderson, et al., 2003). Metropolis et al.'s algorithm simulates the change in energy of a physical system when subjected to a cooling process until it reaches a state of thermal equilibrium.

Later, Kirkpatirick et al. (1983) suggested the use of simulated annealing to search for feasible solutions of an optimization problem in order to reach an optimal solution. Simulated annealing algorithm searches a wide solution space by accepting inferior solutions with a probability that decreases as the algorithm progresses. By sometimes accepting a non-improving solution, the simulated annealing algorithm avoids being trapped in local optima and can search different areas in the solution space for the global optima.
Since the work of Metropolis, a significant amount of work has been done on simulated annealing and its applications in different fields of combinatorial optimization.

The general simulated annealing algorithm for a minimization problem is outlined below.

Select an initial solution s_0, an initial temperature t_0, temperature reduction factor a, and number of iterations at each temperature NI_k, current solution $s=s_0$, counter k=0, best solution $s_{min}=s_0$, best objective function value $f_{min}=f(s_0)$

Repeat
Set counter n=0
    Repeat
        Generate a solution s' ε N(s), where N(s) is neighborhood of solution s
        $\Delta_{s,s'} = f(s)-f(s')$
        if $\Delta_{s,s'} \geq 0$, then $s \leftarrow s'$. If $f(s') < f(s_{min})$ then $f_{min}=f(s')$ and $s_{min}=s'$
        if $\Delta_{s,s'} < 0$, then $s \leftarrow s'$ with probability $\exp(\Delta_{s,s'} / t_k)$
        n=n+1
    until $n = NI_k$
    k=k+1
    set $t_k = a.t_{k-1}$
Until stopping condition = true

The stopping condition can be expressed in terms of minimum temperature or maximum number of iterations allowed. If the temperature of the system is less than or

equal to the minimum temperature desired or the maximum number of iterations has been reached, the algorithm stops.

4.3.2 SA Heuristic Algorithm

The simulated annealing algorithm takes the final route obtained from the construction algorithm RCA and searches for better solutions in the solution space. This is performed by moving arcs between routes. The common terms used in describing the algorithm are discussed below.

Terms used:
Required arc/ Serviced Arc - Arc that is required to be serviced
Unrequired arc - Arc that can be used for traveling but doesn't require servicing (deadheading)
Routex1 - The route from which one or more arcs are removed with or without insertion of one or more arcs from other routes
Routex2 - The route in which one or more arcs are inserted with or without removal of one arc or more arcs from this route.

As explained previously, the simulated annealing search process starts with the solution obtained from RCA and searches a defined neighborhood of this solution for a better one. As simulated annealing allows for uphill moves, i.e. moves that give solutions worse than the current solutions, there is a chance that the process can escape local minima and find better solutions. Different types of moves were tried on a few problems and five types of moves performed the best. These are:

1. One arc move- Moving one arc from Routex1 to Routex2 (Move1)
2. One to one exchange- Exchanging one arc between Routex1 and Routex2 (Move2)
3. Two to one exchange- Moving two arcs from Routex1 to Routex2, and one arc from Routex2 to Routex1 (Move3)
4. Three to one exchange- Moving three arcs from Routex1 to Routex2, and one arc from Routex2 to Routex1 (Move4)

5. Three to two exchange- Moving three arcs from Routex1 to Routex2, and two arcs from Routex2 to Routex1 (Move5)

When arc(s) are removed or inserted in a route, a Route Improvement (RI) algorithm is used to connect the arcs in the route starting and ending at the depot node. The next section describes RI in detail.

4.3.2.1 Route Improvement Algorithm (RI)

The input for the RI is a set of arcs, which are obtained as a result of different types of moves as explained above. As an example, consider two routes A and B with Route A consisting of 5 arcs: {A1, A2, A3, A4, A5} and Route B consisting of 4 arcs: {B1, B2, B3, B4} in that sequence. Both routes start and end at the depot node. If Move2 is performed on these two routes and arc A3 in route A is exchanged with arc B2 in route B, new routes need to be traced through the new sets of arcs. The 2 sets of arcs now become:

Set 1: {A1, A2, A4, A5, B2}
Set 2: {B1, B3, B4, A3}

RI works on each set of arcs, one set at a time to create a low cost route starting and ending at the depot. The algorithm works by first tracing a route starting and ending at the depot and connecting all the arcs in the set. Then the route is divided at multiple points, which results in creation of multiple elements. The sequence of elements in the route is changed by moving the elements within the route. This results in multiple sequences or routes. The output of this algorithm is the route (sequence of elements) having the cheapest cost among the evaluated (rearranged) sequences of elements. The following terms are used in describing RI, depending on the number of points at which an initial or intermediate route (obtained after rearranging) is divided.

Level 1 routes: Set of routes obtained by splitting the input route at 2 points in all possible ways and rearranging the split segments. The analysis performed for obtaining these routes is called Level 1 analysis.

As an example, consider a route starting and ending at the depot node (node 1) that contains 4 required one-way arcs A1, A2, A3 and A4 as shown below:

1-A1-A2-A3-A4-1

Not considering the depot node, Table 4-4 shows the points where the routes can be split and the possible rearrangements of segments created.

Table 4-4 Possible Splitting Points and Rearrangements

No	Splitting points (denoted by ~)	Segments created	Possible Rearrangements
1	A1~A2~A3-A4	A1, A2, A3-A4	A1-A2-A3-A4, A1-A3-A4-A2, A2-A1-A3-A4, A2-A3-A4-A1, A3-A4-A1-A2, A3-A4-A2-A1
2	A1-A2~A3~A4	A1-A2, A3, A4	A1-A2-A3-A4, A1-A2-A4-A3, A3-A1-A2-A4, A3-A4-A1-A2, A4-A1-A2-A3, A4-A3-A1-A2
3	A1~A2-A3~A4	A1, A2-A3, A4	A1-A2-A3-A4, A1-A4-A2-A3, A2-A3-A1-A4, A2-A3-A4-A1, A4-A1-A2-A3, A4-A2-A3-A1

Level 2 routes: Set of routes obtained by splitting the input route at 3 points in all possible ways and rearranging the split segments. The analysis performed for obtaining these routes is called Level 2 analysis. This is done in a similar fashion to the example shown for Level 1 routes.

N1: The number of best (cheapest) Level 1 routes tracked. The cost of each route created during Level 1 analysis is evaluated and the N1 cheapest routes are tracked for further analysis.

N2: The number of best (cheapest) Level 1 routes selected for splitting at 3 points (for creating Level 2 routes). The cost of each route created during Level 2 analysis is evaluated and the N2 cheapest routes are tracked for further analysis.

The steps followed in RI are explained below:

1. Create a route starting at the depot and connecting each required arc in the set and ending at the depot, using a construction algorithm. This algorithm is based on the greedy heuristic approach called "next best" (NB), originally applied to job scheduling problems (Gavett, 1965). This method works by selecting the task, which has the least cost after the current task is completed. This procedure was modified for tracing a route through a set of arcs starting and ending at the depot node. An example that shows the details of this algorithm is presented in section 4.3.2.1.1.
2. The route obtained in step 1 is split at two points in all possible ways (Level 1 analysis). Splitting at two points creates three segments. The sequence in which the three segments are serviced is changed, i.e. the split segments are rearranged in all possible ways and the cost of each route is evaluated. While rearranging the elements, both orientations of an arc are considered if the arc is a two-way arc (allowing service in both directions).
3. During Level 1 analysis, new routes are found. The N1 lowest cost routes are tracked and stored in the Level 1 list. If a route cheaper than a route in Level 1 list is found, the new route is included in the list and the list is updated.
4. The top N2 routes are selected from the level 1 list. These routes are called Level 2 routes and are stored in the Level 2 list. Each Level 2 route is then split at 3 points in all possible ways, thus creating four segments. The sequence in which the four segments are serviced is changed, i.e. the split segments are rearranged in all possible ways and the cost of each route is evaluated considering both orientations of an arc if the arc is a two-way arc (allowing service in both directions).
5. If a new route is obtained during Level 2 analysis and it is cheaper than a route in Level 1 list (top N1 routes), Level 1 analysis (splitting at two points) is performed on the new route. If Level 1 analysis on this new route creates a route that is cheaper than a top N2 route in the Level 2 list, Level 2 analysis is performed on the newly created route.
6. The cheapest cost route obtained after Level 1 and Level 2 analysis is the final route produced by the algorithm.

This algorithm is illustrated with an example in the next section.

4.3.2.1.1 RI Example

Consider a route represented as:

1-(1-2)-(3-4)-(4-5)-(5-6)-7-8-(9-10)-10-11-1

Parenthesis in the above route implies that the arc inside parenthesis is a required arc. The above route can be explained as:

- Start at depot (node 1)
- Service arc 1-2
- Move from 2 to 3 by the shortest distance (which is 2-3)
- Service arc 3-4
- Service arc 4-5
- Service arc 5-6
- Move from node 6 to 9 via the shortest distance (which is 6-7-8-9)
- Service arc 9-10
- Move from 10 to depot via the shortest distance (which is 10-11-1)

Suppose a required arc 12-13 is moved from another route Routex1 to this route as a result of performing Move1 (one arc move). Considering only the required arcs, the arcs in the route above can be denoted as shown in Table 4-5.

Table 4-5 List of Arcs Including the Inserted Arc

Arc	Denoted by:
1-2	A1
3-4	A2
4-5	A3
5-6	A4
9-10	A5
12-13	A6 (inserted arc)

Using this notation, the route (without the inserted arc) can be represented as:
1-A1-A2-A3-A4-A5-1

Using the NB algorithm, a route is created using the above arcs (including the inserted arc). A square matrix is created with dimension of (1 + no of arcs), where each cell denotes the distance from the row element to the column element for that cell. Each cell in the first row denotes the distance from the depot node to the end of the arc of the column element and each cell in the first column denotes the distance from the end of the arc in the row element to the depot node. All other cells consist of distance values from the end of the arc in the row element to the end of the arc of the column element.

The construction algorithm starts at the depot node, and traces the partial route from depot node to the minimum cost column element. The first row and the selected column are then covered. Next, the minimum cost uncovered element in the row corresponding to the column element (excluding column 1) is found and that element is inserted into the route and the corresponding row and column are covered. This step is repeated till all the columns, except column 1, are covered; the route is completed by connecting the end of the last arc to the depot. If two or more elements have the same minimum cost, the next element for insertion in the route is selected randomly. Table 4-6 represents an example of the representation of the square matrix.

Table 4-6 Matrix for Creating a Route in the RI Example

From\To	1	A1	A2	A3	A4	A5	A6
1	-	3	6	5	7	10	9
A1	3	-	3	4	6	12	12
A2	4	7	-	1	3	9	13
A3	5	8	5	-	2	8	14
A4	7	10	7	4	-	6	17
A5	9	12	13	10	10	-	11
A6	4	7	10	9	11	9	-

The steps for generating the initial route using the above matrix are outlined below:

1. Starting at depot node 1, find the column element with the smallest value. The smallest value is 3 for element A1.
   Partial solution: 1-A1, Partial cost: 3
   Cover row 1 and column A1.
2. From row element A1, find the column element (excluding the first column) not covered before and with the smallest value. The smallest value is 3 for the element between row A1 and column A2.
   Partial solution: 1-A1-A2, Partial cost: 6
   Cover row A1 and column A2.
3. From row element A2, find the column element (excluding the first column) not covered before and with the smallest value. The smallest value is 1 for the element from A2 to A3.
   Partial solution: 1-A1-A2-A3, Partial cost: 7
   Cover row A2 and column A3.
4. From row element A3, find the column element (excluding the first column) not covered before and with the smallest value. The smallest value is 2 for the element from A3 to A4.
   Partial solution: 1- A1-A2-A3-A4, Partial cost: 9
   Cover row A3 and column A4.
5. From row element A4, find the column element (excluding the first column) not covered before and with the smallest value. The smallest value is 6 for the element from A4 to A5.
   Partial solution: 1-A1-A2-A3-A4-A5, Partial cost: 15
   Cover row A4 and column A5.
6. From row element A5, find the column element (excluding the first column) not covered before and with the smallest value. The smallest value is 11 for the element from A5 to A6.
   Partial solution: 1-A1-A2-A3-A4-A5-A6, Partial cost: 26
   Cover row A5 and column A6.
7. Since all columns (except column 1) have been covered, go back to the depot node 1 from the row element A6. The cost of traveling from A6 to 1 is 4.

Final solution: 1-A1-A2-A3-A4-A5-A6-1, Final cost: 30

In this way, an initial route is traced connecting the depot node with the set of arcs listed in Table 4-5.
The generated route is:
1-A1-A2-A3-A4-A5-A6-1……(16)

In (16), the starting and ending 1 denote the depot node.
The next step is to divide this route at multiple points. As an example, the generated route can be divided at two points denoted by "~" below:

1-A1~A2-A3~A4-A5-A6-1

Splitting at 2 points creates 3 elements (not considering the depot node):
A1, A2-A3, A4-A5-A6

These elements are rearranged, i.e. sequence is changed. The possible sequences are:

1. (A1, A2-A3, A4-A5-A6) – same as original route
2. (A1, A4-A5-A6, A2-A3)
3. (A2-A3, A1, A4-A5-A6)
4. (A2-A3, A4-A5-A6, A1)
5. (A4-A5-A6, A1, A2-A3)
6. (A4-A5-A6, A2-A3, A1)

In this way a route is split in all possible ways at 2 points, rearranged and the cost is evaluated. While rearranging the elements, both orientations of an arc are considered if the arc is a two-way arc (allowing service in either direction). The N1 routes with least cost routes (obtained after rearranging the elements in all possible ways) are tracked (Level 1 routes). The top N2 (Level 2 routes) of these N1 routes are further analyzed by splitting at 3 points in all possible ways and rearranging the elements obtained after splitting. If a cheaper route than any of the top N1 routes is found during Level 2 analysis, then Level 1 analysis is performed on that route. If Level 1 analysis on this new route creates a route that is cheaper

than a top N2 route in the Level 2 list, Level 2 analysis is performed on the newly created route. The cheapest cost route obtained after Level 1 and Level 2 analysis is the final output of the algorithm.

Using RI, low cost routes are determined for both Routex1 and Routex2 routes obtained as a result of one of the five moves explained in Section 4.3.2. The next section describes the five type of moves performed in detail.

4.3.2.2 Types of Moves

A brief introduction was given in section 4.3.2 about the five types of moves performed on the routes obtained by RCA. These moves help in exploring the solution space in depth. Each of these moves is performed for a specific number of iterations as part of the simulated annealing heuristic.

The moves performed are:

1. Move1

This type of move involves moving a single arc from one route to another (from Routex1 to Routex2)

Steps in Move1:

1. Create a candidate list of arcs that can be moved from one route to another without violating the capacity constraint of the route to which it is moved.

   There are 3 categories of such arcs-

   a) Arcs for which both the preceding and succeeding arcs are not serviced arcs.
   b) Arcs for which only one of the preceding or succeeding arc is a serviced arc.
   c) Arcs for which both the preceding and succeeding arcs are serviced arcs.

   For moving an arc from one route to other, the highest preference is given to type (a) arcs followed by type (b) and then type (c) arcs.
2. For each arc that can be moved, all the candidate routes in which the arc can be inserted without violating the route capacity constraint are determined.
3. An arc is selected at random from the three types of candidate arcs listed in step 1, with the highest probability of selection assigned to type (a) arcs followed by type (b) and type (c) arcs.

4. For the arc selected for moving from one route into another, the route in which the arc can be inserted is selected at random from the candidate routes created in step 2. In another version of Move1, the shortest distance from a candidate arc to each of the possible destination routes is found and the route that is closest to the arc is selected. The distance of a route from an arc is calculated by evaluating the minimum value among the distances from each of the two nodes of the arc to all nodes in the candidate destination routes. The destination route that is closest to the arc is selected for insertion.

The route from which the arc is to be removed is designated as Routex1 and the route into which the arc is inserted is designated as Routex2.

2. Move2

This type of move involves exchanging two arcs between two routes without exceeding the route capacity limits of the routes to which the arcs are moved.

Steps in Move2:

1. A list of pair of arcs that can be exchanged between all possible pair of routes in the network is created using complete enumeration, in such a way that exchanging the arcs does not violate the route capacity constraint of either of the routes.
2. A selection is made at random from the list created in step 1. The selected item consists of the two arcs to be exchanged (and the routes they belong to). The routes whose arcs are exchanged are referred to as Routex1 and Routex2. The first selected arc (defined by the selected item) is removed from Routex1 and inserted into Routex2 and at the same time, other selected arc (defined by the selected item) is removed from Routex2 and inserted into Routex1.

3. Move3

This type of move involves moving two arcs from one route (Routex1) to another (Routex2) and moving one arc from Routex2 to Routex1. The exchange of arcs is performed in such a way that the maximum route capacity of the two routes is not exceeded. Steps in Move3 are similar to those of Move2.

4. Move4

This type of move involves moving three arcs from one route (Routex1) to another (Routex2), and moving one arc from Routex2 to Routex1. The exchange of arcs is performed in such a way that the maximum route capacity of the two routes is not exceeded. Steps in Move4 are similar to those of Move2.

5. Move5

This type of move involves moving three arcs from one route (Routex1) to another (Routex2), and moving two arcs from Routex2 to Routex1. The exchange of arcs is performed in such a way that the maximum route capacity of the two routes is not exceeded. Steps in Move5 are similar to those of Move2.

Sets of moves are defined such that each set contains each move only once and the moves are sequenced in a random order (e.g. Move3 - Move4 - Move1 - Move5 - Move2). Within each set, each of the five moves is performed for a fixed number of iterations. Upon completion of the moves in one set, the algorithm proceeds with a different set.

The nomenclature used in the simulated annealing heuristic is given in Table 4-7.

Table 4-7 Simulated Annealing Nomenclature

Variable	**Definition**
T	Temperature
N	Counter (Number of iterations) at the current temperature
N_max	Max number of iterations at each temperature
RCA_Cost	Cost (distance) obtained from the construction heuristic
RCA_Routes	Set of routes obtained from the construction heuristic.
Best_Cost	Lowest cost (for the best solution)
Best_Routes	Route corresponding to the lowest cost
Curr_Cost	Current cost
Curr_Routes	Set of routes corresponding to the current cost
R	Random value between 0 and 1
SA_Cost	Simulated Annealing cost

SA_Routes	Simulated Annealing route set
ΔTC	Value of (Curr_cost – SA_Cost)
a	Temperature reduction factor
N_move	Maximum number of iterations for each type of move
N_iteration	Counter (Number of iterations in the heuristic)
Max_Iteration	Maximum number of iterations in the heuristic

The steps for the SA algorithm are as follows:

1. Set the values of parameters a, T, N_max, N_move, Max_Iteration and set N = 0, N_iteration = 0. Create a Set of moves, Type of move is the first in the set. Cooling schedule is governed by the maximum number of iterations at each temperature N_max and the temperature reduction factor a. The temperature reduction factor ranges between 0 and 1 and determines the rate at which the temperature parameter reduces. This means that after every N_max iterations, the temperature is reduced by a factor a. The initial temperature T should be high enough to allow an almost free exchange of neighboring solutions, which allows the heuristic to search different areas of the solution space.
2. Set Best_Cost = RCA_Cost, and Best_Routes = RCA_Routes. In this step the best cost and route set are set to the solution value and route obtained from the RCA. Also the current cost and route set are set equal to the RCA's cost and route set, i.e. Curr_Cost = RCA_Cost and Curr_Routes = RCA_Routes.
3. Set N_iteration = N_iteration +1. The overall iteration counter is incremented by 1.
4. Set N = N +1. The annealing schedule iteration counter is incremented by 1.
5. Calculate the cost after performing the move and set it equal to SA_Cost. Also, set the new route set obtained as SA_Routes.
6. Calculate the difference between the current cost and the simulated annealing cost, ΔTC= (Curr_Cost – SA_Cost).
7. If ΔTC >= 0 then set Curr_Cost = SA_Cost and Curr_Routes = SA_Routes. In this step, the difference between current cost and the simulated annealing cost is evaluated. If the difference is greater than or equal to 0, then it means that the SA

solution is better than the current solution and the current values are set as SA values. Also, if the simulated annealing solution is less than the best solution obtained so far, then set the best solution as the simulated annealing solution., i.e. if SA_Cost < Best_Cost, set Best_Cost = SA_Cost and Best_Routes = SA_Routes.

8. If ΔTC < 0 and Exp(ΔTC/T) > R, then set Curr_Cost = SA_Cost and Curr_Routes = SA_Routes. If the SA solution obtained is greater than or equal to the current solution, the non improving solution is either accepted or rejected based on a probability function. R is a random value between 0 and 1. As the value ΔTC gets closer to 0, the non-improving solution has a higher chance of being accepted. Also as T gets smaller the probability of accepting a non-improving solution decreases.
9. If N = N_max then set T = a * T and N = 0. If the maximum number of iterations at the current temperature has been reached, the value of temperature is multiplied by a temperature reduction factor a (discussed in detail in Chapter 7). The temperature reduction factor has a value between 0 and 1. The annealing schedule counter N is reset to 0. Go to step 10.
10. If N_iteration = Max_Iteration, the heuristic terminates, else go to step 11.
11. If (N_iteration MOD (5 *N_move)) = 0 then generate a new set of moves. Type of move is the first in the set. Go to step 3.
    Otherwise, go to step 12.
12. If (N_iteration MOD N_move) = 0, then update type of move to the next move within the set of moves. Go to step 3.
    Otherwise, go to step 3.

The process flow chart for the SA heuristic is shown in Figure 4.2.

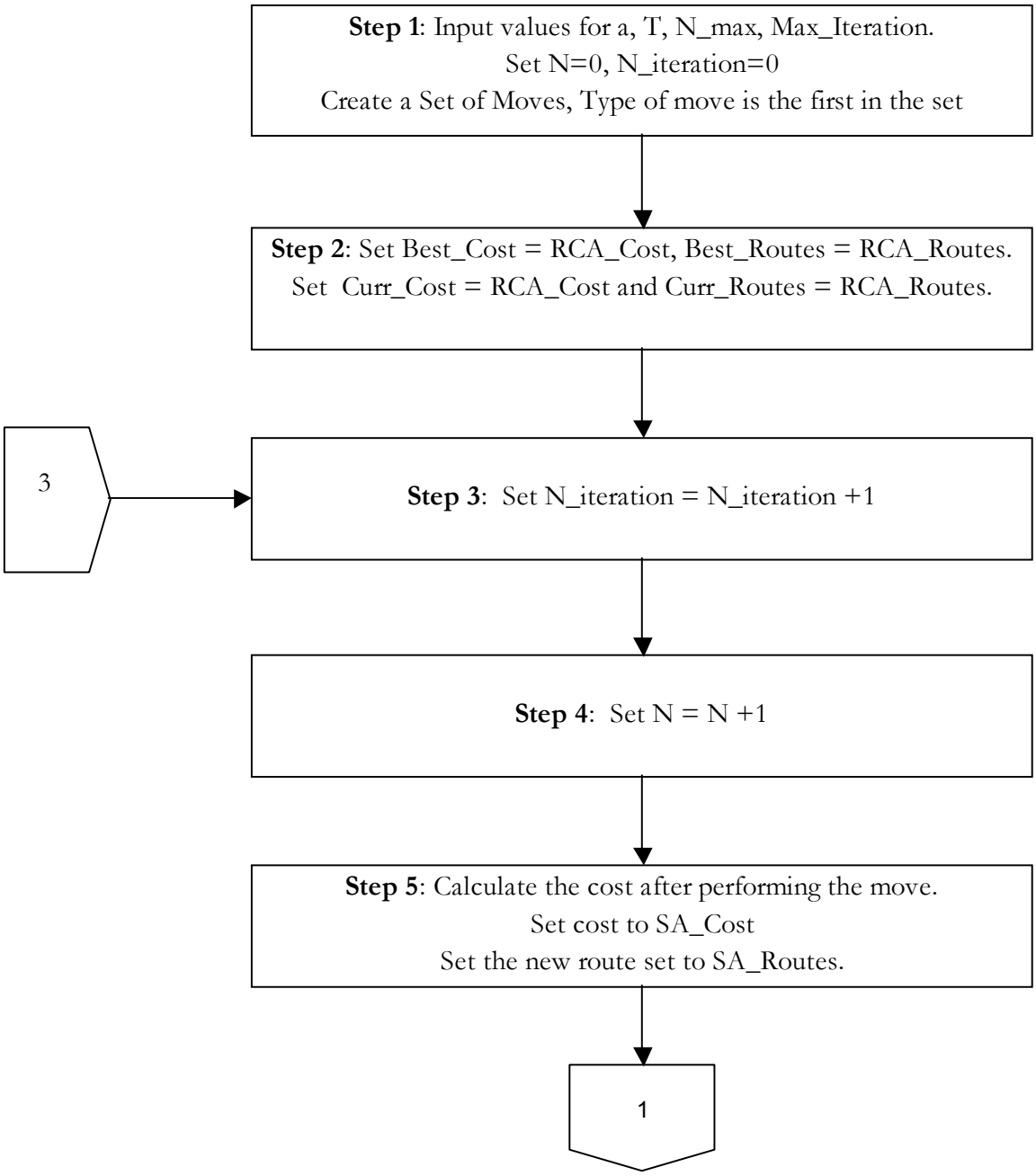

Figure 4.2 SA Flow Diagram

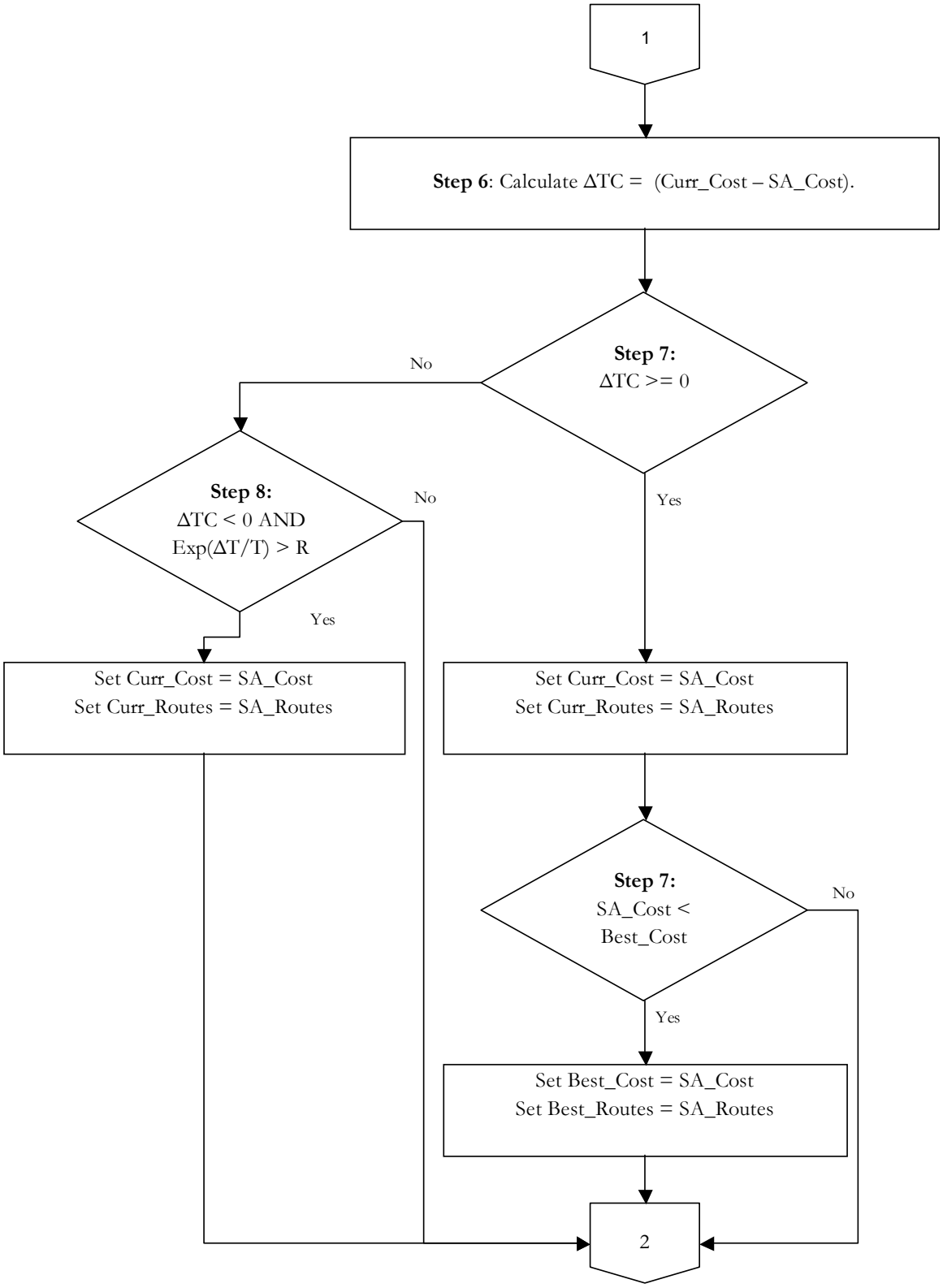

Figure 4.2 SA Flow Diagram (Contd.)

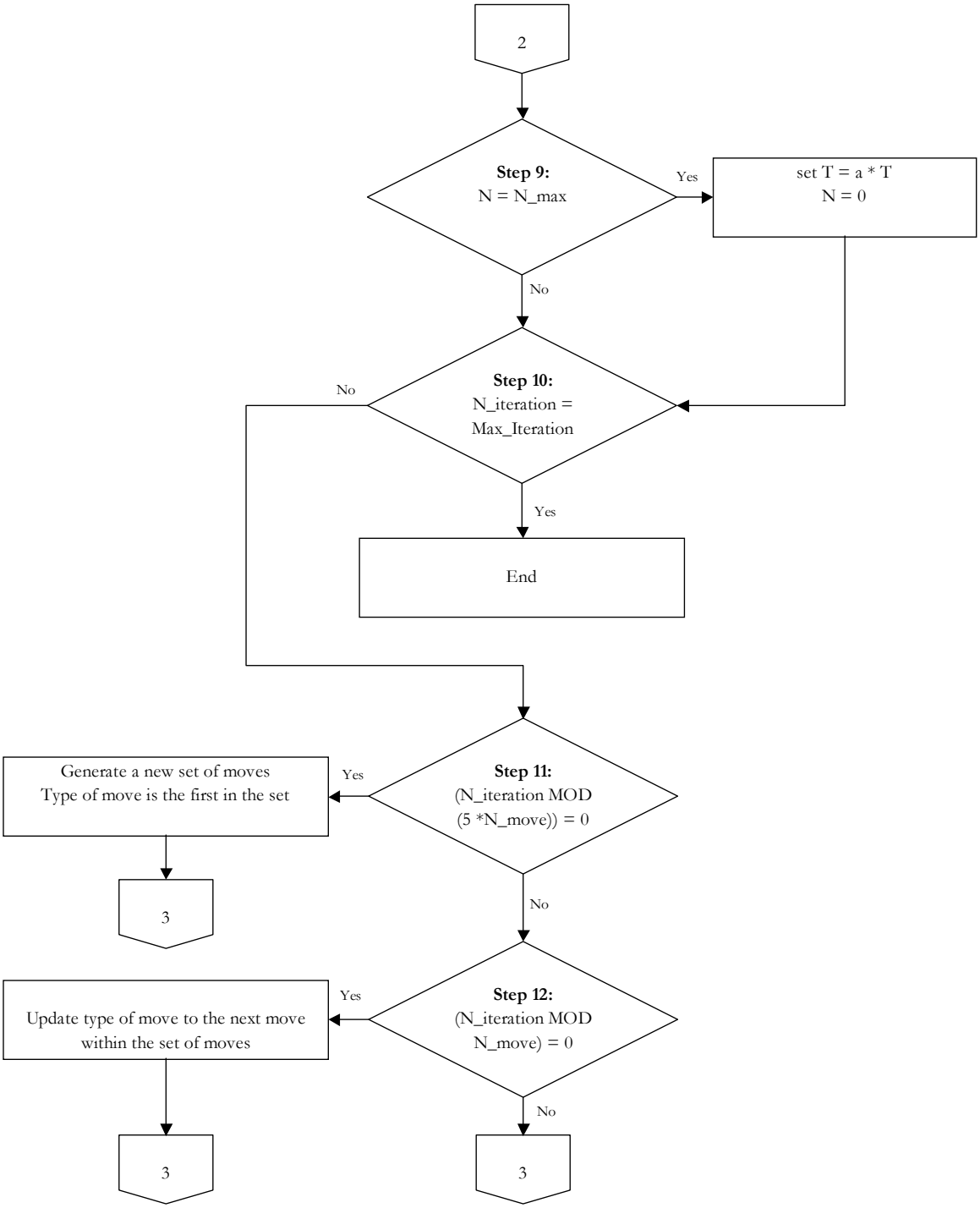

Figure 4.2 SA Flow Diagram (Contd.)

CHAPTER 5

5 Experimentation

5.1 Introduction

In order to evaluate the efficiency of a heuristic algorithm, one needs to compare its results to some standard benchmark. Barr et al. (1995) identified the following two cases:

• Where the optimal solution is known, the heuristic solution can be directly compared to it to measure the effectiveness of the heuristic.

• Where optimal solutions are unknown or unobtainable by current methods, some other benchmark of performance should be offered such as a comparison to a tight lower (upper) bound or comparison to a published solution values on publicly available test problems.

Since the CARP is NP-hard, performing comparisons between the heuristic results and the results obtained from exact methods would only be possible for small size problems. In order to test the solution quality, the heuristic is applied to publicly available test problems and the results are compared to published solution values for those problems.

5.2 Problem Instances

The heuristic was applied to four sets of problem instances provided in the literature. The benchmark sets used and the number of problems in each set is listed in the table below.

Table 5-1 Problem Instances

Set	Name	Number of instances	Built by	Problem sizes
1	gdb	23	DeArmon (1981)	No of nodes: 7-27 No of required arcs: 11-55
2	val	34	Belenguer and Benavent (2003)	No of nodes: 24-50 No of required arcs: 34-97
3	egl	24	Belenguer and Benavent (2003)	No of nodes: 77-140 No of required arcs: 51-190
4	mval	34	Belenguer et al. (2006)	No of nodes: 24-50 No of required arcs: 43-138

The gdb, val and egl datasets have instances with undirected arcs and mval has instances with both directed and undirected arcs (mixed network).

5.3 Experimentation Overview

A program was written in Microsoft Visual Basic 6.0 that performs the steps of the RCA and the SA heuristic. The output of the RCA is the input to the SA heuristic, which should improve the solution obtained by the RCA. After several replications, in which RCA is applied followed by SA, the best overall solution is selected. The following section describes the selection of parameters in the application of the RCA and the simulated annealing heuristic.

5.3.1 Selection of the threshold parameter α in the RCA

As discussed earlier, the value of the threshold parameter (α) controls what elements are included in the RCL from among possible candidates for insertion into the partial route. This parameter controls the randomness and greediness in the algorithm; a value of $\alpha = 0$ corresponds to a pure greedy construction, and a value of $\alpha = 1$ corresponds to a pure random construction for a minimization problem. Experiments were performed on some test problems to evaluate the effect of α on the results. The results obtained showed that no single value of α produced the best results, and that changing the value at the start of each new replication of the multi-start RCA provided better results than keeping the value of α fixed for all replications. At each start, the value of α is selected at random from the set [0,0.2,0.4,0.6,0.8,1] in order to generate a broad range of construction solution values for the local search process. The number of replications was set at 30 for all problem instances.

5.3.2 Selection of the SA parameters

The selection of parameters used in the SA heuristic should be made with care, as they affect the speed of the algorithm and the quality of the solutions obtained. Using the same nomenclature used in Table 4-7, these parameters are listed below.

1. Initial temperature, T
2. Temperature reduction factor, a
3. Maximum number of iterations at each temperature, N_max
4. Maximum number of iterations for each type of move, N_move

5. Stopping criterion- maximum number of iterations in the heuristic, Max_Iteration

The initial temperature (T), the rate at which the temperature is reduced (a), and number of iterations at each temperature (N_max) define the cooling schedule. After every N_max iterations, the temperature is multiplied by the temperature reduction factor a. Initially the temperature should be high enough so that non improving moves are accepted with a reasonable probability, as this would allow a broader solution space to be explored. If a very high initial temperature is used, almost all non-improving moves would be accepted and it would take a long time for the heuristic to converge to a low cost solution. The temperature reduction factor should be selected for sufficiently slow cooling as this would allow for in depth exploration of the solution space at a particular temperature. However, this is done at the expense of the speed of the algorithm. As explained previously, a type of move is selected from the set of moves (having five moves in random order) and applied for N_move iterations during the SA local search. Upon completion of 5*N_move moves (N_move moves performed for each of the five types of moves), the algorithm proceeds with a different set of moves. The stopping criterion is defined in terms of the maximum number of iterations (N_max) in the SA heuristic, and the heuristic stops when that number has been reached. The value of N_max is set in such a way that the temperature gets close to zero when the heuristic stops.

5.3.2.1 Values of parameters used

An initial value of temperature was selected using the following formula:

$$IT = \frac{CostDiff}{\ln(pa)}$$

where,

IT = The initial temperature value

CostDiff = Estimate of the difference between the initial solution and a solution obtained after performing a non-improving move. This value is set as the average cost (length) of all arcs that require service in the network. This provides a rough approximation of the increase in cost acceptable as a result of a non-improving move.

pa = Initial probability of accepting a non-improving solution with a cost equal to the minimum cost + CostDiff. This probability is set as a random value between 0.3 and 0.4. Values of all other parameters were selected after experimenting with a few problem instances. These values are given in the following table.

Table 5-2 Simulated Annealing Parameters

Parameter	**Value**
Temperature reduction factor, a	0.985 – 0.995
Maximum number of iterations at each temperature, N_max	300
Maximum number of iterations for each type of move, N_move	1000
Stopping criterion- maximum number of iterations in the heuristic, Max_Iteration	125,000

Using these parameters, the final value of the temperature, when the stopping criterion is met, is generally close to 0. Although these guidelines were followed for most of the problem instances, slightly different parameter values provided better solutions for some instances during the experimentation phase, and hence were used for those instances.

5.4 Comparisons of the results

The Microsoft Visual Basic 6.0 programming language was used to code the heuristic algorithms developed. The software was run on a computer with central processor unit (CPU) clock speed of 2.99 MHz and a total of 1 gigabyte of random access memory (RAM) to solve the problems listed in Table 5-1 as well as few small test problems that are used for comparison with the mathematical programming model described in Chapter 3.

5.4.1 Comparison of results with the Mathematical Programming (MP) model

A few small problems were solved using the mathematical programming (MP) model described in Chapter 3 and the results were compared to the results obtained with the developed heuristic. Table 5-3 presents the results obtained on three instances.

Table 5-3 Comparison of Results with the MP model

No.	Number of nodes	Number of arcs	MP Model		Developed Heuristic		
			Optimal Solution	Time taken (seconds)	RCA Solution	SA Solution	Time taken (seconds)
1	9	11	73	8	98	73	0
2	8	13	55	280	81	55	0
3	9	12	81	2040	97	81	1

As shown in the table, the time required to run the MP model is extremely high even for very small problems. For example, in problem 3, which has 9 nodes and 12 arcs, the MP model obtained the optimal solution of 81 in approximately 34 minutes (~2040 seconds). For the same problem, the developed heuristic obtained the optimal solution in approximately 1 second.

5.4.2 Comparison of the results with problem instances in literature

In order to evaluate the performance of the heuristic developed, the results obtained for the gdb, val, and egl instances were compared to those obtained with the CARPET heuristic (developed by Hertz et al., 2000) and memetic algorithm, MA (developed by Lacomme et al., 2004). The results obtained by MA are the best known in literature for these problem instances. The results obtained for the mval instances were compared against solution values provided by Belenguer et al. (2006). CARPET is designed for undirected problems and was applied to gdb, val, and egl datasets and not the mval dataset (which has mixed networks). The results obtained by CARPET on these datasets were reported by Lacomme et al. (2004). The results of the heuristic developed and the results obtained by CARPET and MA were also compared against known lower bounds in order to illustrate the quality of the solutions obtained. According to Lacomme et al. (2004), the lower bounds for the gdb instances were obtained by Belenguer and Benavent (2003), except for gdb14, which was obtained by Amberg and Vob (2002). The lower bounds for all val and egl instances were reported by Belenguer and Benavent (2003). For the mval instances, the lower bounds were obtained by Belenguer et al. (2006).

Tables 5-4 through 5-6 show the results obtained for the datasets listed in Table 5-1. Columns (1) to (3) give the lower bound, results obtained by CARPET (2000), and the results obtained by MA, respectively. Column (4) gives the results obtained with the RCA, column (5) gives the results obtained with SA and column (6) gives the percentage improvement obtained by SA over the RCA results. The average deviation from the lower bound for a dataset is calculated as the percentage change in the sum of solution values obtained on all instances over the sum of the lower bounds on all instances in the dataset. The gray shaded cells denote instances in which the results of the developed heuristic match the results obtained by MA, and black shaded cells denote instances in which the results of the developed heuristic beat the results obtained by MA.

Table 5-4 presents the results for the gdb instances. The developed algorithm matched the best-known results in the literature for all 23 instances. The average deviation of results obtained from the best-known lower bounds was 0.21%. The average improvement of SA over the results obtained with RCA was found to be 7.76%, with a lowest improvement of 0.00% and a highest improvement of 17.18%.

The results obtained for the val instances are shown in Table 5-5. The developed algorithm matched the best-known results in the literature for 33 of the 34 instances. The average deviation of results obtained from the best-known lower bounds was 0.62%. In all cases, the SA improved the results obtained from the RCA and the average improvement was found to be 11.82%. The lowest and the highest improvement were found to be 4.42% and 17.86% respectively.

Table 5-6 presents the results obtained for the egl instances. The developed heuristic performed the best on these instances compared to the other heuristics. Out of the 24 instances, 15 of the best-known results obtained with MA (Lacomme et al., 2004) were improved, and on 5 more instances, the results were matched. The average deviation of the results obtained with SA from the best-known lower bounds was 2.61% as opposed to 2.69% deviation for the results obtained by MA. The improvement obtained with SA over RCA was in the range 11.92%-20.12%, and the average improvement was 15.59%.

The results obtained for the mval instances are shown in Table 5-7. The developed heuristic matched the best-known results on 28 of the 34 instances and beat the results on four other instances. The average deviation of the results obtained with SA from the best-known lower bounds was 0.50% compared to 0.55% deviation for the results obtained with

MA (Belenguer et al. 2006). The improvement obtained with SA over RCA was in the range 7.63%-24.42%, and the average improvement was 18.99%.

The results obtained with SA were similar to MA (best in the literature) for small problems (sets gdb & val), but better for larger problems (sets egl & mval), specially for egl which has the largest problems.

Table 5-4 Computational Results for the gdb Instances

	(1)	(2)	(3)	(4)	(5)	(6)
gdb instances	Lower Bound (LB)	CARPET (2000)	MA (2004)	RCA results	SA results*	% Improvement
gdb1	316	316	316	337	316	6.23%
gdb2	339	339	339	374	339	9.36%
gdb3	275	275	275	311	275	11.58%
gdb4	287	287	287	317	287	9.46%
gdb5	377	377	377	397	377	5.04%
gdb6	298	298	298	331	298	9.97%
gdb7	325	325	325	359	325	9.47%
gdb10	344	352	348	402	348	13.43%
gdb11	303	317	303	350	303	13.43%
gdb12	275	275	275	294	275	6.46%
gdb13	395	395	395	447	395	11.63%
gdb14	450	458	458	553	458	17.18%
gdb15	536	544	536	580	536	7.59%
gdb16	100	100	100	102	100	1.96%
gdb17	58	58	58	60	58	3.33%
gdb18	127	127	127	131	127	3.05%
gdb19	91	91	91	91	91	0.00%
gdb20	164	164	164	174	164	5.75%
gdb21	55	55	55	63	55	12.70%
gdb22	121	121	121	129	121	6.20%
gdb23	156	156	156	166	156	6.02%
gdb24	200	200	200	207	200	3.38%
gdb25	233	235	233	246	233	5.28%
TOTAL	5825	5865	5837		5837	
Average deviation from LB		0.69%	0.21%		0.21%	

*No of instances in which best-known solutions were improved by the developed heuristic: 0,
No of instances in which best-known solutions were matched by developed heuristic: 23

Table 5-5 Computational Results for the val Instances

	(1)	(2)	(3)	(4)	(5)	(6)
val instances	Lower Bound (LB)	CARPET (2000)	MA (2004)	RCA results	SA results*	% Improvement
val1a	173	173	173	181	173	4.42%
val1b	173	173	173	203	173	14.78%
val1c	235	245	245	260	245	5.77%
val2a	227	227	227	243	227	6.58%
val2b	259	260	259	277	259	6.50%
val2c	455	494	457	545	457	16.15%
val3a	81	81	81	87	81	6.90%
val3b	87	87	87	102	87	14.71%
val3c	137	138	138	168	138	17.86%
val4a	400	400	400	450	400	11.11%
val4b	412	412	412	473	412	12.90%
val4c	428	428	428	508	428	15.75%
val4d	520	530	530	636	530	16.67%
val5a	423	423	423	461	423	8.24%
val5b	446	446	446	488	446	8.61%
val5c	469	474	474	531	474	10.73%
val5d	571	581	581	668	581	13.02%
val6a	223	223	223	248	223	10.08%
val6b	231	233	233	267	233	12.73%
val6c	311	317	317	372	317	14.78%
val7a	279	279	279	319	279	12.54%
val7b	283	283	283	327	283	13.46%
val7c	333	334	334	382	334	12.57%
val8a	386	386	386	439	386	12.07%
val8b	395	395	395	449	395	12.03%
val8c	517	527	527	606	527	13.04%
val9a	323	323	323	368	323	12.23%
val9b	326	326	326	390	326	16.41%
val9c	332	332	332	370	332	10.27%
val9d	382	391	391	464	391	15.73%
val10a	428	428	428	469	428	8.74%
val10b	436	436	436	487	436	10.47%
val10c	446	446	446	503	446	11.33%
val10d	524	530	528	606	530	12.54%
TOTAL	11651	11761	11721		11723	
Average deviation from LB		0.94%	0.60%		0.62%	

*No of instances in which best-known solutions were improved by the developed heuristic: 0,
No of instances in which best-known solutions were matched by developed heuristic: 33

Table 5-6 Computational Results for the egl Instances

	(1)	(2)	(3)	(4)	(5)	(6)
egl instances	Lower Bound (LB)	CARPET (2000)	MA (2004)	RCA results	SA results*	% Improvement
egl-e1-a	3515	3625	3548	4110	3548	13.67%
egl-e1-b	4436	4532	4498	5136	4498	12.42%
egl-e1-c	5453	5663	5595	6430	5613	12.71%
egl-e2-a	4994	5233	5018	5849	5027	14.05%
egl-e2-b	6249	6422	6340	7539	6317	16.21%
egl-e2-c	8114	8603	8395	10361	8380	19.12%
egl-e3-a	5869	5907	5898	7365	5898	19.92%
egl-e3-b	7646	7921	7816	8985	7801	13.18%
egl-e3-c	10019	10805	10369	11735	10336	11.92%
egl-e4-a	6372	6489	6461	7994	6460	19.19%
egl-e4-b	8809	9216	9021	10339	9019	12.77%
egl-e4-c	11276	11824	11779	13879	11659	16.00%
egl-s1-a	4992	5149	5018	5978	5018	16.06%
egl-s1-b	6201	6641	6435	7460	6394	14.29%
egl-s1-c	8310	8687	8518	9961	8518	14.49%
egl-s2-a	9780	10373	9995	12029	9943	17.34%
egl-s2-b	12886	13495	13174	15815	13338	15.66%
egl-s2-c	16221	17121	16715	20455	16648	18.61%
egl-s3-a	10025	10541	10296	13000	10384	20.12%
egl-s3-b	13554	14291	14028	16201	13984	13.68%
egl-s3-c	16969	17789	17297	20649	17373	15.87%
egl-s4-a	12027	13036	12442	14936	12421	16.84%
egl-s4-b	15933	16924	16531	19310	16505	14.53%
egl-s4-c	20179	21486	20832	24558	20755	15.49%
TOTAL	229829	241773	236019		235837	
Average deviation from LB		5.20%	2.69%		2.61%	

*No of instances in which best-known solutions were improved by the developed heuristic: 14,
No of instances in which best-known solutions were matched by developed heuristic: 5

Table 5-7 Computational Results for the mval Instances

	(1)	(2)	(3)	(4)	(5)	(6)
mval instances	Lower Bound (LB)	CARPET (2000)	MA (2006)	RCA results	SA results*	% Improvement
mval1a	230	N/A	230	249	230	7.63%
mval1b	261	N/A	261	310	261	15.81%
mval1c	309	N/A	315	406	314	22.66%
mval2a	324	N/A	324	440	324	26.36%
mval2b	395	N/A	395	482	395	18.05%
mval2c	521	N/A	526	671	526	21.61%
mval3a	115	N/A	115	141	115	18.44%
mval3b	142	N/A	142	162	142	12.35%
mval3c	166	N/A	166	206	166	19.42%
mval4a	580	N/A	580	710	580	18.31%
mval4b	650	N/A	650	860	650	24.42%
mval4c	630	N/A	630	789	630	20.15%
mval4d	746	N/A	770	970	763	21.34%
mval5a	597	N/A	597	695	597	14.10%
mval5b	613	N/A	613	721	613	14.98%
mval5c	697	N/A	697	842	697	17.22%
mval5d	719	N/A	739	900	739	17.89%
mval6a	326	N/A	326	393	326	17.05%
mval6b	317	N/A	317	400	317	20.75%
mval6c	365	N/A	371	488	370	24.18%
mval7a	364	N/A	364	450	364	19.11%
mval7b	412	N/A	412	544	412	24.26%
mval7c	424	N/A	426	544	426	21.69%
mval8a	581	N/A	581	688	581	15.55%
mval8b	531	N/A	531	650	531	18.31%
mval8c	617	N/A	638	832	635	23.68%
mval9a	458	N/A	458	560	459	18.04%
mval9b	453	N/A	453	552	453	17.93%
mval9c	428	N/A	429	540	429	20.56%
mval9d	514	N/A	520	663	520	21.57%
mval10a	634	N/A	634	749	634	15.35%
mval10b	661	N/A	661	769	661	14.04%
mval10c	623	N/A	623	789	623	21.04%
mval10d	643	N/A	649	832	651	21.75%
TOTAL	16046		16143		16134	
Average deviation from LB			0.60%		0.55%	

*No of instances in which best-known solutions were improved by the developed heuristic: 4,
No of instances in which best-known solutions were matched by developed heuristic: 28

CHAPTER 6

6 Conclusion

This chapter provides a summary of the research performed, the conclusions obtained, and recommendations for future research.

6.1 Overview of Research

The objective of this research was to minimize the total distance required to service all the required arcs in a network with vehicles having limited servicing capacity. The goals of this research were:

1. To develop a mathematical model of the problem.
2. To develop a construction heuristic to generate initial feasible solutions to the problem, then use a simulated annealing algorithm to improve the initial solution
3. To compare the results obtained with the developed heuristic to the results provided by other authors on benchmark problems in literature

It is not practical to solve this problem optimally. A mathematical model was developed for the problem, and used to solve small problems. Even very small problems, with as few as 12 arcs, took more than 34 minutes to solve. A heuristic algorithm based on a multi-start meta-heuristic, and a Greedy Randomized Adaptive Search Procedure (GRASP) (Resende and Ribeiro, 2003) was developed to solve the problem. In each iteration of GRASP, an initial solution was built using a construction heuristic (RCA), and local search was performed on the solutions obtained using a simulated annealing (SA) based heuristic. The best overall solution obtained from several iterations of GRASP is considered as the final result.

RCA traces a route beginning at the depot node and incrementally adds an arc based on constraints and certain criteria at each iteration, until a route is completed. The final solution obtained using RCA is a set of routes that cover all the required arcs in the network. A SA based local search process is applied to this solution to improve it. Five types of moves that exchange arcs between pairs of routes are performed to evaluate the solution space in search of a better solution.

In order to test the proposed algorithm's performance, the developed heuristic was applied to publicly available test problems, and the results obtained were compared with the best-known solutions in literature. The tests were performed on 4 data sets (gdb, val, egl, and mval), having a total of 115 problem instances. The heuristic was able to improve the best-known solutions in 18 of those instances, and matched the best-known solutions in 89 other instances. The following table presents a summary of the results on the 4 datasets.

Table 6-1 Summary of Results

Dataset	No. of instances	No. of instances previous best-known solutions improved	No. of instances previous best-known solutions matched	Deviation from Lower Bound
gdb	23	0	23	0.00%-1.78%
val	34	0	33	0.00%-4.26%
egl	24	14	5	0.49%-3.59%
mval	34	4	28	0.00%-2.92%

6.2 Contribution to Literature

The contributions made by this research to the existing body of knowledge are as follows:

- A mathematical model was developed for the mixed network capacitated arc routing problem having arcs with multiple service requirements, and maximum distance and time limits on routes.
- Developed a heuristic algorithm based on GRASP; each iteration in GRASP consists of a construction heuristic followed by a simulated annealing heuristic to improve the solution of the construction heuristic.

6.3 Recommendations for Future Research

The following recommendations are made for future work:

- Other meta-heuristics such as Tabu Search, Genetic algorithms, Ants Colony, and combination of meta-heuristics (hybrid meta-heuristics) may be used for the search process and the results obtained can be compared with the results obtained in this research.
- Add extensions to the CARP such as different vehicle capacities, road priorities, multiple depots, intermediate facilities for refilling the vehicles, etc. and modify the

developed heuristic for those cases. For a discussion on practical applicability of the developed heuristic, please refer to Appendix C.

- Apply parallel processing implementation of the SA search process. This can be achieved by various ways as suggested by Aarts and Korst (1989); one of the ways they suggested was to allow multiple processors to proceed with annealing using different random numbers until the temperature is reduced. The best result from all the processors is chosen and the heuristic continues with the new temperature setting. Applying parallel computing can improve the effectiveness of the search process and significantly reduce the computation time.
- Add enhancements to the construction heuristic such as path relinking and bias functions. Instead of selecting a candidate randomly from the RCL for insertion into a route, a probability function can be introduced to bias the selection towards particular candidates.

  Path relinking is an intensification strategy for GRASP and was originally proposed by Laguna and Marti (1999). In path relinking, two solutions are selected from a set of high quality solutions obtained during previous search processes. A path is then generated from one solution to another using the fewest number of moves needed. Once the path is completed, one or more of the intermediate solutions are used to initiate a new search process.

REFERENCES

Aarts, E.H.L. and Korst, J. (1989), "Simulated Annealing and Boltzmann Machine: A Stochastic Approach to Combinatorial Optimization and Neural Computing", John Wiley and Sons, England

Amberg, A., Domshke, W. and Voss, S. (2000), "A Multiple Center Capacitated Arc Routing Problems: A Tabu Search Algorithm Using Capacitated Trees", European Journal of Operational Research 124, 360-378

Assad, A.A., Pearn, W.L. and Golden, B.L. (1987), "The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases", American Journal of Mathematics and Management Science 7, 63-88

Barr, R.S., B. L. Golden, J. P. Kelly, M.G.C. Resende, W. R. Stewart (1995), "Designing and Reporting on Computational Experiments with Heuristic Methods", J. Heuristics 1, 9-32.

Belenguer, J.M. and Benavent, E. (1998), "The Capacitated Arc Routing Problem: Valid Inequalities and Facets", Computational Optimization and Applications 10, 165-187

Belenguer, J.M. and Benavent, E. (2003), "A Cutting Plane Algorithm for the Capacitated Arc Routing Problem", Computers and Operations Research 30, 705-728

Belenguer, J.M., Benavent, E., Lacomme, P., Prins, C. and Ramdane-Cherif, W. (2006), "Heuristics and Lower Bound for the Mixed Capacitated Arc Routing Problem", Computers and Operations Research 33(12)

Benavent, E., Campos, V., Corberan, A. and Mota, E. (1990), "The Capacitated Arc Routing Problem. A Heuristic Algorithm", QUESTIIO 14, 107-122

Benavent, E., Campos, V., Corberan, A. and Mota, E. (1992), "The Capacitated Arc Routing Problem. Lower Bounds", Networks 22, 669-690

Beullens , P., Muyldermans, L., Cattrysse, D. and Oudheusden, D.V. (2003), "A Guided Local Search for the Capacitated Arc Routing Problem", European Journal of Operational Research 147, 629-643

Bodin, L. and Berman, L. (1979), "Routing and Scheduling of School Buses by Computer", Transportation Science 13, 113-129

Bodin, L. and Golden, B. (1981), "Classification in Vehicle Routing and Scheduling", Networks . 11, 97-108

Bodin, L. and Sexton, T. (1979), "The Subscriber Dial-a-Ride Problem", Final Report, U.S Department of Transportation, Urban Mass Transportation Administration, Washington D.C

Bodin, L., Golden, B.L., Assad, A. and Ball, M. (1983), "Routing and Scheduling of Vehicles and Crews: The State of the Art, Computers and Operations Research 10(2), 63-211

Bodin, L.D. and Kursh, S.J. (1978), "A Computer- Assisted System for the Routing and Scheduling of Street Papers", Operations Research 26 (4), 525-537

Bodin, L.D. and Kursh, S.J. (1979), "A Detailed Description of a Computer System for the Routing and Scheduling of Street Sweepers", Computers and Operations Research 6, 181-198

Campbell, J.F and Langevin, A. (1995), "Operations Management for Urban Snow Removal and Disposal", Transportation Research- A, 29(5), 359-370

Campbell, J.F and Langevin, A. (2000), "Roadway Snow and Ice Control" in Dror, M. (ed.), Arc Routing: Theory, Solutions, and Applications, Kluwer Academic Publishers 2000, 389-418

Chapleau, L., Ferland, and Rousseau, J.-M. (1985), "Clustering for Routing in Densely Populated Areas", European Journal of Operational Research 20, 48-57

Chapleau, L., Ferland, J.A., Lapalme, G., and Rousseau, J.-M. (1984), "A Parallel Insert Method for the Capacitated Arc Routing Problem", Operations Research Letters 3, 95-99

Christofides (1976), "The Vehicle Routing Problem" R.A.I.R.O., Recherche Operationelle 10, 55-70

Christofides, N. (1973), "The Optimum-Traversal of a Graph, Omega 1, 719-732

Christofides, N., Campos, V., Corberan, A. and Mota, E. (1981), "An Algorithm for the Rural Postman Problem", Imperial College Report, IC.O.R.81.5, London

Christofides, N., Campos, V., Corberan, A. and Mota, E. (1986), "An Algorithm for the Rural Postman Problem on a Directed Graph", Math. Prog. Study 26, 155-166

Christofides, N., Mingozzi, A., and Toth, P. (1981), "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations", Math. Prog., 255-282

Clarke, G. and Wright, J. (1964), "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", Operations Research 12, 568-581

Cook, T.M. and Alprin, B.S. (1976), "A Snow and Ice Removal in an Urban Environment", Management Science 23, 227-234

Corberan, A. and Sanchis, J.M. (1994), "A Polyhedral Approach to the Rural Postman Problem", European Journal of Operational Research 27, 183-203

Cortina, R.S., and Low, T.A. (2001), "Development of New Routes for Snow and Ice Control", Public Works 132 (9), August 2001, 20–23

Dantzig, G. and Ramser, J. (1959), "The Truck Dispatching Problem", Management Science 6, 81-91

DeArmon, J.S. (1981), "A Comparison of Heuristics for the Capacitated Chinese Postman Problem", Master's Thesis, The University of Maryland at College Park, Maryland

Dror, M. (2000) (Ed.), "Arc Routing: Theory, Solutions, and Applications", Kluwer Academic Publishers 2000

Dror, M., Stern, H., and Trudeau, P. (1987), "Postman Tour on a Graph with Precedence Relation on Arcs", Networks 17, 283-294

Edmonds, J. (1967), "Optimum Branching", J.Res. Natl. Bur. Stand., Section B. 71, 233-240

Eglese, R.W. (1994), "Routing Winter Gritting Vehicles" Discrete Applied Mathematics 48, 231-244

Eglese, R.W. and Li, L.Y.O. (1992), "Efficient Routing for Winter Gritting", Journal of Operational Research Society 43, 1031-1034

Eglese, R.W. and Murdock, H. (1991), "Routeing Road Sweepers in a Rural Area", Journal of Operational Research Society, 42(4), 281-288

Eiselt, H.A., Gendreau, M. and Laporte, G. (1995a), "Arc Routing Problems, Part I: The Chinese Postman Problem", Operations Research 43(2), 231-242

Eiselt, H.A., Gendreau, M. and Laporte, G. (1995b), "Arc Routing Problems, Part II: The Rural Postman Problem", Operations Research 43(3), 399-414

Euler, J. (J.R.Newman, Ed.) (1953), "Leonhard Euler and the Koenigsberg Bridges", Sci. Am. 189, 66-70

Euler, L. (1736), "Solution Problematis and Geometrian Situs Perinenis", Commentarii Academiae Scientarum Petropolitanae 8, 128-140

Evans, J.R. (1990), "Design and Implementation of a Vehicle Routing and Planning System for Snow and Ice Control" Working Paper, College of Business Administration, University of Cincinnati, Cincinnati, OH

Evans, J.R. and Weant, M. (1990), "Strategic Planning for Snow and Ice Control Vehicles using Computer-based Routing Software", Public Works 121(4), 60-64

Fisher, M.L. (1995), "Vehicle Routing", Handbooks in Operations Research and Management Science 8, Amsterdam, New York, Elsevier

Fisher, M.L. and Jaikumar, R.(1981), "A Generalized Assignment Heuristic for Vehicle Routing", Networks 11, 109-124

Fleischner, H. (1990), "Eulerian Graphs and Related Topics", Part 1, Vol. 1, Annals of Discrete Mathematics 45, North-Holland, Amsterdam

Frederickson G.N. (1979), "Approximation Algorithms for Some Routing Problems", J. Assoc. Comput. Math. 26, 538-554

Frederickson, G.N., Hecht, M.S., and Kim, C.E. (1978), "Approximation Algorithms for Some Rural Postman Problems", Siam Journal of Computing 7, 178-193

Gavett, J.W., "Three Heuristic Rules For Sequencing Jobs To A Single Production Facility", Management Science 11 (8), B-166-B-176

Gelinas, E. (1992), "Le probleme du postier chinois avec constraints generales de preseance, M.Sc.A. Dissertation, Ecole Polytechnique de Montreal, Canada

Gendreau, M., Hertz, A., Laporte, G. (1994), "A Tabu Search Heuristic for the Vehicle Routing Problem", Operations Research 40 (10), 1276-1290

Gendreau, M., Hertz, A., Laporte, G. (1992), "New Insertion and Post Optimization Procedures for the Traveling Salesman Problem", Operations Research 40 (6), 1086-1094

Gendreau, M., Laporte, G. and Yell, S. (1997), "Efficient Routing of Service Vehicles", Engineering Optimization 28, 263-271

Gilbert, J. (1989), "Generateur d'itineraires d' enlevement de la neige", Publication CRT-648, Centre for Research and Transportation, University of Montreal

Gillett, B and Johnson, T. (1976), "Multi-Terminal Vehicle Dispatch Algorithm", Omega 4, 711-718

Gillett, B and Miller, L. (1974), "A Heuristic Algorithm for the Vehicle Dispatch Problem", Operations Research 22, 340-349

Ghiani, G., Improta, G., and Laporte, G. (2001), "The Capacitated Arc Routing Problem with Intermediate Facilities", Networks 37(3), 134-143

Golden, B., Assad, A., Doyle, T. and Stewart, W. (1980), "Approximate Traveling Salesman Algorithms", Operations Research 28, 694-711

Golden, B.L. and Wong, R.T. (1981), "Capacitated Arc Routing Problems", Networks . 11, 305-315

Golden, B.L., DeArmon, J.S. and Baker, E.K. (1983). "Computational Experiments with Algorithms for a Class of Routing Problems." Computers and Operation Research 10(1), 47–59.

Goode, L. and Nantung, T. (1995), "CASPER: The friendly, efficient snow routes planner", TR News 181 (Nov-Dec 1995), 20-21

Gray, D.M. and Male, D.H. (1981), Handbook of Snow, Principles, Processes, Management, and Use, Pergammon Press, Toronto, Canada

Greistorfer P. (2003), "A Tabu Scatter Search Metaheuristic for the Arc Routing Problem", Computers and Industrial Engineering 44 (2), 249-266

Haghani, A. and Qiao, H. (2001), "Decision Support System for Snow Emergency Vehicle Routing", Transportation Research Record 1771, 172-178

Haghani, A. and Qiao, H. (2002), "Snow Emergency Vehicle Routing with Route Continuity Constraints", Transportation Research Record 1783, 119-124

Haslam, E. (1988), " The Application of Routing Technologies to the Problem of Snow Removal", Master's Thesis, Purdue University

Haslam, E. and Wright, J.R. (1991), "Application of Routing Technologies to Rural Snow and Ice Control", Transportation Research Record 1304, 202-211

Held, M and Karp, R. (1970), "The Traveling-Salesman Problem and Minimum Spanning Trees", Operations Research 18, 1138-1162

Held, M and Karp, R. (1971), "The Traveling-Salesman Problem and Minimum Spanning Trees, Part II", Mathematical Programming 1, 6-25

Henderson, D., Jacobson, S.H. and Johnson, A.W. (2003), Handbook of Metaheuristics, Glover, F. and Kochenberger, G.A. (Eds), Kluwer Academic Publishers

Hertz, A and Mittaz, M. (2001), "A Variable Neighborhood Descent Algorithm for the Undirected Capacitated Arc Routing Problem", Transportation Science 35(4), 425-434

Hertz, A., Laporte, G. and Mittaz, M. (2000), "A Tabu Search Heuristic for the Capacitated Arc Routing Problem", Operations Research 48(10), 129-135

Hertz, A., Laporte, G. and Nanchen Hugo, P. (1999), "Improvement Procedures for the Undirected Rural Postman Problem", INFORMS Journal on Computing 11(1), 53-62

Hierholzer, C. (1873), "Uber die Moglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren", Mathematische Annalen VI, 30-32

Kearney, A. (1980), "Improving Productivity in Physical Distribution", Report undertaken for CPDM, London

Keseling, J.R. (1994), "Maryland's Strategies for Fighting Winter Storms", Public Works, 40-42

Keyser, J.H. (1981), "Chemicals and Abrasives for Snow and Ice Control" in Gray, D.M. and Male, D.H. (Eds.), Handbook of Snow, Pergamon Press, Toronto, Canada, 580-612

Krolak, P., Felts, W. and Nelson, J. (1971), "A Man-Machine Approach Toward Solving the Traveling Salesman Problem", Comm. ACM 14, 327-334

Krolak, P., Felts, W. and Nelson, J. (1972), "A Man-Machine Approach Toward Solving the Generalized Truck Dispatching Problem", Transportation Science 6, 149-169

Lacomme, P., Prins,C. and Ramdane-Cherif, W. (2001a), "A Genetic Algorithm for the Capacitated Arc Routing Problem and its Extensions", in: Applications of Evolutionary Computing (E.J.W. Boers, Ed.), 473-483, Lecture Notes in Computer Science 2037, Springer

Lacomme, P., Prins,C. and Ramdane-Cherif, W. (2001b), "General Arc Routing Problems Solved by a Cutting Plane Algorithm and a Genetic Algorithm", IFAC (15th Triennial World Congress of the International Federation of Automatic Control), Barcelona, 2002

Lacomme, P., Prins,C. and Ramdane-Cherif, W. (2002), "Fast Algorithms for General Arc Routing Problems", IFORS 2002 Conference, Edinburgh, UK

Lacomme, P., Prins,C. and Ramdane-Cherif, W. (2004), "Competitive Memetic Algorithms for Arc Routing Problems", Annals of Operations Research 121, 159-185

Laguna, M. and Marti, R. (1999), "GRASP and Path Relinking for 2-layer Straight Line Crossing Minimization", INFORMS Journal on Computing 11, 44-52

Lawler, E.L. (1976), Combinatorial Optimization: Networks and Matroids, New York, Rinehart and Winston.

Lemieux, P.F. and Campagna, L. (1984), "The Snow Plowing Problem Solved by a Graph Theory Algorithm", Civil Engineering Systems 1, 337-341

Lenstra, J.K. and Rinnooy Kan, A.H.G. (1976), "On General Routing Problems", Networks 6, 272-280

Levy, L. and Bodin, L.D. (1989), "The Arc Oriented Location Routing Problem", INFOR 27, 74-94

Li, L.Y.O (1992), "Vehicle Routing for Winter Gritting", Ph.D. Dissertation, Department of OR and OM, Lancaster University

Li, L.Y.O and Eglese, R.W. (1996), "An Interactive Algorithm for Vehicle Routing for Winter-Gritting", Journal of Operational Research Society 47, 217-228

Liebling, T.M. (1970), "Graphentheorie in Planungs-und Tourenprobleme", Lecture Notes in Operations Research and Mathematical Systems 21, Springer, Berlin

Liebling, T.M. (1973), "Routing Problems for Street Cleaning and Snow Removal", in Deininger, R. (ed.), Models for Environmental Pollution Control, 363-374

Lin, S. (1965), "Computer Solutions of the Traveling Salesman Problem", Bell Syst. Tech. J. 44, 2245-2269

Lin, S. and Kernighan, B. (1973), "An Effective Heuristic Algorithm for the Traveling Salesman Problem", Operations Research 21, 498-516

Lindsey, R.K. and Seely, M.S. (1999), "Resource Allocation Study For Snow Removal", Transportation Research Record 1672, 23-27

Lotan, T.D, Cattrysee, D., Oudheusden, D.V., and Leuven, K.U. (1996), "Winter Gritting in the Province of Antwerp: A Combined Location and Routing Problem", Belgian Journal of Operations Research 36, 141-157

Magnanti, T.L. (1981), "Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects", Networks 11, 179-213

Male, J.W., Liebman, J.C., and Orloff, C.S. (1977), "An Improvement of Orloff's General Routing Problem", Networks 7, 89-92

Marks, H.D. and Stricker, R. (1971), "Routing for Public Service Vehicles", ASCE Journal of the Urban Planning and Development Division 97, 165-178

McBride, R. (1982), "Controlling Left and U-turns in the Routing of Refuse Collection Vehicles", Computers and Operations Research 9, 145-152

Metropolis, N., Rosenbluth, A., Teller, M., Teller, E. (1953), "Equation of State Calculations by Fast Computing Machines", Journal of Chemical Physics 21, 1087-1092

Mei-Ko Kwan (1962), "Graphic Programming using Odd and Even Points", Chinese Math. 1, 273-277

Minsk, L.D (1979), "A Systems Study of Snow Removal", US Army Cold Regions Research and Engineering Lab, 220-226

Minsk, L.D. (1970), "A Short History of Man's Attempts to Move Through Snow", in Snow Removal and Ice Control Research, Special Report 115, Proceedings of an International Symposium held April 8-10, 1970, Highway Research Board, National Academy of Sciences, Washington D.C. 1970, 1-7

Mourao, M.C. and Amado, L. (2005), "Heuristic Method for a Mixed Capacitated Arc Routing Problem: A Refuse Collection Application", European Journal of Operational Research 160, 139-153

Newton, R. and Thomas, W. (1974), "Bus Routing in a Multi-School System", Computers and Operations Research 1, 213-222

Orloff, C. (1974), "A Fundamental Problem in Vehicle Routing", Networks 4, 35-64

Pearn, W.L. Assad, A., and Golden, B.L. (1987) "Transforming arc routing into node routing problems", Computers and Operations Research 14(4), 285-288

Pearn, W.L. (1988), "New Lower Bounds for the Capacitated Arc Routing Problem", Computers and Operations Research 16, 589-600

Pearn, W.L. (1989), "Approximate Solutions for the Capacitated Arc Routing Problem", Computers and Operations Research 16, 589-600

Pearn, W.L. (1991), "Augment-Insert Algorithms for the Capacitated Arc Routing Problem", Computers and Operations Research 18, 189-198

Resende, M.G.C. and Ribeiro, C.C. (2003), "Greedy Randomized Adaptive Search Procedures", State of the Art Handbook in Meta-heuristics , Glover, F. and Kochenberger, G.A. (Eds.), Kluwer Academic Publishers, 287-319

Russell, G.L. and Sorenson, H.K. (1979), "A Value Engineering Study of Snow and Ice Control", Transportation Research Board Special Report No. 185, 66-73

Sanchis, J.M. (1990), "El Poliedro del Problemo del Cartero Rural", Ph.D. Thesis, Universidad de Valencia, Spain.

Stewart, W. and Golden, B. (1979), "A Vehicle Routing Algorithm Bases on Generalized Lagrange Multipliers", Proceedings of the AIDS 1970 Annual Convention, L. Moore, K. Monroe, B.Taylor, Eds., New Orleans, LA, Vol. 2, 108-110

Tan K.C., Lee L.H., Zhu Q.L.and Ou, K. (2001), "Heuristic methods for vehicle routing problem with time windows", Artificial Intelligence in Engineering 15, 281-295

Tucker, W.B. and Clohan, G.M. (1979), "Computer Simulation of Urban Snow Removal", Transportation Research Board Special Research Report Number 185, 293-302

Ulusoy, G. (1985), "The Fleet Size and Mix Problem for Capacitated Arc Routing". European Journal of Operational Research 22, 329-337

Wang, J.-Y. (1992), "Computer Aided System for Planning Efficient Routes", Ph.D. Dissertation, Purdue University

Wang, J-Y. and Wright, J.R. (1994), " Interactive Design of Service Routes", Journal of Transportation Engineering 120 (6), 897-913

Wang, Y., Kandula, P. and Wright, J.R. (1995), "Evaluation of Computer-Generated Routes for Improved Snow and Ice Control", Transportation Research Record 1509, 15-21

Win, Z. (1987), "Contributions to the Routing Problems", Doctoral Dissertation, Universitai Augsburg, Germany

Wisconsin Department of Transportation report in Better Roads Magazine (October 2003): http://www.betterroads.com/articles/oct03d.htm

Wright, J.R. (1993), "The Computer Aided System for Planning Efficient Routes", Joint Highway Research Project Draft Final Report FHWA/IN/JHRP-93-8, Purdue University, West Lafayette, Indiana.

APPENDIX A

Mathematical Programming Code for the Model

```
TITLE
    Arc_routing_pb

INDEX
    i:=1..9
    j:=1..9
    p:=1..3
    r:=1..9

DATA

    c[i,j]:=SPARSEFILE("Final_DIST.dat");  !Distance values stored in a DAT file (shown on next page)
    n[i,j]:=SPARSEFILE("Final_nij.dat"); !Requirement values stored in a DAT file (shown on next page)
    t[i,j]:=1/25 * SPARSEFILE("Final_DIST.dat");
    g[i,j]:=1/37.5 * SPARSEFILE("Final_DIST.dat");


DECISION VARIABLE
X[i,j,p]
WHERE (c[i,j] >0);
l[i,j,p]
WHERE (c[i,j] >0);
f[i,j,p]
WHERE (c[i,j] >0);

MODEL
    Min Z =SUM(i,j,p: c[i,j]*X[i,j,p]);

SUBJECT TO

    C1[i,p]:
    SUM(r: X[i:=r,j:=i,p]) - SUM(r: X[i,j:=r,p]) =0;

    C2[i,j] where (n[i,j] < 1):
        SUM (p: l[i,j,p]) + sum(p: l[i:=j,j:= i,p]) = 1;

    C3[i,j] where (n[i,j] >= 1):
        SUM(p: l[i,j,p]) = n[i,j];

    C4 [p]:
        SUM(i,j: l[i,j,p] *  c[i,j]) <= 18;

    C5[p]:
        SUM(i,j:t[i,j]*X[i,j,p])+SUM(i,j:g[i,j]*l[i,j,p]) <= 2

    C6[i,j,p]:
        X[i,j,p] >= l[i,j,p];

    C7[p]:
        SUM(i,j: X[i,j,p] * c[i,j]) <= 30;
    C8[i,p] Where i<>1:
```

```
        SUM(r:f[i,j:=r,p])- SUM(r:f[i:=r,j:=i,p])= SUM(j:l[i,j,p]);

    C9[i,j,p]:
        f[i,j,p] <= sqr(count(i)) * X[i,j,p];

    C10[i,j,p]:
        f[i,j,p] >= 0;

    C11[i,j,p]:
        l[i,j,p] >= 0;

    C12[i,j,p]:
        X[i,j,p] >= 0;

    BINARY
    X[i,j,p], l[i,j,p];

END
```

Content of the file "Final_DIST.dat" (having distance values from node i to j):

```
1,2,2,
2,1,2,
1,9,3,
9,1,3,
1,8,2,
8,1,2,
2,3,3,
3,2,3,
3,4,2,
3,5,3,
5,3,3,
3,9,6,
4,6,5,
6,4,5,
4,7,4,
7,4,4,
5,6,4,
6,5,4,
5,9,6,
9,5,6,
7,8,7,
8,7,7
```

Content of the file "Final_nij.dat" (having requirement values for arc (i,j)):

```
1,2,0.5,
1,9,0.5,
1,8,0.5,
2,3,0.5,
3,4,1,
3,5,0.5,
3,9,1,
4,6,0.5,
4,7,0.5,
5,6,1,
5,9,0.5,
7,8,0.5,
6,5,1
```

APPENDIX B

Floyd's Algorithm

Floyd's Algorithm is designed to compute the shortest distances between all nodes in a graph. It has a complexity of $O(n^3)$. The input for the algorithm is the initial distance matrix (D_{ij}^{0}), in which each row represents the starting node (i) and each column represents an ending node (j). Each cell in the matrix denotes the distance between the starting node and the ending node if there is an arc between the starting and ending node. If there is no arc between the starting node (i) and the ending node, a very large value is used in position (i,j) in the matrix.

The pseudo-code for Floyd's Algorithm is shown below. Let n be the total number of nodes in the graph.

Algorithm Floyd (Input: Initial Distance Matrix- D_{ij}^{0})

    For k=1 to n

        For i=1 to n

            For j=1 to n

                D_{ij}^{k} = Minimum [D_{ij}^{k-1}, D_{ik}^{k-1} + D_{kj}^{k-1}]

    Return matrix D_{ij}^{n} with shortest distances.

End

The path corresponding to the shortest distance between nodes i and j is the shortest path between those nodes.

APPENDIX C

Practical Applications of the Developed Heuristic

The results obtained by the developed heuristic prove that the heuristic can be applied to practical problems. It can be adapted for use in networks having multiple depots, intermediate facilities for refilling, non-homogeneous vehicle capacities, road priorities and other practical features. The heuristic can be applied for real life problems having operational constraints such as one-way streets, restricted junctions, different costs for servicing, and traveling without servicing. The heuristic may also be integrated with a GIS system to provide a visual representation of the routes. Typically, benefits obtained with a routing software include improved fleet efficiency, enhanced service, even the distribution of the workload, and reduction in fuel costs.

The price of a route optimization package can quickly pay for itself even in a small fleet. In addition to gasoline savings achieved by the reduced mileage, other costs such as labor costs due to nonproductive time and fixed cost of trucks can also be reduced (by reducing number of routes). A cost/benefit analysis may be performed to compare the estimated savings to be realized from having better routings of the vehicles, against the investment made in purchasing a route optimization software.