Algorithms for vehicle routing problem with pickup and delivery

Algorithms for vehicle routing problem with pickup and delivery

By

Yuvraj Gajpal

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

© Copyright by Yuvraj Gajpal, May, 2008

Doctor of Philosophy (2008) McMaster University

(Management Science) Hamilton, Ontario

TITLE: Algorithms for vehicle routing problem with pickup and delivery

AUTHOR: Yuvraj Gajpal, M.Tech. (Indian Institute of Technology Madras, India), B.E.(Government Engineering College Raipur, India)

SUPERVISOR: Professor Prakash Abad

NUMBER OF PAGES: xvii, 224

ABSTRACT

In this thesis, we have considered the vehicle routing problem with pickup and delivery which is a generalization of the capacitated vehicle routing problem (CVRP). The vehicle routing problem with pickup and delivery (VRPPD) arises whenever pickup demand and delivery demand is to be satisfied by the same vehicle. The problem is encountered in many real life situations including reverse logistics. We consider three variants of VRPPD, namely, the vehicle routing problem with back-hauls (VRPB), the vehicle routing problem with back-hauls and mixed-load (VRPBM) and the vehicle routing problem with simultaneous pickup and delivery (VRPSPD).

The inherent complexity of VRPPD makes it an $\mathcal{NP}$-hard problem. It is not possible to solve an $\mathcal{NP}$-hard problem in polynomial time unless $\mathcal{P} = \mathcal{NP}$. Therefore, heuristics and metaheuristics are used to produce a good quality solution within reasonable CPU time. We develop ant colony algorithms for VRPB, VRPBM and VRPSPD. We have improved the existing ant-colony algorithms by applying better local search schemes and by adding new features such as construction rule and the trail updating criteria. We also develop saving based heuristics for single and multi-depot versions of VRPSPD. Checking feasibility of a given route is an important issue in VRPSPD because of the fluctuating load on the vehicle. We have proposed the cumulative net-pickup approach for this purpose. One important feature of this approach is that it checks the feasibility of an altered route in constant time.

The proposed heuristics and metaheuristics are evaluated by solving benchmark problem instances available in literature and then comparing the solutions with the solutions

produced by the existing algorithms. Our computational experiment has shown that the proposed heuristics and metaheuristics give better or equally good results in comparison to the existing solution procedures.

Dedicated

to

My Parents

Smt. Poona Gajpal and Shri Aghanu Ram Gajpal

and

The people of

my village Parsatthi & Mujagahan

ACKNOWLEDGEMENTS

I received help from many individuals during the course of this thesis. First and foremost, I sincerely thank my supervisor Professor Prakash Abad, for his guidance, constant encouragement, countless discussion and for timely feedback. His confidence on my abilities and his patience have been the essential factors to the success of this research.

I express my gratitude to my supervisory committee members, Dr. George Steiner and Dr. John Miltenburg whose discussions and comments have helped me tremendously in developing this thesis.

I wish to thank my fellow students and my friends for making my days enjoyable at McMaster.

Finally, I would like to thank my wife Neha and my family members for their love and moral support during the course of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACS	Ant colony system	11
ACO	Ant colony optimization	8
ALT	the heuristic algorithm of Salhi et al., 1999	168
ARPD	Average Relative percentage deviation	59
BTS:	New tabu search of Brandao, 2006	58
CVRP	Capacitated vehicle routing problem	1
DA	Deterministic annealing	8
FPTAS	Fully polynomial time approximation scheme	5
GA	Genetic algorithm	8
GSA	Generalized saving algorithm	162
HA	The hybrid algorithm of Crispim et al., 2005	133
HeuSDP	The algorithm of Chen et al., 2006	133
ITP	Iterated tour partitioning	6
LNS	Large neighborhood search by Ropke, S. and D. Pisinger (2006)	58
LC-INS	the heuristic algorithm of Salhi et al., 1999	168
MACS	Multi ant colony system	15
NBA	Nearest depot based algorithm	191
PBA	Partition based algorithm	181
PTAS	Polynomial time approximation scheme	5
QITP	Quadratic iterated tour partitioning	6

RCRS	The heuristic algorithm of Dethloff, 2001	168
RPD	Relative percentage deviation	59
RTS-AMP:	Reactive tabu adaptive memory programming of Wassan, 2004	21
SA	Simulated annealing	8
TA	Tillman, 1969 algorithm based on saving algorithm for multi-depot VRP	190
TS	Tabu search	8
TSP	Traveling salesman problem	6
UITP	Unequal iterated tour partitioning	6
VRP	Vehicle routing problem	1
VRPB	Vehicle routing problem with back-hauls	3
VRPBM	Vehicle routing problem with back-hauls and mixed-load	3
VRPPD	Vehicle routing problem with pickup and delivery	2
VRPSPD	Vehicle routing problem with simultaneous pickup and delivery	4

CHAPTER 1

Introduction

One of the important problems in logistics and supply chain management is the routing of a set of vehicles that move goods from a warehouse to retailers and/or customers. Transportation is a key element in supply chain management because goods are rarely produced and consumed at the same place. The growth of e-commerce and the increased trade liberalization has fueled business across the globe and has made the world economies increasingly interdependent. The transportation cost in a distribution network contributes typically 10% to 20% of the final cost of goods [Toth and Vigo (2001)] and more than 45% of the total logistic costs [Osman and Laporte (1996)]. In order to control the transportation cost, the dispatcher must route and schedule vehicles to serve customer demands in an efficient way. The underlying combinatorial problem is often called vehicle routing problem (VRP). Given its inherent complexity, VRP has received attention from many researchers, academics, and practitioners in the last 50 years.

VRP is encountered in industries such as courier services, supply of industrial goods, supply of food and beverages, public transport, urban solid waste collection, etc. In real word applications of VRP in North America, it has been shown that the efficient planning and scheduling of vehicles can reduce the transportation cost by 5% to 20% [Toth and Vigo (2001)].

The most basic vehicle routing problem is the classical capacitated vehicle routing problem (CVRP). Under CVRP a set of customers are served by a fleet of identical

vehicles each having limited capacity. Each vehicle starts from the depot/warehouse and after serving a set of customers it returns back to the depot/warehouse. CVRP entails determining the route for each vehicle so that the total tour length (i.e., the total distance traveled by all vehicles) is minimized. The total demand of any vehicle must not exceed the vehicle capacity. Also, the total route duration (travel time and service time) for a vehicle may not exceed the specified upper bound. In this thesis, we use the abbreviations CVRP and VRP interchangeably.

The classical CVRP arises in supply chain management when the focus is mainly on the forward flow of material. In recent years, environmental regulations and the increased incentives for returning and reusing products have led to reverse flows of materials. The management of return flows has opened a new field in supply chain management called reverse logistics. Reverse logistics generates additional revenue by recapturing the value otherwise lost or underutilized in the supply chain.

The design of supply chain for forward and reverse logistics is often carried out independently. There is thus a need for integrating the forward and reverse flows in a supply chain. The vehicle planner and dispatcher in reverse logistics need to consider the delivery and pickup of products from a customer by a single vehicle. The vehicle routing problem with pickup and delivery (VRPPD) arises in many real life situations whenever the pickup demand and the delivery demand is to be satisfied by the same vehicle. In VRPPD, delivery demand of a customer is fulfilled by the product originating from the depot and all pickups are dropped back to the depot. In this thesis, we will focus upon the vehicle routing problem with pickup and delivery.

1.1 Classification of vehicle routing problem with pickup and delivery

The vehicle routing problem with pickup and delivery (VRPPD) is an $\mathcal{NP}$-hard problem because when we set all pickup or all delivery amounts to zero, the problem becomes the CVRP, which is a known $\mathcal{NP}$-hard problem. See Papadimitriou and Steiglitz (1998) to know more about $\mathcal{NP}$-hard problem. The vehicle routing problem with pickup and delivery (VRPPD) can be classified in three categories: (i) vehicle routing problem with backhauls; (ii) vehicle routing problem with backhauls and mixed-load; (iii) vehicle routing problem with simultaneous pickup and delivery.

Vehicle routing problem with backhauls (VRPB) consists of two groups of customers: linehaul or delivery customers and backhaul or pickup customers. A linehaul customer requires a given quantity of product to be delivered while a backhaul customer requires a given quantity of product to be picked up. In VRPB, all deliveries for linehaul customers must be made before any pickup from a backhaul customer. This restriction is imposed when arranging goods is difficult if goods are picked up before all deliveries are made. The objective is to design a set of minimum cost routes ensuring that the total delivery demand of linehaul customers and the total pickup demand of backhaul customers served by the vehicle does not exceed vehicle capacity.

Similar to VRPB, vehicle routing problem with backhauls and mixed-load (VRPBM) consists of linehaul and backhaul customers. However, the restriction of serving all linehaul customers before serving any backhaul customer is relaxed. In other words deliveries and pickups are allowed in any order of sequence in VRPBM. One consequence of relaxing the sequencing restriction is that there is fluctuating load on the

vehicle during the trip. Thus in VRPBM, the objective is to design a set of minimum cost routes so that the capacity of the vehicle is not violated anywhere in the route.

In vehicle routing problem with simultaneous pickup and delivery (VRPSPD), each customer is a linehaul as well as a backhaul customer. In VRPSPD each customer not only requires a given quantity of products to be delivered but also requires a given quantity of products to be picked up. A complete service (i.e., delivery and pickup) to the customer is provided by a vehicle in a single visit. One reason for simultaneous pickup and delivery can be the high handling cost. The objective is to design a set of minimum cost routes so that the load on the vehicle is below its capacity anywhere in the route. VRPBM is a special case of VRPSPD because when we set either delivery or pickup demand of each customer to zero, VRPSPD reduces to VRPBM.

1.2 Solution Method

Solution methods for an $\mathcal{NP}$-hard problem can be divided in four categories. We give a brief introduction to the four methods in the next subsections.

1.2.1 Exact methods

An exact method refers to a method that computes the optimal solution to the problem if there is sufficient computer time. Let Q be the vehicle capacity. VRP can be solved exactly in polynomial time when $Q = 2$ by transforming it to a minimum weight matching problem. However, the problem is $\mathcal{NP}$-hard when $Q \geq 3$ (see Bompadre et al. (2006)). It is not possible to solve an $\mathcal{NP}$-hard problem exactly in polynomial time unless $\mathcal{P} = \mathcal{NP}$ (Garey and Johnson (1979)).

The branch and bound is the widely used exact algorithm for solving VRP (e.g., Laporte et al. (1986), Ralphs (2003) and Fukasawa et al. (2006)). Dynamic programming (e.g., Magnanti (1981) and Kolen et al. (1987)) and Lagrangean relaxation procedures (e.g., Magnanti (1981) and Stewart and Golden (1984)) are other exact approaches that have been used to solve VRP and its variants.

A branch and bound method for VRPB has been designed by Toth and Vigo (1997) and Mingozzi et al. (1999). They used the branch-and-bound algorithm to solve the problem exactly for up to 40 customers. Dell'Amico et al. (2006) developed a branch-and-price approach for VRPSPD. They used the approach to solve the problem exactly for up to 40 customers.

1.2.2 Approximation Algorithm

It is not possible to solve an $\mathcal{NP}$-hard problem in polynomial time unless $\mathcal{P} = \mathcal{NP}$. A polynomial time approximation algorithm can find a feasible solution in polynomial time for the problem whose value is within factor α of the unknown optimal solution ($\alpha \geq 1$ for minimization problem and $\alpha \leq 1$ for maximization problem). The factor α is called the performance guarantee/ratio or the approximation ratio of the algorithm. An approximation algorithm is evaluated by the worst case over all possible instances of the problem. There are two classes of approximation algorithms which return a solution within the desired precision: *polynomial time approximation scheme* (PTAS) and *fully polynomial time approximation scheme* (FPTAS). PTAS is defined as a family of polynomial time approximation algorithms A_ε, $\varepsilon > 0$, such that algorithm A_ε provides a

solution with performance guarantee of $\alpha = 1 + \varepsilon$. The running time of the PTAS depends upon ε and is polynomial in the input size of the problem. This means that the error can be as small as possible but computational time will increase with decreasing error. If the running time of algorithm A_ε is bounded by a polynomial in the input size and by $1/\varepsilon$, then it is called a *fully polynomial time approximation scheme* (FPTAS). FPTAS is stronger than PTAS because FPTAS $\subseteq$ PTAS. A strong $\mathcal{NP}$-hard problem can not have an FPTAS unless $\mathcal{P}=\mathcal{NP}$ (Garey and Johnson (1979)).

The first PTAS for the capacitated vehicle routing problem with equal demand (ECVRP) was proposed by Haimovich and Kan (1985) and later modified by Altinkemer and Gavish (1990). The iterated tour partitioning (ITP) heuristic of Altinkemer and Gavish (1990) can produce a solution at most $1+(1-1/Q)\alpha$ times the optimal solution, assuming that the approximation ratio of the traveling salesman tour is $1+\alpha$. Here Q is the vehicle capacity. Altinkemer and Gavish (1987) proposed unequal iterated tour partitioning (UITP) heuristic for CVRP. UITP has a performance guarantee of $2+(1-2/Q)\alpha$, given that Q is even. The worst case bounds of ITP and UITP heuristics depend on the approximation ratio of the input TSP tour. Recently Bompadre et al. (2006) have presented the quadratic iterated tour partitioning (QITP) heuristic which improves the approximation ratio of ITP for ECVRP and of UITP for CVRP. They were able to reduce the performance ratio by amount $b(\alpha,Q)$, where $b(\alpha,Q)$ is a function of α and vehicle capacity Q. In our knowledge, there are no approximation algorithms for other variants of VRP. Given the complexity, it is very difficult to design an approximation algorithm for a variant of VRP.

1.2.3 Heuristic

A heuristic is an algorithm that provides a good quality feasible solution within reasonable computer time but there is no guarantee on finding the optimal solution. One may able to design a special instance for which the heuristic provides a poor solution and/or runs slowly. Since the special instances usually are not likely in real life, the heuristic may work well in real life applications. Laporte and Semet (2001) classify heuristics for VRP in three classes. In the first class, the heuristic is characterized by a constructive method that iteratively builds vehicle routes until all customers are served by one of the vehicles (Clarke and Wright (1964), Mole and Jameson (1976), Nelson et al. (1985), Paessens (1988) and Altinkemer and Gavish (1991)). In the second class the heuristic is characterized by a two-step method. A two-step method uses either route-first-cluster-second strategy or cluster-first-route-second strategy. In a route first-cluster second strategy, customers are first ordered to form a TSP (i.e., traveling salesman problem) tour and then this order is partitioned to produce feasible routes for individual vehicles (Beasley (1983), Haimovich and Kan (1985)). In a cluster first-route second strategy, the group of customers to be served by a vehicle is identified as a cluster and then an efficient route is designed for the cluster (Gillett and Miller (1974), Fisher and Jaikumar (1981)). In the third class, the heuristic uses improvement schemes based upon local search techniques (Stewart and Golden (1984)).

1.2.4 Metaheuristics

Metaheuristcs are widely used for solving vehicle routing problems in practice. A metaheuristic is a general algorithmic framework which explores the solution space in an

intelligent way. According to Dorigo (2004), a metaheuristic can be seen as a general-purpose heuristic method designed to guide a problem-specific heuristic. A metaheuristic is a general algorithmic framework, which can be applied to different optimization problems. There are five types of metaheuristics applied to VRP: Simulated Annealing (SA), Deterministic Annealing (DA), Tabu search (TS), Genetic Algorithm (GA) and Ant colony optimization (ACO). These metaheuristics are briefly described below.

Simulated annealing

Simulated annealing was introduced by Kirkpatrick et al. (1983). Simulated annealing is a randomized improvement algorithm. As its name implies, Simulated Annealing (SA) exploits the analogy between the way metal cools and freezes into the minimum energy crystalline structure (the annealing process) and the search for the minimum solution to a combinatorial problem. Solids are annealed by raising the temperature level at which particles randomly arrange in liquid phase and then are cooled so that the particles fall into a low-energy state of the regular lattice. At high temperatures, all possible crystalline states are attainable with some probability. Lowering the temperature decreases the number of accessible states and puts the system into its ground state.

In a combinatorial problem, each configuration has a cost level associated with it. Similar to the annealing of a solid, one can statistically model the evolution of the system into the state that corresponds to the minimum value of the cost function. SA's major advantage over other heuristic methods is its ability to avoid a local minimum. The algorithm employs random search, which may accept an increase in the cost. The inferior

solutions are accepted with probability $p = \exp(-\delta f / T)$ where δf the increase in the objective function value and T is is a control parameter representing the system temperature. The search thus avoids the local minimum by jumping out of it early in the computation. Towards the end of the computation, the temperature is low and the probability of accepting an inferior solution is nearly zero. The slower the cooling, the higher the chance of finding the optimal solution, but the longer the run time. Thus effective use of this technique depends on finding a cooling schedule that finds good solutions without using excessive CPU time.

Deterministic Annealing

Deterministic Annealing (DA) is similar to simulated annealing but DA uses a deterministic rule instead of a probabilistic rule to accept the new solution. There are two types of deterministic rules used in DA: threshold accepting and record-to-travel. Threshold accepting leaves out the stochastic element in accepting an inferior solution by introducing a deterministic threshold, T_h. Let the current solution at iteration t be x_t and the solution at the next iteration be x_{t+1}. The inferior solution is accepted if $f(x_{t+1}) < f(x_t) + T_h$. Threshold parameter T_h is gradually lowered similar to lowering the temperature in SA. Record-to-travel is similar to threshold accepting but here an inferior solution is accepted if $f(x_{t+1}) < T_h * f(x_t)$. See Golden et al. (1998) for an application of DA to VRP.

Tabu search

The basic concept of tabu search was proposed by Glover (1986) and Hansen (1986) independently. Tabu search is a neighborhood search method which uses memory to avoid cycling and thus escape a local minimum. In each iteration it moves from the current solution x_t to a new solution x_{t+1} by searching the neighborhood $N(x_t)$ of x_t. In moving from one iteration to the next, tabu search looks for the neighborhood solution that improves the solution most. If there is no improved solution in the neighborhood, it chooses the solution, which gives the least deterioration. To avoid visiting the same solution again or in a cyclic manner, recently visited solutions are forbidden or discarded. The set of forbidden (tabu) solutions is temporarily stored in the list called "tabu list". Usually it is not possible to store a complete solution in the list, hence only some attributes of the solution are stored. The duration for which a particular solution is stored in the tabu list is called tabu-tenure. Tabu tenure can be varied over time.

Other sophisticated features used in tabu search are aspiration criteria, diversification approach and intensification procedure. An aspiration criterion is used to override the status of an attribute stored in the tabu list. Intensification is used to intensify the search in the region that is likely to contain acceptable solutions. Intensification is carried out by giving a higher priority to solutions that share common features with the current solution. In contrast to intensification, diversification is used to direct the search towards a region not yet explored. The diversification and intensification are used alternately during the search.

Genetic Algorithm

Genetic Algorithm (GA) was first introduced by Holland (1975). GA is a population-based algorithm that simulates the evolutionary process observed in nature. Unlike other metaheuristics, GA explores a population of solutions to produce a better solution. Individual solutions which constitute the population are called chromosomes. GA uses the survival of the fittest chromosome to select different solutions in the population. The good solutions are used to reproduce new solutions and form a pool of better solutions, while the bad solutions are discarded. The basic steps of GA are described below:

1. Select Mates: Select two solutions called parent chromosomes from the population either randomly or by using a fitness performance criterion.
2. Recombine parents: Each selected pair is combined using a crossover operator to produce two new solutions called offsprings or child chromosomes.
3. Mutation: Few solutions are randomly selected and a small random modification is carried out in each solution to avoid the stagnation of the population.
4. New population: The new population is created by replacing some of the solutions in the current population with the new solutions produced during crossover.

Ant colony system

Ant colony system (ACS) is a memory based algorithmic approach derived from the foraging behavior of real ants. ACS algorithm makes use of simple agents called ants which construct solutions to the combinatorial optimization problem. The generation (or construction) of a solution by an ant is guided by the artificial pheromone trails and by

the problem-specific information. An individual ant constructs a complete solution by starting with the null solution and by iteratively adding solution components. After the construction of the solution, the ant gives feedback by depositing pheromone (i.e., updating trail intensity) for each solution component. Typically, solution components that are part of better solutions or solution components that have been used by ants over many iterations receive higher amounts of pheromone. These solution components in turn are more likely to be used in future iterations. ACS uses evaporation factor or persistence of trail in updating the trail intensity. See Stuetzle and Hoos (2000) for details on the application of ACS to combinatorial optimization problems. Bullnheimer et al. (1999) were the first to construct an ACS algorithm for VRP. The algorithm is further improved by Doerner et al. (2002) by combining it with the saving heuristic. Basic steps of the ant colony system are as follows.

Step 1. Initialize the trail intensities and parameters and generate an initial solution either randomly or using the problem specific heuristic.

Step 2. While (the termination condition is not met) do the following:

- Generate an ant-solution for each ant using the trail intensities.
- Improve each ant-solution by carrying out a local search.
- Update the trail intensities.

Step 3. Return the best solution found so far.

1.3 Motivation

In this thesis, we will apply ant colony optimization metaheuristic to the vehicle routing problem with pickup and delivery (VRPPD). Some real word applications of VRPPD are:

1. Grocery Industry where supermarkets and shops are demand customers and grocery suppliers or production sites are pick up customers.
2. Bottles from a blood bank are taken to the hospitals and blood collected from blood donation camps is taken to the blood bank.
3. Soft drink industry where new bottles have to be delivered and empty bottles have to be picked up.
4. In foundries, the sand used to form moulds usually contains toxic materials. The sand needs to be recycled. Collecting used sand is serving a pickup customer and delivering purified sand is serving a demand customer.

Apart from the above applications, VRPPD is also possible in industries in which dairy products, agriculture items, pharmaceuticals goods, dry cleaning supplies, electronic appliances, hardware items, etc. are distributed (Min et al. (1992)).

Technology has affected the logistics function more than any other business function. The increased use of computers and information technology requires the logistics manager to take decisions in a short span of time. Given the complexity, generally it is not possible to solve VRPPD optimally in reasonable CPU time. A simple heuristic procedure may produce a solution in reasonable CPU time but without adequate quality. A metaheuristic can generate a solution with sufficient quality. The only

drawback of a metaheuristic is the increased CPU time. However, the recent evolution of IT has made it possible to produce high quality solutions within reasonable CPU time using a metaheuristic. It is not surprising that most commercial software for solving VRP problems are based on metaheuristics. Metaheuristics provide a general framework which can handle a family of problems. In order to make the general framework work for a particular problem, one has to design problem specific routines such as feasibility check, local search technique, etc.

Tabu search (TS) is a widely used metaheuristic approach for solving vehicle routing problems. To date, it has proven to be the best approach for some variants of vehicle routing problems. There are many applications of tabu search to CVRP. Each uses a different move to generate a neighborhood solution (e.g., Osman (1993), Taillard (1993), Gendreau et al. (1994), Rochat and Taillard (1995b), Xu and Kelly (1996) and Rego (2001)). Tabu search has also been used to solve variants of VRP (e.g., Osman and Wassan (2002), Wassan (2004), Brandao (2006), Crispim and Brandao (2005), Chen and Wu (2006) and Montane and Galvao (2005)).

In recent times, attempts have been made to solve CVRP using ant colony system (e.g., Bullnheimer et al. (1999), Doerner et al. (2002), Reimann et al. (2002) and Reimann et al. (2004)). Although the solution quality reported in the above studies is not as good as other metaheuristics, ACS seems to have potential. Gendreau et al. (2001) consider the results obtained by ACS to be quite encouraging given the limited applications of ACS to VRP. The local search is an important part of an ant colony

system and we believe that it has not been fully exploited yet. An ant colony system can be improved by applying better local search techniques and by adding newer features.

1.4 Overview of the thesis

The rest of the thesis consists of seven chapters. Chapter 2 presents a survey of literature on vehicle routing problem with pickup and delivery (VRPPD). Chapters 3 to 7 are the main body of this thesis. Each of these chapters is focused on a specific topic and/or the methodology used to solve the particular version of the pickup and delivery problem. In each chapter, we start with a brief introduction of the problem. We then describe the methodology used to solve the problem, the numerical results, and a concluding summary for the chapter.

Chapter 3 presents the multi-ant colony system (MACS) for solving VRPB. Ant colony system (ACS) is an algorithmic approach inspired by the foraging behavior of real ants. Artificial ants are used to construct a solution by using pheromone information generated in the previous iterations. In MACS, we use two types of ants to construct a solution. The first type of ant is used to assign customers to vehicles and the second type of ant is used to construct a route for a vehicle given the assigned customers; i.e., to solve the underlying traveling salesman problem. An interesting property of MACS is the correlation between the CPU time and the solution quality. An extensive numerical experiment is performed on benchmark problem instances available in the literature.

Chapter 4 presents cumulative net-pickup approach for checking the feasibility of a given route in VRPSPD. This chapter also presents two multi-route local search schemes for VRPSPD. The existing approach to check feasibility of a given route while performing

the local search requires linear time. The cumulative net-pickup approach does the check in constant time. The local search schemes and the cumulative net-pickup approach presented in this chapter are used in an ant-colony system (ACS) algorithm presented in chapter 5. We provide several lemmas associated with the cumulative net-pickup approach for each local search scheme.

The proposed ACS algorithm in chapter 5 uses a construction rule as well as two multi-route local search schemes described in chapter 4. The ACS algorithm can also solve the vehicle routing problem with backhaul and mixed-load (VRPBM). An extensive numerical experiment is performed on benchmark problem instances available in literature.

In Chapter 6, we present saving based heuristics for VRPSPD. The saving heuristic of Clarke and Wright (1964) has been successfully used in solving CVRP and its variants. The parallel saving heuristic of Altinkemer and Gavish (1991) for CVRP combines the saving approach with a matching based procedure. Although the saving heuristic of Clarke and Wright (1964) is used widely to solve CVRP, no such heuristics exist for VRPSPD. In this Chapter, we fill this gap and design saving based as well as parallel saving heuristics for VRPSPD. The heuristics involve creating a new route by merging two existing routes. We have used the cumulative net-pickup approach for checking the feasibility of such a route.

Chapter 7 presents saving based heuristics for the multi-depot version of VRPSPD. We developed four algorithms for the problem. These algorithms are as follows: 1) partition based algorithm, 2) nearest depot algorithm, 3) saving algorithm and

4) Tillman's saving algorithm. In the partition based algorithm, customers are divided into borderline and non-borderline customers. Non-borderline customers are first assigned to their nearest depot and the problem is then solved as a single depot VRPSPD for each depot. Borderline customers are then inserted into the vehicle routes. In the nearest depot algorithm, customers are first assigned to their nearest depot. Then a single depot VRPSPD is solved for each depot using the saving based heuristic described in chapter 6. In the saving algorithm, we modify the saving algorithm of Clarke and Wright (1964) to make it suitable for multi-depot VRPSPD. In Tillman's saving algorithm, we extend the saving algorithm of Tillman (1969) designed for the multi-depot capacitated vehicle routing problem (CVRP) to the multi-depot version of VRPSPD.

In chapter 8, we present the conclusions of this thesis and highlight the scope for future work.

CHAPTER 2

Literature Survey

We survey the literature in six parts. The first part is the capacitated vehicle routing problem (CVRP). The second part is the vehicle routing problem with backhaul (VRPB) where all linehaul customers are served before serving a backhaul customer. The third part is the vehicle routing problem with backhaul and mixed load (VRPBM) where the linehaul and backhaul customers are served in any sequence. The fourth part is the vehicle routing problem with simultaneous pickup and delivery (VRPSPD) in which a customer requires some quantity of materials to be delivered as well as a load to be picked up during the same visit. The fifth part is the multi-depot version of the capacitated vehicle routing problem where vehicles are situated in more than one depot. The sixth part is the multi-depot version of the vehicle routing problem with simultaneous pickup and delivery.

2.1 Capacitated Vehicle Routing Problem (CVRP)

The Capacitated vehicle routing problem (CVRP) is an $\mathcal{NP}$-hard problem and polynomial time algorithms for finding the optimal solution are unlikely to exist. Hence exact solution techniques (see e.g., Christofides et al. (1981), Laporte et al. (1985), Agarwal et al. (1989)) are limited to solving small size problem in terms of the number of customers.

Limited success with exact algorithms has led to approximate and heuristic solution procedures. The heuristic solutions described in the literature on VRP can be categorized in three classes. The first class of heuristic solutions is characterized by a constructive method that iteratively builds vehicle routes until all customers are served by one of the vehicles (Clarke and Wright (1964); Mole and Jameson (1976); Nelson et al. (1985); Paessens (1988); Altinkemer and Gavish (1991)). The second class of heuristic solutions is characterized by two-step methods. The two-step methods use either a route first-cluster second method or a cluster first-route second method. In a route first-cluster second method, the n customers are first ordered to form a TSP (i.e., traveling salesman problem) tour and then this order is partitioned to produce the feasible routes for individual vehicles (Beasley (1983); Haimovich and Kan (1985)). In a cluster first-route second method, the group of customers to be served by each vehicle is identified to form a cluster and then an efficient route is designed for each cluster (Gillet and Miller (1976); Fisher and Jaikumar (1981)). The third class uses improvement schemes based upon the local search technique which improves the given solution (Stewart and Golden (1984)).

The exact algorithms have limitations in solving large size problems whereas simple greedy heuristic approaches have limitations in terms of solution quality. A metaheuristic attempts to improve upon exact as well as simple heuristics by relaxing the computing time restriction. A metaheuristic procedure is a guided local search technique which explores neighborhood solutions in a systematic way. The search should overcome local optimality. A metaheuristic may take longer CPU time but would produce a good quality solution. Tabu search (TS) is a widely used metaheuristic procedure in vehicle

routing problem and until now it has proven to be the best approach for many variants of vehicle routing problems. There are numerous attempts in literature on successful application of Tabu search to CVRP each one using different type of moves to generate neighborhood solutions (see Osman (1993); Taillard (1993); Gendreau et al. (1994); Rochat and Taillard (1995a); Xu and Kelly (1996); Rego (2001)).

In recent times, attempts have been made to solve CVRP by using ant colony system (ACS). For an introduction to ACS approach see Dorigo et al. ((1996)). Bullenheimer, Hartl and Strauss ((1999)) designed for the first time an ant colony system for the vehicle routing problem. This ant colony system is further improved by Doerner et al. ((2002)) by combining it with a problem based savings heuristic. Recently Reimann et al. ((2004)) proposed decomposition based approach (D-Ants) for CVRP. In the D-Ants system, the problem is decomposed into several small sub-problems. Each sub-problem is then solved using the saving based ant colony system.

2.2 Vehicle Routing Problem with Backhauls (VRPB)

Many exact and heuristic methods are proposed in literature to solve VRPB with the objective of minimizing the total distance traveled by the vehicles. Goetschalckx and Jacobsblecha (1989) propose a two phase approach for VRPB. In the first phase, an initial solution is generated based upon space filling curves for linehaul and backhaul customers. In the second phase, these solutions are merged to get a final set of vehicle routes. This algorithm is not able to handle the case where the number of available vehicles is fixed. Goetschalckx and Jacobsblecha (1993) extend the generalized assignment heuristic of VRP to VRPB. Toth and Vigo (1996) developed a "cluster first

and route second" algorithm based upon the generalized assignment approach of VRP. Further, Toth and Vigo (1999) proposed a new cluster method which exploits the information associated with the lower bound based on Lagrangian relaxation. This heuristic is used for both the symmetric as well as asymmetric versions of the problem (in the symmetric version of the problem, the distance from customer 1 to customer 2 is the same as distance from customer 2 to customer 1 whereas in asymmetric version, the two distances are not equal). Apart from the above, few studies have investigated mixed service and other variations of VRPB (e.g., Thangiah et al. (1996), Anily (1996), Wade and Salhi (2002)).

Yano et al. (1987) developed an exact procedure for VRPB using the set covering approach based upon Lagrangian relaxation and branch and bound framework. They applied the procedure to the routing problem for retail chain stores where the number of customers served was small. Toth and Vigo (1997) and Mingozzi et al. (1999) used a branch and bound algorithm to solve the problem exactly for up to 100 customers. Toth and Vigo (1997) used Lagrangian relaxation of the underlying integer programming formulation to obtain a lower bound whereas Mingozzi et al. (1999) used a heuristic to obtain a lower bound in their branch and bound algorithm.

Recently several attempts have been made to solve VRPB using Tabu search. Osman and Wassan (2002) proposed Reactive Tabu Search (RTS) for VRPB. Further Wassan (2004) combined adaptive memory programming with tabu search and proposed Reactive Tabu Adaptive Memory Programming search (RTS-AMP). Both tabu search heuristics use λ–interchange of customers as proposed by Osman (1993). RTS-AMP

maintains a set of solutions called elitist solutions and uses these solutions to turn the direction of search towards an unexplored region of the solution space. Recently, Brandao (2006) has proposed a new tabu search algorithm where the initial solution is obtained from a pseudo lower bound based upon the *K-tree* approach. Other attempts of metaheuristics are based on different variants of VRPB (e.g. Potvin et al. (1996), Thangiah et al. (1996), Duhamel et al. (1997), Reimann et al. (2002), Wade and Salhi (2004)). Reimann and Ulrich (2006) designed an ant colony algorithm for the vehicle routing problem with backhauls and time windows. Although there are several studies on using tabu search in VRPB, to our knowledge, there have been no published studies on using the ant colony system in VRPB in which all linehaul customers are to be served before any backhaul customers.

2.3 Vehicle Routing Problem with Backhauls and mixed load (VRPBM)

There are relatively few papers which deal with VRPBM. Deif and Bodin (1984) extended the classical saving heuristic of Clarke and Wright to solve VRPBM. They calculate the saving between the linehaul and backhaul customer by adding a penalty term to postpone the insertion of the backhaul customer in the route. Casco et al. (1988) combined the saving method with load based insertion method to solve VRPBM. Golden et al. (1985), Casco et al. (1988) and Salhi and Nagy (1999) combined Clarke and Wright saving heuristic with an insertion based heuristic. These papers first solve CVRP consisting only of the linehaul customers and then insert backhaul customers in the route. They differ in the way a backhaul customer is inserted in the existing route. Golden et al. (1985) consider the number of linehaul customers left on the route after the insertion of a

backhaul customer whereas Casco et al. (1988) consider the remaining load to be delivered after the insertion of a backhaul customer. Salhi and Nagy (1999) adopted a similar approach but they insert two backhaul customers at a time in the route. Mosheiov (1998) proposed a heuristic based upon the tour partitioning approach. Wade and Salhi (2004) designed an ant colony system algorithm for VRPBM. Recently Bianchessi and Righini (2007) have proposed a tabu search algorithm using complex and variable neighborhood approach.

2.4 Vehicle routing problem with simultaneous pickup and delivery (VRPSPD)

The problem of VRPSPD was first defined by Min (1989) when he formulated a real life problem of transporting books between libraries. He used a two-phase approach to solve the problem. In the first phase, customers are clustered into groups in such a way that the total delivery demand or total pickup load for each cluster is below vehicle capacity Q. In the second phase, a traveling salesman problem (TSP) is solved exactly and if the TSP route violates the vehicle capacity constraint then the TSP route is resolved by penalizing the arcs that violate the vehicle capacity constraint. Salhi and Nagy (1999) used the saving and insertion based heuristic to solve VRPSPD. They first solve the CVRP problem consisting only the linehaul customers and then insert the backhaul customers in the route. Dethloff (2001) was first to highlight the application of VRPSPD to reverse logistics. He proposed heuristics based on three different insertion criteria: 1) the extra travel distance if a customer is inserted between two customers, 2) the residual capacity which represents the extra amount that can be delivered or picked up after the insertion of a given customer, 3) the surcharge due to the late and unfavorable

insertion of a remotely located customer. Finally, Nagy and Salhi (2005) have developed a heuristic that treats pickups and deliveries in an integrated manner. First, the problem is solved as a CVRP in such a way that the total delivery demand or the total pickup amount is less than the vehicle capacity. Hence the initial solution is weakly feasible. In a *weakly feasible solution*, the load of vehicle may not be necessarily below its capacity at each point in the route however, the total delivery demand or the total pickup amount is less than the vehicle capacity Q. In a *strongly feasible solution*, the load on vehicle is below its capacity at each point in the route. The initial weakly feasible solution is then modified to make it strongly feasible which in turn is improved using different routines. Recent attempts to solve VRPSPD are based upon metaheuristic approaches (e.g., Crispim and Brandao (2005), Chen and Wu (2006), Montane and Galvao (2005) , and Bianchessi and Righini (2007)). An exact method based upon the branch and price method is proposed by Dell'Amico et al. (2006).

2.5 Multi depot version of Capacitated Vehicle Routing Problem (CVRP)

The first heuristic for multi-depot CVRP was proposed by Tillman (1969) using the saving algorithm of Clarke and Wright (1964). The heuristic constructs an initial solution by assigning each customer to the nearest depot and by using a separate truck to serve each customer. Two customers are combined and are served by a single truck only if they are assigned to the same depot. Tillman and Hering (1971) modified this heuristic further by incorporating a look-ahead criterion which considers some additional options before selecting the customer and assigning him permanently to a particular depot. Tillman and Cain (1972) present an upper bound algorithm where the criteria for joining

two routes is the maximum saving. Golden et al. (1977) improve the algorithm of Tillman and Cain (1972) by providing an efficient data structure and by adding a penalty function. Wren and Holliday (1972) develop an algorithm which allows construction of multiple routes starting from several depots simultaneously. A customer is added iteratively to an existing route or placed on a new route to build an initial solution. The initial solution is then refined using several improvement heuristics. Gillet and Miller (1976) solve multi-depot VRP in two stages. In the first stage each customer is assigned to a depot. In the second stage, each sub problem is solved as a single-depot problem using the sweep algorithm of Gillett and Miller (1974). Raft (1982) uses a five-phase approach for the problem. The five phases are: 1) the route assignment, 2) the depot assignment, 3) the vehicle assignment, 4) the delivery period and 5) the detailed route design period. Each phase is considered and solved separately and then combined using an iterative procedure.

Cassidy and Bennett (1972) solve a real life catering service problem which is closely related to multi-depot CVRP. They consider the problem of transporting meals between different schools in the Inner London Education Authority. An exact branch and bound algorithm for multi-depot CVRP is proposed by Laporte et al. (1984) for the symmetric case and by Laporte et al. (1988) for the asymmetric case. Laporte et al. (1984) were able to solve a problem with up to 40 customers while Laporte et al. (1988) were able to solve a problem with up to 80 customers. Chao et al. (1993) use record-to-record metaheuristic of Dueck (1993) to solve multi-depot CVRP. Renaud et al. (1996) and Cordeau et al. (1997) designed a tabu search procedure for multi-depot CVRP.

Thangiah (2001) presents a generalized clustering method based on genetic algorithm for solving multi-depot CVRP.

2.6 Multi depot version of vehicle routing problem with simultaneous pickup and delivery

Multi-depot VRPSPD has not received much attention in literature. Salhi and Nagy (1999) and Nagy and Salhi (2005) extend the heuristic developed for single-depot VRPSPD to multi-depot VRPSPD. To our knowledge, there is no independent study in the literature for multi-depot VRPSPD although multi-depot VRPB has been studied by Min et al. (1992). Min et al. (1992) developed a heuristic solution by decomposing the problem in three phases: 1) forming clusters of customers, 2) assigning a cluster to a depot and 3) building a TSP route for each cluster and the assigned depot.

CHAPTER 3

Multi-ant colony system (MACS) for a vehicle routing problem with backhauls

3.1 Introduction

In vehicle routing problem with backhaul (VRPB), there are two groups of customers: linehaul or delivery customers and backhaul or pickup customers. A linehaul customer requires a given quantity of product to be delivered while a backhaul customer requires a given quantity of product to be picked up. The problem can be defined using graph theory as follows. Consider a complete undirected graph $G = (N, A)$ where $N = \{0\} \cup L \cup B$ is a set of $l+b+1$ vertices and subsets $L = \{1,\ldots\ldots,l\}$ and B = $\{l+1, l+2, \ldots\ldots, l+b\}$ correspond to the linehaul and backhaul customers, respectively. Set $A = \{(i, j), i, j \in N\}$ is a set of arcs. In VRPB, arc (i, j) represents the distance between the customer i and the customer j. Associated with arc (i, j), there is a nonnegative cost c_{ij} and associated with vertex i there is a non-negative quantity d_i. Vertex 0 represents the depot (with fictitious demand 0) where v identical vehicles, each with capacity Q are located. In order to make the problem feasible, it is assumed that the number of available vehicles $v \geq \max(v_L, v_B)$, where v_L and v_B are the minimum number of vehicles needed to serve all linehaul and all backhaul customers, separately. The values of v_L and v_B can be calculated by solving the Bin Packing Problems (BPP) associated with the subsets of linehaul and backhaul customers. Although BPP is an $\mathcal{NP}$-hard problem in strong

sense, it can be solved optimally within reasonable CPU time even when the problem involves hundreds of items/customers (see Silvano and Paolo (1990)).

In this chapter, we do not deal with mixed service; i.e., we assume that each vehicle is unloaded first by satisfying the demands of linehaul customers and later it is loaded by collecting the loads from backhaul customers. Also a route consisting entirely of backhaul customers generally is not allowed (Toth and Vigo (2001)). If this restriction is not imposed, we may get a solution where all linehaul customers are served by one set of vehicles and all backhaul customers are served by another set of vehicles. In VRPB, the objective is to design a set of minimum cost routes so that the total load of linehaul or the total load of backhaul customers does not exceed the vehicle capacity on each route. VRPB is an $\mathcal{NP}$-hard problem in a strong sense since by setting $B = \phi$, the problem becomes CVRP, which is known to be an $\mathcal{NP}$-hard problem.

3.2 Ant Colony System for the CVRP

An ACS algorithm makes use of simple agents called *ants* that iteratively construct solutions to the combinatorial optimization problem. The generation (or construction) of a solution by an ant is guided by (artificial) pheromone trails and problem-specific heuristic information. An individual ant constructs a complete solution by starting with the null solution and by iteratively adding solution components. After the construction of the solution, each ant gives feedback by depositing pheromone (i.e., updating trail intensity) for each solution component. Typically, solution components that are part of better solutions or solution components that are used by ants over many iterations receive higher amounts of pheromone. These solution components in turn are

more likely to be used in future iterations. This is achieved by using the 'evaporation factor' or 'persistence of trail' while updating the trail intensities. See Stuetzle and Hoos (2000) for details on an application of ACS to combinatorial optimization problems. Bullnheimer et al. (1999) have first constructed an ACS algorithm for VRP. The algorithm is further improved by Doerner et al. (2002) by combining it with the saving heuristic. Basic steps of the saving based ant colony system are

Step1: Initialize the trail intensities and parameters.

Step2: While (termination condition is not met) do the following:

- Generate an ant-solution for each ant using the trail intensities.
- Improve each ant-solution by carrying out local search.
- Update trail intensities.

Step3: Return the best solution found so far.

The main component of this algorithm, step 2, is described in detail below.

3.2.1 Generation of a solution by an ant

An artificial ant constructs a solution by successively serving customers until all customers are served. The choice of the next customer is affected by two factors: 1) the trail intensity which stores the information on the selection of this customer in previous iterations, 2) the saving value s_{ij} which represents the saving (in terms of the distance traveled) when customers i and j are served jointly by one vehicle instead of two separate vehicles. The next customer to visit is chosen from m best candidates in terms of the attractiveness value ξ_{ij} which is calculated as

$$\xi_{ij} = [\tau_{ij}]^{\alpha}[s_{ij}]^{\beta}. \quad (3.1)$$

Here, τ_{ij} is the trail intensity of visiting customer j immediately after customer i and parameters α and β determine the relative biases for the trail intensity and the saving value, respectively. Let Ω_m denote the set of m best feasible customers based upon the attractiveness value. Then the next customer to be served is generated randomly using the following *random proportional rule*:

$$p_{ij} = \begin{cases} \dfrac{\xi_{ij}}{\sum_{l \in \Omega_m} \xi_{il}} & \text{if } j \in \Omega_m \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

A complete solution is built by choosing customers one by one without exceeding the capacity of the vehicle. Once a vehicle is filled, another one is used. The construction process stops when all customers are served. The *random proportional rule* used to select the next customer is explained below with a numerical example.

Suppose set Ω_m consists of three customers 1, 2 and 3. Let the probability of visiting customer 1 after customer i be 0.2, i.e., $p_{i1} = 0.20$. Similarly let $p_{i2} = 0.30$ and $p_{i3} = 0.50$. The cumulative probabilities for visiting customers 1, 2 and 3 are 0.20, 0.50 and 1, respectively. We generate a uniform random number u in the range of [0, 1] and use the following table to select the next customer.

$$\text{Next Customer} = \begin{cases} 1 & \text{if} \quad 0 < u \leq 0.2 \\ 2 & \text{if} \quad 0.2 < u \leq 0.5 \\ 3 & \text{if} \quad 0.5 < u \leq 1 \end{cases}$$

3.2.2 Local search

After a solution is constructed by an ant, the solution is improved using the 2-*opt* local search scheme. This scheme was originally proposed by Lin (1965) for improving the route in a traveling salesman problem (TSP). The 2-*opt* scheme is one of the best known local search scheme for TSP and it is based on the arc exchange approach. In CVRP, each vehicle route is a traveling salesman problem. The 2-*opt* scheme starts with a given route and breaks it at two places to generate three segments. The route is then reconstructed by reversing the middle segment (the segment which does not contain the depot). A given route is broken at each combination of two places and is updated whenever there is an improvement. The process is repeated until there is no further improvement in the solution (i.e., until there is no improvement in the solution in one complete cycle of the search). In VRPB, we would need to treat the linehaul and backhaul customers separately while using the 2-*opt* scheme.

3.2.3 Updating trail intensities

After all ants construct their solutions, trail intensities are updated using the solutions of γ elitist ants. Elitist ants are defined as σ best ant-solutions found so far. Given the γ elitist ant-solutions, the trail intensity of visiting customer j immediately after customer i is updated as follows:

$$\tau_{ij}^{new} = \rho \tau_{ij}^{old} + \sum_{\mu=1}^{\gamma-1} \Delta\tau_{ij}^{\mu} + \gamma \Delta\tau_{ij}^{*} \qquad i = 1,2,.....,n; \quad j = 1,2,...,n; \quad i \neq j. \tag{3.3}$$

Here ρ is the trail persistence (with $0 < \rho < 1$). The first term in (3.3) represents the information carried out from previous iterations. The second term is the deposition of pheromones laid by γ -1 elitist ants excluding the best elitist ant and is defined as:

$$\Delta\tau_{ij}^{\mu} = \begin{cases} (\gamma - \mu)/L_{\mu} & \text{if arc}(i, j) \text{ is traversed by the } \mu^{th} \text{ elitist ant} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

Here L_{μ} is the total tour length of the μ^{th} elitist ant. The third term in (3.4) is the deposition of pheromone by the best elitist ant. This term is defined as:

$$\Delta\tau_{ij}^{*} = \begin{cases} 1/L^{*} & \text{if arc}(i, j) \text{ is traversed by the best elitist ant} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

Here L^{*} is the total tour length in the best elitist ant solution found so far.

3.3 Multi-ant colony system (MACS) for VRPB

The ant colony system for CVRP described in section 3.2 is not suitable when the number of vehicles is fixed. It can generate an infeasible solution in terms of the number of vehicles required. We use multi-ant colony system (MACS) to deal with the infeasibility. The concept of multiple types of ants was first proposed by Gambardella et al. (1999) who designed an ant colony system to solve the vehicle routing problem with time windows. They used two types of ants to minimize the two different objective functions: the first type of ant minimizes the number of vehicles used while the second type of ant

minimizes the total tour length. Our multi-ant approach is inspired by "cluster first and route second" procedure of VRP. We also use two types of ants to construct a solution. The first type of ant is used to assign customers to vehicles and the second type of ant is used to construct a route for a vehicle given the assigned customers; i.e., to solve the underlying traveling salesman problem. An outline of the algorithm is given below and a more detailed description of the algorithm is provided in the next sub-sections.

Step1: Initialize the trail intensities and parameters by generating an initial solution. Set COUNT = 1.

Step2: While COUNT ≤ 5 do the following:

- Generate an ant-solution for each ant using the trail intensities.
- Improve each ant-solution by carrying out the following local search:
  1. Use 2-opt local search once
  2. Use the following two local search schemes until there is no further improvement:
     i. Customer insertion/interchange multi-route scheme,
     ii. Sub-path exchange multi-route scheme.
- Update elitist ants.
- If there is no change in the elitist ant solutions in the last 10 iterations,

  Then

  Reinitialize the trail intensity and destroy all elitist ant solutions. Increase COUNT by 1, i.e., set COUNT = COUNT + 1.

  Else

Update trail intensities.

Step3: Return the best solution found so far.

In MACS, we have added the feature of reinitialization of trail intensities and destroying the elitist ant solutions. This feature is similar to the re-start feature of Genetic Algorithm used by Prins (2004). When the solution quality varies significantly from run to run, the restart feature is helpful. Instead of executing the algorithm just once for a long time period, it is better to re-start the algorithm multiple times and execute it for short time periods.

3.3.1 The initial solution and the reinitialization of trail intensities

The trail intensities are initialized using an initial solution, which is generated using a two-phase procedure. In the first phase, customers are randomly assigned one by one to v vehicles ignoring the capacity restriction. In the second phase, a route is constructed for each vehicle by iteratively adding the assigned customers. The initial solution plays an important role in many metaheuristic approaches such as simulated annealing (Johnson et al. (1989)). However, the initial solution does not play a significant role in ant colony system because the ant-solutions generated in each iteration are almost independent of the initial solution. Hence, we avoid using an elaborate time-consuming procedure for constructing a feasible initial solution.

We use two types of ants: *vehicle-ants* and *route-ants*. The trail intensity of a *vehicle-ant* $\tau 1_{ik}$, represents the intensity of customer i being served by vehicle k. The trail intensity of a *route-ant* $\tau 2_{ij}$, is the intensity of visiting customer j immediately after i. Suppose L is

the objective function value (i.e., the total tour length) in the initial solution. Initially, we set $\tau 1_{ik} = 1/L$; $i = 1,\ldots,n$; k = 1,...,v. and $\tau 2_{ij} = 1/L$; $i = 1,\ldots,n$; $j = 1,\ldots,n$; $i \neq j$.

In step 2 of the algorithm, trail intensities are reinitialized 5 times. We use the objective function value of the best solution found to date to reinitialize the trail intensities. Thus, during re-initialization all $\tau 1_{ik}$ and $\tau 2_{ij}$ are set to $1/L_{best}$, where L_{best} is the objective function value (i.e., the total tour length) in the best solution found to date.

3.3.2 Generation of a solution by an ant

The construction process used in ACS described in section 2 for solving CVRP can produce an infeasible solution in terms of the number of vehicles used. In MACS, we keep the number of vehicles fixed. We use a two phase approach based upon *vehicle-ants* and *route-ants*. The approach is described in detail below.

Assigning customers to vehicles

In this phase, we assign customers to vehicles insuring that the capacity of each vehicle is not exceeded. Let

l	total number of linehaul customers.
b	total number of backhaul customers.
$n = l+b$	total number of customers which includes linehaul and backhaul customers.
Ω	the set of customers not yet served.
C_k	the set of customers to be served by vehicle k.
v	the number of available vehicles.
Q	capacity of each vehicle.

AC_k^L available capacity of vehicle k for assigning a linehaul customer.

AC_k^B available capacity of vehicle k for assigning a backhaul customer.

Note that as stated in section 3.1, v is known and its value is such that feasible solutions exist. The procedure given below should return a feasible solution as long as such a solution exists.

The steps of assigning customers to vehicles are

Step1: Initialize the sets and the available capacities:

$\Omega = \{1,2, 3, \ldots,n\}$,

$C_k = \phi$; $k = 1,2, \ldots,v$,

$AC_k^L = AC_k^B = Q$; $k = 1,2\ldots,v$.

Step2: Randomly choose a customer, say customer i, who is not yet assigned to a vehicle. Remove customer i from set Ω.

Step3: Let Ψ be the set of feasible vehicles. A vehicle is feasible for assigning customer i if available capacity AC_k^L is greater than or equal to the demand for customer i and i is a linehaul customer, or if AC_k^B is greater than the load for customer i and i is a backhaul customer. We use the following *random-proportional rule* for choosing the vehicle to serve customer i.

$$p_{ik} = \begin{cases} \dfrac{[\tau 1_{ik}]^{\alpha 1}[\eta_k]^{\beta 1}}{\sum_{h \in \Psi}[\tau 1_{ih}]^{\alpha 1}[\eta_h]^{\beta 1}} & \text{if vehicle } k \in \Psi \\ 0 & \text{otherwise} \end{cases} \qquad (3.6)$$

where

$$\eta_k = \begin{cases} \dfrac{AC_k^L}{Q} & \text{if customer } i \text{ is a linehaul customer} \\ \dfrac{AC_k^B}{Q} & \text{if customer } i \text{ is a backhaul customer} \end{cases} \quad (3.7)$$

Here η_k is called dummy visibility since it insures that a high probability is given to a vehicle that has high available capacity. The dummy visibility increases the chance of generating a feasible solution since it enhances the probability of assigning at least one linehaul customer to each vehicle. Parameters α1 and $\beta 1$ reflect the relative influence of trail intensity and dummy visibility. AC_k^L or AC_k^B is updated once a vehicle is chosen for serving customer i. Customer i is then added to set C_k.

Step 4: if $\Omega = \phi$ then stop else go to step 2

There is a chance of getting an infeasible solution in step 3 because of the limited capacity of a vehicle. In particular, while assigning the last few customers, we may find that no vehicle can accommodate the customer; i.e., $\Psi = \phi$. In this case, steps 1 to 4 are repeated wherein we set customers in Ω in non-increasing order of demand. We repeat cycles of steps 1 to 4 until a feasible solution is found.

Arranging the customers in non-increasing order of demand increases the likelihood of obtaining a feasible solution in a given cycle. Since customers are assigned to vehicles according to the random proportion rule, the solution would vary from cycle

to cycle. Since we are assuming that feasible solutions exist, repeated cycles of steps 1 to 4 would eventually lead to a feasible solution. For some hard and unfriendly problem instances, our algorithm may take excessive CPU time. However, this may happen with other solution techniques as well.

A numerical example on assigning customers to vehicles.

Consider a problem instance with 3 linehaul and 2 backhaul customers. There are two vehicles available to serve these five customers, each having a capacity of 50. The demand (delivery or pickup) for the five customers are given below. Customers 1, 2 and 3 are linehaul whereas customers 4 and 5 are backhaul customers. The unit of measurement for delivery demand, pickup demand and capacity is assumed to be the same. Hence, we do not write it.

Customer	Demand (d_i)
1	30
2	15
3	20
4	40
5	20

Thus for the above problem, $l = 3$, $b = 2$, $n = 3+2 = 5$, $v = 2$ and $Q = 50$. Let the initial solution be S = {{0-1-3-4-0}, {0-2-5-0}} and suppose the total tour length L of this solution is 1000. Then we set all $\tau 1_{ik} = 1/L = 0.001$ and all $\tau 2_{ij} = 1/L = 0.001$.

Now we start the steps for assigning customers to vehicles.

Step 1: Initialization

The set of customers not yet served, $\Omega = \{1,2, 3,4,5\}$,

The set of customer to be served by vehicle k, $C_1 = \{\phi\}$; $C_2 = \{\phi\}$,

Available capacity of vehicle k, AC_1^L=50, AC_2^L=50, AC_1^B=50, and AC_2^B=50.

Iteration 1:

Step 2: Suppose the customer chosen randomly from Ω is customer 3. Update Ω to Ω = $\{1,2, 4,5\}$.

Step 3: Since $AC_1^L > d_3$ and $AC_2^L > d_3$, $\Psi = \{1,2\}$. The dummy visibilities for vehicles 1 and 2 are $\eta_1 = \frac{AC_1^L}{Q} = 50/50 = 1$ and $\eta_2 = \frac{AC_2^L}{Q} = 50/50 = 1$. Let $\alpha 1 = \beta 1 = 3$.

The probability p_{3k} for choosing vehicle k to serve customer 3 is calculated as

$$p_{31} = \frac{[\tau 1_{31}]^{\alpha 1}[\eta_1]^{\beta 1}}{[\tau 1_{31}]^{\alpha 1}[\eta_1]^{\beta 1} + [\tau 1_{32}]^{\alpha 1}[\eta_2]^{\beta 1}} = \frac{[0.001]^3[1]^3}{[0.001]^3[1]^3 + [0.001]^3[1]^3} = 0.5$$

$$p_{32} = \frac{[\tau 1_{32}]^{\alpha 1}[\eta_2]^{\beta 1}}{[\tau 1_{31}]^{\alpha 1}[\eta_1]^{\beta 1} + [\tau 1_{32}]^{\alpha 1}[\eta_2]^{\beta 1}} = \frac{[0.001]^3[1]^3}{[0.001]^3[1]^3 + [0.001]^3[1]^3} = 0.5$$

Suppose after using the random proportional rule, the vehicle chosen to serve customer 3 is vehicle 1. Hence, we update AC_1^L to AC_1^L = 50-20 = 30 and C_1 to $C_1 = \{3\}$.

Step 4: Since $\Omega \neq \phi$, we go back to step 2.

Iteration 2:

Note,

$\Omega = \{1,2,4,5\}$,

$C_1 = \{3\}$; $C_2 = \{\phi\}$,

$AC_1^L = 30$; $AC_2^L = 50$,

$AC_1^B = 50$; and $AC_2^B = 50$.

Step 2: Suppose customer chosen randomly from Ω is customer 4. Update Ω to $\Omega = \{1,2,5\}$.

Step 3: Note customer 4 is a backhaul customer and d_4 =40. Since $AC_1^B > d_4$ and $AC_2^B > d_4$, $\Psi = \{1, 2\}$. The dummy visibilities for vehicles 1 and 2 are $\eta_1 = \frac{AC_1^B}{Q} = 50/50 = 1$ and $\eta_2 = \frac{AC_1^B}{Q} = 50/50 = 1$. The probability p_{4k} for choosing vehicle k to serve customer 4 is calculated as

$$p_{41} = \frac{[\tau 1_{41}]^{\alpha 1}[\eta_1]^{\beta 1}}{[\tau 1_{41}]^{\alpha 1}[\eta_1]^{\beta 1} + [\tau 1_{42}]^{\alpha 1}[\eta_2]^{\beta 1}} = \frac{[0.001]^3[1]^3}{[0.001]^3[1]^3 + [0.001]^3[1]^3} = 0.5$$

$$p_{42} = \frac{[\tau 1_{42}]^{\alpha 1}[\eta_2]^{\beta 1}}{[\tau 1_{41}]^{\alpha 1}[\eta_1]^{\beta 1} + [\tau 1_{42}]^{\alpha 1}[\eta_2]^{\beta 1}} = \frac{[0.001]^3[1]^3}{[0.001]^3[1]^3 + [0.001]^3[1]^3} = 0.5$$

Suppose after using the random proportional rule, the vehicle chosen to serve customer 4 is vehicle 2. Hence we update AC_2^B to AC_2^B = 50-40 = 10 and C_2 to $C_2 = \{4\}$.

Step 4: Since $\Omega \neq \phi$, we go back to step 2.

Iteration 3:

At the beginning of iteration 3,

$\Omega = \{1,2,5\}$,

$C_1 = \{3\}$; $C_2 = \{4\}$,

AC_1^L=30, AC_2^L=50,

AC_1^B=50, and AC_2^B=10.

Step 2: Suppose customer chosen randomly from Ω is customer 2. Update Ω to Ω = {1,5}.

Step 3: Since $AC_1^L > d_2$ and $AC_2^L > d_2$, $\Psi = \{1, 2\}$. The dummy visibilities for vehicles 1 and 2 are η_1 = 30/50 = 0.6 and η_2 = 50/50 = 1. The probability p_{2k} for choosing vehicle k to serve customer 2 is calculated as

$$p_{21} = \frac{[\tau 1_{21}]^{\alpha 1}[\eta_1]^{\beta 1}}{[\tau 1_{21}]^{\alpha 1}[\eta_1]^{\beta 1} + [\tau 1_{22}]^{\alpha 1}[\eta_2]^{\beta 1}} = \frac{[0.001]^3[0.6]^3}{[0.001]^3[0.6]^3 + [0.001]^3[1]^3} = 0.17$$

$$p_{22} = \frac{[\tau 1_{22}]^{\alpha 1}[\eta_2]^{\beta 1}}{[\tau 1_{21}]^{\alpha 1}[\eta_1]^{\beta 1} + [\tau 1_{22}]^{\alpha 1}[\eta_2]^{\beta 1}} = \frac{[0.001]^3[1]^3}{[0.001]^3[0.6]^3 + [0.001]^3[1]^3} = 0.83$$

Suppose after using the random proportional rule, the vehicle chosen to serve customer 2 is vehicle 2. Update AC_2^L to AC_2^L = 50 -15 = 35 and C_2 to C_2= {4,2}.

Step 4: Since $\Omega \neq \phi$, we go back to step 2.

Iteration 4:

At the beginning of iteration 4,

$\Omega = \{1,5\}$,

$C_1 = \{3\}$; $C_2 = \{4,2\}$,

$AC_1^L = 30,\ AC_2^L = 35,$

$AC_1^B = 50$, and $AC_2^B = 10.$

Step 2: Suppose customer chosen randomly from Ω is customer 1. Update Ω to $\Omega = \{5\}$.

Step 3: Since $AC_1^L \geq d_1$ and $AC_2^L > d_2$, $\Psi = \{1, 2\}$. The dummy visibilities for vehicles 1 and 2 are $\eta_1 = 30/50 = 0.6$ and $\eta_2 = 35/50 = 0.7$. The probability p_{1k} for choosing vehicle k to serve customer 1 is calculated as

$$p_{11} = \frac{[\tau 1_{11}]^{\alpha 1}[\eta_1]^{\beta 1}}{[\tau 1_{11}]^{\alpha 1}[\eta_1]^{\beta 1} + [\tau 1_{12}]^{\alpha 1}[\eta_2]^{\beta 1}} = \frac{[0.001]^3[0.6]^3}{[0.001]^3[0.6]^3 + [0.001]^3[0.7]^3} = 0.39$$

$$p_{12} = \frac{[\tau 1_{12}]^{\alpha 1}[\eta_2]^{\beta 1}}{[\tau 1_{11}]^{\alpha 1}[\eta_1]^{\beta 1} + [\tau 1_{12}]^{\alpha 1}[\eta_2]^{\beta 1}} = \frac{[0.001]^3[0.7]^3}{[0.001]^3[0.6]^3 + [0.001]^3[0.7]^3} = 0.61$$

Suppose after using the random proportional rule, the vehicle chosen to serve customer 1 is vehicle 1. Update AC_1^L to $AC_1^L = 30 - 30 = 0$ and C_1 to $C_1 = \{3,1\}$.

Step 4: Since $\Omega \neq \phi$, we go back to step 2.

Iteration 5:

At the beginning of iteration 5,

$\Omega = \{5\},$

$C_1 = \{3,1\};\ C_2 = \{4,2\},$

$AC_1^L = 0,\ AC_2^L = 35,$

$AC_1^B = 50$, and $AC_2^B = 10.$

Step 2 : Only customer 5 is available, hence we choose customer 5 for assignment and update Ω to $\Omega = \{\phi\}$.

Step 3: Since, $AC_1^B \geq d_5$ but $AC_2^B < d_5$, $\Psi = \{1\}$. Thus, only vehicle 1 can be assigned to customer 5. We update $AC_1^B = 50 - 20 = 30$ and $C_1 = \{3,1,5\}$.

Step 4: Since $\Omega = \phi$, we stop.

At the end of the algorithm, the assignments are $C_1 = \{3,1,5\}$ and $C_2 = \{4,2\}$.

Construction of routes for individual vehicles

Once customers are assigned to vehicles, a route is constructed for each vehicle by solving the underlying traveling salesman problem. First, all linehaul customers assigned to vehicle k are considered for visit. When all linehaul customers are visited, the backhaul customers assigned to vehicle k are considered. The ant starts from the depot and successively builds the solution by choosing the next customer j probabilistically from the set of feasible customers, $\Re$. Initially $\Re$ is set to linehaul customers assigned to vehicle k and it is updated as linehaul customers are visited. When $\Re$ becomes empty, it is set to backhaul customers assigned to vehicle k. The attractiveness value of visiting customer j immediately after customer i, ξ_{ij} is defined as

$$\xi_{ij} = \begin{cases} [\tau 2_{ij}]^{\alpha 2}[s_{ij}]^{\beta 2} & \text{if } i \text{ is a customer} \\ [\tau 2_{ij}]^{\alpha 2}[1/c_{ij}]^{\beta 2} & \text{if } i \text{ is a depot} \end{cases} \qquad (3.8)$$

Here parameter $\alpha 2$ represents the relative influence of the trail intensity and parameter $\beta 2$ represents the relative influence of the saving value. The saving value s_{ij} represents the savings (in terms of distance traveled) when customers i and j are served

jointly by one vehicle instead of two separate vehicles. The saving value s_{ij} is calculated as follows:

$$s_{ij} = c_{0i} + c_{0j} - c_{ij} \tag{3.9}$$

Here c(i , j) is the distance between customer i and j. We use the following *random proportional rule* to select the next customer j to visit immediately after customer i

$$p_{ij} = \begin{cases} \dfrac{\xi_{ij}}{\sum_{l \in \Re} \xi_{il}} & \text{for } j \in \Re \\ \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

The process is continued until the route for the vehicle is built. Then the procedure is repeated for other vehicles.

A numerical example on construction of routes for individual vehicles

We continue with the numerical example used to illustrate assigning customers to vehicles. Note that we have $C_1 = \{3,1,5\}$ and $C_2 = \{4,2\}$. Let $\alpha 2 = 1$ and $\beta 2 = 1$.

First consider the construction of the route for vehicle 1. Initially $\Re$ is set to linehaul customers assigned to vehicle 1, i.e., $\Re = \{3,1\}$. The ant starts from the depot, hence our partial route so far is $R_1 = \{0\}$.

The attractiveness values are calculated as

$$\xi_{03} = [\tau 2_{03}]^{\alpha 2} [1/c(0,3)]^{\beta 2} = [0.001]^1 [1/100]^1 = 0.00001$$

$$\xi_{01} = [\tau 2_{01}]^{\alpha 2} [1/c(0,1)]^{\beta 2} = [0.001]^1 [1/120]^1 = 0.000008$$

The probability to visit customer j immediately from the depot is:

$$p_{03} = \frac{\xi_{03}}{\xi_{03} + \xi_{01}} = \frac{0.00001}{0.00001 + 0.000008} = 0.55$$

$$p_{01} = \frac{\xi_{01}}{\xi_{03} + \xi_{01}} = \frac{0.000008}{0.00001 + 0.000008} = 0.45$$

Suppose after using the random proportional rule, the customer chosen to visit next is customer 1. Update $\Re$ to $\Re = \{3\}$, and R_1 to $R_1 = \{0\text{-}1\}$.

After customer 1, there is only one linehaul customer left, namely customer 3. Thus we update R_1 to $R_1 = \{0\text{-}1\text{-}3\}$ and set $\Re = \phi$.

Now fill $\Re$ with all backhaul customers who are assigned to vehicle 1, i.e., let $\Re = \{5\}$. Thus $R_1 = \{0\text{-}1\text{-}3\text{-}5\}$. The ant returns back to the depot after customer 5. The final route of vehicle 1 is $R_1 = \{0\text{-}1\text{-}3\text{-}5\text{-}0\}$

The construction of the route for vehicle 2 is trivial since $C_2 = \{4,2\}$; i.e., we have just one linehaul and one backhaul customer assigned to vehicle 2. Hence, the ant starts the route from the depot, visits customer 2 and then customer 4, and finally returns back to the depot. Thus, $R_2 = \{0\text{-}2\text{-}4\text{-}0\}$ and the final solution is S = {{0-1-3-5-0}, {0-2-4-0}}.

3.3.3 Local search

After the construction of a solution by an ant, the solution is improved by local search. Three types of local search schemes are used including two multi-route schemes. We use 2-*opt* local search scheme described earlier to improve the solution generated. Other two local search schemes are the "customer insertion/interchange multi-route

scheme" and the "sub-path exchange multi-route scheme". These two local search schemes are described in detail in the next sub-sections. These two schemes are used iteratively until both are not able to improve the solution. The best solution obtained by the insertion/interchange multi-route scheme is used as an initial solution for the sub-path exchange multi-route scheme and the best solution obtained by the sub-path exchange multi-route scheme is used as an initial solution for the insertion/interchange multi-route scheme in the subsequent cycle. The process is repeated until both multi-route schemes are not able to improve the solution. During our experiment, we found that generally a local minimum is reached after three or four cycles.

Customer insertion/interchange multi-route scheme

This approach uses two types of operations for a given customer. These two operations are described below for customer i assigned to vehicle p:

1. Insertion Operation: Customer i is removed from his original position and inserted in all vehicle routes (including the same vehicle route). Vehicle capacity is checked before the insertion of customer i to another vehicle. Our variant of VRPB requires that each vehicle should serve at least one linehaul customer and the linehaul customers on a route must be served before backhaul customers. In order to maintain this feasibility, the insertion operation for a linehaul customer is used only when the remaining number of linehaul customers assigned to vehicle p is more than 1. To insure feasibility, a linehaul customer is allowed to be inserted at the following positions in the route.
    - just after the depot

- between two linehaul customers
- between the last linehaul customer and the first backhaul customer

Similarly, a backhaul customer is allowed to be inserted at the following positions in the route.

- between the last linehaul customer and the first backhaul customer
- between two backhaul customers
- just before the depot

2. Interchange Operation: In this operation customer i (from vehicle p) is shifted to another vehicle q and customer j from vehicle q is shifted to vehicle p. We place customer i on vehicle q at its *best* insertion place in terms of total tour length and similarly we place customer j on vehicle p at his *best* insertion place.

In order to reduce CPU time, the following neighborhood criterion is used before an insertion/interchange operation is performed. Let Γ be some fixed number and let N_Γ be the set of Γ nearest customers (in term of the distance) of customer i. The interchange operation between customer i and customer j is performed only if at least one of the Γ neighbors of customer i is served by vehicle q and at least one of the Γ neighbors of customer j is served by vehicle p. Similarly, the insertion of customer i on vehicle q is performed only if one of the Γ neighbors of customer i is served by vehicle q.

The steps in the local search are:

Step 1: Initialize $\Omega = \{1,2,\ldots,n\}$.

Step 2: Randomly choose a customer, say customer i from vehicle p for evaluation, and remove this customer from set Ω.

Step 3: Perform insertion of customer i to determine the best insertion position and evaluate the total tour length of the resulting solution.

Step 4: Consider also the interchange of customer i and each customer j. Both customers i and j must be either linehaul customers or backhaul customers and the interchange should be feasible in terms of the vehicle capacity. Identify the best interchange customer j and the corresponding solution after performing the interchange operation.

Step 5: Choose the best solution obtained from steps 3 and 4 and compare it with the current solution. If the best solution is better than the current solution then update the current solution.

Step6: If $\Omega = \phi$ then stop and report the current solution, else go to step 2.

The customer insertion/interchange multi-route scheme is similar to the insertion/exchange schemes used by other researchers (e.g., Osman (1993), Van Breedam (1995), Kindervater and Savelsbergh (1997)). Our insertion operation is similar with previous work but our interchange operation differs in the way a customer is placed in the new route. In previous work, the positions of customer i from vehicle p and customer j from vehicle q are interchanged. In our approach, customer i from vehicle p is placed in the route of vehicle q at his *best* insertion place. Similarly, customer j from vehicle q is placed in the route of vehicle p at his *best* insertion place.

The cost of the solution generated by the interchange as well as the insertion operation can be determined in constant time. In the absence of the customer

neighborhood criterion, the theoretical computational complexity of a single iteration of the insertion/interchange multi-route scheme can be calculated as follows.

The insertion operation has complexity of $O(l^2 + b^2)$. The interchange operation considers a customer on a given route for interchange with each customer from another route. Hence, for linehaul customers, the number of pairs for possible interchange are of order $O(l^2)$. Each customer from a given pair is placed at his/her best insertion place. Thus, each customer is evaluated at maximum of l places to find his/her best insertion place. In a given pair, finding the best insertion place for one customer is independent of finding the best insertion for the second customer. Therefore, the overall complexity of an interchange operation involving linehaul customers is:

$$O(l^2) \times (O(l) + O(l)) = O(l^3).$$

Similarly the overall complexity of an interchange operation involving backhaul customers is $O(b^3)$. Hence, the overall complexity for the insertion/interchange multi-route scheme is:

$$O(l^2 + b^2) + O(l^3) + O(b^3) = O(l^3 + b^3).$$

The introduction of the customer neighborhood criterion should reduce CPU time in practice. However, in the worst case scenario, there may still be l^2 pairs of linehaul customers and b^2 pairs of backhaul customers for interchange even in the presence of the customer neighborhood criterion. Thus, the theoretical computational complexity of the scheme remains the same with or without the customer neighborhood criterion.

Sub-path exchange multi-route scheme

The customer insertion/interchange multi-route scheme considers shifting a given customer to another vehicle route. The sub-path exchange multi-route scheme can shift more than one customer from one route to another. Suppose $S = \{R_1, R_2, \ldots, R_p, \ldots R_q, \ldots, R_{v-1}, R_v\}$ represents a solution to VRPB. Here R_p denote the route of vehicle p. The sub-path exchange multi-route scheme considers two routes R_p and R_q and combines them in two new routes R'_p and R'_q to produce a new solution $S' = \{R_1, R_2, \ldots, R'_p, \ldots, R'_q, \ldots, R_{v-1}, R_v\}$. Consider sub-path $(e,\ldots,f)$ belonging to route R_p and sub-path $(g,\ldots,h)$ belonging to R_q. To maintain feasibility, all customers in sub-path $(e,\ldots,f)$ and sub-path $(g,\ldots,h)$ are either linehaul or backhaul customers. Furthermore, the feasibility in terms of vehicle capacity is checked before an exchange is considered. The new route R'_p is obtained by replacing sub-path $(e,\ldots,f)$ by either sub-path $(g,\ldots,h)$ or reverse sub-path $(h,\ldots,g)$. Similarly, the new route R'_q is obtained by replacing sub-path $(g,\ldots,h)$ by either sub-path $(e,\ldots,f)$ or reverse sub-path $(f,\ldots,e)$.

We again use the customer neighborhood criterion to reduce the CPU time. The exchange of sub-path $(e,\ldots,f)$ of vehicle p with sub-path $(g,\ldots,h)$ of vehicle q is considered only if customer f is among the 15 nearest neighbors of customer g.

Each possible sub-path from route R_p and from route R_q is considered for possible exchange and the best pair of sub-paths is used to produce new routes R'_p and R'_q. The calculation of the cost of the route produced by an exchange can be

performed in constant time. Let γ_i represent the immediate predecessor of customer i and π_i represent the immediate successor of customer i. Then the cost (i.e., total tour length) of new solution S' is calculated as follows:

$$\begin{aligned} Cost(S') = Cost(S) &- c(\gamma_e, e) - c(f, \pi_f) - c(\gamma_g, g) - c(h, \pi_h) \\ &+ \min\left\{\left(c(\gamma_e, g) + c(h, \pi_f)\right), \left(c(\gamma_e, h) + c(g, \pi_f)\right)\right\} \\ &+ \min\left\{\left(c(\gamma_g, e) + c(f, \pi_h)\right), \left(c(\gamma_g, f) + c(e, \pi_h)\right)\right\} \end{aligned} \tag{3.11}$$

The number of sub-paths starting from a given linehaul customer is of order O(l). There are l linehaul customers. Hence, the total number of sub-paths containing linehaul customers is of order O(l^2). Any of these l^2 sub-paths can be exchanged with other l^2 - 1 sub-paths. Thus, the overall complexity of the sub-path exchange multi-route scheme for linehaul customers in the absence of the neighborhood criterion is:

$$O(l^2) \times O(l^2 - 1) = O(l^4).$$

Similarly, the overall complexity of the sub-path exchange multi-route scheme for backhaul customers is O(b^4). Therefore, the overall complexity of the sub-path exchange multi-route scheme in absence of the neighborhood criterion is O($l^4 + b^4$).

With the customer neighborhood criterion, a given sub-path (containing linehaul customers) can be exchanged with a maximum of $\Gamma \times l$ sub-paths from the other vehicles, where Γ is constant. Thus, the overall complexity of the sub-path exchange multi-route scheme for linehaul customers with the neighborhood criterion is:

$$O(l^2) \times O(\Gamma \times l) = O(l^3).$$

Similarly, the overall complexity of the sub-path exchange multi-route scheme for backhaul customers is O(b^3). Therefore, the overall complexity of the sub-path exchange multi-route scheme with the neighborhood criterion is O($l^3 + b^3$).

The sub-path exchange multi-route scheme is similar to CROSS exchange heuristic of Taillard et al. (1997) and ICROS heuristic of Bräysy and Dullaert (2003). Our sub-path exchange multi-route scheme falls between the above two heuristics. Considering all possible sub-paths for possible exchange is similar to CROSS heuristic whereas connecting the reverse sub-paths is similar to ICROSS heuristic.

3.3.4 Updating elitist ants

We use elitist ants to update trail intensities. We use γ elitist ants which are distinct from one another (See subsection 3.4.1 for setting the value of γ). Elitist ants are updated by comparing the present elitist ant solutions with the current ant solution. If the current ant solution is found to be better than the μ^{th} elitist ant solution, then the current ant solution becomes the new μ^{th} elitist ant solution. The ranks of the previous μ^{th} elitist ant and the elitist ants below it are lowered by one. All current solutions are considered for updating the elitist ants. In order to keep σ elitist ants distinct from one another, the current solution is compared only if the total tour length in the current solution is not equal to the total tour length in any of the elitist ant solutions. The hashing function used by Osman and Wassan (2002) can be used instead of the total tour length for identifying the identical solutions, but calculation of the hashing function would require more CPU time.

3.3.5 Updating trail intensities

Vehicle-trail intensities and *route-trail* intensities are updated using the elitist ant solutions. *Vehicle-trail* intensities are updated as follows:

$$\tau 1_{ik}^{new} = \rho \tau 1_{ik}^{old} + \Delta \tau 1_{ik}^{*} + \sum_{\mu=1}^{\gamma} \Delta \tau 1_{ik}^{\mu} \qquad i=1,2,.....,n;\ k=1,2,.....,v. \tag{3.12}$$

where

$$\Delta \tau 1_{ik}^{\mu} = \begin{cases} 1/L^{\mu} & \text{if customer } i \text{ is served by vehicle } k \text{ in the } \mu^{th} \text{ elitist ant solution} \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

$$\Delta \tau 1_{ik}^{*} = \begin{cases} 1/L^{*} & \text{if customer } i \text{ is served by vehicle } k \text{ in the best solution found to date} \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

Here ρ is the trail persistence and L^{μ} is the total tour length in the μ^{th} elitist ant solution and L^{*} is the total tour length in the best solution found to date.

Similarly, *Route-trail* intensities are updated as follows:

$$\tau 2_{ij}^{new} = \rho \tau 2_{ij}^{old} + \Delta \tau 2_{ij}^{*} + \sum_{\mu=1}^{\gamma} \Delta \tau 2_{ij}^{\mu} \qquad i=1,2,.....,n;\ j=1,2,.....,n \quad ;\ i \neq j \tag{3.15}$$

where

$$\Delta \tau 2_{ij}^{\mu} = \begin{cases} 1/L^{\mu} & \text{if customer } j \text{ is visited after customer } i \text{ in the } \mu^{th} \text{ elitist ant solution} \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

$$\Delta \tau 2_{ij}^{*} = \begin{cases} 1/L^{*} & \text{if customer } j \text{ is visited after customer } i \text{ in the best solution found so far} \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

The scheme allows all elitist ants to lay trails with *equal importance* as opposed to the updating scheme used in an ant colony system algorithm for VRP by Bullnheimer et al. (1999). The strategy of giving more importance to the best elitist ant solution and less importance to the solutions of other elitist ants can divert the search towards the global minimum but it can also increase the risk of being trapped at a local minimum.

The best solution found so far will be equivalent to the top elitist ant solution when COUNT=0 but it may differ from the top elitist ant solution when COUNT >0.

Numerical Example for updating trail intensity

Consider the example used in the previous section. Suppose the number of elitist ants is equal to 2 and $\rho = 0.95$. Suppose the best solution found so far is:

S(Best) = { {0-2-1-4-0}, {0-3-5-0}}, $L^* = 500$

Similarly, suppose the two elitist ant solutions are:

S(Elitist 1) = { {0-3-1-5-0}, {0-2-4-0}}, $L^1 = 700$

S(Elitist 2) = {{0-1-3-4-0}, {0-2-5-0}}, $L^2 = 1000$

In the above solutions, the first route is for vehicle 1 and the second route is for vehicle 2. Also assume that all trail intensities are initially set to 0.001 (i.e, all $\tau 1_{ik} = 0.001$ and all $\tau 2_{ij} = 0.001$). The *Vehicle-trail* intensities for vehicle 1 are updated as follows.

$\tau 1_{i1}^{new} =$	$\rho * \tau 1_{i1}^{old} +$	$\Delta\tau 1_{i1}^{*} +$	$\Delta\tau 1_{i1}^{1} +$	$\Delta\tau 1_{i1}^{2} =$	
$\tau 1_{11}^{new} =$	0.95*0.001 +	1/500 +	1/700 +	1/1000 =	0.0053
$\tau 1_{21}^{new} =$	0.95*0.001 +	1/500 +	0+	0 =	0.0029

$\tau 1_{31}^{new} =$	0.95*0.001 +	0+	1/700 +	1/1000 =	0.0034
$\tau 1_{41}^{new} =$	0.95*0.001 +	1/500 +	0+	1/1000 =	0.0039
$\tau 1_{51}^{new} =$	0.95*0.001 +	0 +	1/700 +	0 =	0.0024

Similarly, the *Vehicle-trail* intensities for vehicle 2 are updated as follows

$\tau 1_{i2}^{new} =$	$\rho * \tau 1_{i2}^{old} +$	$\Delta\tau 1_{i2}^{*} +$	$\Delta\tau 1_{i2}^{1} +$	$\Delta\tau 1_{i2}^{2} =$	
$\tau 1_{12}^{new} =$	0.95*0.001 +	0 +	0 +	0 =	0.00095
$\tau 1_{22}^{new} =$	0.95*0.001 +	0 +	1/700+	1/1000 =	0.0034
$\tau 1_{32}^{new} =$	0.95*0.001 +	1/500 +	0 +	0 =	0.0029
$\tau 1_{42}^{new} =$	0.95*0.001 +	0 +	1/700+	0 =	0.0024
$\tau 1_{52}^{new} =$	0.95*0.001 +	1/500 +	0 +	1/1000 =	0.0039

The *Route*-trail intensities are updated in a similar way. The route-trail intensities for a route starting at customer 1 are updated as:

$\tau 2_{1j}^{new} =$	$\rho * \tau 2_{1j}^{old} +$	$\Delta\tau 2_{1j}^{*} +$	$\Delta\tau 2_{1j}^{1} +$	$\Delta\tau 2_{1j}^{2} =$	
$\tau 2_{12}^{new} =$	0.95*0.001 +	0 +	0 +	0 =	0.00095
$\tau 2_{13}^{new} =$	0.95*0.001 +	0 +	0 +	1/1000 =	0.00195
$\tau 2_{14}^{new} =$	0.95*0.001 +	1/500 +	0 +	0 =	0.00295
$\tau 2_{15}^{new} =$	0.95*0.001 +	0 +	1/700+	0 =	0.00239

Updating trail intensities for the other customers will be similar to the calculations above.

3.4 Numerical analysis

This section presents the numerical results for MACS and compares MACS with the existing heuristic and metaheuristic approaches.

3.4.1 Parameter settings

The quality of the solution depends on the number of ants used which ultimately affects the CPU time of the algorithm. We use 25 artificial ants of each type in MACS, i.e., at each iteration, 25 solutions are generated and each solution is improved by local search. We reinitialize the trail intensities and destroy the elitist ants if there is no change in the elitist ant solutions in the last 10 iterations. Our algorithm stops after the trail intensities are reinitialized 5 times.

Parameters $\alpha 1$, $\beta 1$, $\alpha 2$, $\beta 2$, ρ and γ are set by performing sensitivity analysis carried out within limited CPU time. See Dorigo (2004) for more insight on setting values for the parameters and the effect parameter values can have on solution quality. In our experiment we found that the solution is not sensitive with respect to parameters $\alpha 1$, $\beta 1$, $\alpha 2$, $\beta 2$, ρ and γ. For computational purposes, we use $\alpha 1 = \beta 1 = 3$, $\alpha 2 = \beta 2 = 1$, $\rho = 0.95$, $\gamma = 6$. We thus use 6 elitist ants to update the pheromone.

We found that the performance of MACS is sensitive to parameter Γ. Our customer interchange local search scheme uses Γ of size 10. We found that Γ of 10 gave us the best tradeoff between the CPU time and the solution quality. If none of the 10 neighboring customers are served by the given vehicle, then the probability is low that

this customer is served by the same vehicle in the optimal solution. Similarly, our sub-path exchange multi-route scheme uses Γ of size 15.

3.4.2. Benchmark problem

The numerical experiment is performed using two sets of problem instances available in literature. The first set (set-1) was proposed by Goetschalckx and Jacobsblecha (1989). This data set includes 62 instances where the total number of customers ranges from 25 to 150 with backhaul customers being 25 to 50% of the linehaul customers. In problem instances for set-1, customer coordinates are uniformly distributed in interval [0, 24000] for the x coordinate, and in interval [0, 32000] for the y coordinate. The depot is located centrally at (x=12000, y=16000). Customer demands are generated from Normal distribution with mean of 500 and standard deviation of 200. For each problem instances, the vehicle capacity is defined so that approximately 3 to 12 vehicles are used to serve all the demands.

The second set (set-2) was proposed by Toth and Vigo (1997). This set consists of 33 instances with 21 to 100 customers. These instances are generated from 11 problems chosen from the VRP literature. Three instances are generated from each problem with backhaul customers being 50, 66 and 80 % of the linehaul customers.

Euclidean distance between two customers is calculated in two ways in the literature. In the first approach, the distance is multiplied by 10 and is rounded to the nearest integer. The total distance is then divided by 10 and rounded to the nearest integer. In the second approach, the distance is used without rounding but the total tour length is rounded to two

decimal places. Since many studies use the distance without rounding, we use only approach 2.

3.4.3. Existing heuristics for VRPB

We compare MACS results with the following metaheuristic algorithms for which there are reported results:

RTS-AMP: reactive tabu adaptive memory programming of Wassan (2004)

LNS: large neighborhood search by Ropke, S. and D. Pisinger (2006)

BTS: new tabu search of Brandao (2006)

We do not include reactive tabu search (RTS) in our table because RTS-AMP has produced either similar or better quality solutions on all instances at a lower computational cost compared to RTS. Brandao (2006) developed three types of tabu search; namely *open*, *K-tree* and *K-tree_r* tabu search. Out of these three methods, *K-tree_r* seems to yield better solutions on average compared to the other two methods. Therefore, we use the solutions obtained by *K-tree_r* method alone and call it BTS for comparison purposes. Similarly Ropke and Pisinger (2006) developed three configurations of large scale neighborhood search; namely *standard*, *6R-no learning* and *6R-normal learning*. Out of these three methods, we use the solutions obtained by *6R-no learning* configuration and call it LNS in our comparison. Each of the studies reports solutions in a different way. RTS-AMP reports the best solution from 8 runs, LNS reports the best solution from 10 runs and BTS reports a single run solution but uses the best run from 5 runs.

Generally, it is difficult to compare the CPU times of different algorithms as the computational experiments have been performed on different machines. A comparison of CPU time can be done using Mflops (million floating point calculation per second) values of different computers given in the Linpack benchmark report of Dongarra (2006). Specifications of different computers and their Mflops values are presented in Table 3.5. We use the Mflops values to scale the running time of an algorithm in reference to our computer (i.e., Intel Xeon 2.4 GHz). The scaled CPU time represents the approximate CPU time if Intel Xeon 2.4 GHz were to be used to run the particular algorithm. We report the average CPU time of MACS over 8 runs. In order to keep the CPU time of MACS comparable with other algorithms, we have restricted the number of ants of each type to 25.

We also use average relative percentage deviation (ARPD) to evaluate the performance of a metaheuristic. The ARPD value, the average objective function value and the average CPU time are reported at the bottom of each table. The ARPD value is based on relative percentage deviation (RPD) calculations. The RPD values are not shown in the table. For a given problem instance, the RPD value is calculated as

RPD = {(Heuristic Solution – Best Known Solution)/Best Known Solution} * 100..(3.18)

3.4.4. Computational experiment and performance analysis of the algorithms

The proposed MACS was coded in C and implemented on an Intel Xeon with 2.4 GHz computer system. We run MACS 8 times and report the results under different criterion. The detailed results of MACS are reported in Table 3.3 and Table 3.4 for data set-1 and

set-2 respectively. The detailed results for other algorithms are reported in Table 3.8, Table 3.9, Table 3.10 and Table 3.11. The ARPD values for different algorithms are given at the bottom of Tables 3.3, 3.4, 3.8 and 3.9.

We summarize the overall results of MACS in Tables 3.1 and 3.2 using the following performance measures.

1. Overall average solution: The overall average cost over 62 problem instances for set-1 and 33 problem instances for set-2. For each problem instance, the average cost is calculated from costs over 8 runs.
2. Average standard deviation: The average of the standard deviation over 62 problem instances for set-1 and 33 problem instances for set-2. For each problem, the standard deviation is computed using the solutions from 8 runs.
3. Average best solution: The average of the best solutions over 62 problem instances for set-1 and 33 problem instances for set-2. For each problem instance, the cost associated with the best solution is obtained from 8 runs.
4. Average best run: We execute the algorithm for 8 runs. The best run is the run that gives the lowest average over 62 problem instances for set-1 and 33 problem instances for set-2. The lowest average is called "average best run". We use this measure to compare ACS with BTS since BTS reports the best run solution for each problem instance.
5. Average scaled CPU time: The average of the scaled CPU time per run over 62 problem instances for set-1 and 33 problem instances for set-2. For each problem

instance, the scaled CPU time per run is calculated using the scaled CPU time over 8 runs.

We present summary results of different algorithms in Table 3.2. A blank space in Table 3.2 indicates that the corresponding summary value was not reported in the original paper.

Table 3.1: Average performance of MACS over 62 instances for set-1 and 33 instances for set-2

	Overall average solution	Average standard deviation	Average best solution	Average best run	Average scaled CPU time
Set-1	290920.90	276.04	290655.29	290838.73	67.57
Set-2	702.35	0.7985	701.48	701.76	25.64

Table 3.2: Comparison of MACS with different heuristics for VRPB

		RTS-AMP	LNS	BTS	MACS
Set-1	Overall average solution	-	291823.34	291305.7	290920.90
	Average best solution	290981.8	291014.7	-	290655.29
	Average best run	-	-	291160.4	290838.73
	Average scaled CPU time	20.20	26.22	66.84	67.57
Set-2	Overall average solution	-	704.54	702.50	702.35
	Average best solution	706.48	701.18	-	701.48

	Average best run	-	-	702.15	701.76
	Average scaled CPU time	7.95	15.55	24.40	25.64

The results for set-1 reported in Tables 3.1 and 3.2 indicate that the proposed multi-ant colony system (MACS) gives better results than the other heuristics. Table 3.2 shows that MACS has the lowest overall average solution followed by BTS and LNS. Also, Table 3.2 indicate that MACS is on average better than the other algorithms in terms of the best solution and the best run. In terms of CPU time, MACS and BTS are using equivalent CPU times. LNS takes less than half of the CPU time than MACS and RTS-AMP takes one third of the CPU time than MACS. However, MACS does have a control over solution quality and the CPU time and this feature is described in section 5.

The computational results for set-2 have similar trend in terms of the average solution. MACS has the lowest overall average solution followed by BTS and LNS. The average best run solution of MACS is slightly better than BTS. The overall average solution of MACS is better than LNS and RTS-AMP but the average best solution of MACS is slightly worse than the average best solution of LNS. The trend of CPU time is similar to the trend for set-1. The comparison using ARPD performance criterion has almost the same trend.

Finally note that we have found 4 new best known solutions for set-1 and 1 new best known solution for set-2. These solutions are highlighted with bold font in Tables 3.3. and 3.4.

Table 3.3. Total Tour Length obtained by MACS for data set -1

No	Instance	Average Solution	Standard Deviation	Best Solution	Best Run	Overall Best Solution*	Average CPU time
1	A1	229885.65	0	229885.65	229885.65	229885.65	1.00
2	A2	180119.21	0	180119.21	180119.21	180119.21	1.75
3	A3	163405.38	0	163405.38	163405.38	163405.38	3.00
4	A4	155796.41	0	155796.41	155796.41	155796.41	1.88
5	B1	239080.16	0	239080.16	239080.16	239080.15	2.13
6	B2	198047.77	0	198047.77	198047.77	198047.77	2.50
7	B3	169372.29	0	169372.29	169372.29	169372.29	2.00
8	C1	250556.77	0	250556.77	250556.77	250556.77	3.88
9	C2	215020.23	0	215020.23	215020.23	215020.23	4.13
10	C3	199345.96	0	199345.96	199345.96	199345.96	4.88
11	C4	195366.63	0	195366.63	195366.63	195366.63	3.88
12	D1	322530.13	0	322530.13	322530.13	322530.13	6.13
13	D2	316708.86	0	316708.86	316708.86	316708.86	6.25
14	D3	239478.63	0	239478.63	239478.63	239478.63	5.63
15	D4	205831.94	0	205831.94	205831.94	205831.94	6.50
16	E1	238879.58	0	238879.58	238879.58	238879.58	6.75
17	E2	212263.11	0	212263.11	212263.11	212263.11	6.50
18	E3	206659.17	0	206659.17	206659.17	206659.17	10.38
19	F1	263173.96	0	263173.96	263173.96	263173.97	11.13
20	F2	265214.16	0	265214.16	265214.16	265214.16	9.13
21	F3	241484.85	420.14	241120.78	241120.78	241120.77	11.25
22	F4	233861.85	0	233861.85	233861.85	233861.84	15.00
23	G1	307007.11	190.04	306536.78	307074.3	306305.40	18.00
24	G2	245440.99	0	245440.99	245440.99	245440.99	10.38
25	G3	229507.48	0	229507.48	229507.48	229507.48	14.25
26	G4	232521.25	0	232521.25	232521.25	232521.25	21.75
27	G5	221730.35	0	221730.35	221730.35	221730.35	20.38
28	G6	213457.45	0	213457.45	213457.45	213457.45	20.63
29	H1	268996.28	178.8	268933.06	268933.06	268933.06	24.50
30	H2	253365.50	0	253365.50	253365.5	253365.50	21.50
31	H3	247449.04	0	247449.04	247449.04	247449.04	20.25
32	H4	250220.77	0	250220.77	250220.77	250220.77	27.13
33	H5	246121.31	0	246121.31	246121.31	246121.31	26.00
34	H6	249135.32	0	249135.32	249135.32	249135.32	30.25

35	I1	350395.33	219.17	350245.28	350245.28	350245.28	41.63
36	I2	310318.17	1058.78	309943.84	309943.84	309943.84	37.50
37	I3	294839.85	510.65	294507.38	294507.38	294507.38	43.88
38	I4	296129.01	260.39	295988.45	295988.45	295988.44	46.63
39	I5	301826.52	894.41	301236.01	302441.47	301236.00	47.38
40	J1	335124.95	218.99	335006.68	335479.75	335006.69	66.63
41	J2	310417.21	0	310417.21	310417.21	310417.21	59.25
42	J3	279343.35	351.11	279219.21	279219.21	279219.21	77.63
43	J4	296584.68	145.71	296533.16	296533.16	296533.16	62.75
44	K1	396139.64	676.66	395075.67	395075.67	**394071.16**	104.88
45	K2	362563.92	462.14	362130.00	362130	362130.00	96.88
46	K3	366709.20	873.68	365694.08	365694.08	365694.09	114.25
47	K4	350317.26	751.71	349870.36	349870.36	348949.39	106.00
48	L1	420063.78	1307.78	417921.58	418828.93	**417896.72**	148.38
49	L2	401360.36	318.64	401247.70	401247.7	401228.81	142.00
50	L3	404315.09	1896.83	402677.72	403990.54	402677.72	160.63
51	L4	384834.32	274.73	384636.33	385225.91	384636.34	159.50
52	L5	390329.49	1192.23	387564.55	390987.74	387564.56	158.00
53	M1	399120.67	587.75	398729.82	398729.82	398593.19	229.50
54	M2	398156.41	563.83	397324.15	397664.54	396916.97	183.38
55	M3	377812.03	296.44	377328.75	377328.75	**375695.41**	183.75
56	M4	348464.23	69.28	348417.94	348512.42	**348140.16**	164.13
57	N1	408168.11	124.97	408100.62	408100.62	408100.62	213.88
58	N2	408246.94	246.84	408065.44	408065.44	408065.44	214.38
59	N3	394701.38	864.08	394337.86	396827	394337.86	199.75
60	N4	394866.92	108.42	394788.36	394788.36	394788.37	219.00
61	N5	374120.15	561.91	373723.46	373723.46	373476.31	272.00
62	N6	374791.55	1488.52	373758.65	373758.65	373758.656	255.00
Average		290920.92	276.04	290655.29	290838.73	290576.218	67.567
ARPD		0.101	-	0.029	0.0778	0.007	-

* Overall best solution is the cost associated with the solution found from different runs while performing overall execution of MACS.

Table 3.4. Total Tour Length obtained by MACS for data set -2

No	Instance	Average Solution	Standard Deviation	Best Solution	Best Run	Overall Best Solution*	Average CPU time
1	eil22_50	371.00	0.00	371	371	371	1
2	eil22_66	366.00	0.00	366	366	366	0.13
3	eil22_80	375.00	0.00	375	375	375	0.5
4	eil23_50	682.00	0.00	682	682	682	1.5

5	eil23_66	649.00	0.00	649	649	649	0.63
6	eil23_80	623.00	0.00	623	623	623	3
7	eil30_50	501.00	0.00	501	501	501	1.38
8	eil30_66	537.00	0.00	537	537	537	2
9	eil30_80	514.00	0.00	514	514	514	3
10	eil33_50	738.00	0.00	738	738	738	4.25
11	eil33_66	750.00	0.00	750	750	750	2.88
12	eil33_80	736.00	0.00	736	736	736	2
13	eil51_50	559.00	0.00	559	559	559	8.25
14	eil51_66	548.00	0.00	548	548	548	10.75
15	eil51_80	565.00	0.00	565	565	565	13.25
16	eilA76_50	739.25	0.46	739	739	739	19
17	eilA76_66	768.00	0.00	768	768	768	23.63
18	eilA76_80	782.63	3.85	781	781	781	53.5
19	eilB76_50	801.00	0.00	801	801	801	21.25
20	eilB76_66	873.13	0.35	873	873	873	25.38
21	eilB76_80	920.13	2.80	919	919	919	38.25
22	eilC76_50	713.00	0.00	713	713	713	18.75
23	eilC76_66	734.00	0.00	734	734	734	27.88
24	eilC76_80	736.13	1.55	734	734	733	36
25	eilD76_50	690.00	0.00	690	690	690	27.88
26	eilD76_66	715.00	0.00	715	715	715	28.88
27	eilD76_80	698.63	1.19	696	699	694	42.13
28	eilA101_50	834.75	2.60	831	832	831	56.13
29	eilA101_66	846.00	0.00	846	846	846	59.13
30	eilA101_80	866.88	5.14	860	860	857	88.25
31	eilB101_50	927.63	2.62	925	930	**923**	67
32	eilB101_66	1008.88	5.03	1002	1002	988	84.88
33	eilB101_80	1008.50	0.76	1008	1008	1008	73.88
Average		702.34	0.80	701.48	701.76	700.82	25.644
ARPD		0.177	-	0.076	0.11	0.003	-

* Overall best solution is the cost associated with the solution found from different runs while performing overall execution of MACS.

Table 3.5: Summary of the CPU's used in testing various heuristics and rough conversion factors (r) relative to a 2.4 GHz Intel Xeon Processor with 884 Mflops

Reference	CPU Used	Mflops	r
HTV96 and TV99 (Toth and Vigo (1996, 1999)	IBM 386/20	1.3	0.001
RTS-AMP (Wassan 2004)	Sun Sparc1000 at 50 MHz	10	0.011

BTS (Brandao , 2006)	Pentium III at 500 MHz	72.5	0.082
LNS (Ropke and Psinger, 2006)	Pentium IV at 1.5 GHz	326	0.369
MACS	Intel Xeon 2.4 GHz	884	1

Effect of trail intensities and local search schemes on MACS

In order to see the effect of trail intensities and local search schemes on MACS, we executed MACS and calculate the average solution over 8 runs under each setting. In Table 3.6, we report the overall average solution and its percentage deviation from the average best known solution for each setting. The average best known solution is the average of the best known solutions for 62 problem instances for set-1 and 33 problem instances for set-2 as reported in the literature. For each setting, we keep the parameter values the same and keep the number of ants at 25. However, we vary the stopping criteria to keep CPU time comparable. The first row in Table 3.6 reports the solution without any change in the algorithm. The second row reports the solution when 25 randomly generated vehicle routes are improved by three local search schemes used in MACS. In this case, the average solution deteriorates from 0.13 % to 0.99 % for set-1 and from 0.22 % to 1.41 % for set-2. This result indicates that the trail intensities play a crucial role for diverting the solution towards the global optimum and shows that a purely random search cannot explore the promising regions in an effective way. When sub-path exchange multi-route scheme is not used in MACS, the percentage deviation deteriorates to 0.38% for set-1 and 0.59% for set-2. Similarly, when the customer insertion/interchange multi-route scheme is not used, the percentage deviation value deteriorates to 1.35% for set-1 and 1.42% for set-2. These two results indicate that the

customer insertion/interchange multi-route scheme is more effective than the sub-path exchange multi-route scheme. It is interesting to note that a combination of the two schemes is necessary to produce better quality solutions in the same CPU time. In addition, a hybrid local search that combines different strategies such as insertion, interchange, exchange etc. seems to produce better solutions than the individual strategies.

Table3.6: Effect of trail intensities and local search on solution quality

Setting	Set-1		Set-2	
	Overall average solution	% deviation from the average best known sol.	Overall average solution	% deviation from the average best known sol.
No change from current MACS	290920.90	0.13	702.35	0.22
Local search applied on 25 randomly generated solutions	293435.69	0.99	710.70	1.41
In the absence of the sub-path exchange multi-route scheme	291661.34	0.38	704.92	0.59
In the absence of the insertion/interchange multi-route scheme	294465.94	1.35	710.73	1.42

3.4.5 Effect of number of ants on MACS

An interesting feature of MACS is that the quality of the solution can be controlled by varying the number of ants. In Table 3.7, we vary the number of ants and report the overall average solution and the average of best solution. It is evident that the solution does not deteriorate too much when we reduce the number of ants and thereby reduce the CPU time.

Table 3.7 also allows us to compare MACS with other heuristic solutions keeping the CPU time compatible. The CPU time of MACS is comparable with the CPU time of LNS when the number of ants is set to 10. The solution quality of MACS when the number of ants is 10 is still better than the solution quality of LNS in terms of the average solution for both set-1 and set-2. Similarly, the CPU time of RTS-AMP is comparable with the CPU time of MACS when number of ants is set to 5. The solution quality of MACS is still better than the solution quality of RTS-AMP under the criteria of best solution for both set-1 and set-2.

Table 3.7 : The effect of number of ants on solution quality and CPU time

Number of ants of each type	Set -1			Set-2		
	Overall average solution	Average best solution	Average CPU time (seconds)	Overall average solution	Average best solution	Average CPU time (seconds)
5	291430.90	290827.20	15.63	703.72	701.73	6.52
10	291128.80	290724.20	30.25	703.08	701.76	11.86
15	291049.90	290719.00	44.25	702.70	701.70	15.81
20	291010.80	290658.40	57.03	702.58	701.67	20.81
25	**290920.90**	**290655.29**	**67.57**	**702.35**	**701.48**	**25.64**
30	290894.10	290662.20	80.16	702.17	701.36	29.92
35	290891.00	290622.90	91.34	702.11	701.18	36.30
40	290868.10	290642.80	103.75	702.05	701.33	41.08
45	290861.10	290629.60	116.94	701.88	701.15	44.18
50	290848.40	290622.80	127.12	701.80	701.15	49.76

Table 3.8. Total Tour Length obtained by different algorithms for data set -1

No	Inst.	Size	L	B	Veh Cap	Veh No	E-MBG	E-TV	HTV96	RTS-AMP	BTS		LNS		
											Best Run	Best Sol.	Avg. Sol.	Best . Sol.	Overall Best sol.
Column	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	A1	25	20	5	1550	8	229886	229886	229886	229885.65	229886	229886	229885.66	229885.65	229885.65
2	A2	25	20	5	2550	5	180119	180119	180117	180119.21	180119	180119	180119.21	180119.21	180119.21
3	A3	25	20	5	4050	4	163405	163405	163405	163405.38	163405	163405	163405.38	163405.38	163405.38
4	A4	25	20	5	4050	3	155796	155796	155795	155796.41	155796	155796	155796.41	155796.41	155796.41
5	B1	30	20	10	1600	7	239080	239080	239077	239080.15	239080	239080	239080.16	239080.15	239080.15
6	B2	30	20	10	2600	5	198048	198048	198045	198047.77	198048	198048	198047.77	198047.77	198047.77
7	B3	30	20	10	4000	3	169372	169372	169368	169372.29	169372	169372	169372.29	169372.29	169372.29
8	C1	40	20	20	1800	7	249448	249448	250557	250556.77	250557	250557	250560.15	250556.77	250556.77
9	C2	40	20	20	2600	5	215020	215020	215019	215020.23	215020	215020	215020.23	215020.23	215020.23
10	C3	40	20	20	4150	5	199346	199346	199344	199345.96	199346	199346	199345.96	199345.96	199345.96
11	C4	40	20	20	4150	4	195367	195366	195365	195366.63	195366	195366	195366.63	195366.63	195366.63
12	D1	38	30	8	1700	12	322530	322530	322533	322530.13	322530	322530	322530.13	322530.13	322530.13
13	D2	38	30	8	1700	11	316709	316709	316711	316708.86	316709	316709	316708.86	316708.86	316708.86
14	D3	38	30	8	2750	7	239479	239479	239482	239478.63	239479	239479	239478.63	239478.63	239478.63
15	D4	38	30	8	4075	5	205832	205832	205832	205831.94	205832	205832	205831.94	205831.94	205831.94
16	E1	45	30	15	2650	7	238880	238880	238880	238879.58	238880	238880	238879.58	238879.58	238879.58
17	E2	45	30	15	4300	4	212263	212263	212262	212263.11	212263	212263	212263.11	212263.11	212263.11
18	E3	45	30	15	5225	4	206659	206659	206658	206659.17	206659	206659	206697.72	206659.17	206659.17
19	F1	60	30	30	3000	6	263173	263173	263175	264299.6	263173	263173	268430.58	267060.43	267060.43
20	F2	60	30	30	3000	7	265213	265213	265214	265214.16	265493	265213	265214.16	265214.16	265214.16
21	F3	60	30	30	4400	5	241120	241120	241487	241120.77	241120	241120	241969.77	241969.77	241969.77
22	F4	60	30	30	5500	4	233861	233861	233861	233861.85	233861	233861	235528.13	235175.20	235175.20
23	G1	57	45	12	2700	10	306305	307274†	307272	306305.4	306306	306305	306322.98	306305.40	306305.40
24	G2	57	45	12	4300	6	245441	245441†	245441	245440.99	245441	245441	245440.99	245440.99	245440.99
25	G3	57	45	12	5300	5	229507†	229507	230170	229507.48	229507	229507	230737.17	229507.48	229507.48
26	G4	57	45	12	5300	6	232521†	233184†	232646	232521.25	232521	232521	233006.36	232521.25	232521.25
27	G5	57	45	12	6400	5	221730†	221730	222025	221730.35	221730	221730	222435.96	221730.35	221730.35
28	G6	57	45	12	8000	4	-	213457	213457	213457.45	213457	213457	214090.55	213457.45	213457.45
29	H1	68	45	23	4000	6	268933†	268933	270525	268933.06	268933	268933	269467.78	268933.06	268933.06
30	H2	68	45	23	5100	5	253365	253365	253366	253365.5	253365	253365	253462.09	253365.50	253365.50

31	H3	68	45	23	6100	4	247449	247449	247449	247449.04	247449	247449	247508.59	247449.04	247449.04
32	H4	68	45	23	6100	5	250221	250221	250221	250220.77	250221	250221	250269.07	250220.77	250220.77
33	H5	68	45	23	7100	4	246121	246121	246121	246121.31	246121	246121	246767.73	246121.31	246121.31
34	H6	68	45	23	7100	5	249135	249135	249136	249135.32	249135	249135	249231.92	249135.32	249135.32
35	I1	90	45	45	3000	10	354021†	-	356381	351190.09	350435	350246	350852.85	350245.28	350245.28
36	I2	90	45	45	4000	7	309943	-	313917	309955.04	309944	309944	311016.93	309943.84	309943.84
37	I3	90	45	45	5700	5	294833†	-	297318	294507.38	294507	294507	294858.13	294507.38	294507.38
38	I4	90	45	45	5700	6	295988†	-	295988	295988.44	295988	295988	296159.12	295988.44	295988.44
39	I5	90	45	45	5700	7	301226†	-	302708	301380.7	301236	301236	301909.59	301236.00	301236.00
40	J1	94	75	19	4400	10	335006†	-	341984	335593.42	335007	335007	336522.31	335006.68	335006.68
41	J2	94	75	19	5600	8	-	-	316308	310417.21	310793	310417	312458.56	310417.21	310417.21
42	J3	94	75	19	8200	6	-	-	282535	279219.21	279306	279219	279423.74	279219.21	279219.21
43	J4	94	75	19	6600	7	-	-	298184	296533.16	296860	296553	297781.22	296533.16	296533.16
44	K1	113	75	38	4100	10	394637†	-	407939	394389.63	394974	394376	395993.78	394375.63	394375.63
45	K2	113	75	38	5200	8	362360†	-	370840	362778.95	363829	362130	362998.61	362130.00	362130.00
46	K3	113	75	38	5200	9	365693†	-	371322	366222.05	366246	365694	366218.02	365694.08	365694.08
47	K4	113	75	38	6200	7	-	-	359642	348949.39	351345	348950	349266.17	348949.39	348949.39
48	L1	150	75	75	4400	10	-	-	449271	424891.17	426401	425772	427658.80	426013.41	426013.41
49	L2	150	75	75	5000	8	-	-	407445	401652.65	402152	401228	401587.25	401228.80	401228.80
50	L3	150	75	75	5000	9	-	-	413806	403394.96	404391	402720	403029.19	402677.72	402677.72
51	L4	150	75	75	6000	7	-	-	390247	384833.66	384999	384637	385207.32	384636.34	384636.34
52	L5	150	75	75	6000	8	-	-	394576	388061.69	389044	387928	388677.62	387564.55	387564.55
53	M1	125	100	25	5200	11	-	-	407072	400574.49	400384	398593	401540.39	398913.70	398913.70
54	M2	125	100	25	5200	10	-	-	411132	397655.71	398924	396917	401724.68	399336.27	398827.67
55	M3	125	100	25	6200	9	-	-	383448	377352.81	377433	376309	378502.30	377212.23	376159.13
56	M4	125	100	25	8000	7	-	-	356311	348437.62	349091	348418	348663.06	348417.94	348417.94
57	N1	150	100	50	5700	11	-	-	428328	408723.89	409531	408101	414044.03	410789.32	409210.18
58	N2	150	100	50	5700	10	-	-	429521	408572.07	408287	408066	413124.59	409385.19	409385.19
59	N3	150	100	50	6600	9	-	-	412220	394337.86	394338	394338	399363.23	394337.86	394337.86
60	N4	150	100	50	6600	10	-	-	410694	395817.21	399029	396055	402131.56	398965.12	394788.36
61	N5	150	100	50	8500	7	-	-	389349	375347.27	376522	373477	377447.83	373476.30	373476.30
62	N6	150	100	50	8500	8	-	-	384461	377064.3	374774	374691	376612.61	373758.65	373758.65
Average Solution							-	-	295045.96	290981.8	291160.4	290764.77	291823.34	291014.7	290896.7
ARPD							-	-	1.191	0.114	0.159	0.054	0.362	0.130	0.100

* Bold letter stands for better than the best know solution

† Total travel length for the best VRPB solution (i.e., optimality is not proved) obtained by E-MBG and E-TV

Table 3.9. Total Tour Length obtained by different algorithms for data set -2

No	Instance	Size	L	B	Veh Cap	Veh No	E-MBG	E-TV	HTV99	RTS-AMP	BTS		LNS		
											Best Run	Best Sol.	Avg. Sol.	Best . Sol.	Overall Best Sol.
1	eil22_50	21	11	10	6000	3	371	371	371	371	371	371	371	371	371
2	eil22_66	21	14	7	6000	3	366	366	366	366	366	366	366	366	366
3	eil22_80	21	17	4	6000	3	375	375	375	375	375	375	375	375	375
4	eil23_50	22	11	11	4500	2	682	682	682	729	682	682	682	682	682
5	eil23_66	22	15	7	4500	2	649	649	649	658	649	649	649	649	649
6	eil23_80	22	18	4	4500	2	623	623	623	675	623	623	623	623	623
7	eil30_50	29	15	14	4500	2	501	501	501	503	501	501	501	501	501
8	eil30_66	29	20	9	4500	3	537	537	537	537	537	537	537	537	537
9	eil30_80	29	24	5	4500	3	514	514	522	514	514	514	514	514	514
10	eil33_50	32	16	16	8000	3	738	738	742	738	738	738	738	738	738
11	eil33_66	32	22	10	8000	3	750	750	753	750	750	750	750	750	750
12	eil33_80	32	26	6	8000	3	736	736	761	736	736	736	736	736	736
13	eil51_50	50	25	25	160	3	559	559	562	559	559	559	559	559	559
14	eil51_66	50	34	16	160	4	548	548	553	548	548	548	550	548	548
15	eil51_80	50	40	10	160	4	565	565	574	565	565	565	571	565	565
16	eilA76_50	75	37	38	140	6	739	739	756	739	739	739	739	739	739
17	eilA76_66	75	50	25	140	7	768	768	780	768	768	768	774	769	768
18	eilA76_80	75	60	15	140	8	781†	781†	833	801	781	781	794	783	783
19	eilB76_50	75	37	38	100	8	801	801	825	801	801	801	802	801	801
20	eilB76_66	75	50	25	100	10	873	873	891	873	873	873	875	873	873
21	eilB76_80	75	60	15	100	12	919	919	948	929	919	919	924	919	919
22	eilC76_50	75	37	38	180	5	713	713	715	713	714	713	713	713	713
23	eilC76_66	75	50	25	180	6	734	734†	745	734	734	734	739	734	734
24	eilC76_80	75	60	15	180	7	733†	733†	759	733	737	733	741	736	734
25	eilD76_50	75	37	38	220	4	690	690	691	690	690	690	696	690	690
26	eilD76_66	75	50	25	220	5	715†	715†	717	715	715	715	716	715	715
27	eilD76_80	75	60	15	220	6	694†	703†	710	694	694	694	699	695	694
28	eilA101_50	100	50	50	200	4		843†	852	835	841	834	840	831	831
29	eilA101_66	100	67	33	200	6	846	846	868	848	850	846	848	846	846
30	eilA101_80	100	80	20	200	6		916†	900	870	866	862	869	857	857
31	eilB101_50	100	50	50	112	7	933†		962	926	932	925	928	925	925
32	eilB101_66	100	67	33	112	9			1040	998	993	987	1010	989	989
33	eilB101_80	100	80	20	112	11	1022		1060	1021	1010	1008	1021	1010	1008
Average Solution									714.12	706.48	702.15	701.09	704.54	701.18	701
Average Relative Percentage Deviation (ARPD)									1.593	0.777	0.151	0.029	0.453	0.041	0.018

† Total travel length for the best VRPB solution (i.e., optimality is not proved) obtained by E-MBG and E-TV

Table 3.10: CPU time used by different algorithms for data set-1

No	Instance	HTV 96		RTS-AMP		BTS		LNS	
		Actual	Scaled	Actual	Scaled	Actual	Scaled	Actual	Scaled
1	A1	16.8	0.02	9	0.10	40	3.28	7	2.58
2	A2	11.4	0.01	6	0.07	25	2.05	8	2.95
3	A3	7.8	0.01	6	0.07	25	2.05	10	3.69
4	A4	6.2	0.01	3	0.03	14	1.15	10	3.69
5	B1	27.6	0.03	12	0.13	53	4.35	9	3.32
6	B2	14.6	0.01	4	0.04	24	1.97	10	3.69
7	B3	1.8	0.00	30	0.33	18	1.48	14	5.17
8	C1	39	0.04	46	0.51	75	6.15	14	5.17
9	C2	23.2	0.02	60	0.66	80	6.56	16	5.90
10	C3	11.8	0.01	17	0.19	37	3.03	20	7.38
11	C4	12.8	0.01	1	0.01	37	3.03	19	7.01
12	D1	49.6	0.05	19	0.21	115	9.43	12	4.43
13	D2	51.6	0.05	13	0.14	113	9.27	12	4.43
14	D3	34.8	0.03	9	0.10	136	11.15	13	4.80
15	D4	24.2	0.02	9	0.10	99	8.12	16	5.90
16	E1	45.8	0.05	27	0.30	134	10.99	18	6.64
17	E2	28.8	0.03	55	0.61	172	14.10	23	8.49
18	E3	25.8	0.03	103	1.13	123	10.09	27	9.96
19	F1	67.8	0.07	97	1.07	249	20.42	30	11.07
20	F2	73.6	0.07	62	0.68	210	17.22	29	10.70
21	F3	56	0.06	167	1.84	138	11.32	36	13.28
22	F4	45	0.05	59	0.65	201	16.48	44	16.24
23	G1	116.8	0.12	169	1.86	342	28.04	23	8.49
24	G2	69.2	0.07	61	0.67	371	30.42	28	10.33
25	G3	57.2	0.06	188	2.07	196	16.07	32	11.81
26	G4	58.2	0.06	302	3.32	183	15.01	32	11.81
27	G5	57	0.06	167	1.84	242	19.84	36	13.28
28	G6	37	0.04	301	3.31	213	17.47	42	15.50
29	H1	86.6	0.09	93	1.02	363	29.77	42	15.50
30	H2	66.2	0.07	178	1.96	398	32.64	49	18.08
31	H3	61.2	0.06	134	1.47	345	28.29	55	20.30
32	H4	67.4	0.07	469	5.16	167	13.69	53	19.56
33	H5	54	0.05	346	3.81	188	15.42	58	21.40
34	H6	61.4	0.06	130	1.43	161	13.20	57	21.03
35	I1	337.6	0.34	490	5.39	648	53.14	54	19.93
36	I2	190.4	0.19	696	7.66	542	44.44	65	23.99
37	I3	130	0.13	487	5.36	574	47.07	83	30.63
38	I4	136.4	0.14	1259	13.85	630	51.66	77	28.41

39	I5	163	0.16	936	10.30	504	41.33	75	27.68
40	J1	290.2	0.29	1101	12.11	965	79.13	58	21.40
41	J2	245.8	0.25	1223	13.45	946	77.57	67	24.72
42	J3	187.6	0.19	558	6.14	764	62.65	87	32.10
43	J4	228.6	0.23	1877	20.65	859	70.44	74	27.31
44	K1	479	0.48	887	9.76	1334	109.39	83	30.63
45	K2	330.6	0.33	1787	19.66	1302	106.76	97	35.79
46	K3	380.2	0.38	2069	22.76	1204	98.73	97	35.79
47	K4	331.6	0.33	4612	50.73	1150	94.30	111	40.96
48	L1	1115.6	1.12	1703	18.73	2366	194.01	153	56.46
49	L2	833.8	0.83	5964	65.60	2477	203.11	181	66.79
50	L3	878.8	0.88	8422	92.64	2333	191.31	176	64.94
51	L4	709.2	0.71	2275	25.03	2287	187.53	207	76.38
52	L5	767	0.77	6722	73.94	2258	185.16	211	77.86
53	M1	651.8	0.65	2754	30.29	1858	152.36	104	38.38
54	M2	566.8	0.57	4330	47.63	1913	156.87	102	37.64
55	M3	503	0.50	2178	23.96	1878	154.00	115	42.44
56	M4	447.8	0.45	3146	34.61	1858	152.36	140	51.66
57	N1	938.8	0.94	6142	67.56	2468	202.38	156	57.56
58	N2	866.6	0.87	9877	108.65	2430	199.26	155	57.20
59	N3	787.6	0.79	3828	42.11	2441	200.16	177	65.31
60	N4	848.4	0.85	11046	121.51	2298	188.44	172	63.47
61	N5	547.2	0.55	7704	84.74	2494	204.51	214	78.97
62	N6	543	0.54	16406	180.47	2468	202.38	211	77.86
Average		256.526	0.257	1835.984	20.196	815.097	66.838	71.065	26.223

Table 3.11: CPU time used by different algorithms for data set-2

No	Instance	HTV 96		RTS-AMP		BTS		LNS	
		Actual	Scaled	Actual	Scaled	Actual	Scaled	Actual	Scaled
1	eil22_50	5.10	0.01	0.80	0.01	11.00	0.90	8.00	2.95
2	eil22_66	4.80	0.01	0.39	0.00	6.00	0.49	8.00	2.95
3	eil22_80	7.00	0.01	1.00	0.01	11.00	0.90	8.00	2.95
4	eil23_50	2.60	0.00	0.03	0.00	25.00	2.05	12.00	4.43
5	eil23_66	5.50	0.01	0.06	0.00	10.00	0.82	13.00	4.79
6	eil23_80	3.90	0.01	0.15	0.00	13.00	1.07	12.00	4.43
7	eil30_50	3.30	0.00	0.06	0.00	22.00	1.80	19.00	7.01
8	eil30_66	7.40	0.01	0.57	0.01	44.00	3.61	14.00	5.16
9	eil30_80	7.50	0.01	0.52	0.01	55.00	4.51	14.00	5.16
10	eil33_50	16.40	0.02	1.00	0.01	96.00	7.87	20.00	7.38
11	eil33_66	15.80	0.02	5.00	0.06	27.00	2.21	17.00	6.27

12	eil33_80	15.90	0.02	44.00	0.50	54.00	4.43	15.00	5.53
13	eil51_50	40.40	0.06	12.00	0.14	117.00	9.60	39.00	14.38
14	eil51_66	48.50	0.07	8.00	0.09	148.00	12.14	31.00	11.43
15	eil51_80	53.10	0.08	188.00	2.13	163.00	13.37	29.00	10.69
16	eilA76_50	164.30	0.24	127.00	1.44	237.00	19.44	50.00	18.44
17	eilA76_66	148.30	0.22	219.00	2.48	259.00	21.24	44.00	16.23
18	eilA76_80	238.20	0.35	43.00	0.49	491.00	40.27	40.00	14.75
19	eilB76_50	240.70	0.35	171.00	1.93	245.00	20.09	42.00	15.49
20	eilB76_66	241.00	0.35	560.00	6.33	412.00	33.79	38.00	14.01
21	eilB76_80	240.70	0.35	954.00	10.79	609.00	49.95	38.00	14.01
22	eilC76_50	110.50	0.16	625.00	7.07	239.00	19.60	61.00	22.50
23	eilC76_66	148.70	0.22	25.00	0.28	297.00	24.36	51.00	18.81
24	eilC76_80	219.40	0.32	1204.00	13.62	451.00	36.99	48.00	17.70
25	eilD76_50	93.70	0.14	1352.00	15.29	285.00	23.37	75.00	27.66
26	eilD76_66	89.70	0.13	801.00	9.06	291.00	23.87	60.00	22.13
27	eilD76_80	190.60	0.28	3056.00	34.57	523.00	42.89	55.00	20.28
28	eilA101_50	213.50	0.31	654.00	7.40	679.00	55.69	137.00	50.52
29	eilA101_66	240.60	0.35	3358.00	37.99	563.00	46.17	100.00	36.88
30	eilA101_80	241.00	0.35	565.00	6.39	914.00	74.96	87.00	32.08
31	eilB101_50	241.60	0.36	2070.00	23.42	740.00	60.69	79.00	29.13
32	eilB101_66	241.30	0.35	5179.00	58.59	961.00	78.82	66.00	24.34
33	eilB101_80	241.60	0.36	1980.00	22.40	819.00	67.17	61.00	22.50
Average		114.624	0.169	703.169	7.954	297.485	24.398	42.152	15.545

3.6 Conclusions

This chapter presents a multi-ant colony system (MACS) algorithm for solving VRPB. MACS contains the following features:

1. Two types of ants, vehicle-ants and route-ants are used. Similarly, two types of trail intensities--vehicle trail intensity and route trail intensity--are used to construct a feasible solution.
2. An inherent part of ant colony system is the local search. Two multi-route local search schemes--the customer insertion/interchange multi-route scheme and the sub-path exchange multi-route scheme--are used.

3. While updating trail intensities, no distinction is made between the elitist ants; i.e., equal importance is given to each elitist ant. This reduces the risk of being trapped at a local minimum.
4. The trail intensities are reinitialized and elitist ants are destroyed when there is no change in the elitist ant solutions in consecutive 10 iterations. The best solution found to date is used in reinitializing the trail intensities.

Extensive computational experiment on benchmark problems has shown that on the overall proposed multi-ant colony system (MACS) gives better results than the other algorithms. In addition, MACS has produced 5 new best known solutions for the benchmark problem instances available in the literature. Another interesting feature of MACS is that the solution quality and the CPU time of MACS can be controlled by varying the number of ants. Further research can be done to improve the local search schemes in ant colony systems. Hybrid local search schemes can make the ant colony system approach more effective for the other variants of the vehicle routing problem.

CHAPTER 4

Cumulative net-pickup approach for checking the feasibility of a route in vehicle routing problem with simultaneous pickup and delivery

4.1 Introduction

The vehicle routing problem with simultaneous pickup and delivery (VRPSPD) is an extension of the classical capacitated vehicle routing problem (CVRP). In VRPSPD, a customer requires a given shipment to be delivered as well as a given load to be picked up simultaneously. Complete service (i.e., delivery and pickup) to the customer is provided by a vehicle in a single visit. The objective is to design a set of minimum cost routes so that the load on each vehicle is below its capacity at every point in the route.

The vehicle routing problem with simultaneous pickup and delivery is characterized by the following: a fleet of identical vehicles located at the depot to be used to serve customers distributed geographically in the area. Capacity of each vehicle is Q. A customer requires a given shipment to be delivered and another load to be picked up during the single visit of the vehicle. The objective is to design a set of minimum cost routes so that the total load on a vehicle is below the vehicle capacity Q at each point in the route and the total distance traveled is less than or equal to the allowable maximum distance. VRPSPD is an $\mathcal{NP}$-hard problem in strong sense because when we set either all pickup demands or all delivery demands to zero, the problem reduces to CVRP which

is known to be an $\mathcal{NP}$-hard problem. VRPSPD is closely related to the vehicle routing problem with backhauls (VRPB) which requires that all deliveries to linehaul customers must be made before any pickup from a backhaul customer. When the deliveries and pickups are allowed in any order of sequence, the problem is called vehicle routing problem with backhaul and mixed load (VRPBM). VRPBM is a special case of VRPSPD because when we set the delivery or pickup demand of each customer to zero, the problem reduces to VRPBM.

Berbeglia et al. (2007) have provided the classification scheme for a static pickup and delivery problem. In this classification scheme, the problems of VRPSPD and VRPBM fall under the category of one to one pickup and delivery problem.

In this chapter, we provide two multi-route local search schemes. A major issue in VRPSPD is checking the feasibility of a given route. We propose the cumulative net-pickup approach for checking the feasibility of a vehicle route during local search. These local search schemes and the cumulative net-pickup approach presented in this chapter are used in the ant-colony system proposed in chapter 5. Cumulative net-pickup approach is a constant time approach for checking the feasibility of a route during local search. By using an appropriate data structure, the feasibility of a single move during local search can be checked without examining the entire route. Kindervater and Savelsbergh (1997) proposed a constant time approach for checking the feasibility of the tour in a TSP problem during local search. They, also describe three quantities that are sufficient for checking the feasibility of the tour in a TSP with simultaneous pickup and delivery. The three quantities are the maximum load, minimum load and the final load.

Another constant time approach for checking the feasibility of a route in VRP is proposed by Irnich (2006). He provides a uniform framework for VRP with complex side constraints. He suggested using a resource-extension functions (REFs) to check the feasibility during local search. The properties of REFs have been described in detail by Irnich (2007). The concept of REFs is based upon the cumulative consumption of resources from the start node to a given node. Irnich (2006) has hinted that the REFs can be used in VRPSPD, by representing a customer by two nodes, one for the delivery, and one for the pickup, with an additional pairing constraint guaranteeing that both nodes are served consecutively on the same route. The REFs approach would use the calculation of cumulative load and cumulative pickup in VRPSPD.

The implementation of any approach to check the feasibility of a route must be tailored to each local search scheme. In local search, the base route is altered through a move. The challenge therefore is to develop a condition for checking the feasibility of the altered route for each type of move.

4.2. Notation and checking feasibility of a route

VRPSPD and VRPBM have the added difficulty because of the fluctuating load over the route. There is a need for an effective way to check the feasibility of a given vehicle-route within a local search scheme. Most of the papers seem to have overlooked the feasibility issue while performing the local search. The studies that consider the issue of checking feasibility are the papers by Nagy and Salhi (2005) and Bianchessi and Righini (2007). They use the load-on-arc approach to check the feasibility of a given route. In the load-on-arc approach, each arc is examined to make sure that the altered

route is feasible. Checking each arc to ensure feasibility of a route is a time consuming process. In this chapter, we propose a new approach based upon the notion of cumulative net-pickup load. Cumulative net-pickup approach is a constant time approach for checking the feasibility of a route during local search. As stated earlier, other constant time approaches available in literature are by Kindervater and Savelsbergh (1997) and by Irnich (2007). In cumulative net-pickup approach, the feasibility of a new route is checked using a single expression and thereby reducing the complexity to O(1).

In this section, we first describe the notation used in the cumulative net-pickup approach. We also provide a necessary and sufficient condition for the feasibility of a given route.

Let R_r denote the route of vehicle r and let S = { R_1, R_2,..., R_r,..., R_s,..., R_{v-1}, R_v } represent a solution to VRPSPD. Here v is the number of vehicles used in solution S. Let the number of customers in route R_r be n_r. Route R_r thus contains the list of n_r customers plus two copies of the depot numbered 0 and n_{r+1}. Now let

D_r total demand to be delivered by vehicle r over its route; i.e., over R_r.

$\sigma_r(i)$ the i^{th} customer in route R_r (i.e., the customer in the i^{th} position in the route of vehicle r).

$d\sigma_r(i)$ quantity of product to be delivered to customer $\sigma_r(i)$.

$p\sigma_r(i)$ quantity of product to be picked up from customer $\sigma_r(i)$.

$f_r(i)$ net-pickup of customer $\sigma_r(i)$.

$F_r(i)$ cumulative net-pickup by vehicle r just after visiting customer $\sigma_r(i)$.

$VL_r(i)$ the load on vehicle r after visiting customer $\sigma_r(i)$.

$DD_r(i)$ the total load delivered by vehicle r after visiting customer $\sigma_r(i)$.

$X_r(i, j)$ the maximum cumulative net-pickup encountered by vehicle r between customers $\sigma_r(i)$ and $\sigma_r(j-1)$.

$Y_r(i, j)$ the minimum cumulative net-pickup encountered by vehicle r between customers $\sigma_r(i)$ and $\sigma_r(j-1)$.

Note indexes i and j are used here to denote positions in a single route or in two different routes. In chapter 3, indexes i and j were used to refer to customer i and customer j. The following relationships must hold given the above notation:

$$f_r(i) = p\sigma_r(i) - d\sigma_r(i),$$

$$F_r(0) = 0 \quad \forall r = 1,2,...,v,$$

$$DD_r(0) = 0 \quad \forall r = 1,2,...,v,$$

$$F_r(i) = F_r(i-1) + f_r(i) \qquad \forall\ i = 1,2,...,n_r,\ \forall r = 1,2,...,v,$$

$$DD_r(i) = DD_r(i-1) + d\sigma_r(i) \qquad \forall\ i = 1,2,...,n_r,\ \forall r = 1,2,...,v,$$

$$VL_r(i) = D_r + F_r(i) \qquad \forall\ i = 0,1,2,...,n_r,\ \forall r = 1,2,...,v,$$

$$X_r(i, j) = Max\ \{ F_r(i), F_r(i+1), ..., F_r(j-1) \},$$

$$Y_r(i, j) = Min\ \{ F_r(i), F_r(i+1), ..., F_r(j-1) \}.$$

We use three special cases of $X_r(i, j)$. The first case is $X_r(0, j)$ which represents the maximum cumulative net-pickup experienced by vehicle r after starting the route at the depot and until reaching customer $\sigma_r(j-1)$. The second case is $X_r(i, n_r+1)$ which

represents the maximum cumulative net-pickup experienced by vehicle r just after visiting customer $\sigma_r(i)$ and until reaching the end of the tour. The third special case is $X_r(0, n_r+1)$ which represents the maximum cumulative net-pickup experienced by vehicle r on its entire route R_r. Similarly we use three special cases $Y_r(0, j)$, $Y_r(i, n_r+1)$, and $Y_r(0, n_r+1)$ of $Y_r(i, j)$.

Now, we explain the cumulative net-pickup with a numerical example. Consider a subset of five customers from some problem instance with known delivery demands and pickup demands as given below.

Table 4.1. Input data for a subset of customers from some problem instance

Customer	Delivery Demand	Pickup Demand
1	20	40
2	40	10
3	50	10
4	40	30
5	10	60

Consider route R_r = {0-4-1-3-5-2-0} where 0 represent the depot. Given the input values from the above table , quantities associated with route R_r in the cumulative net-pickup approach are as follows.

Table 4.2: Calculation of quantities used in the cumulative net-pickup approach

Quantity \ position	0	1	2	4	5	6	7
$\sigma_r(i)$	0	4	1	3	5	2	0
$d\sigma_r(i)$	0	40	20	50	10	40	0
$p\sigma_r(i)$	0	30	40	10	60	10	0
$f_r(i)$	0	-10	20	-40	50	-30	0
$F_r(i)$	0	-10	10	-30	20	-10	-10
$DD_r(i)$	0	40	60	110	120	160	160
$VL_r(i)$	160	150	170	130	180	150	150
$X_r(0,i)$	0	0	10	10	20	20	20
$Y_r(0,i)$	0	-10	-10	-30	-30	-30	-30

Other values in the cumulative net-pickup are calculated as follows:

$D_r = 40 + 20 + 50 + 10 + 40 = 160$

$X_r(0, n_r + 1) = Max\{0, -10, 10, -30, 20, -10, -10\} = 20$

$Y_r(0, n_r + 1) = Min\{0, -10, 10, -30, 20, -10, -10\} = -30$

We describe altered routes using prime (i.e., ') and double prime (i.e., ") as superscripts. Thus, the notations associated with an altered route will be the same as described above except for ' or " used as a superscript.

Lemma 4.1 A necessary and sufficient condition for route R_r to be feasible is that $D_r + X_r(0, n_r + 1) \leq Q$.

Proof. *Necessary condition*

Suppose that route R_r is feasible. Thus the load on vehicle r does not exceed the vehicle capacity, Q at any point in the route, i.e., $VL_r(0) \leq Q, VL_r(1) \leq Q, \ldots, VL_r(n_r) \leq Q$

Or, $Max\{ VL_r(0), VL_r(1), \ldots, VL_r(n_r - 1), VL_r(n_r) \} \leq Q$

$\Rightarrow Max\{ D_r + F_r(0), D_r + F_r(1), \ldots, D_r + F_r(n_r - 1), D_r + F_r(n_r) \} \leq Q$

$\Rightarrow D_r + Max\{ F_r(0), F_r(1), \ldots, F_r(n_r - 1), F_r(n_r) \} \leq Q$

$\Rightarrow D_r + X_r(0, n_r + 1) \leq Q$

Proof of Sufficient Condition

The proof is similar to the above except that the steps will be in the reverse order.

□

Consider route R_r = {0-4-1-3-5-2-0} for which a numerical example was presented on the last two pages. Feasibility of route R_r can be checked using lemma 1. Suppose the capacity of vehicle r is 200 (i.e., Q = 200). Then given lemma 1, route R_r is feasible since 160 + 20 ≤ 200.

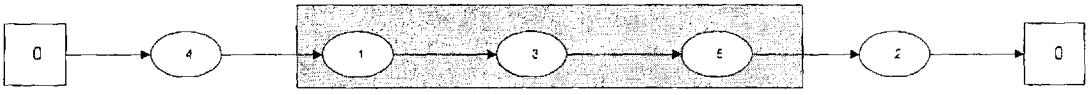
4.3. Local Search

Local search is an important part of metaheuristics including ant colony systems. In this section, we present three types of local search schemes: 2-opt scheme, the customer insertion/interchange multi-route scheme and the sub-path exchange multi-route

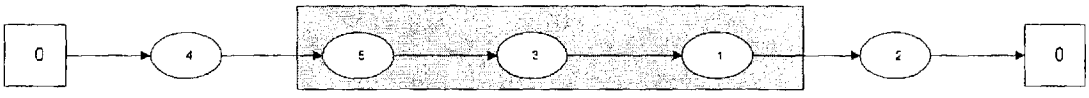
scheme. The local search schemes are used in the ant colony system (ACS) algorithm presented in chapter 5. The local search schemes are described in detail in the next subsections along with the cumulative net-pickup approach for checking the feasibility of a given route.

4.3.1 2-Opt local search scheme

The 2-Opt local search scheme was originally proposed by Lin (1965) for improving the traveling salesman problem (TSP) solution. It is one of the best-known local search schemes for TSP and it is based on the arc exchange approach. In CVRP, each vehicle route is a traveling salesman problem. The 2-opt scheme starts with a given route and breaks it at two places to generate three segments. The route is then reconstructed by reversing the middle segment. A given route is broken at each combination of two places and is updated whenever there is an improvement. The process is continued until there is no further improvement. The 2-opt local search scheme is illustrated below.



a) Before 2-opt



b) After 2-opt

Figure 4.1: 2-opt local search scheme with the reversal of the middle segment

In CVRP, a new route is always feasible while performing the 2-opt local search since the total demand of a given route does not change. However, this is not true for VRPSPD. In VRPSPD, the load on the vehicle fluctuates and it may exceed the vehicle capacity at some arc. In this thesis we define the feasible route as a route in which the load along each arc is less than or equal to the vehicle capacity and the total distance traveled is less than or equal to the allowable maximum distance. The following lemmas are useful in checking the feasibility of a new route generated during the 2-opt local search.

Lemma 4.2 Let $R_r^{'}$ represent the altered route obtained by reversing the sub-route $\{\sigma_r(i),\ldots,\sigma_r(j)\}$ in the current route R_r. Then

a) $F_r^{'}(k)=F_r(k)$ *for* $k=0,1,\ldots,(i-1)$ *and*

b) $F_r^{'}(k)=F_r(k)$ *for* $k=j, j+1,\ldots,n_r+1$.

Proof.

Part a)

The proof follows from the fact that sub-route $\{\sigma_r^{'}(0),\ldots,\sigma_r^{'}(i-1)\}$ in the altered route $R_r^{'}$ and sub-route $\{\sigma_r(0),\ldots,\sigma_r(i-1)\}$ in the current route R_r are identical.

Part b)

case b1: $k=j$,

We know that the customer at position l, i$\leq l \leq j$ in the altered route $R_r^{'}$ is the customer at position $(i+j)$ - l in the original route R_r; i.e., $\sigma_r^{'}(l) = \sigma_r(i+j-l)$. The cumulative net-pickup by the vehicle in the altered route $R_r^{'}$ after visiting customer $\sigma_r^{'}(j)$ is

$$F_r'(j) = F_r'(i-1) + \{ f_r'(i) + f_r'(i+1) + \ldots + f_r'(j-1),\ f_r'(j) \}$$

$$= F_r(i-1) + \{ f_r(i+j-i) + f_r(i+j-(i+1)) + \ldots + f_r(i+j-(j-1)) + f_r(i+j-j) \}$$

$$= F_r(i-1) + \{ f_r(j) + f_r(j-1) + \ldots + f_r(i+1) + f_r(i) \}$$

$$= F_r(i-1) + \{ f_r(i) + f_r(i+1) + \ldots + f_r(j-1) + f_r(j) \}$$

$$= F_r(j)$$

Case b2: $j+1 \le k \le n_r+1$.

We know that $\sigma_r'(l) = \sigma_r(l)$; $j+1 \le l \le k$. Also from case b1, $F_r'(j) = F_r(j)$. Hence,

$$F_r'(k) = F_r'(j) + f_r'(j+1) + \ldots + f_r'(k)$$

$$= F_r(j) + f_r(j+1) + \ldots + f_r(k)$$

$$= F_r(k)$$

□

Lemma 4.3 Let R_r' represent the altered route obtained by reversing sub-route $\{ \sigma_r(i), \ldots, \sigma_r(j) \}$ in the current route R_r. Then

a) The maximum cumulative net-pickup between customers $\sigma_r'(i)$ and $\sigma_r'(j)$ in the altered route R_r'; i.e., $X_r'(i, j+1)$ is equal to $F_r(i-1) + F_r(j) - Y_r'(i-1, j)$.

b) If original route R_r is feasible and $D_r + F_r(i-1) + F_r(j) - Y_r(i-1, j) \le Q$ then the altered route R_r' is also feasible.

Proof.

Part a)

Consider position $i + k$ with $0 \leq k \leq j - i$ in the altered route R_r'. The cumulative net-pickup after visiting customer $\sigma_r'(i+k)$ in the altered route R_r' is

$$F_r'(i+k) = F_r'(i-1) + \{ f_r'(i) + f_r'(i+1) + \ldots + f_r'(i+k-1) + f_r'(i+k) \}$$

$$= F_r'(i-1) + \{ f_r'(i) + f_r'(i+1) + \ldots + f_r'(i+k-1) + f_r'(i+k) \}$$

$$+ \{ f_r'(i+k+1) + f_r'(i+k+2) + \ldots + f_r'(j-1) + f_r'(j) \}$$

$$- \{ f_r'(i+k+1) + f_r'(i+k+2) + \ldots + f_r'(j-1) + f_r'(j) \}$$

$$= F_r'(j) - \{ f_r'(i+k+1) + f_r'(i+k+2) + \ldots + f_r'(j-1) + f_r'(j) \}$$

Note that $\sigma_r'(i+l) = \sigma_r(j-l)$ and $f_r'(i+l) = f_r(j-l)$; $l = 0,1,2,\ldots,j - i$. Also from lemma 4.2, case b1, $F_r'(j) = F_r(j)$. Hence

$$F_r'(i+k) = F_r(j) - \{ f_r(j-k-1) + f_r(j-k-2) + \ldots + f_r(i+1) + f_r(i) \}$$

$$= F_r(j) - \{ f_r(i) + f_r(i+1) + \ldots + f_r(j-k-2) + f_r(j-k-1) \}$$

$$= F_r(j) - \{ F_r(i-1) + f_r(i) + f_r(i+1) + \ldots + f_r(j-k-2) + f_r(j-k-1) \}$$

$$+ F_r(i-1)$$

$$= F_r(j) - F_r(j-k-1) + F_r(i-1)$$

$$= F_r(j) + F_r(i-1) - F_r(j-k-1)$$

The maximum cumulative net-pickup between customers $\sigma_r'(i)$ and $\sigma_r'(j)$ in the altered route R_r' is

$$X_r'(i, j+1) = Max\{ F_r'(i), F_r'(i+1), \ldots, F_r'(j-1), F_r'(j) \}$$

$$= Max\{ F_r'(i+0), F_r'(i+1), \ldots, F_r'(i+(j-i-1)), F_r'(i+(j-i)) \}$$

Now as shown before $F_r'(i+k) = \{ F_r(j) + F_r(i-1) - F_r(j-k-1)$ for $0 \le k \le j - i$. Hence

$$X_r'(i, j+1) = F_r(j) + F_r(i-1) + Max\{-F_r(j-0-1), -F_r(j-1-1), \ldots, -F_r(j-(j-i-1)-1), -F_r(j-(j-i)-1)\}$$

$$= F_r(j) + F_r(i-1) - Min\{F_r(j-1), F_r(j-2), \ldots, F_r(i), F_r(i-1)\}$$

$$= F_r(j) + F_r(i-1) - Min\{F_r(i-1), F_r(i), \ldots, F_r(j-2), F_r(j-1)\}$$

$$= F_r(i-1) + F_r(j) - Y_r(i-1, j)$$

Part b)

Note that $D_r' = D_r$ and $n_r' = n_r$. The maximum cumulative net-pickup of altered route R_r' is

$$X_r'(0, n_r'+1) = Max\{F_r'(0), \ldots, F_r'(i-1), F_r'(i), \ldots, F_r'(j), F_r'(j+1), \ldots, F_r'(n_r+1)\}$$

$$= Max\{Max\{F_r'(0), \ldots, F_r'(i-1)\}, Max\{F_r'(i), \ldots, F_r'(j)\}, Max\{F_r'(j+1), \ldots, F_r'(n_r'+1)\}\}$$

Now using lemma 4.2 and part a) of this lemma, we have

$$X_r'(0, n_r'+1) = Max\{Max\{F_r(0), \ldots, F_r(i-1)\}, F_r(i-1) + F_r(j) - Y_r(i-1, j), Max\{F_r(j+1), \ldots, F_r(n_r+1)\}\}$$

Note that route R_r is feasible. Hence from lemma 4.1, $D_r + X_r(0, n_r+1) \le Q$.

Thus, feasibility of R_r implies

$D_r + Max\{F_r(0), \ldots, F_r(i-1)\} \le Q$ and

$D_r + Max\{F_r(j+1), \ldots, F_r(n_r+1)\} \le Q$

Also, we have assumed that $D_r + F_r(i-1) + F_r(j) - Y_r(i-1, j) \leq Q$

Combining the above relationships, we have

$$D_r + Max\{Max\{F_r(0),...,F_r(i-1)\}, F_r(i-1) + F_r(j) - Y_r(i-1, j), Max\{F_r(j+1),...,$$

$$F_r(n_r+1)\} \leq Q .$$

$$\Rightarrow D_r' + X_r'(0, n_r'+1) \leq Q.$$

Hence, given lemma 4.1, route R_r' is feasible.

□

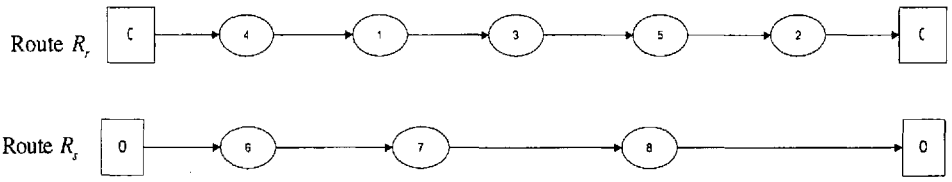
If the original route is feasible and it is altered by a single 2-opt operation, the feasibility of the altered route can be checked using part b) of lemma 4.3.

4.3.2 Customer insertion/interchange multi-route scheme

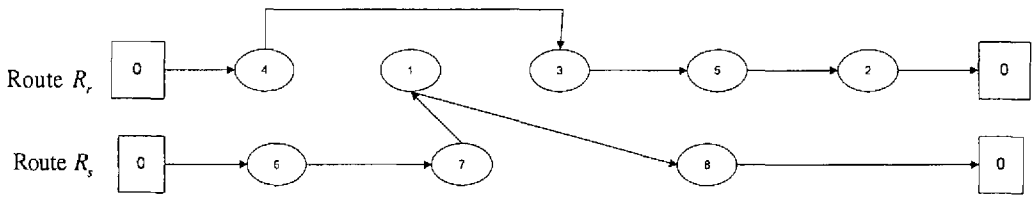
This approach uses two types of operations for a given customer. The two operations are described below for customer $\sigma_r(i)$.

1. Insertion Operation: In this operation customer $\sigma_r(i)$ is removed from the i^{th} position in route R_r and is inserted at each other position in R_r as well as at each position in route R_s, $s \neq r$. For an example of an insertion operation, see Figure 4.2. Here customer 1 from route R_r is inserted between customers 7 and 8 in route R_s.
2. Interchange Operation: In this operation customer $\sigma_r(i)$ from route R_r is shifted to his best position k in route R_s and customer $\sigma_s(k)$ from route R_s is shifted to

his best position in R_r. For an example of an interchange operation, see Figure 4.3. Here customer 1 from route R_r and customer 6 in route R_s are interchanged.

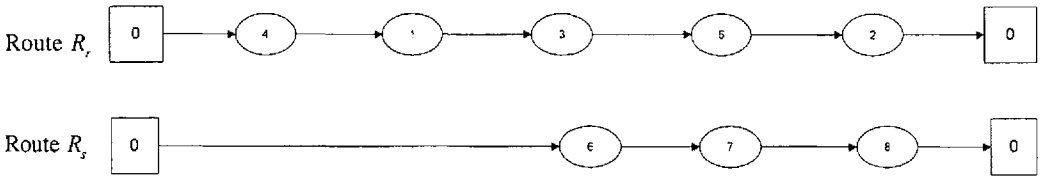


a) Before Insertion of customer 1

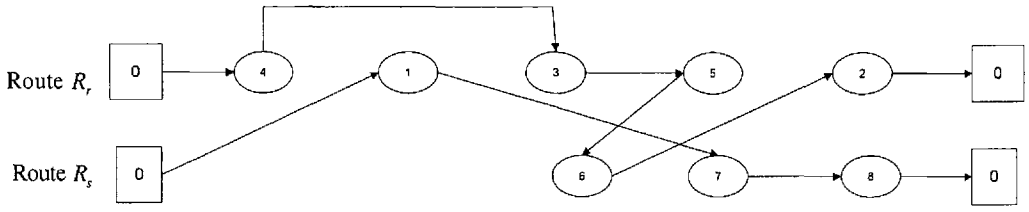


b) After Insertion of customer 1 from route R_r to route R_s

Figure 4.2 Insertion operation involving customer 1



a) Before Interchange operation



b) After Interchange operation between customers 1 and 6

Figure 4.3 Interchange operation between customers 1 and 6

Lemmas 4.4 to 4.17 presented in this subsection facilitate checking the feasibility of a new route generated by an insertion or interchange operation. These lemmas show that different expressions/approaches are needed for positive and negative net-pickup customers while checking the feasibility of a route during local search.

Lemma 4.4 Let $R_s^{'}$ represent the altered route when customer $\sigma_r(i)$ is inserted in route R_s immediately after customer $\sigma_s(j)$. Then the cumulative net-pickup $F_s^{'}(k)$ after visiting customer $\sigma_s^{'}(k)$ in the altered route $R_s^{'}$ is

Case 1: $F_s^{'}(k) = F_s(k)$ for $k = 0,1,...,j$,

Case 2: $F_s^{'}(k) = F_s(k-1) + f_r(i)$ for $k = j+1,...,n_s+1$.

Proof

Case1: for k =0,1,...,j

$$F_s^{'}(k) = f_s^{'}(0) + f_s^{'}(1) + \ldots + f_s^{'}(k-1) + f_s^{'}(k)$$

$$= f_s(0) + f_s(1) + \ldots + f_s(k-1) + f_s(k)$$

$$= F_s(k)$$

Case 2: For $k = j+1,..., n_s+1$

In this case $\sigma_s^{'}(j+1) = \sigma_r(i)$, $f_s^{'}(j+1) = f_s(i)$ and $\sigma_s^{'}(l) = \sigma_s(l-1), f_s^{'}(l) = f_s(l-1)$ for $l = j+2,...,k$. Now,

$$F_s'(k) = F_s'(j) + f_s'(j+1) + f_s'(j+2) + \ldots + f_s'(k-1) + f_s'(k)$$

$$= F_s(j) + f_r(i) + f_s(j+1) + \ldots + f_s(k-2) + f_s(k-1)$$

$$= \{ F_s(j) + f_s(j+1) + \ldots + f_s(k-2) + f_s(k-1) \} + f_r(i)$$

$$= F_s(k-1) + f_r(i)$$

□

Lemma 4.5 Consider the insertion of customer $\sigma_r(i)$ with $f_r(i) \geq 0$ in route R_s. If the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$, $0 \leq j \leq n_s$ is infeasible, then the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j-1)$ is also infeasible.

Proof. Let R_s'' be the altered route of vehicle s when customer $\sigma_r(i)$ from route R_r is inserted in route R_s after customer $\sigma_s(j)$. Note that $n_s'' = n_s + 1$. Then the maximum cumulative net-pickup experienced by vehicle s on its entire route R_s'' is

$$X_s''(0, n_s''+1) = Max\{ F_s''(0), \ldots, F_s''(j-1), F_s''(j), F_s''(j+1), \ldots, F_s''(n_s'') \}$$

$$= Max\{ F_s(0), \ldots, F_s(j-1), F_s(j), (F_s(j) + f_r(i)), \ldots, (F_s(n_s) + f_r(i)) \}$$

Since $f_r(i) \geq 0$, $F_s(j) + f_r(i) \geq F_s(j)$. Hence

$$X_s''(0, n_s''+1) = Max\{ F_s(0), \ldots, F_s(j-1), , (F_s(j) + f_r(i)), \ldots, (F_s(n_s) + f_r(i)) \}$$

Let R_s' represent the altered route of vehicle s when customer $\sigma_r(i)$ from route R_r is inserted in route R_s after customer $\sigma_s(j-1)$. Note that $n_s' = n_s + 1$. Then the maximum cumulative net-pickup experienced by vehicle s on its entire route R_s' is,

$$X_s'(0, n_s'+1) = Max\{ F_s'(0), \ldots, F_s'(j-1), F_s'(j), F_s'(j+1), \ldots F_s'(n_s') \}$$

$$= Max\{ F_s(0),\ldots,F_s(j-1),(F_s(j-1)+f_r(i)),(F_s(j)+f_r(i)),\ldots,$$

$$(F_s(n_s)+f_r(i)\}$$

$$= Max\{ X_s^{'}(0,n_s^{''}+1), F_s(j-1)+f_r(i)\}$$

Clearly $X_s^{'}(0,n_s^{'}+1) \geq X_s^{''}(0,n_s^{''}+1)$.

Since $D_s^{'} = D_s^{''}$, $D_s^{'}+X_s^{'}(0,n_s^{'}+1) \geq D_s^{''}+X_s^{''}(0,n_s^{''}+1)$.

Thus, $D_s^{''}+X_s^{''}(0,n_s^{''}+1) \geq Q$ implies that $D_s^{'}+X_s^{'}(0,n_s^{'}+1) \geq Q$. Hence, if route $R_s^{''}$ is infeasible, route $R_s^{'}$ is also infeasible.

□

Lemma 4.6 Consider the insertion of customer $\sigma_r(i)$ with $f_r(i) \geq 0$ in route R_s. If the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is infeasible, then the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(k)$ is infeasible for $0 \leq k \leq j-1$.

Proof. Lemma 4.5 implies that if the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is infeasible, then the insertion of customer $\sigma_r(i)$ after $\sigma_s(j-1)$ is also infeasible. Now the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j-1)$ is infeasible. Hence, again from lemma 4.5, the insertion of customer $\sigma_r(i)$ after $\sigma_s(j-2)$ is also infeasible. By repeating the above steps it can be proven that the insertion after customer $\sigma_s(k)$ is infeasible for $0 \leq k \leq j-1$.

□

Lemma 4.7 Consider the insertion of customer $\sigma_r(i)$ with $f_r(i) \geq 0$ in route R_s of vehicle s. If the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j-1)$ is feasible, then the insertion of customer $\sigma_r(i)$ is feasible

a) after customer $\sigma_s(j)$ and

b) after customer $\sigma_s(k)$; $j+1 \leq k \leq n_s$.

Proof.

Part a) Proof is by contradiction.

Suppose the route of vehicle s is feasible when customer $\sigma_r(i)$ is inserted after customer $\sigma_s(j-1)$ but it is not feasible when customer $\sigma_r(i)$ is inserted after customer $\sigma_s(j)$. Lemma 4.5 implies that if it is not feasible to insert customer $\sigma_r(i)$ after customer $\sigma_s(j)$, then it is also not feasible to insert customer $\sigma_r(i)$ after customer $\sigma_s(j-1)$. But the latter conclusion is contradictory to the assumption in the proof.

Part b)

We know from part a) that when the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j-1)$ is feasible, his insertion after customer $\sigma_s(j)$ is also feasible. Now insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is feasible, hence, the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j+1)$ is also feasible. By repeating the above steps it is proven that insertion of customer $\sigma_r(i)$ after $\sigma_s(k)$ is feasible for $j+1 \leq k \leq n_s$.

□

Lemma 4.8 Consider customer $\sigma_r(i)$ with $f_r(i) \geq 0$ in the route of vehicle R_r. Assume that route R_r is feasible. Let $R_r^{'}$ denote the altered route of vehicle R_r when customer $\sigma_r(i)$ is removed from his original position i and is reinserted after customer $\sigma_r(j)$. Then the altered route $R_r^{'}$ is feasible for

Case 1: $j \geq i+1$,

Case 2: $j = i-1$,

Case 3: $j \leq i-2$, provided $D_r + F_r(j) + f_r(j) \leq Q$ and the insertion of customer $\sigma_r(i)$ after customer $\sigma_r(j+1)$ is feasible.

Proof.

Case 1: Let $R_r^{''}$ represent the route of vehicle r when customer $\sigma_r(i)$ is removed from route R_r. Note that $\sigma_r^{''}(j) = \sigma_r(j)$ for $j \leq i-1$ and $\sigma_r^{''}(j) = \sigma_r(j+1)$ for $j \geq i$. If $\sigma_r(i)$ is reinserted in route $R_r^{''}$ after customer $\sigma_r^{''}(i-1)$ (i.e, $\sigma_r(i-1)$), the resulting route is the original route R_r. Also, if customer $\sigma_r(i)$ is reinserted in route $R_r^{''}$ after customer $\sigma_r^{''}(j-1)$ (i.e., $\sigma_r(j)$) for $j \geq i+1$, the resulting route is $R_r^{'}$. Now R_r is feasible. Thus inserting customer $\sigma_r(i)$ after customer $\sigma_r^{''}(i-1)$ (i.e., $\sigma_r(i-1)$) is feasible. Hence from lemma 4.7, part a), inserting customer $\sigma_r(i)$ after customer $\sigma_r^{''}(i)$ (i.e., $\sigma_r(i+1)$) is also feasible. By repeatedly applying part a) of lemma 4.7, it can be proven that insertion of customer $\sigma_r(i)$ after $\sigma_r(j)$, $j \geq i+1$ is feasible.

Case 2: Let $R_r^{''}$ again represent the route of vehicle r when customer $\sigma_r(i)$ is removed from route R_r. If $\sigma_r(i)$ is reinserted in route $R_r^{''}$ after customer $\sigma_r^{''}(i-1)$ (i.e, $\sigma_r(i-1)$), the resulting route is the original route R_r, which is assumed to be feasible.

Case 3: Now let $R_r^{''}$ denote the altered route of vehicle r when customer $\sigma_r(i)$ is removed from the original position i and is reinserted after original customer $\sigma_r(j+1)$; $j \le i-2$.

Note that $n_r^{''} = n_r$ and $D_r^{''} = D_r$. Then

$$X_r^{''}(0, n_r^{''}+1) = Max\{ F_r^{''}(0),\ldots, F_r^{''}(j), F_r^{''}(j+1), F_r^{''}(j+2), F_r^{''}(j+3),\ldots, F_r^{''}(i), F_r^{''}(i+1) \ldots, F_r^{''}(n_r^{''}) \}$$

$$= Max \{ F_r(0),\ldots, F_r(j), F_r(j+1), (F_r(j+1)+f_r(i)), (F_r(j+2)+f_r(i)),\ldots, (F_r(i-1)+f_r(i)), F_r(i+1),\ldots, F_r(n_r) \}$$

Since $F_r(j+1)+f_r(i) \ge F_r(j+1)$

$$X_r^{''}(0, n_r^{''}+1) = Max\{ F_r(0),\ldots, F_r(j), (F_r(j+1)+f_r(i)), (F_r(j+2)+f_r(i)),\ldots, (F_r(i-1)+f_r(i)), F_r(i+1),\ldots, F_r(n_r) \}$$

Now $R_r^{'}$ is the altered route when customer $\sigma_r(i)$ is removed from the original position and is reinserted after customer $\sigma_r(j)$; $j \le i-1$. Note that $n_r^{'} = n_r$ and $D_r^{'} = D_r$. The maximum cumulative net-pickup experienced by vehicle r in $R_r^{'}$ is

$$X_r^{'}(0, n_r^{'}+1) = Max\{ F_r^{'}(0),\ldots, F_r^{'}(j), F_r^{'}(j+1), F_r^{'}(j+2),\ldots, F_r^{'}(i), F_r^{'}(i+1),\ldots, F_r^{'}(n_r^{'}) \}$$

$$= Max\{ F_r(0),\ldots, F_r(j), (F_r(j)+f_r(i)), (F_r(j+1)+f_r(i)),\ldots, (F_r(i-1)+f_r(i)), F_r(i+1),\ldots, F_r(n_r) \}$$

$$= Max\{\, X_r''(0, n_r'' + 1),\ F_r(j) + f_r(i) \,\}$$

Given lemma 4.1 and since route R_r'' is feasible, $D_r'' + X_r''(0, n_r'' + 1) \leq Q$. Also, $D_r + F_r(j) + f_r(i) \leq Q$ and $D_r'' = D_r' = D_r$. Thus, $D_r + Max\{\, X_r''(0, n_r'' + 1),\ F_r(j) + f_r(i) \,\} \leq Q$ implies that $D_r' + X_r'(0, n_r' + 1) \leq Q$. Hence, given lemma 4.1, route R_r' is feasible.

□

Lemma 4.9 Consider the insertion of customer $\sigma_r(i)$ with $f_r(i) \leq 0$ in route R_s. If the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is infeasible, then the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j+1)$ is also infeasible.

Proof Let R_s'' denote the altered route of vehicle s when customer $\sigma_r(i)$ from route R_r is inserted in route R_s after customer $\sigma_s(j)$. Note that $n_s'' = n_s + 1$. Then the maximum cumulative net-pickup experienced by vehicle s on its entire route R_s'' is,

$$X_s''(0, n_s'' + 1) = Max\{\, F_s''(0), \ldots, F_s''(j), F_s''(j+1), F_s''(j+2), \ldots, F_s''(n_s'') \,\}$$

$$= Max\{\, F_s(0), \ldots, F_s(j), (F_s(j) + f_r(i)), (F_s(j+1) + f_r(i)), \ldots, (F_s(n_s) + f_r(i)) \}$$

Since $f_r(i) \leq 0$, $F_s(j) \geq F_s(j) + f_r(i)$. Hence

$$X_s''(0, n_s'' + 1) = Max\{\, F_s(0), \ldots, F_s(j), (F_s(j+1) + f_r(i)), \ldots, (F_s(n_s) + f_r(i)) \}$$

Let R_s' denote the altered route of vehicle s when customer $\sigma_r(i)$ is inserted in route R_s after customer $\sigma_s(j+1)$. Note that $D_s' = D_s''$. Then the maximum cumulative net-pickup experienced by vehicle s on its entire route R_s' is

$$X_s'(0,n_s'+1) = Max\{ F_s'(0),\ldots, F_s'(j),\ F_s'(j+1),\ F_s'(j+2),\ldots,\ F_s'(n_s') \}$$

$$= Max\{ F_s(0),\ldots,\ F_s(j),\ F_s(j+1),\ (F_s(j+1)+f_r(i)),\ldots,(F_s(n_s)+f_r(i))\}$$

$$= Max\{ X_s''(0,n_s''+1),\ F_s(j+1) \}$$

Clearly $X_s'(0,n_s'+1) \geq X_s''(0,n_s''+1)$.

Since $D_s' = D_s''$, $D_s' + X_s'(0,n_s'+1) \geq D_s'' + X_s''(0,n_s''+1)$.

Thus, $D_s'' + X_s''(0,n_s''+1) \geq Q$ implies that $D_s' + X_s'(0,n_s'+1) \geq Q$. Hence, if route R_s'' is infeasible, then route R_s' is also infeasible.

□

Lemma 4.10 Consider the insertion of customer $\sigma_r(i)$ with $f_r(i) \leq 0$ in route R_s. If the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is infeasible, then the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(k)$ is infeasible for $j+1 \leq k \leq n_s$.

Proof. Lemma 4.9 implies that if the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is infeasible then the insertion of customer $\sigma_r(i)$ after $\sigma_s(j+1)$ is also infeasible. Now since the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j+1)$ is infeasible, then the insertion of customer $\sigma_r(i)$ after $\sigma_s(j+2)$ is also infeasible. By repeating the above steps it can be proven that the insertion is infeasible for $j+1 \leq k \leq n_s$.

□

Lemma 4.11 Consider the insertion of customer $\sigma_r(i)$ with $f_r(i) \leq 0$ in route R_s of vehicle s. If the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j+1)$ is feasible, then the insertion of customer $\sigma_r(i)$ is feasible

a) after customer $\sigma_s(j)$ and

b) after customer $\sigma_s(k)$; $0 \leq k \leq j-1$.

Proof.

Part a) Proof is by contradiction:

Suppose insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j+1)$ is feasible but his insertion after customer $\sigma_s(j)$ is infeasible. Lemma 4.9 implies that if insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is infeasible, then his insertion after customer $\sigma_s(j+1)$ is also infeasible. The latter conclusion is contradictory to the assumption in the proof.

Part b)

Now we know that insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is feasible. Hence the insertion of customer $\sigma_r(i)$ after $\sigma_s(j-1)$ is feasible. By repeating the above steps it can be proven that insertion of customer $\sigma_r(i)$ after $\sigma_s(k)$ is feasible for $0 \leq k \leq j-1$.

□

Lemma 4.12 Consider customer $\sigma_r(i)$ with $f_r(i) \leq 0$ in the route of vehicle R_r. Assume that route R_r is feasible. Let R_r' denote the altered route of vehicle r when customer $\sigma_r(i)$ is removed from his original position i and is inserted after customer $\sigma_r(j)$. Then the altered route R_r' is feasible for

Case 1: $j \leq i-2$.

Case 2: $j = i-1$.

Case 3: $j \geq i+1$, provided $D_r + F_r(j) - f_r(j) \leq Q$ and the insertion of customer $\sigma_r(i)$ after $\sigma_r(j-1)$ is feasible.

Proof.

Proof is similar to proof of lemma 4.8.

□

Lemma 4.13 Consider the insertion of customer $\sigma_r(i)$ with $f_r(i) \geq 0$ in route R_s, ($s \neq r$). The insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j)$ is feasible for

Case 1: $j = n_s$, provided $D_s + d\sigma_r(i) + Max\{X_s(0, n_s+1), F_s(n_s) + f_r(i)\} \leq Q$.

Case 2: $0 < j < n_s$, provided $D_s + d\sigma_r(i) + F_s(j) + f_r(i) \leq Q$ and the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j+1)$ is feasible.

Proof.

Case 1: Let $R_s^{'}$ represent the altered route of vehicle s when customer $\sigma_r(i)$ is inserted after customer $\sigma_s(n_s)$. Note that $n_s^{'} = n_s + 1$. Then the maximum cumulative net-pickup experienced by vehicle s on its entire route $R_s^{'}$ is

$$X_s^{'}(0, n_s^{'}+1) = Max\{ F_s^{'}(0),\ F_s^{'}(1), \ldots, F_s^{'}(n_s^{'}-1),\ F_s^{'}(n_s^{'}) \}$$

$$= Max\{ F_s(0), F_s(1), \ldots, F_s(n_s), F_s(n_s) + f_r(i) \}$$

$$= Max\{ Max\{ F_s(0), F_s(1), \ldots, F_s(n_s) \}, F_s(n_s) + f_r(i) \}$$

$$= Max\{ X_s(0, n_s+1), F_s(n_s) + f_r(i) \}$$

The total demand delivered by vehicle s in route R_s' is $D_s' = D_s + d\sigma_r(i)$. Thus $D_s + d\sigma_r(i) + Max\{ X_s(0,n_s+1), F_s(n_s)+f_r(i) \} \leq Q$ implies that $D_s' + X_s'(0,n_s'+1) \leq Q$. Hence, from lemma 4.1, route R_s' is feasible.

Case 2: Let R_s'' represent the altered route of vehicle s when customer $\sigma_r(i)$ is inserted after customer $\sigma_s(j+1)$. Note that $n_s'' = n_s + 1$ and $D_s'' = D_s + d\sigma_r(i)$. Then the maximum cumulative net-pickup experienced by vehicle s on route R_s'' is

$$X_s''(0,n_s''+1) = Max\{ F_s''(0),\ldots,F_s''(j), F_s''(j+1), F_s''(j+2),\ldots,F_s''(n_s) \}$$

$$= Max\{ F_s(0),\ldots,F_s(j), F_s(j+1), (F_s(j+1)+f_r(i)),\ldots,(F_s(n_s)+f_r(i)) \}$$

Since $f_r(i) \geq 0$, $F_s(j+1)+f_r(i) \geq F_s(j+1)$. Hence

$$X_s''(0,n_s''+1) = Max\{ F_s(0),\ldots,F_s(j), (F_s(j+1)+f_r(i)),\ldots,(F_s(n_s)+f_r(i)) \}$$

Let R_s' represent the altered route of vehicle s when customer $\sigma_r(i)$ is inserted after customer $\sigma_s(j)$. Note that $n_s' = n_s + 1$ and $D_s' = D_s + d\sigma_r(i)$. Then the maximum cumulative net-pickup experienced by vehicle s on its entire route R_s' is

$$X_s'(0,n_s'+1) = Max\{ F_s'(0),\ldots,F_s'(j), F_s'(j+1), F_s'(j+2),\ldots,F_s'(n_s') \}$$

$$= Max\{ F_s(0),\ldots,F_s(j),(F_s(j)+f_r(i)), (F_s(j+1)+f_r(i)), \ldots,(F_s(n_s) + f_r(i)) \}$$

$$= Max\{ X_s''(0,n_s''+1),(F_s(j)+f_r(i)) \}$$

Now route R_s'' is feasible. Hence from lemma 4.1, $D_s'' + X_s''(0,n_s''+1) \leq Q$.

Also, $D_s^{'} = D_s^{'} = D_s + d\sigma_r(i)$ and $D_s + d\sigma_r(i) + F_s(j) + f_r(i) \le Q$. Thus, $D_s + d\sigma_r(i) +$ $Max\{ X_s^{''}(0, n_s^{''}+1), (F_s(j) + f_r(i))\} \le Q$. implies that $D_s^{'} + X_s^{'}(0, n_s^{'}+1) \le Q$.

Hence, from lemma 4.1, route $R_s^{'}$ is feasible.

□

Lemma 4.14 Consider the insertion of customer $\sigma_r(i)$ with $(f_r(i) \le 0)$ in route R_s $(s \neq r)$. The insertion of customer $\sigma_r(i)$ in R_s after customer $\sigma_s(j)$ is feasible for

Case 1: $j = 0$, provided $D_s + d\sigma_r(i) + Max\{0, X_s(0, n_s+1) + f_r(i)\} \le Q$.

Case 2: $1 \le j \le n_s$, provided $D_s + d\sigma_r(i) + F_s(j) \le Q$ and the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j-1)$ is feasible.

Proof.

Case 1: Let $R_s^{'}$ represent the altered route of vehicle s when customer $\sigma_r(i)$ is inserted after customer $\sigma_s(0)$. Note that $n_s^{'} = n_s + 1$, $D_s^{'} = D_s + d\sigma_r(i)$ and $F_s^{'}(0) = F_s(0) = 0$. Then the maximum cumulative net-pickup experienced by vehicle s on route $R_s^{'}$ is

$$\begin{aligned} X_s^{'}(0, n_s^{'}+1) &= Max\{ F_s^{'}(0), F_s^{'}(1), F_s^{'}(2), \ldots, F_s^{'}(n_s^{'}) \} \\ &= Max\{ F_s(0), (F_s(0) + f_r(i)), (F_s(1) + f_r(i)), \ldots, (F_s(n_s) + f_r(i))\} \\ &= Max\{0, Max\{ F_s(0), F_s(1), \ldots, F_s(n_s) \} + f_r(i) \} \\ &= Max\{0, X_s(0, n_s+1) + f_r(i) \} \end{aligned}$$

Thus $D_s + d\sigma_r(i) + Max\{0, X_s(0, n_s+1) + f_r(i)\} \le Q$ implies that $D_s^{'} + X_s^{'}(0, n_s^{'}+1) \le Q$. Hence given lemma 4.1, $R_s^{'}$ is feasible.

Case 2: Let R_s'' represent the altered route of vehicle s when customer $\sigma_r(i)$ is inserted after customer $\sigma_s(j-1)$. Note that $n_s'' = n_s + 1$, $D_s'' = D_s + d\sigma_r(i)$. Then the maximum cumulative net-pickup experienced by vehicle s on R_s'' is

$$X_s''(0,n_s''+1) = Max\{ F_s''(0),\ldots,F_s''(j-1), F_s''(j), F_s''(j+1),\ldots,F_s''(n_s) \}$$

$$= Max\{ F_s(0),\ldots,F_s(j-1),(F_s(j-1)+f_r(i)),(F_s(j)+f_r(i)),\ldots,(F_s(n_s) + f_r(i))\}$$

Since $f_r(i) \le 0, F_s(j-1) \ge F_s(j-1)+f_r(i)$. Hence

$$X_s''(0,n_s''+1) = Max\{ F_s(0),\ldots,F_s(j-1), (F_s(j)+f_r(i)),\ldots,(F_s(n_s) + f_r(i))\}$$

Let R_s' represent the altered route of vehicle s when customer $\sigma_r(i)$ is inserted after customer $\sigma_s(j)$. Note that $n_s' = n_s + 1$, $D_s'' = D_s' = D_s + d\sigma_r(i)$. Then the maximum cumulative net-pickup experienced by vehicle s on its entire route R_s' is

$$X_s'(0,n_s'+1) = Max\{ F_s'(0),\ldots,F_s'(j-1), F_s'(j), F_s'(j+1),\ldots,F_s'(n_s') \}$$

$$= Max\{ F_s(0),\ldots,F_s(j-1), F_s(j), (F_s(j)+f_r(i)), \ldots,(F_s(n_s) + f_r(i))\}$$

$$= Max\{ X_s''(0,n_s''+1), F_s(j) \}$$

Since R_s'' is feasible, $D_s + d\sigma_r(i) + X_s''(0,n_s''+1) \le Q$. Also $D_s + d\sigma_r(i) + F_s(j) \le Q$. Thus, $D_s + d\sigma_r(i) + Max\{ X_s''(0,n_s''+1), F_s(j) \} \le Q$ implies that $D_s' + X_s'(0,n_s'+1) \le Q$. Hence, given lemma 4.1, route R_s' is feasible.

□

Lemma 4.15 Consider an interchange operation in which customer $\sigma_r(i)$ from vehicle r is shifted to vehicle s at some position and customer $\sigma_s(k)$ from vehicle s is shifted to vehicle r at some position. Then shifting of customer $\sigma_r(i)$ on vehicle s is not feasible if

Case 1: $f_r(i) \geq 0$ and $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ X_s(0,k), (X_s(k+1,n_s+1) - f_s(k)), (F_s(n_s) - f_s(k) + f_r(i))\} \geq Q$

Case 2: $f_r(i) \leq 0$ and $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ -f_r(i), X_s(0,k), (X_s(k+1,n_s+1) - f_s(k))\} + f_r(i) \geq Q$

Proof.

Case 1: We can conclude from lemma 4.5 that the insertion of customer $\sigma_r(i)$ with $f_r(i) \geq 0$ is not feasible anywhere in the route of vehicle s if his insertion on vehicle s after customer $\sigma_s(n_s)$ is not feasible. Let R_s' represent the altered route of vehicle s when customer $\sigma_s(k)$ is removed from R_s and customer $\sigma_r(i)$ is inserted after customer $\sigma_s(n_s)$. Note that $n_s' = n_s$, $D_s' = D_s - d\sigma_s(k) + d\sigma_r(i)$. Then the maximum cumulative net-pickup of route R_s' is

$$X_s'(0,n_s'+1) = Max\{ F_s'(0),\ldots,F_s'(k-1), F_s'(k),\ldots,F_s'(n_s'-1), F_s'(n_s') \}$$

$$= Max\{ F_s(0),\ldots,F_s(k-1),(F_s(k+1) - f_s(k)),\ldots,(F_s(n_s) - f_s(k)),(F_s(n_s) - f_s(k) + f_r(i))\}$$

$$= Max\{Max\{ F_s(0),\ldots,F_s(k-1)\}, Max\{ F_s(k+1),\ldots,F_s(n_s)\} - f_s(k), (F_s(n_s) - f_s(k) + f_r(i))\}$$

$$= Max\{ X_s(0,k) ,(X_s(k+1,n_s+1) - f_s(k)), (F_s(n_s) - f_s(k) + f_r(i))\}$$

Thus, $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ X_s(0,k), (X_s(k+1,n_s+1) - f_s(k), (F_s(n_s) - f_s(k) + f_r(i)\} \geq Q$ implies that $D_s' + X_s'(0,n_s'+1) \geq Q$. Hence, given lemma 4.1, route R_s' is infeasible.

Case 2: We can conclude from lemma 4.9 that the insertion of customer $\sigma_r(i)$ with $f_r(i) \leq 0$ is not feasible anywhere in R_s if his insertion on vehicle s after customer $\sigma_s(0)$ is not feasible. Let R_s' represent the altered route of vehicle s when customer $\sigma_s(k)$ is removed from R_s and customer $\sigma_r(i)$ is inserted after customer $\sigma_s(0)$. Note that $n_s' = n_s$, $D_s' = D_s - d\sigma_s(k) + d\sigma_r(i)$. Then the maximum cumulative net-pickup of route R_s' is

$$X_s'(0,n_s'+1) = Max\{ F_s'(0), F_s'(1), F_s'(2),\ldots, F_s'(k-1), F_s'(k),\ldots, F_s'(n_s')\}$$

$$= Max\{ F_s(0),(F_s(0) + f_r(i)),(F_s(1) + f_r(i)),\ldots,(F_s(k-1) + f_r(i)), (F_s(k+1) + f_r(i) - f_s(k)),\ldots,(F_s(n_s) + f_r(i) - f_s(k))\}$$

Note that $F_s(0) = 0 = f_r(i) - f_r(i)$. Hence,

$$X_s'(0,n_s'+1) = Max\{- f_r(i), Max\{ F_s(0),\ldots, F_s(k-1)\}, Max\{ F_s(k-1),\ldots, F_s(n_s)\}- f_s(k)\} + f_r(i)$$

$$= Max\{ - f_r(i), X_s(0,k), (X_s(k+1,n_s+1) - f_s(k))\}+ f_r(i)$$

Thus, $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{- f_r(i), X_s(0,k),(X_s(k+1,n_s+1) - f_s(k))\}+ f_r(i) \geq Q$ implies that $D_s' + X_s'(0,n_s'+1) \geq Q$. Hence, given lemma 4.1 route R_s' is infeasible.

□

Lemma 4.16 Consider an interchange operation in which customer $\sigma_r(i)$ from vehicle r is shifted to vehicle s at some position and customer $\sigma_s(k)$ from vehicle s is shifted to vehicle r at some position. In this setting consider the shifting of customer $\sigma_r(i)$ with $f_r(i) \geq 0$ after customer $\sigma_s(j)$ in route R_s. The shifting of customer $\sigma_r(i)$ in R_s after customer $\sigma_s(j)$ is feasible for

Case 1. $j = n_s$, provided $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ X_s(0,k), (X_s(k+1,n_s+1) - f_s(k), (F_s(n_s) - f_s(k) + f_r(i)\} \leq Q$

Case 2. $k \leq j < n_s$, provided $D_s - d\sigma_s(k) + d\sigma_r(i) + F_s(j) + f_r(i) - f_s(k) \leq Q$ and the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j+1)$ in route R_s is feasible.

Case 3. $0 \leq j < k$, provided $D_s - d\sigma_s(k) + d\sigma_r(i) + F_s(j) + f_r(i) \leq Q$ and the shifting of customer $\sigma_r(i)$ after customer $\sigma_s(j+1)$ in route R_s is feasible.

Proof.

Case 1: Let R_s' represent the altered route of vehicle s when customer $\sigma_s(k)$ is removed from R_s and customer $\sigma_r(i)$ is inserted after customer $\sigma_s(n_s)$. Note that $n_s' = n_s$, $D_s' = D_s - d\sigma_s(k) + d\sigma_r(i)$. Then the maximum cumulative net-pickup of route R_s' is

$$X_s'(0,n_s'+1) = Max\{ F_s'(0),\ldots,F_s'(k-1), F_s'(k),\ldots,F_s'(n_s'-1), F_s'(n_s')\}$$

$$= Max\{ F_s(0),\ldots,F_s(k-1),(F_s(k+1) - f_s(k)),\ldots,(F_s(n_s) - f_s(k)),(F_s(n_s) - f_s(k) + f_r(i))\}$$

$$= Max\{Max\{ F_s(0),\ldots, F_s(k-1)\}, Max\{ F_s(k+1),\ldots, F_s(n_s)\} - f_s(k),$$

$$(F_s(n_s) - f_s(k) + f_r(i))\}$$

$$= Max\{ X_s(0,k), (X_s(k+1,n_s+1) - f_s(k)), (F_s(n_s) - f_s(k) + f_r(i))\}$$

Thus, $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ X_s(0,k), (X_s(k+1,n_s+1) - f_s(k), (F_s(n_s) - f_s(k) + f_r(i)\} \leq Q$ implies that $D_s' + X_s'(0,n_s'+1) \leq Q$. Hence, given lemma 4.1, route R_s' is feasible.

Case 2: Let R_s'' represent the altered route of vehicle s when customer $\sigma_s(k)$ is removed from R_s and customer $\sigma_r(i)$ is shifted in R_s after customer $\sigma_s(j+1)$. Note that $n_s'' = n_s$, $D_s'' = D_s - d\sigma_s(k) + d\sigma_r(i)$. Then the maximum cumulative net-pickup of vehicle s in R_s'' is

$$X_s''(0,n_s''+1) = Max\{ F_s''(0),\ldots, F_s''(k-1), F_s''(k),\ldots, F_s''(j-1), F_s''(j), F_s''(j+1),\ldots,$$

$$F_s''(n_s'')\}$$

$$= Max\{ F_s(0),\ldots, F_s(k-1), (F_s(k+1) - f_s(k)),\ldots,(F_s(j) - f_s(k)), (F_s(j+1) -$$

$$f_s(k)), (F_s(j+1) - f_s(k) + f_r(i)), \ldots, (F_s(n_s) - f_s(k) + f_r(i))\}$$

Since $f_r(i) \geq 0$, $F_s(j+1) - f_s(k) \leq F_s(j+1) - f_s(k) + f_r(i)$. Hence,

$$X_s''(0,n_s''+1) = Max\{ F_s(0),\ldots, F_s(k-1), (F_s(k+1) - f_s(k)),\ldots,(F_s(j) - f_s(k)), (F_s(j+1) -$$

$$f_s(k) + f_r(i)), \ldots, (F_s(n_s) - f_s(k) + f_r(i))\}$$

Let $R_s^{'}$ represent the altered route of vehicle s when customer $\sigma_s(k)$ is removed from R_s and customer $\sigma_r(i)$ is inserted in R_s after customer $\sigma_s(j)$. Note that $n_s^{'} = n_s$. Then the maximum cumulative net-pickup of vehicle s in $R_s^{'}$ is

$$X_s^{'}(0,n_s^{'}+1) = Max\{ F_s^{'}(0),\ldots,F_s^{'}(k-1),F_s^{'}(k),\ldots,F_s^{'}(j-1),F_s^{'}(j),F_s^{'}(j+1),\ldots,F_s^{'}(n_s^{'}) \}$$

$$= Max\{ F_s(0),\ldots,F_s(k-1),(F_s(k+1)-f_s(k)),\ldots,(F_s(j)-f_s(k)),(F_s(j)-f_s(k)+f_r(i)),(F_s(j+1)-f_s(k)+f_r(i)),\ldots,(F_s(n_s)-f_s(k)+f_r(i))\}$$

$$= Max\{ X_s^{''}(0,n_s^{''}+1),(F_s(j)-f_s(k)+f_r(i))\}$$

Given lemma 4.1 and since $R_s^{''}$ is feasible, $D_s - d\sigma_s(k) + d\sigma_r(i) + X_s^{''}(0,n_s^{''}+1) \le Q$.

Also, $D_s - d\sigma_s(k) + d\sigma_r(i) + F_s(j) + f_r(i) - f_s(k) \le Q$. Thus, $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ X_s^{''}(0,n_s^{''}+1),(F_s(j)+f_r(i)-f_s(k))\} \le Q$ implies that $D_s^{'} + X_s^{''}(0,n_s^{''}+1) \le Q$. Hence, given lemma 4.1, route $R_s^{'}$ is feasible.

Case 3: Let $R_s^{''}$ represent the altered route of vehicle s when customer $\sigma_s(k)$ is removed from R_s and customer $\sigma_r(i)$ is inserted in R_s after customer $\sigma_s(j+1)$. Note that $n_s^{''} = n_s$, $D_s^{''} = D_s - d\sigma_s(k) + d\sigma_r(i)$. Then the maximum cumulative net-pickup of vehicle s in its entire route $R_s^{''}$ is

$$X_s^{''}(0,n_s^{''}+1) = Max\{ F_s^{''}(0),\ldots,F_s^{''}(j),F_s^{''}(j+1),F_s^{''}(j+2),\ldots,F_s^{''}(k),F_s^{''}(k+1),\ldots,F_s^{''}(n_s^{''}) \}$$

$$= Max\{ F_s(0),\ldots,F_s(j),F_s(j+1),(F_s(j+1)+f_r(i)),\ldots,(F_s(k-1)+f_r(i)),(F_s(k+1)+f_r(i)-f_s(k)),\ldots,(F_s(n_s)-f_s(k)+f_r(i))\}$$

Since $f_r(i) \geq 0$, $F_s(j+1) \leq F_s(j+1) + f_r(i)$. Hence,

$$X_s''(0, n_s''+1) = Max\{ F_s(0), \ldots, F_s(j), (F_s(j+1) + f_r(i)), \ldots, (F_s(k-1) + f_r(i)), (F_s(k+1) + f_r(i) - f_s(k)), \ldots, (F_s(n_s) - f_s(k) + f_r(i))\}$$

Let R_s' represent the altered route of vehicle s when customer $\sigma_s(k)$ is removed from R_s and customer $\sigma_r(i)$ is inserted in R_s after customer $\sigma_s(j)$. Note that $n_s' = n_s$ and $D_s' = D_s - d\sigma_s(k) + d\sigma_r(i)$. Then the maximum cumulative net-pickup of vehicle s in its entire route R_s' is

$$X_s'(0, n_s'+1) = Max\{ F_s'(0), \ldots, F_s'(j-1), F_s'(j), F_s'(j+1), \ldots, F_s'(k), F_s'(k+1), \ldots, F_s'(n_s')\}$$

$$= Max\{ F_s(0), \ldots, F_s(j-1), F_s(j), (F_s(j) + f_r(i)), \ldots, (F_s(k-1) + f_r(i)), (F_s(k+1) + f_r(i) - f_s(k)), \ldots, (F_s(n_s) - f_s(k) + f_r(i))\}$$

$$= Max\{ X_s''(0, n_s''+1), (F_s(j) + f_r(i))\}$$

Given lemma 4.1 and since R_s'' is feasible, $D_s'' + X_s''(0, n_s''+1) = D_s - d\sigma_s(k) + d\sigma_r(i) + X_s''(0, n_s''+1) \leq Q$. Also $D_s - d\sigma_s(k) + d\sigma_r(i) + F_s(j) + f_r(i) \leq Q$. Thus, $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ X_s''(0, n_s''+1), (F_s(j) + f_r(i))\} \leq Q$ implies that $D_s' + X_s'(0, n_s'+1) \leq Q$. Hence, given lemma 4.1, route R_s' is feasible.

□

Lemma 4.17 Consider an interchange operation in which customer $\sigma_r(i)$ from vehicle r is shifted to vehicle s at some position and customer $\sigma_s(k)$ from vehicle s is shifted to vehicle r at some position. In this setting, consider shifting of customer $\sigma_r(i)$ with $f_r(i)$

≤ 0 after customer $f_s(j)$ in route R_s. If $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ - f_r(i), X_s(0,k), (X_s(k+1,n_s+1) - f_s(k))\} + f_r(i) \leq Q$, then the shifting of customer $\sigma_r(i)$ in route R_s after customer $\sigma_s(j)$ is feasible for

Case 1. $j = 0$, provided $D_s - d\sigma_s(k) + d\sigma_r(i) + Max\{ - f_r(i), X_s(0,k), (X_s(k+1,n_s+1) - f_s(k))\} + f_r(i) \leq Q$,

Case 2. $0 < j < k$, provided $D_s - d\sigma_s(k) + d\sigma_r(i) + F_s(j) \leq Q$ and the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j-1)$ in route R_s is feasible,

Case 3. $k \leq j < n_s$, provided $D_s - d\sigma_s(k) + d\sigma_r(i) + F_s(j) - f_s(k) \leq Q$ and the insertion of customer $\sigma_r(i)$ after customer $\sigma_s(j-1)$ in route R_s is feasible.

Proof.

Proof is similar to proof of lemma 4.16

□

In order to reduce CPU time, the following neighborhood criterion is used before performing an insertion/interchange operation. Let Γ be some fixed number and let N_Γ be the set of Γ nearest customers (in term of the distance) of customer $\sigma_r(i)$. The interchange operation between customer $\sigma_r(i)$ and customer $\sigma_s(k)$ is performed only if at least one of the Γ neighbors of customer $\sigma_r(i)$ is served by vehicle s and at least one of the Γ neighbors of customer $\sigma_s(k)$ is served by vehicle r. Similarly, the insertion of customer $\sigma_r(i)$ in the route of vehicle s is performed only if one of the Γ neighbors of customer $\sigma_r(i)$ is served by vehicle s.

The basic steps in the local search are:

Step 1. Initialize $\Omega = \{1,2,\ldots,n\}$.

Step 2. Randomly choose a customer, say $\sigma_r(i)$, in R_r from set Ω. Remove customer $\sigma_r(i)$ from set Ω.

Step 3. Perform the insertion of customer $\sigma_r(i)$ at his best position in R_r or in R_s, $s \neq r$ and evaluate the total tour length of the resulting solution.

Step 4. Consider the interchange of customer $\sigma_r(i)$ in R_r and customer $\sigma_s(k)$ in R_s, $s \neq r$; $k = 1,2,\ldots,n_s$. Identify the best interchange customer and the corresponding solution.

Step 5. Choose the best solution obtained from steps 3 and 4 and compare it with the current solution. If the best solution from steps 3 and 4 is better than the current solution, then update the current solution.

Step 6. If $\Omega = \phi$ then stop and report the current solution, else go to step 2.

We illustrate steps 3 and 4 with a numerical example. Consider a problem where the total number of customers is 8. Let the current complete solution be S = {{0-4-1-3-5-2-0}, {0-6-7-8-0}}. Thus the current solution consists of two vehicle routes R_r = {0-4-1-3-5-2-0} with tour length 300 and route R_s = {0-6-7-8-0} with tour length 200 giving us the total tour length of 500. Let us assume that customer $\sigma_r(2) = 1$ is chosen for evaluation in step 2 of the algorithm. We consider the depot to be at position 0. Then steps 3 and 4 are as follows:

Step 3: Perform the insertion of customer 1 at every other position in R_r and at every position in R_s. Evaluate the total tour length L of the resulting solution as follows:

S1 = {{0-**1**-4-3-5-2-0}, {0-6-7-8-0}} $L = 310 + 200 = 510$

S2 = {{0-4-3-**1**-5-2-0}, {0-6-7-8-0}} $L = 320 + 200 = 520$

S3 = {{0-4-3-5-**1**-2-0}, {0-6-7-8-0}} $L = 310 + 200 = 510$

S4 = {{0-4-3-5-2-**1**-0}, {0-6-7-8-0}} $L = 305 + 200 = 505$

S5 = {{0-4-3-5-2-0}, {0-**1**-6-7-8-0}} $L = 250 + 270 = 520$

S6 = {{0-4-3-5-2-0}, {0-6-**1**-7-8-0}} $L = 250 + 280 = 530$

S7 = {{0-4-3-5-2-0}, {0-6-7-**1**-8-0}} $L = 250 + 240 = 490$

S8 = {{0-4-3-5-2-0}, {0-6-7-8-**1**-0}} $L = 250 + 260 = 510$

Since S7 gives us the best solution, we keep it for the comparison in step 5.

Step 4: Consider now the interchange of customer **1** with all other customers form route R_s. Suppose we consider the interchange between customer **1** and customer 6 of route R_s. First, find out the best insertion place for customer 1 in route R_s. Here,

R_s^1 = {0-**1**-7-8-0}, $L_s = 210$

R_s^2 = {0-7-**1**-8-0}, $L_s = 230$

R_s^3 = {0-7-8-**1**-0}, $L_s = 240$

The best insertion place for customer 1 in route R_s is position 1 (we consider the depot to be at position 0). Now find out the best insertion place for customer 6 in route R_r. Here,

R_r^1 = {0-**6**-4-3-5-2-0}, L_r = 320

R_r^2 = {0-4-**6**-3-5-2-0}, L_r = 330

R_r^3 = {0-4-3-**6**-5-2-0}, L_r = 300

R_r^4 = {0-4-3-5-**6**-2-0}, L_r = 280

R_r^5 = {0-4-3-5-2-**6**-0}, L_r = 290

The best insertion place for customer 6 in route R_r is position 4. Evaluate the solution obtained by shifting customer 1 from route R_r to position 3 in route R_s and by shifting customer 6 from route R_s to position 4 in R_r. The resulting solution is S = {{0-4-3-5-**6**-2-0},{0-**1**-7-8-0}} with the total tour length L = 280 + 210 = 490.

Similarly, the interchange of customer 1 with other customers in route R_s is considered. We identify the best interchange customer for customer 1 and the corresponding solution.

Step 5: Chose the best solution obtained from steps 3 and 4 and compare it with the current solution.

Checking the feasibility of an insertion operation

In step 3 of the local search, an insertion operation for customer $\sigma_r(i)$ is performed to obtain his best feasible insertion place in the complete solution. According to lemma 4.5, the insertion of customer $\sigma_r(i)$ with $f_r(i) \geq 0$ is not feasible anywhere in route R_s if his insertion is not feasible in route R_s after customer $\sigma_s(n_s)$. Thus, the feasibility is checked in reverse order starting with the insertion after customer $\sigma_s(n_s)$ to the insertion after customer $\sigma_s(0)$. The process of checking the feasibility is stopped as soon as an infeasible insertion is encountered. From lemma 4.6, the insertion of customer $\sigma_r(i)$ at any point before customer $\sigma_s(j)$ will be infeasible if his insertion after customer $\sigma_s(j)$ is infeasible.

Similarly, for customer $\sigma_r(i)$ with $f_r(i) \leq 0$, feasibility is checked in the forward order starting with the insertion after customer $\sigma_s(0)$. Note that lemma 4.9 implies that the insertion is not feasible at any point in route R_s if the insertion is not feasible after customer $\sigma_s(0)$. The process of checking the feasibility of an insertion operation is stopped as soon as an infeasible insertion is encountered.

The following four situations arises in an insertion of customer $\sigma_r(i)$:

Case 1. Customer $\sigma_r(i)$ with $f_r(i) \geq 0$ is inserted in the same route R_r.

In this case the insertion of customer $\sigma_r(i)$ is always feasible anywhere after customer $\sigma_r(i-1)$. The feasibility of the insertion before customer $\sigma_r(i-1)$ is

checked in a reverse order starting from the insertion after customer $\sigma_r(i-2)$. We use the condition given in case 3 of lemma 4.8 to check the feasibility here.

Case 2. Customer $\sigma_r(i)$ with $f_r(i) \leq 0$ is inserted in the same route R_r.

In this case the insertion of customer $\sigma_r(i)$ is always feasible anywhere before customer $\sigma_r(i)$. The feasibility of the insertion after customer $\sigma_r(i)$ is checked in forward order starting from the insertion after customer $\sigma_r(i+1)$ using the condition given in case 3 of lemma 4.12.

Case 3. Customer $\sigma_r(i)$ with $f_r(i) \geq 0$ is inserted in another route R_s.

In this case the feasibility is checked in reverse order starting from the insertion after customer $\sigma_s(n_s)$ to the insertion after customer $\sigma_s(0)$ using conditions given in lemma 4.12.

Case 4. Customer $\sigma_r(i)$ with $f_r(i) \geq 0$ is inserted in another route R_s.

In this case, the feasibility is checked in forward order starting from the insertion after customer $\sigma_s(0)$ to the insertion after customer $\sigma_s(n_s)$ using conditions given in lemma 4.14.

Checking the feasibility in an Interchange Operation

In step 4 of the local search, an interchange operation between customers $\sigma_r(i)$ and $\sigma_s(k)$ is performed. In the interchange operation, customer $\sigma_r(i)$ from vehicle r is placed in the route of vehicle s at his best insertion place and customer $\sigma_s(k)$ from vehicle s is placed in the route of vehicle r at his best insertion place. Thus an insertion

operation is performed for both customers $\sigma_r(i)$ and $\sigma_s(k)$. Lemma 4.15 provides the basic condition for the interchange operation between customers $\sigma_r(i)$ and $\sigma_s(k)$. The interchange operation is considered only when the infeasibility condition of lemma 4.15 is not met for both customers $\sigma_r(i)$ and $\sigma_s(k)$. The feasibility condition of the insertion operation for $f_r(i) \geq 0$ is checked using the condition given in lemma 4.16 in reverse order starting with the insertion after customer $\sigma_s(n_s)$. The feasibility condition of an insertion operation for customer $\sigma_r(i)$ with $f_r(i) \leq 0$ is checked using the condition given in lemma 4.17 in the forward order starting with the insertion after customer $\sigma_s(0)$.

The customer insertion/interchange multi-route scheme is similar to the insertion/exchange heuristics used by other researchers (e.g., Osman (1993), Van Breedam (1995), Kindervater and Savelsbergh (1997)). Our insertion operation is similar to the previous work but our interchange operation differs in the way the customers are placed in new routes. In the previous work, the positions of customer $\sigma_r(i)$ from vehicle r and customer $\sigma_s(k)$ from vehicle s are interchanged. In our approach, customer $\sigma_r(i)$ from vehicle r is placed in the route of vehicle s at his best insertion place. Similarly, customer $\sigma_s(k)$ from vehicle s is placed in the route of vehicle r at his best insertion place.

The cost of a solution generated by an interchange or insertion operation can be determined in constant time. In the absence of the customer neighborhood criteria, the theoretical computational complexity of one iteration of the insertion/interchange multi-route scheme can be calculated as follows.

The insertion operation has complexity of $O(n^2)$. The interchange operation considers a customer on a given route for interchange with each customer from another route. Hence, the number of pairs for possible interchanges are of order $O(n^2)$. Each customer from a given pair is placed at his/her best insertion place. Thus, the insertion of a customer is evaluated at a maximum of n places to find his/her best insertion place. In a given pair, finding the best insertion place for one customer is independent of finding the best insertion for the second customer. Therefore, the overall complexity of an insertion/interchange multi-route scheme is

$O(n^2) + O(n^2) \times (O(n) + O(n)) = O(n^2) + O(n^3) = O(n^3)$.

The introduction of the customer neighborhood criterion should reduce CPU time in practice. However, in the worst-case scenario, there may still be n^2 pairs of customers to be considered even with the customer neighborhood criterion. Hence, the theoretical computational complexity of the scheme remains the same with or without customer neighborhood criterion.

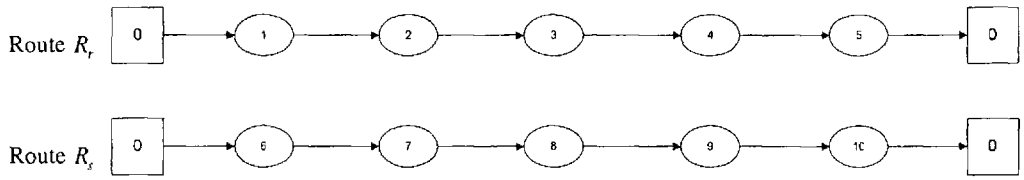
4.3.3 Sub-path *exchange* multi-route scheme

The customer insertion/interchange multi-route scheme considers shifting a given customer to another vehicle route. The sub-path exchange multi-route scheme can shift more than one customer from a given route to another route. Let S = { $R_1, R_2, \ldots, R_r, \ldots, R_s, \ldots, R_{v-1}, R_v$ } represent a solution to VRPSPD. The sub-path exchange multi-route scheme considers two routes R_r and R_s and combines them in two new routes R_r' and R_s' to produce a new solution S' = { $R_1, R_2, \ldots, R_r', \ldots, R_s', \ldots R_{v-1}, R_v$ }.

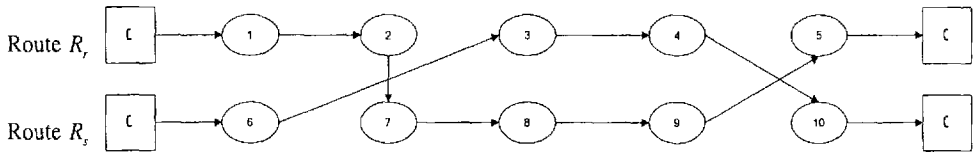
Consider sub-path $\{\sigma_r(i),...,\sigma_r(j)\}$ belonging to route R_r and sub-path $\{\sigma_s(k),...,\sigma_s(l)\}$ belonging to R_s. The new route R_r' is obtained by replacing sub-path $\{\sigma_r(i),...,\sigma_r(j)\}$ by either sub-path $\{\sigma_s(k),...,\sigma_s(l)\}$ or reverse sub-path $\{\sigma_s(l),...,\sigma_s(k)\}$. Similarly the new route R_s' is obtained by replacing sub-path $\{\sigma_s(k),...,\sigma_s(l)\}$ by either sub-path $\{\sigma_r(i),...,\sigma_r(j)\}$ or the reverse sub-path $\{\sigma_r(j),...,\sigma_r(i)\}$. The total delivery demand D_s' of the new route R_s' is calculated as follows:

$$D_s' = D_s + (DD_r(j) - DD_r(i-1)) - (DD_s(l) - DD_s(k-1))$$

For an example, consider the exchange between sub-path {3-4} from route R_r and sub-path {7-8-9} from route R_s as shown in figure 4.4.



a) Before sub-path exchange



b) After sub-path exchange

Figure 4.4 An illustration of sub-path exchange multi-route scheme

The maximum cumulative net-pickup of the new route $R_s^{'}$ is calculated using the following lemmas

Lemma 4.18 When sub-path $\{\sigma_s(k),...,\sigma_s(l)\}$ of vehicle s is replaced by sub-path $\{\sigma_r(i),...,\sigma_r(j)\}$, $(i \le j)$ of vehicle r, the maximum cumulative net-pickup of the altered route $R_s^{'}$ is $Max\ \{X_s(0,k),\ (F_s(k-1)+X_r(i,j+1)-F_r(i-1)),\ (F_s(k-1)+F_r(j)-F_r(i-1)+X_s(l+1,n_s+1)-F_s(l))\}$.

Proof.

Let $n1$ be the total number of customers in sub-path $\{\sigma_s(k),...,\sigma_s(l)\}$, $n2$ be the total number of customers in sub-path $\{\sigma_r(i),...,\sigma_r(j)\}$ and $n_s^{'}$ be the total number of customers in $R_s^{'}$. Note that $n1 = l - k + 1$, $n2 = j - i + 1$ and $n_s^{'} = n_s - n1 + n2$. The maximum cumulative net-pickup of the altered route $R_s^{'}$ is

$$X_s^{'}(0,n_s^{'}+1) = Max\{F_s^{'}(0),...,F_s^{'}(k-1),F_s^{'}(k),...,F_s^{'}(k+n2),F_s^{'}(k+n2+1),...,F_s^{'}(n_s^{'})\}$$

$$= Max\{Max\{F_s^{'}(0),...,F_s^{'}(k-1)\},Max\{F_s^{'}(k),...,F_s^{'}(k+n2)\},$$

$$Max\{F_s^{'}(k+n2+1),...,F_s^{'}(n_s^{'})\}\}$$

Now consider each term in the above expression separately. Note

$$Max\{F_s^{'}(0),...,F_s^{'}(k-1)\} = Max\{F_s(0),...,F_s(k-1)\} = X_s(0,k)$$

Also,

$$Max\{F_s^{'}(k),...,F_s^{'}(k+n2)\}$$

$$= Max\{(F_s(k-1)+f_r(i)),...,(F_s(k-1)+f_r(i)+f_r(i+1)+...+f_r(j))\}$$

$$= F_s(k-1) + Max\{ (f_r(i)),\ldots, (f_r(i)+f_r(i+1)+\ldots+f_r(j))\}$$

$$= F_s(k-1) + Max\{ (F_r(i) - F_r(i-1)),\ldots,(F_r(j) - F_r(i-1))\}$$

$$= F_s(k-1) + Max\{ F_r(i),\ldots, F_r(j) \} - F_r(i-1)$$

$$= F_s(k-1) + X_r(i, j+1) - F_r(i-1)$$

And

$Max\{ F_s'(k+n2+1) ,\ldots, F_s'(n_s') \}$

$$= Max\{ (F_s'(k+n2) + f_s(l+1)),\ldots,(F_s'(k+n2) + f_s(l+1) + f_s(l+2) +\ldots + f_s(n_s))\}$$

$$= F_s'(k+n2) + Max\{ (f_s(l+1)),\ldots,(f_s(l+1) + f_s(l+2) +\ldots + f_s(n_s))\}$$

$$= F_s'(k+n2) + Max\{ (F_s(l+1) - F_s(l)),\ldots,(F_s(n_s) - F_s(l))\}$$

$$= (F_s(k-1) + f_r(i)+\ldots+f_r(j)) + Max\{ F_s(l+1),\ldots, F_s(n_s) \} - F_s(l)$$

$$= (F_s(k-1) + F_r(j) - F_r(i-1)) + X_s(l+1, n_s+1) - F_s(l)$$

Combining the above three terms completes the proof.

□

Lemma 4.19 Let R_s'' represent the route of vehicle s in which sub-path $\{\sigma_s(k),\ldots,\sigma_s(l)\}$ of vehicle s is replaced by reverse sub-path $\{\sigma_r(j),\ldots,\sigma_r(i)\}$ of vehicle r. Then the maximum cumulative net-pickup of route R_s'' is $Max\{ X_s(0,k), (F_s(k-1) + F_r(j) - Y_r(i-1, j)), (F_s(k-1) + F_r(j) - F_r(i-1) + X_s(l+1, n_s+1) - F_s(l))\}$.

Proof.

As in lemma 4.18, let $R_s^{'}$ represent the route of vehicle s in which sub-path $\{ \sigma_s(k),...,\sigma_s(l) \}$ of vehicle s is replaced by sub-path $\{ \sigma_r(i),...,\sigma_r(j) \}$ of vehicle r. Let $n1$ be the total number of customers in sub-path $\{ \sigma_s(k),...,\sigma_s(l) \}$, $n2$ be the total number of customers in sub-path $\{ \sigma_r(i),...,\sigma_r(j) \}$ and $n_s^{'}$ be the total number of customers in route $R_s^{'}$. Note that $n1 = l - k + 1$, $n2 = j - i + 1$ and $n_s^{'} = n_s - n1 + n2$. The maximum cumulative net-pickup of the altered route $R_s^{''}$ is

$$X_s^{''}(0, n_s^{''}+1) = Max\{ F_s^{''}(0),\ldots,F_s^{''}(k-1), F_s^{''}(k),\ldots,F_s^{''}(k+n2), F_s^{''}(k+n2+1),\ldots, F_s^{''}(n_s^{''}) \}$$

$$= Max\{Max\{ F_s^{''}(0),\ldots,F_s^{''}(k-1) \}, Max\{ F_s^{''}(k),\ldots,F_s^{''}(k+n2) \}, Max\{ F_s^{''}(k+n2+1),\ldots, F_s^{''}(n_s^{''}) \}\}$$

The maximum cumulative net-pickup for route $R_s^{'}$ is derived in lemma 4.18. If sub route $\{ \sigma_s^{'}(k),...,\sigma_s^{'}(k+n2) \}$ in route $R_s^{'}$ is reversed, then the resulting route is $R_s^{''}$. Thus given lemma 4.2 we get following expression.

$$X_s^{''}(0, n_s^{''}+1) = Max\{Max\{ F_s^{'}(0),\ldots,F_s^{'}(k-1) \}, Max\{ F_s^{''}(k),\ldots,F_s^{''}(k+n2) \}, Max\{ F_s^{'}(k+n2+1),\ldots, F_s^{'}(n_s^{''}) \}\}$$

First consider route $R_s^{'}$. The minimum cumulative net-pickup of the middle path $\{ \sigma_s^{'}(k-1),...,\sigma_s^{'}(k+n2-1) \}$ is

$$Y_s^{'}(k-1, k+n2) = Min\{ F_s^{'}(k-1), F_s^{'}(k),\ldots,F_s^{'}(k+n2-1) \}$$

$$= Min\{ F_s(k-1), (F_s(k-1)+f_r(i)),\ldots,(F_s(k-1)+f_r(i)+\ldots+f_r(j-1)) \}$$

$$= F_s(k-1) + Min\{0,(f_r(i)),\ldots,(f_r(i)+\ldots+f_r(j-1))\}$$

$$= F_s(k-1) + Min\{\ (F_r(i-1) - F_r(i-1)),\ (F_r(i) - F_r(i-1)),\ldots,(F_r(j-1)\ -$$

$$F_r(i-1))\}$$

$$= F_s(k-1) + Min\{\ F_r(i-1),\ F_r(i),\ldots,F_r(j-1)\ \} - F_r(i-1)$$

$$= F_s(k-1) + Y_r(i-1,\ j) - F_r(i-1)$$

Applying lemma 4.3 to routes $R_s^{''}$ and $R_s^{'}$, the maximum cumulative net-pickup for the middle sub-path of route $R_s^{''}$ is

$$Max\{\ F_s^{''}(k),\ldots,F_s^{''}(k+n2)\ \} = X_s^{''}(k,k+n2+1)$$

$$= F_s^{'}(k-1) + F_s^{'}(k+n2) - Y_s^{'}(k-1,k+n2)$$

Given the definition of $R_s^{'}$ and given the expression for $Y_s^{'}(k-1,k+n2)$,

$$X_s^{''}(k,k+n2+1) = F_s(k-1) + (F_s(k-1) + F_r(j) - F_r(i-1)) - (F_s(k-1) + Y_r(i-1,\ j) -$$

$$F_r(i-1))$$

$$= F_s(k-1) + F_r(j) - Y_r(i-1,\ j)$$

Also from lemma 4.18,

$$Max\{\ F_s^{'}(0),\ldots,F_s^{'}(k-1)\ \} = X_s(0,k) \text{ and}$$

$$Max\{\ F_s^{'}(k+n2+1)\ ,\ldots,F_s^{'}(n_s^{'})\ \} = F_s(k-1) + F_r(j) - F_r(i-1) + X_s(l+1,n_s+1) - F_s(l)$$

Substituting the above, we have

$$X_s^{''}(0,n_s^{''}+1) = Max\{\ X_s(0,k),\ (F_s(k-1) + F_r(j) - Y_r(i-1,\ j)),\ (F_s(k-1) + F_r(j) -$$

$$F_r(i-1) + X_s(l+1,n_s+1) - F_s(l))\}$$

□

We use the customer neighborhood criteria to reduce the CPU time. The exchange of sub-path $\{ \sigma_r(i),...,\sigma_r(j) \}$ of vehicle r with sub-path $\{ \sigma_s(k),...,\sigma_s(l) \}$ of vehicle s is considered only if customer $\sigma_s(k)$ is among the 5 nearest neighbors of customer $\sigma_r(i)$.

Each possible sub-path from route R_r and each possible sub-path from route R_s (subject to the nearest neighborhood criteria) is considered for possible exchange and the best pair is used to produce new routes R_r' and R_s'.

Let $c[a , b]$ be the distance between customers a to b. The evaluation of the cost (i.e., total tour length) of route S' produced by an exchange of sub-paths can be performed in constant time; i.e., $Cost(S')$

$= Cost(S) - c[\sigma_r(i-1),\sigma_r(i)] - c[\sigma_r(j),\sigma_r(j+1)] - c[\sigma_r(k-1),\sigma_r(k)] - c[\sigma_r(l),\sigma_r(l+1)] + Min\{ c[\sigma_r(i-1),\sigma_s(k)] + c[\sigma_s(l),\sigma_r(j+1)], c[\sigma_r(i-1), \sigma_s(l)] + c[\sigma_s(k),\sigma_r(j+1)] \} + Min\{ c[\sigma_s(l-1),\sigma_r(i)] + c[\sigma_r(j),\sigma_s(k+1)], c[\sigma_s(l-1),\sigma_r(j)] + c[\sigma_r(i),\sigma_s(k+1)] \}$

The number of sub-paths starting from a given customer is of order O(n). There are n customers. Hence the total number of sub-paths is of order O(n^2). Any of these n^2 sub-paths can be exchanged with the other n^2 -1 sub-paths. Thus, the overall complexity of the sub-path exchange multi-route scheme in the absence of the neighborhood criteria is:

$$O(n^2) \times O(n^2 - 1) = O(n^4).$$

With the customer neighborhood criterion, a given sub-path can be exchanged with a maximum of $\Gamma \times n$ sub-paths. Here Γ is a constant. Thus, when a neighbourhood criterion is used, the overall complexity of the sub-path exchange multi-route scheme is:

$$O(n^2) \times O(\Gamma \times n) = O(n^3).$$

4.4 Conclusions

The vehicle routing problem with simultaneous pickup and delivery (VRPSPD) is important given the need for integrating forward and reverse flows of materials in supply chains. This chapter presents two multi-route local search schemes for VRPSPD. The local search is an important part of a metaheuristic approach. The local search schemes developed in this chapter are used in the ACS algorithm described in chapter 5.

While performing local search, insuring the feasibility of a new route is a major issue in VRPSPD given the fluctuating load on the vehicle. We have proposed the cumulative net-pickup approach for checking the feasibility of a route. For each move of the three local search schemes, we have developed a condition for checking the feasibility of the altered route. Thus the check of feasibility can be carried out in constant time. Although other general constant time approaches such as REFs are available in the literature, to our knowledge this is the first time an approach is implemented to VRPSPD.

The cumulative net-pickup approach for VRPSPD has the following special features: 1) The insertion scheme for a positive net-pickup customer is completely different from the insertion scheme for a negative net-pickup customer; 2) It provides a condition for stopping the insertion and pruning the local search. It appears that the above

two features are not available with the other constant time approaches described in section 4.1.

Although the cumulative net-pickup approach described in this chapter is mainly designed for VRPSD, it is also applicable to VRPBM given that VRPBM is a special case of VRPSPD. In addition, the local search schemes and the corresponding conditions for checking the feasibility of the altered routes can be used in other metaheuristic approaches.

CHAPTER 5

An Ant colony system (ACS) for Vehicle routing problem with simultaneous pickup and delivery

5.1 Introduction

Chapter 4 introduced the cumulative net-pickup approach for checking feasibility of a route in the vehicle routing problem with simultaneous pickup and delivery (VRPSPD). In this chapter, we use an ant colony system (ACS) algorithm to solve VRPSPD. Ant colony system (ACS) is an algorithmic approach inspired by the foraging behavior of real ants. Artificial ants are used to construct a solution for the problem by using the pheromone information from previously generated solutions. See Stuetzle and Hoos (2000) for details on the application of ACS to combinatorial optimization problems. In comparison to VRPB, the number of available vehicles is not fixed in VRPSPD. For this reason, we have used only a single type of ant in ACS.

5.2 Ant Colony System for VRPSPD

Basic steps of an ACS are:

Step 1. Initialize the trail intensities and parameters using an initial solution based upon the nearest neighborhood heuristic.

Step 2. While (termination condition is not met) do the following:

- Generate an ant-solution for each ant using the trail intensities.

- Improve each ant-solution by carrying out local search.
- Update elitist ants.
- Update trail intensities.

Step 3. Return the best solution found so far.

5.2.1 Initial Solution

We construct an initial solution using the nearest neighbor heuristic by iteratively adding customers to a vehicle route. We start a route from the depot and visit the nearest customer in terms of the distance. We keep track of the total load, $DD_r(i)$, and the maximum cumulative net-pickup $X_r(0,i)$ at the i^{th} position in the route. Given lemma 4.1 in chapter 4, if $DD_r(i) + X_r(0,i) \leq Q$ then the existing route is feasible in terms of the vehicle load and we move to the next nearest customer; otherwise we return back to the depot and start another route from the depot. Likewise, we proceed until all customers are served. Let L be the total tour length of the initial solution. Consider any two customers a and b. The trail intensity of visiting customer b immediately after customer a is denoted by τ_{ab}. Initially all τ_{ab} are set to $1/L$.

5.2.2 Generation of a solution by an ant

An artificial ant constructs a complete solution (i e., a set of routes of a tour) by successively visiting customers until all customers are visited. Each ant begins the tour from either the first or the last served customer in a vehicle route in the best solution found to date. Thus if v is the number of vehicles used in the best solution to date, we use

$2v$ ants in the current iteration. Each ant constructs a complete solution. The choice of the next customer to be visited is affected by two factors: 1) the trail intensity τ_{ab} which stores the information on the selection of this solution component in previous iterations, 2) the saving value $s(a, b)$ which represents the saving (in terms of the distance traveled) when customers a and b are served jointly by one vehicle instead of two separate vehicles. Here $s(a, b)$ is the problem specific information used in the ACS algorithm. See Clarke and Wright (1964) for a detailed description of the saving algorithm. The next customer to visit is chosen from the m best candidates in terms of the attractiveness value ξ_{ab} which is calculated as follows:

$$\xi_{ab} = [\tau_{ab}]^{\alpha}[s(a,b)]^{\beta} \tag{5.1}$$

Here, parameters α and β determine the relative biases for the trail intensity and the saving value, respectively (see section 5.3.1 for setting the value of parameters α and β).

Let Ω_m denote the set of the m best ranked customers in terms of the attractiveness value. If the number of customers not yet visited is less than m, then Ω_m contains all customers not yet visited by the ant. The next customer to be visited is selected using the following

$$p_{ab} = \begin{cases} \dfrac{\xi_{ab}}{\sum_{l \in \Omega_m} \xi_{al}} & \text{if } j \in \Omega_m \\ 0 & \textit{otherwise} \end{cases}$$

Existing ant colony algorithms by Bullnheimer et al. (1999) and Doerner et al. (2002) allow Ω_m to include only the feasible customers. The proposed construction rule

allows Ω_m to include all remaining neighborhood customers for a possible move and the feasibility of the route is checked only after the next customer to be visited is selected. If visiting customer *b* after visiting customer *a* leads to an infeasible solution (by violating the condition of lemma 4.1 in chapter 4) or in terms of the total travel time by the vehicle, then the ant returns to the depot and starts another route in which the first customer to be visited is customer *b*. A complete solution is built by choosing customers one by one and the construction process ends when all customers are served. The *random proportional rule* used to select the next customer is explained in section 3.2.1 with a numerical example. A numerical example to explain the construction of a route by an ant is presented in section 3.3 in chapter 3.

5.2.3 Local Search

After the construction of a solution by the ant, the solution is improved by local search. The following three local search schemes are used in our ACS approach:

1. 2-Opt local search scheme,
2. The customer insertion/interchange multi-route scheme, and
3. The sub-path exchange multi-route scheme.

The local search schemes were described in detail in chapter 4 along with the cumulative net-pickup approach for checking the feasibility of a given route. The three local search schemes are used iteratively until there is no further improvement in the solution. During our experiment, we found that a local minimum is usually reached after three or four cycles.

5.2.4 Updating elitist ants

Elitist ants are used to update trail intensities. We use γ elitist ants which are distinct from one another. Elitist ants are defined as γ best ant solutions found so far. Elitist ants are updated by comparing the present elitist ant solutions with the current ant solution. See subsection 3.3.4 in chapter 3 for more details on updating elitist ants.

5.2.5 Updating trail intensities

After all ants construct their solutions, trail intensities are updated using the solutions of γ elitist ants. The trail intensity of visiting customer b immediately after customer a is updated as follows:

$$\tau_{ab}^{new} = \rho \times \tau_{ab}^{old} + \sum_{\mu=1}^{\gamma} \Delta\tau_{ij}^{\mu} \qquad a=1,2,.....,n;\ \ b=1,2,...,n;\quad a \neq b. \tag{5.2}$$

Here ρ is the trail persistence factor (with $0 < \rho < 1$) (see section 5.3.1 for setting the value of parameter ρ). The first term in equation (5.2) represents the information carried from previous iterations. The second term is the deposition of pheromone by γ elitist ants where

$$\Delta\tau_{ab}^{\mu} = \begin{cases} 1/L^{\mu} & \text{if arc (a,b) is traversed by the } \mu^{\text{th}} \text{ elitist ant} \\ 0 & \text{otherwise} \end{cases}$$

This scheme allows all elitist ants to lay trails with equal importance as opposed to the updating scheme used in an ACS algorithm for CVRP by Bullnheimer et al. (1999). Our experiment shows that the strategy of giving more importance to the best solution may divert the search toward the global minimum but it can also increase the risk of being trapped at a local minimum.

5.3 Numerical analysis for VRPSPD

This section presents numerical results for the proposed ACS algorithm for VRPSPD and compares it with the existing metaheuristic approaches.

5.3.1 Parameter settings

The quality of a solution depends upon the number of ants used which ultimately affect the CPU time of the algorithm. We use $2v$ artificial ants; i.e., at each iteration, $2v$ new solutions are generated and each new solution is improved by local search. Here v is the number of vehicles. Our algorithm stops when the total number of iterations reaches 100. Parameters α, β, γ and ρ are set by performing sensitivity analysis carried out within limited CPU time. We found that the solution is not sensitive with respect to parameters α, β, γ and ρ. See Dorigo (2004) for more insights on setting the values for these parameters and the effect parameter values can have on solution quality. We use $\alpha = 5$, $\beta = 5$, $\gamma = 10$, $\rho = 0.95$ for our computational purpose. We thus use 10 elitist ants to update the pheromone. Our customer interchange multi-route local search scheme uses a Γ value of 10; we consider only the 10 closest customers in the neighbourhood of a given customer as possible candidate for the interchange. Our sub-path exchange multi-route local search scheme uses Γ of size 5.

In our experiment, we found that the local search scheme has a significant impact on the CPU time and the solution quality of the algorithm. The quality and CPU time of the proposed ACS is sensitive to parameter Γ. Increasing the value of Γ improves the solution quality, however it also increases the CPU time. Thus, the value for Γ should be

based on the tradeoff between CPU time and solution quality. We found the best tradeoff for $\Gamma = 10$ for insertion/interchange local search scheme and $\Gamma = 5$ for sub-path exchange multi-route scheme.

5.3.2 Benchmark problems

The numerical experiment is performed using three sets of problem instances available in the literature for VRPSPD. The first data set is provided by Min (1989). It is a single instance taken from a real life problem of transporting books between the main library and the branch libraries. There are two trucks each with capacity of 10,500 pounds available for delivery and pick up of books from 22 branch libraries.

The second data set is provided by Dethloff (2001). This data set consists of 40 problem instances, each containing 50 customers. In this data set, two different geographical scenarios are used. In the first scenario referred to as SCA, both the x and y coordinates of the customers are uniformly distributed over the interval [0, 100]; thus, customers are scattered in a square of size 100×100. In the second scenario referred to as CON, one-half of the customers are distributed in the same way as in SCA and the coordinates of the other half are uniformly distributed over the interval [100/3, 200/3]. The delivery demands, d_a are generated from uniform distribution [0, 100] and the pickup demands, p_a are generated as $p_a = (0.5 + r_a)d_a$, where r_a is a random number uniformly distributed over interval [0, 1].

The third data set is provided by Salhi and Nagy (1999). They generated problem instances from 14 benchmark problems of Christofides et al. (1979) using the following

three steps: 1) keep x and y co-ordinates for each customer to be the same as in the original problem, 2) for customer a having co-ordinates x_a and y_a and the original demand d_a^{ori} compute the ratio $rt_a = \min(x_a/y_a, y_a/x_a)$, 3). They calculate the new delivery demand for customer a as $d_a = rt_a \times d_a^{ori}$ and the new pickup demand for customer a as $p_a = (1-rt_a)\times d_a^{ori}$. In this way, a set of 14 instances referred to as set CMTX is generated. Another set of 14 instances referred to as set CMTY is generated from set CMTX by exchanging the pickup and delivery demand for each customer.

Altogether we use 69 problem instances for VRPSPD: 1 instance from Min (1989), 40 instances from Dethloff (2001) and 28 instances from Salhi and Nagy (1999). The problem instance by Min (1989) is referred to as Min, Dethloff (2001) instances are referred to as SCA and CON, and instances of Salhi and Nagy (1999) are referred to as CMTX/Y.

5.3.3 Computational experiment and the performance analysis of ACS

The proposed ACS was coded in C and implemented on Intel Xeon with 2.4 GHz computer.

We compare ACS with the following algorithms for which there are reported results.

HA	the hybrid algorithm of Crispim and Brandao (2005)
HeuSDP	the algorithm of Chen and Wu (2006)
TS	tabu search of Montane and Galvao (2005)
LNS	large neighborhood search by Ropke and Pisinger (2006)

Generally, it is difficult to compare CPU times since different algorithms are tested on different machines. A comparison of CPU times can be done using Mflops (million floating point calculation per second) values for different computers as given in Linpack benchmark report of Dongarra (2006). Table 5.1 presents Mflops values for the computers relevant to our study. Since some of the computers are not listed in the Linpack report, we used an equivalent machine listed in the report to obtain an approximate Mflops value for the particular machine. We use the Mflops values to scale the running time of an algorithm relative to our computer (i.e., Intel Xeon 2.4 GHz). The conversion factor r is the ratio of the Mflops value of a given computer to the Mflops value of our computer.

Table 5.1 : Mflops of different computers and the conversion factor (r)

Algorithm	Actual computer used	Approximate equivalent computer used	Mflops	r
HA	Pentium II 350 MHz	Pentium II 333 MHz	69	0.078
HeuSDP	Pentium IV 1.6 GHz PC	PIV 1.7 GHz	363	0.41
TS	Athlon 2 GHz PC	Pentium IV 2.53 GHz	1190	1.346
LNS	Pentium VI 1.5 GHz	Pentium VI 1.5 GHz	326	0.368
ACS	Intel Xeon 2.4 GHz	Intel Xeon 2.4 GHz	884	1

Generally, metaheuristic solutions tend to be better than the heuristic solutions. Hence, we do not include the heuristic solutions of Salhi and Nagy (1999), Dethloff (2001) and Nagy and Salhi (2005) in the comparison. Ropke and Pisinger (2006) carried out three large scale neighborhood search (LNS), namely standard, 6R-no learning and

6R-normal learning. Since on average, 6R-no learning yields a better solution than the others, we use 6R-no learning in our comparison. LNS uses the best solution over 10 runs while other algorithms use the single run solution. Thus, we report the ACS solution for the single run as well as the best solution over 10 runs for each problem instance. We also report the average solution over 10 runs for each problem instance. We use the best solution and the average solution over 10 runs to compare ACS with LNS while we use the single run solution to compare ACS with TS, HA and HeuSDP.

We did obtain the total tour length of 88.0 for the problem instance Min. This solution is proved to be the exact optimal solution as reported by Halse (1992), see Ropke and Pisinger (2006).

Numerical results for VRPSPD instances of Dethloff (2001)

The detailed results of different algorithms for 40 VRPSPD instances of Dethloff (2001) are reported in Table 5.8. The average solution values over the 40 problem instances of Dethloff (2001) for a single run of TS and ACS are reported in Table 5.2. Performance measures for ACS and LNS are given in Table 5.3.

Table 5.2 shows that the average solution of ACS is better than the average solution of TS. Table 5.3 shows that ACS is better than LNS in terms of the average solution as well as the average best solution. Also, the scaled CPU time of ACS is much less than the scaled CPU time of LNS.

ACS has also produced *9 new best known solutions* for Dethloff (2001) instances. The new best known solutions are reported in bold in Table 5.4.

Table 5.2: Comparison of ACS with TS over 40 VRPSPD instances of Dethloff (2001) over single run.

	TS	ACS
Average solution	764.25	759.02
Average CPU time	3.73	9.29
Average scaled CPU time	5.02	9.29

Table 5.3: Comparison of ACS with LNS over 40 VRPSPD instances of Dethloff (2001) over 10 runs

	LNS	ACS
Average best solution	761.07	758.64
Average solution	767.40	759.10
Average CPU time	157.93	9.29
Average scaled CPU time	58.12	9.29

Table 5.4: Total Tour Length obtained by different algorithms and corresponding CPU time for 40 instances of Dethloff (2001)

No	Inst.	TS		LNS		ACS		
		Single run solution	CPU time	Best solution	CPU time	Single run solution	Best solution	CPU time
1	SCA3-0	640.55	3.37	636.1	232	635.62	**635.62**	6
2	SCA3-1	697.84	3.25	697.8	218	697.84	697.84	6
3	SCA3-2	659.34	3.52	659.3	203	659.34	659.34	6
4	SCA3-3	680.04	3.31	680.6	241	680.04	680.04	6.1
5	SCA3-4	690.5	3.43	690.5	208	690.5	690.5	5.7
6	SCA3-5	659.9	3.67	659.9	226	659.9	659.9	5.1

7	SCA3-6	653.81	3.35	651.1	233	651.09	651.09	6.1
8	SCA3-7	659.17	3.33	666.1	206	659.17	659.17	6.8
9	SCA3-8	719.47	3.4	719.5	190	719.47	719.47	5.4
10	SCA3-9	681	3.41	681	220	681	681	6
11	SCA8-0	981.47	4.14	975.1	98	961.5	**961.5**	11
12	SCA8-1	1077.44	4.27	1052.4	95	1050.38	**1049.65**	11.5
13	SCA8-2	1050.98	4.2	1044.5	94	1044.48	1042.69	11.9
14	SCA8-3	983.34	4.17	999.1	94	983.34	983.34	11.3
15	SCA8-4	1073.46	4.13	1065.5	93	1065.49	1065.49	11.1
16	SCA8-5	1047.24	4.02	1027.1	96	1027.08	1027.08	11.3
17	SCA8-6	995.59	3.85	977	94	971.82	**971.82**	12
18	SCA8-7	1068.56	4.22	1061	92	1063.15	1052.17	12.5
19	SCA8-8	1080.58	3.85	1071.2	98	1071.18	1071.18	11
20	SCA8-9	1084.8	4.2	1060.5	92	1061.23	1060.5	11.5
21	CON3-0	631.39	3.64	616.5	215	616.52	616.52	8.3
22	CON3-1	554.47	3.31	554.5	245	554.47	554.47	7.1
23	CON3-2	522.86	3.45	521.4	232	519.11	**518**	6.9
24	CON3-3	591.19	3.28	591.2	231	591.19	591.19	7.2
25	CON3-4	591.12	3.47	588.8	221	588.79	588.79	6
26	CON3-5	563.7	3.38	563.7	209	563.7	563.7	6.9
27	CON3-6	506.19	3.32	500.8	225	499.05	499.05	7.3
28	CON3-7	577.68	3.51	576.5	227	576.48	576.48	7
29	CON3-8	523.05	3.66	523.1	237	523.05	523.05	7.4
30	CON3-9	580.05	3.36	586.4	207	578.25	578.25	6.8
31	CON8-0	860.48	4.19	857.2	94	857.17	857.17	12.3
32	CON8-1	740.85	3.89	740.9	94	740.85	740.85	12
33	CON8-2	723.32	3.76	716	94	712.89	**712.89**	13
34	CON8-3	811.23	4.12	811.1	98	811.07	811.07	13.9
35	CON8-4	772.25	3.75	772.3	95	772.25	772.25	11.9
36	CON8-5	756.91	3.99	755.7	94	754.88	**754.88**	12.4
37	CON8-6	678.92	4.04	693.1	96	678.92	678.92	12.4
38	CON8-7	814.5	4	814.8	94	811.96	**811.96**	13
39	CON8-8	775.59	3.74	774	94	767.53	**767.53**	12.5
40	CON8-9	809	4.13	809.3	92	809	809	12.9
Average		764.25	3.73	761.07	157.93	759.02	758.64	9.29

Numerical results for VRPSPD instances of Salhi and Nagy (1999)

The original data set used by Salhi and Nagy (1999) was not available. Hence, similar to other researchers, we recreated the data set. The numerical results for ACS and LNS over 28 VRPSPD instances of Salhi and Nagy (1999) are reported in Table 5.7. We do not include TS, HA and HeuSDP algorithms in Table 5.9 for the following reasons: 1) TS and HeuSDP have used integer values for generating delivery and pickup demands

while the other metaheuristics have used fractional values; 2) TS solutions assume integer values for the distance matrix and do not consider fixed service time in the problem instances with the maximum tour length constraint; 3) HA and HeuSDP do not report the solutions for the problem instances with the maximum tour length constraint.

We compare ACS with HA, HeuSDP and TS over only the 14 problem instances in which the maximum tour length constraint is not imposed. The average solution over 14 problem instances is reported in Table 5.5. It is seen that in terms of the average solution, ACS clearly gives better results than the other algorithms.

Summary results over 10 runs of ACS and LNS are given in Table 5.6. Results from Table 5.6 indicate that ACS is better than LNS in terms of solution quality as well as CPU time. ACS has also produced *22 new best known solutions* and these new best known solutions are highlighted in bold in Table 5.7.

Table 5.5: Comparison between different algorithms over 14 problem instances (without maximum distance constraint) of Salhi and Nagy (1999)

	HA	HeuSDP	TS	ACS
Average solution	829.93	770.13	777.86	755.84
Average CPU time	39	261.28	9.31	151.54
Average scaled CPU time	3.04	107.3	12.58	151.54

Table 5.6: Comparison between LNS and ACS over 28 problem instances of Salhi and Nagy (1999)

	LNS	ACS
Average best solution	932.62	893.22
Average solution	949.6	895.31
Average CPU time	813.33	151.54
Average scaled CPU time	299.31	151.54

Table 5.7: Total Tour Length obtained by LNS and ACS and corresponding CPU time for 28 VRPSPD instances of Salhi and Nagy (1999)

No	Inst.	LNS		ACS		
		Best solution	CPU time	Single run solution	Best solution	CPU time
1	CMT1X	467	221	466.77	466.77	5.00
2	CMT2X	704	294	688.05	**684.21**	20.75
3	CMT3X	731	863	721.4	**721.4**	41.25
4	CMT4X	879	1676	857.19	**854.12**	131.75
5	CMT5X	1108	2340	1035.03	**1034.87**	377.50
6	CMT6X	559	113	555.43	**555.43**	14.00
7	CMT7X	901	167	901.11	**900.12**	47.75
8	CMT8X	866	413	865.5	865.5	80.75
9	CMT9X	1205	765	1161.97	**1161.54**	300.00
10	CMT10X	1462	1275	1401.13	**1386.29**	773.50
11	CMT11X	837	1821	844.52	839.66	57.25
12	CMT12X	685	684	663.09	**663.01**	36.25
13	CMT13X	1578	563	1542.86	**1542.86**	160.25
14	CMT14X	885	387	821.75	**821.75**	78.50
15	CMT1Y	467	235	466.77	466.77	5.00
16	CMT2Y	685	331	688.26	684.94	22.25
17	CMT3Y	738	708	724.54	**721.4**	43.75
18	CMT4Y	876	1788	860.85	855.76	140.25
19	CMT5Y	1146	2177	1039.99	**1037.34**	393.50

20	CMT6Y	559	101	555.43	**555.43**	13.75
21	CMT7Y	952	166	901.22	**900.54**	46.25
22	CMT8Y	873	398	865.5	**865.5**	77.75
23	CMT9Y	1271	757	1161.97	**1161.54**	291.75
24	CMT10Y	1552	1255	1400.68	**1395.04**	757.50
25	CMT11Y	920	1376	859.57	**840.19**	52.75
26	CMT12Y	675	539	663.5	**663.5**	39.25
27	CMT13Y	1602	547	1542.86	**1542.86**	160.25
28	CMT14Y	-	-	821.75	**821.75**	74.75
Average		932.70	813.33	895.67	893.22	151.54

5.4 Numerical analysis for VRPBM

We implemented the algorithm designed for VRPSPD also on VRPBM without any modification to the algorithm. In VRPSPD, each customer is a linehaul as well as a backhaul customer while in VRPBM a customer is either a linehaul customer or a backhaul customer. We keep the parameter settings for VRPBM to be the same as for VRPSPD. This section compares the performance of ACS with the existing metaheuristic approaches.

5.4.1 Benchmark problems

The numerical experiment is performed using the data set of Salhi and Nagy (1999) for VRPBM. They generated three problem subsets referred to as T, Q and H from the 14 problem instances of VRP (Christofides et al. (1979)). In set T, every tenth customer is declared a backhaul customer and is assigned pickup demand equal to the original delivery demand. Similarly, for sets Q and H, every fourth and every second customer is declared to be a backhaul customer. Thus, the proportion of backhaul customers is 10% in problem set T, 25% in problem set Q and 50% in problem set H. In

our comparison, we refer to the three subsets T, Q and H as CMTT, CMTH and CMTQ, respectively.

5.4.2 Computational experiment and performance analysis of the algorithms

We compare ACS with the following algorithms for which there are reported results.

HA the hybrid algorithm of Crispim and Brandao (2005)

LNS large neighborhood search by Ropke and Pisinger (2006)

The numerical results of LNS and ACS for 42 problem instances are presented in Table 5.10. We do not include HA in Table 5.10 because the solutions for 21 problem instances which include the maximum tour length constraint are not reported by HA. LNS reports the best solution over 10 runs while HA uses the single run solution. Hence, the solution from a single run as well as the best solution over 10 runs are reported in Table 5.10 for ACS. We also report the average solution over 10 runs for each problem instance for ACS.

Table 5.8 gives the average solution as well as the average scaled CPU time over 21 problem instances for HA and ACS. The average solution of ACS is much better than HA but the scaled CPU time for ACS is greater than the scaled CPU time for HA.

Table 5.9 gives the performance measures for ACS and LNS. ACS is competitive with LNS in terms of the average best solution, however in terms of the overall average solution, ACS is better than LNS. ACS has also produced *6 new best known* solutions and these solutions are highlighted in Table 5.10 in bold. For comparison purposes, we calculated the average of the best known solutions over 42 VRPBM problem instances.

This average is 918.47. Table 5.10 shows that the average best solution over 10 runs for LNS is 919.61 and it is 919.57 for ACS. Thus, the average best solutions for ACS and LNS are very close to the average best known solution.

Table 5.8: Comparison between HA and ACS algorithms over 21 VRPBM instances (without maximum distance constraint) of Salhi and Nagy (1999)

Percentage of Backhaul customers	Performance measures	HA	ACS
50%	Average solution	797.43	734.44
	Average CPU time	45.03	88.37
	Average scaled CPU time	3.51	88.37
25%	Average solution	879.57	813.63
	Average CPU time	41.46	112.14
	Average scaled CPU time	3.24	112.14
10%	Average solution	945.29	873.2
	Average CPU time	35.81	117.91
	Average scaled CPU time	2.8	117.91

Table 5.9: Comparison between LNS and ACS algorithms over 42 VRPBM instances of Salhi and Nagy (1999)

Percentage of Backhaul customers	Performance measures	LNS	ACS
50%	Average Best Solution	880.93	881.02
	Overall Average Solution	884.21	882.97

	CPU time	178.14	150.74
	Scaled CPU time	65.7	150.74
25 %	Average Best Solution	922.43	922.97
	Overall Average Solution	925.3	924.49
	CPU time	143.07	154.93
	Scaled CPU time	52.76	154.93
10%	Average Best Solution	955.43	954.74
	Overall Average Solution	958.71	955.89
	CPU time	133.21	148.47
	Scaled CPU time	49.12	148.47

Table 5.10: Total Tour Length obtained by LNS and ACS and corresponding CPU time for 42 VRPBM instances of Salhi and Nagy (1999)

No	Instances	LNS		ACS		
		Best solution	CPU time	Single run solution	Best solution	CPU time
1	CMT1H	465	51	465.02	465.02	5.6
2	CMT2H	663	78	663.92	662.63	22
3	CMT3H	701	186	702.94	701.31	35.6
4	CMT4H	829	345	831.39	831.39	125.4
5	CMT5H	983	514	1007.99	992.37	351.4
6	CMT6H	555	31	555.43	555.43	13
7	CMT7H	900	54	900.84	900.84	50
8	CMT8H	866	95	865.5	865.50	85.6
9	CMT9H	1166	177	1163.85	1161.63	306.4
10	CMT10H	1393	296	1392.39	**1383.78**	791
11	CMT11H	818	303	823.06	820.35	45.8
12	CMT12H	629	150	646.74	629.37	32.8
13	CMT13H	1543	125	1542.97	1542.86	164.2

14	CMT14H	822	89	821.75	821.75	81.6
15	CMT1Q	490	41	489.74	489.74	6
16	CMT2Q	733	65	733.15	732.76	26.2
17	CMT3Q	747	128	747.46	747.15	39.8
18	CMT4Q	918	244	918.57	**913.93**	153
19	CMT5Q	1119	381	1137.58	1134.72	451.8
20	CMT6Q	555	30	555.43	555.43	12.8
21	CMT7Q	901	53	902.95	900.69	46.8
22	CMT8Q	866	93	865.5	865.50	74.4
23	CMT9Q	1162	171	1161.97	1161.51	289.6
24	CMT10Q	1389	288	1392.23	**1386.54**	730.2
25	CMT11Q	939	196	939.36	939.36	66.2
26	CMT12Q	729	108	729.55	729.46	42
27	CMT13Q	1543	120	1542.97	1542.97	157.8
28	CMT14Q	822	85	821.75	821.75	72.4
29	CMT1T	520	34	520.06	520.06	7
30	CMT2T	783	57	785	782.77	26
31	CMT3T	798	109	798.07	798.07	42.6
32	CMT4T	1000	212	990.39	**990.39**	166.8
33	CMT5T	1227	333	1232.53	1232.08	460.8
34	CMT6T	555	31	555.43	555.43	11.6
35	CMT7T	903	52	903.05	903.05	39
36	CMT8T	866	95	865.54	865.54	65.6
37	CMT9T	1164	178	1162.8	**1162.68**	261
38	CMT10T	1402	291	1401.56	1400.22	658.6
39	CMT11T	1000	164	998.8	998.80	70.2
40	CMT12T	788	96	787.52	787.52	52
41	CMT13T	1544	127	1542.97	**1542.97**	152.8
42	CMT14T	827	86	826.77	826.77	64.6
Average		**919.60**	**151.48**	**921.25**	**919.57**	**151.38**

5.5 Conclusions

While there are many solution procedures for CVRP, only a few solution procedures exist for VRPSPD. This chapter presents an ant colony system (ACS) algorithm for solving VRPSPD. ACS contains the following new features as compared to the existing ant colony systems:

1. A new construction rule is used to generate an ant solution. The rule checks for the feasibility of the route only *after* the next customer to be served is selected.
2. An inherent part of an ant colony system is local search. Two new local search schemes, the customer insertion/interchange multi-route scheme and the sub-path exchange multi-route scheme, are used.
3. We do not distinguish among the elitist ants while updating the trail intensities; equal importance is given to each elitist ant. This reduces the risk of being trapped at a local minimum.

Computational experiment with benchmark problem instances has shown that in general the proposed ant colony system (ACS) gives better results for VRPSPD both in terms of the solution quality and the CPU time. The reduced CPU time of ACS for VRPSPD instances is attributed to the cumulative net-pickup approach for checking the feasibility of a given route. Overall, ACS has produced 31 new best known solutions for VRPSPD problem instances available in the literature. Further research can be done to improve the local search schemes in ant colony systems. Since VRPBM is a special case of VRPSPD, we tested our ACS on VRPBM instances as well. We have found that ACS is competitive to LNS metaheuristic in solving VRPBM problem instances. ACS has also produced 6 new best known solutions for VRPBM instances.

CHAPTER 6

Saving based algorithms for vehicle routing problem with simultaneous pickup and delivery (VRPSPD)

6.1 Introduction

The VRPSPD is characterized by the following: a fleet of identical vehicles located at the depot to serve the customers distributed geographically in the area. The capacity of each vehicle is Q. A customer requires a given shipment to be delivered and another load to be picked up during a single visit of the vehicle. A non-negative cost c_{ij} is associated with each arc (i, j) and without loss of generality, it is the distance between customer i and j. The objective is to design a set of minimum cost routes so that the load on a vehicle is below the vehicle capacity Q at each point in the route.

There are several successful applications of the saving based approach to the variants of CVRP. As mentioned in literature survey, there are applications of the Clarke and Wright saving algorithm to solve VRPBM. However, the algorithm is applied separately to linehaul and backhaul customers or it is applied in combination with insertion heuristics. In this chapter, we ensure that the saving algorithm is applied to both linehaul and backhaul customers in VRPSPD in an integrated manner. We also apply the parallel saving algorithm (PSA) of Altinkemer and Gavish (1991) to VRPSPD. Since VRPBM is a special case of VRPSPD, we apply our heuristic to VRPBM as well.

Checking the feasibility of a route in VRPSPD is difficult because of the fluctuating load on the route. In this chapter, we propose a cumulative net-pickup approach for the saving

heuristic in VRPSPD. In the saving heuristic, a new route is created by merging the two existing routes. The proposed approach checks for the feasibility when two existing routes are merged.

6.2. Cumulative net-pickup approach for checking the feasibility of a new route

In vehicle routing problem with simultaneous pickup and delivery (VRPSPD), checking feasibility is not easy since one needs to insure that the load on the vehicle does not exceed its capacity at each point in the route. The saving heuristics developed in this chapter involves merging of two routes. We propose a cumulative net-pickup approach to check feasibility of a route that is obtained by merging two routes. Cumulative net-pickup approach is a constant time approach for checking the feasibility of a route. Other constant time approaches available in literature are by Kindervater and Savelsbergh (1997) and by Irnich (2007). Kindervater and Savelsbergh (1997) proposed a constant time approach for checking the feasibility of the tour in a TSP with simultaneous pickup and delivery. They describe three quantities that are sufficient for the task: the maximum load, the minimum load and the final load. In the cumulative net-pickup approach, we check the feasibility of a route using a single expression and this reduces the complexity to O(1).

Let R_r denote the route of vehicle r. Let the number of customers on route R_r be n_r. Route R_r contains the list of n_r customers plus two copies of the depot numbered 0 and n_r+1. To describe the cumulative net-pickup approach, we use the following notation:

D_r total demand to be delivered by vehicle r.

$\sigma_r(i)$ the i^{th} customer in route R_r (i.e., the customer in the i^{th} position in the route of vehicle r).

$d_r(i)$ quantity of product to be delivered to customer $\sigma_r(i)$.

$p_r(i)$ quantity of product to be picked up from customer $\sigma_r(i)$.

$f_r(i)$ $= p_r(i) - d_r(i)$ net-pickup of customer $\sigma_r(i)$.

$F_r(i)$ cumulative net-pickup by vehicle r after visiting customer $\sigma_r(i)$.

X_r maximum cumulative net-pickup encountered by vehicle r (The notation of X_r is equivalent to the notation $X_r(0, n_r+1)$ used in chapter 4).

Y_r minimum cumulative net-pickup encountered by vehicle r (The notation Y_r is equivalent to the notation $Y_r(0, n_r+1)$ used in chapter 4).

$VL_r(i)$ load of vehicle r after visiting customer $\sigma_r(i)$ in R_r.

The following relationship must hold given the above notation:

$$F_r(0) = 0,$$

$$F_r(i) = F_r(i-1) + f_r(i), \quad \forall i = 1, 2, ..., n_r+1,$$

$$D_r = \sum_{l=0}^{n_r+1} d_r(l),$$

$$X_r = \text{Max}\{ F_r(0), F_r(1), \ldots, F_r(n_r) \},$$

$$Y_r = \text{Min}\{ F_r(0), F_r(1), \ldots, F_r(n_r) \}.$$

The following lemma is useful in checking the feasibility of a given route.

Lemma 6.1. A necessary and sufficient condition for route R_r to be feasible is that $D_r + X_r \leq Q$.

Proof of necessary condition:

Suppose route R_r is feasible. Thus its load does not exceed vehicle capacity Q at any point in the route; That is

$$Max\{VL_r(0), VL_r(1), ..., VL_r(n_r)\} \leq Q$$

$$\Rightarrow Max\{D_r + F_r(0), D_r + F_r(1), ..., D_r + F_r(n_r)\} \leq Q$$

$$\Rightarrow D_r + Max\{F_r(0), F_r(1), ..., F_r(n_r)\} \leq Q$$

$$\Rightarrow D_r + X_r \leq Q \quad .$$

Proof of sufficient Condition:

The proof is similar to the above except that the steps will be in the reverse order.

□

Let the reverse route of vehicle *r* be denoted as $\bar{R}_r$. In the reverse route $\bar{R}_r$, customers are served in reverse order of the original route R_r. Now let

$\bar{D}_r$ total demand to be delivered by vehicle *r* in reversed route $\bar{R}_r$.

$\bar{\sigma}_r(i)$ the i^{th} customer in route $\bar{R}_r$ (i.e., the customer in the i^{th} position in the reverse route of vehicle *r*).

$\bar{d}_r(i)$ quantity of product to be delivered to customer $\bar{\sigma}_r(i)$.

$\bar{p}_r(i)$ quantity of product to be picked up from customer $\bar{\sigma}_r(i)$.

$\bar{f}_r(i) = \bar{p}_r(i) - \bar{d}_r(i)$ net-pickup of customer $\bar{\sigma}_r(i)$.

$\bar{F}_r(i)$ cumulative net-pickup by vehicle r after visiting customer $\bar{\sigma}_r(i)$.

$\bar{X}_r$ maximum cumulative net-pickup encountered by vehicle r in $\bar{R}_r$.

$\bar{Y}_r$ minimum cumulative net-pickup encountered by vehicle r in $\bar{R}_r$.

The following lemmas are used to calculate $\bar{X}_r$ and $\bar{Y}_r$.

Lemma 6.2: The cumulative net-pickup of vehicle r after visiting customer $\bar{\sigma}_r(i)$ in the reversed route $\bar{R}_r$ is $F_r(n_r) - F_r(n_r - i)$

Proof :

The customer in the i^{th} position in reversed route $\bar{R}_r$ is the customer in position $n_r + 1 - i$ in route R_r. That is $\bar{\sigma}_r(i) = \sigma_r(n_r + 1 - i)$ and $\bar{f}_r(n_r) = f_r(n_r + 1 - i)$. Now, the cumulative net-pickup of customer $\bar{\sigma}_r(i)$ is

$$\bar{F}_r(i) = \bar{f}_r(0) + \bar{f}_r(1) + ... \bar{f}_r(i)$$

$$\bar{F}_r(i) = f_r(n_r + 1) + f_r(n_r) + ... + f_r(n_r + 1 - i)$$

$$= \{f_r(n_r + 1) + f_r(n_r) + ... + f_r(n_r + 1 - i)\} + \{f_r(n_r - i) + f_r(n_r - i - 1) + ... + f_r(0)\} - \{f_r(n_r - i) + f_r(n_r - i - 1) + ... + f_r(0)\}$$

$$= F_r(n_r + 1) - \{f_r(n_r - i) + ... + f_r(0)\}$$

$$= F_r(n_r + 1) - F_r(n_r - i)$$

Since $F_r(n_r + 1) = F_r(n_r) + f_r(n_r + 1) = F_r(n_r) + 0 = F_r(n_r)$

$\bar{F}_r(i) = F_r(n_r) - F_r(n_r - i)$ □

Lemma 6.3: When route R_r of vehicle r is reversed,

a) $\bar{X}_r = F_r(n_r) - Y_r$.

b) $\bar{Y}_r = F_r(n_r) - X_r$

c) $\bar{F}_r(n_r) = F_r(n_r)$

Proof:

a) We know that $\bar{\sigma}_r(i) = \sigma_r(n_r + 1 - i)$. The maximum cumulative net-pickup of route $\bar{R}_r$ is

$$\begin{aligned}
\bar{X}_r &= Max\{\bar{F}_r(0), \bar{F}_r(1), ..., \bar{F}_r(n_r)\} \\
&= Max\{F_r(n_r) - F_r(n_r), F_r(n_r) - F_r(n_r - 1), ..., F_r(n_r) - F_r(0)\} \\
&= F_r(n_r) + Max\{-F_r(n_r), -F_r(n_r - 1), ..., -F_r(0)\} \\
&= F_r(n_r) - Min\{F_r(n_r), F_r(n_r - 1), ..., F_r(0)\} \\
&= F_r(n_r) - Y_r
\end{aligned}$$

b) Proof is similar to part a.

c) From lemma 6.2, we know that the cumulative net pick-up of the i^{th} customer in $\bar{R}_r$ is $\bar{F}_r(n_r) = F_r(n_r) - F_r(n_r - i)$. Thus the cumulative net pick-up after visiting customer $\bar{\sigma}_r(n_r)$ in $\bar{R}_r$ is $\bar{F}_r(n_r) = F_r(n_r) - F_r(n_r - n_r) = F_r(n_r) - F_r(0)$.

By definition $F_r(0) = 0$, therefore

$\bar{F}_r(n_r) = F_r(n_r) - 0 = F_r(n_r)$ □

The saving and matching based heuristics designed in this chapter involve merging of two existing routes. Let us consider routes R_r and R_s of vehicles r and s for

possible merging. Let vehicle q serve the customers of the route obtained by merging R_r and R_s. The routes R_r and R_s can be merged into R_q in the following eight ways.

M1. $R_q = R_r \cup R_s = \{0, \sigma_r(1), ..., \sigma_r(n_r), \sigma_s(1), ..., \sigma_s(n_s), 0\}$,

M2. $R_q = R_r \cup \overline{R}_s = \{0, \sigma_r(1), ..., \sigma_r(n_r), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), 0\}$,

M3. $R_q = \overline{R}_r \cup R_s = \{0, \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \sigma_s(1), ..., \sigma_s(n_s), 0\}$,

M4. $R_q = \overline{R}_r \cup \overline{R}_s = \{0, \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), 0\}$,

M5. $R_q = \overline{R}_s \cup \overline{R}_r = \{0, \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), 0\}$,

M6. $R_q = R_s \cup \overline{R}_r = \{0, \sigma_s(1), ..., \sigma_s(n_s), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), 0\}$,

M7. $R_q = \overline{R}_s \cup R_r = \{0, \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), \sigma_r(1), ..., \sigma_r(n_r), 0\}$, and

M8. $R_q = R_s \cup R_r = \{0, \sigma_s(1), ..., \sigma_s(n_s), \sigma_r(1), ..., \sigma_r(n_r), 0\}$.

Here route R_q obtained by method M5 is the reverse of the route obtained by method M1. We consider the routes obtained by M1 and by M5 separately because it is possible that one of the routes is feasible whereas the other is not.

Similarly, route R_q obtained by methods M6, M7 and M8 is the reverse of the route obtained by methods M2, M3 and M4, respectively.

Lemma 6.4: Suppose routes R_r and R_s are combined by method M1 and the resulting route is served by vehicle q, i.e., $R_q = R_r \cup R_s = \{0, \sigma_r(1), ..., \sigma_r(n_r), \sigma_s(1), ..., \sigma_s(n_s), 0\}$. Then

a) the maximum cumulative net-pickup of R_q is $X_q = Max\{X_r, F_r(n_r) + X_s\}$,

b) the minimum cumulative net-pickup of R_q is $Y_q = Min\{Y_r, F_r(n_r) + Y_s\}$ and

c) the cumulative net-pickup after visiting the last customer in R_q is $F_q(n_q) = F_r(n_r) + F_s(n_s)$.

Proof:

a) $X_q = Max\{F_q(0), ..., F_q(n_r), F_q(n_r+1), ..., F_q(n_r+n_s)\}$

$= Max\{F_r(0), ..., F_r(n_r), (F_r(n_r) + f_s(1)), ..., (F_r(n_r) + f_s(1) + ... + f_s(n_s))\}$

$= Max\{F_r(0), ..., F_r(n_r), F_r(n_r), (F_r(n_r) + f_s(1)), ..., (F_r(n_r) + f_s(1) + ... + f_s(n_s))\}$

$= Max\{Max\{F_r(0), ..., F_r(n_r)\}, Max\{F_r(n_r), (F_r(n_r) + f_s(1)), ..., (F_r(n_r) + f_s(1) + ... + f_s(n_s))\}\}$

$= Max\{X_r, F_r(n_r) + Max\{0, (f_s(1)), ..., (f_s(1) + ... + f_s(n_s))\}\}$

Since $f_s(0) = 0$

$X_q = Max\{X_r, F_r(n_r) + Max\{f_s(0), (f_s(0) + f_s(1)), ..., (f_s(0) + f_s(1) + ... + f_s(n_s))\}\}$

$= Max\{X_r, F_r(n_r) + Max\{F_s(0), F_s(1), ..., F_s(n_s)\}\}$

$= Max\{X_r, F_r(n_r) + X_s\}$

b) Proof is similar to part a.

c) $F_q(n_q) = f_q(0) + ... + f_q(n_r) + f_q(n_r+1) + ... + f_q(n_r+n_s)$

$= f_r(0) + ... + f_r(n_r) + f_s(1) + ... + f_s(n_s)$

$= (f_r(0) + ... + f_r(n_r)) + (f_s(0) + f_s(1) + ... + f_s(n_s))$

$= F_r(n_r) + F_s(n_s)$ □

Note that given lemma 6.1, route R_q obtained by method M1 is feasible if $D_r + D_s + Max\{ X_r, F_r(n_r) + X_s \} \leq Q$.

Lemma 6.5: Suppose routes R_r and R_s are combined by method M2 and the resulting route is served by vehicle q, i.e., $R_q = R_r \cup \overline{R}_s = \{0, \sigma_r(1), ..., \sigma_r(n_r), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), 0\}$. Then

a) the maximum cumulative net-pickup of route R_q is $X_q = Max \{ X_r, F_r(n_r) + F_s(n_s) - Y_s \}$,

b) the minimum cumulative net-pickup of route R_q is $Y_q = Min\{ Y_r, F_r(n_r) + F_s(n_s) - X_s \}$, and

c) the cumulative net-pickup after visiting the last customer in R_q is $F_q(n_q) = F_r(n_r) + F_s(n_s)$.

Proof:

Let R_p represent the reversed route of vehicle s, i.e., $R_p = \overline{R}_s$. Then R_q is obtained by combining routes R_r and R_p, i.e., $R_q = R_r \cup R_p$.

a) Using part a) of lemma 6.4, we get the following expression for the maximum cumulative net-pickup of route R_q.

$X_q = Max\{ X_r, F_r(n_r) + X_p \}$.

We have assumed that $R_p = \overline{R}_s$, hence $X_p = \overline{X}_s$. Thus,

$X_q = Max\{ X_r, F_r(n_r) + \overline{X}_s \}$,

We know from part a) of lemma 6.3 that $\overline{X}_s = F_s(n_s) - Y_s$. Hence,

$X_q = Max\{X_r, F_r(n_r) + F_s(n_s) - Y_s\}$

b) Using part b) of lemma 6.4, we get the following expression for the minimum cumulative net-pickup of route R_q.

$Y_q = Min\{Y_r, F_r(n_r) + Y_p\}$.

We have assumed that $R_p = \overline{R}_s$, hence $Y_p = \overline{Y}_s$. Thus,

$Y_q = Min\{Y_r, F_r(n_r) + \overline{Y}_s\}$

We know from part b) of lemma 6.3 that $\overline{Y}_s = F_s(n_s) - X_s$. Hence,

$Y_q = Min\{Y_r, F_r(n_r) + F_s(n_s) - X_s\}$

c) Using part b) of lemma 6.4, we get the following expression for the minimum cumulative net-pickup of route R_q.

$F_q(n_q) = F_r(n_r) + F_p(n_p)$.

We have assumed that $R_p = \overline{R}_s$, hence $F_p(n_p) = \overline{F}_s(n_s)$. Thus,

$F_q(n_q) = F_r(n_r) + \overline{F}_s(n_s)$

We know from part c) of lemma 6.3 that $\overline{F}_s(n_s) = F_s(n_s)$. Hence,

$F_q(n_q) = F_r(n_r) + F_s(n_s)$. □

The maximum cumulative net-pickup, X_q and the minimum cumulative net-pickup, Y_q of route R_q obtained by any other method M2, M3, M4, M5, M6, M7 and M8

can be derived in a way similar to lemma 6.5. The maximum cumulative net-pickup obtained by each method is given below.

M1. $X_q = Max\{X_r, F_r(n_r) + X_s\}$ (6.1)

M2. $X_q = Max\{X_r, F_r(n_r) + F_s(n_s) - Y_s\}$ (6.2)

M3. $X_q = Max\{F_r(n_r) - Y_r,\ F_r(n_r) + X_s\}$ (6.3)

M4. $X_r = Max\{F_r(n_r) - Y_r,\ F_r(n_r) + F_s(n_s) - Y_s\}$ (6.4)

M5. $X_r = Max\{F_s(n_s) - Y_s,\ F_s(n_s) + F_r(n_r) - Y_r\}$ (6.5)

M6. $X_r = Max\{X_s,\ F_s(n_s) + F_r(n_r) - Y_r\}$ (6.6)

M7. $X_r = Max\{F_s(n_s) - Y_s,\ F_s(n_s) + X_r\}$ (6.7)

M8. $X_r = Max\{X_s,\ F_s(n_s) + X_r\}$ (6.8)

The minimum cumulative net-pickup obtained by each method is as follows.

M1. $Y_q = Min\{Y_r, F_r(n_r) + Y_s\}$ (6.9)

M2. $Y_q = Min\{Y_r, F_r(n_r) + F_s(n_s) - X_s\}$ (6.10)

M3. $Y_q = Min\{F_r(n_r) - X_r,\ F_r(n_r) + Y_s\}$ (6.11)

M4. $Y_r = Min\{F_r(n_r) - Y_r,\ F_r(n_r) + F_s(n_s) - Y_s\}$ (6.12)

M5. $Y_r = Min\{F_s(n_s) - X_s,\ F_s(n_s) + F_r(n_r) - X_r\}$ (6.13)

M6. $Y_r = Max\{Y_s,\ F_s(n_s) + F_r(n_r) - X_r\}$ (6.14)

M7. $Y_r = Min\{F_s(n_s) - X_s,\ F_s(n_s) + Y_r\}$ (6.15)

M8. $Y_r = Max\{Y_s,\ F_s(n_s) + Y_r\}$ (6.16)

Lemma 6.6: Suppose routes R_r and R_s are combined by any method M1 to M8 and the resulting route is served by vehicle q. Then the cumulative net-pickup after visiting the last customer in R_q is $F_q(n_q) = F_r(n_r) + F_s(n_s)$.

Proof:

The cumulative net-pickup after visiting the last customer in R_q is the summation of net-pick of customers served by vehicle q. The number of customers to be served by vehicle q will be the same irrespective of the method by which R_r and R_s are combined to obtain R_q. Thus,

$$\begin{aligned} F_q(n_q) &= f_q(0) + ... + f_q(n_r) + f_q(n_r + 1) + ... + f_q(n_r + n_s) \\ &= f_r(0) + ... + f_r(n_r) + f_s(1) + ... + f_s(n_s) \\ &= (f_r(0) + ... + f_r(n_r)) + (f_s(0) + f_s(1) + ... + f_s(n_s)) \\ &= F_r(n_r) + F_s(n_s) \end{aligned}$$

□

We use lemma 6.1 and the expressions for maximum cumulative net-pickup given in (6.1) to (6.8) for checking the feasibility of route R_q. Using lemma 6.1, the final condition for route R_q to be feasible under each method is given below.

M1. $D_r + D_s + Max\{X_r, F_r(n_r) + X_s\} \le Q$ (6.17)

M2. $D_r + D_s + Max\{X_r, F_r(n_r) + F_s(n_s) - Y_s\} \le Q$ (6.18)

M3. $D_r + D_s + Max\{F_r(n_r) - Y_r,\ F_r(n_r) + X_s\} \le Q$ (6.19)

M4. $D_r + D_s + Max\{F_r(n_r) - Y_r,\ F_r(n_r) + F_s(n_s) - Y_s\} \le Q$ (6.20)

M5. $D_r + D_s + Max\{F_s(n_s) - Y_s,\ F_s(n_s) + F_r(n_r) - Y_r\} \le Q$ (6.21)

M6. $D_r + D_s + Max\{X_s,\ F_s(n_s) + F_r(n_r) - Y_r\} \le Q$ (6.22)

M7. $D_r + D_s + Max\{F_s(n_s) - Y_s,\ F_s(n_s) + X_r\} \le Q$ (6.23)

M8. $D_r + D_s + Max\{X_s,\ F_s(n_s) + X_r\} \le Q$ (6.24)

The above methods M1 to M8 are used in the saving and matching based algorithm described in the next two sections. Once routes R_r and R_s are combined to form route R_q, the maximum cumulative net-pickup X_q, the minimum cumulative net-pickup Y_q and the cumulative net-pickup after visiting the last customer $F_q(n_q)$ are updated using equations 6.1 to 6.16 and Lemma 6.6.

6.3. Saving based algorithms

The saving algorithm of Clarke and Wright (1964) is the most widely used constructive algorithm for the variants of CVRP. The algorithm works on the principle of the maximum saving when two routes are combined and served by a single vehicle instead of two different vehicles. Now suppose $R_r = \{0, \sigma_r(1), \ldots\ldots, \sigma_r(n_r), 0\}$ and $R_s = \{0, \sigma_s(1), \ldots\ldots, \sigma_s(n_s), 0\}$ are combined and served by vehicle q, so that $R_q = R_r \cup R_s = \{0, \sigma_r(1), \ldots\ldots, \sigma_r(n_r), \sigma_s(1), \ldots, \sigma_s(n_s), 0\}$. Let customer $a = \sigma_r(n_r)$ and $b = \sigma_s(1)$. Then the saving value s_{ab} can be calculated as follows:

$$s_{ab} = c_{0a} + c_{0b} - c_{ab}.$$

Here c_{ab} is the distance between customer a and customer b. The saving algorithm of Clarke and Wright (1964) for CVRP works as follows;

Step 1: Let n be the total number of customers. Calculate the saving value for each combination of customers k and l, i.e., $s_{kl} = c_{0k} + c_{0l} - c_{kl}$ for $k,l = 1,...,n$ and $k \neq l$. Initialize vector Ω by storing the saving values s_{kl} in non-increasing order. Assign individual vehicle for each customer and create routes $R_m = \{0, \sigma_m(1), 0\}$ for $m = 1,...,n$.

Step 2. Consider the first element of Ω, say s_{kl}, and remove it from Ω. Determine the vehicles that serve customer k and customer l. Suppose customer k is served by vehicle r and customer l is served by vehicle s. Now check the following conditions:

1. Customer k is the last visited customer in route R_r, i.e., $k = \sigma_r(n_r)$.
2. Customer l is the first visited customer in route R_s, i.e., $l = \sigma_s(1)$.
3. Vehicle r and s are different i.e., $r \neq s$.
4. The combined load of vehicle r and vehicle s is less than the vehicle capacity Q.

If the above conditions are satisfied then merge the routes of vehicles r and s and reassign vehicle r to serve the new route i.e., $R_r = \{0, \sigma_r(1),..., \sigma_r(n_r), \sigma_s(1),..., \sigma_s(n_s), 0\}$. Label vehicle s as an inactive vehicle, i.e., let $R_s = \{\phi\}$.

Step 3. If $\Omega = \phi$ then stop; else go to step 2.

Step 4. Return the solution $S = \{R_1, R_2,..., R_v\}$ where $R_r \neq \{\phi\}$ for $r = 1,...,v$.

6.3.1. Saving Algorithm (SA) for VRPSPD

The saving algorithm for VRPSPD is similar to the saving algorithm for CVRP. In VRPSPD, we consider all eight methods described in section 3 for combining routes R_r and R_s. Here n is the total number of customers in the problem. The saving algorithm for VRPSPD works as follows;

Step 1: Calculate the saving value for each combination of customers k and l, i.e., $s_{kl} = c_{0k} + c_{0l} - c_{kl}$ for $k,l = 1,...,n$ and $k \neq l$. Initialize vector Ω by storing saving values s_{kl} in non-increasing order. Assign an individual vehicle to each customer and create the routes $R_m = \{0, \sigma_m(1), 0\}$; for $m = 1,...,n$.

Step 2. Consider the first element of Ω, say s_{kl} and remove it from Ω. Determine the vehicles that serve customer k and customer l. Suppose customer k is served by vehicle r and customer l is served by vehicle s. Now check the following conditions:

1 Customer k is either the first or the last visited customer in route R_r.

2 Customer l is either the first or the last visited customer in route R_s.

3 Vehicle r and s are different i.e., $r \neq s$.

If the above conditions are satisfied then check the feasibility of the combined route using the four conditions described below. If the conditions are satisfied then assign vehicle q to serve the combined route. Label vehicles r and s as inactive vehicles, i.e., $R_r = \{\phi\}$ and $R_s = \{\phi\}$.

1. Customer k is the last visited customer in route R_r and customer l is the first visited customer in route R_s. If equation (6.17) is satisfied, then use method M1 for combining the two routes; else if equation (6.21) is satisfied, use method M5 for combining the two routes.
2. Customers k and l are the last visited customers in routes R_r and R_s, respectively. If equation (6.18) is satisfied, then use method M2 for combining the two routes; else if equation (6.22) is satisfied, use method M6 for combining the two routes.
3. Customer k and l are the first visited customers in routes R_r and R_s respectively. If equation (6.19) is satisfied, then use method M3 for combining the two routes; else if equation (6.23) is satisfied then use method M7 for combing the two routes.
4. Customer k is the first visited customer in route R_r and customer l is the last visited customer in route R_s. If equation (6.20) is satisfied, then use method M4 for combining two routes; else if equation (6.24) is satisfied then use method M8 for combing the two routes.

Update X_q, Y_q using equations (6.1) to (6.16) on the basis of the method used to obtain R_q. Similarly, update $F_r(n_r)$ using lemma 6.6.

Step 3. if $\Omega = \phi$ then stop; else go to step 2.

Step 4. Return the solution $S = \{R_1, ..., R_p, ..., R_v\}$ such that $R_p \neq \{\phi\}$ for $r = 1, ..., v$.

6.3.2 Generalized Saving Algorithm (GSA) for VRPSPD

The saving algorithm by Clarke and Wright (1964) for CVRP was improved by Yellow (1970) by introducing a route shape parameter λ in the saving calculation. Suppose we are combining routes $R_r = \{0, \sigma_r(1), \ldots\ldots, \sigma_r(n_r), 0\}$ and $R_s = \{0, \sigma_s(1), \ldots\ldots, \sigma_s(n_s), 0\}$. Let customer $a = \sigma_r(n_r)$ and customer $b = \sigma_s(1)$. Then the modification proposed by Yellow (1970) to calculate the saving value s_{ab} is

$$s_{ab} = c_{0a} + c_{0b} - \lambda c_{ab} \qquad (6.25)$$

One drawback of the Clarke and Wright (1964) algorithm is that it tends to produce good routes at the beginning of the solution procedure but less interesting routes toward the end. The shape parameter λ was introduced to overcome this drawback. The larger the value of λ, the more the emphasis is on the distance between the customers that are to be connected.

The generalized saving algorithm is similar to the saving algorithm described in the previous section. The only difference is the calculation of the saving value by equation (6.25) in step 1 of the saving algorithm. Thus, the saving algorithm becomes the special case of the generalized saving algorithm when λ is set to 1. In the generalized saving algorithm, different solutions are generated with different value of λ and the best solution among them is chosen.

6.4. Parallel saving based algorithm

An interesting modification to the classical saving algorithm by Clarke and Wright (1964) is described by Desrochers and Verhoog (1989) and Altinkemer and

Gavish (1991). Both of these studies proposed the parallel saving based algorithm (PSA) for CVRP, which is based upon the concept of optimal matching among different routes. The basic idea of PSA is to merge multiple vehicle routes instead of merging just two routes as in the classical saving algorithm. The classical saving algorithm is a special case of the parallel saving algorithm in the sense that just two routes are combined in a given iteration. We propose the following extension of PSA for solving VRPSPD.

6.4.1 Parallel saving based algorithm for CVRP

Let,

$C(R_r)$ the cost of the optimum traveling salesman tour over customers visited in route R_r,

S_{rs} the saving obtained by combining routes R_r and R_s,

V_t the set of vehicle routes used at the end of iteration t,

M_t a set of dummy nodes at iteration t; $|M_t| = n-(t+1)*T$.

T a parameter that determines the number of pairs to be merged in a given iteration.

In each iteration, a maximum weighted matching problem is solved. In the maximum weighted matching problem, we are given graph $G = (V, E)$, with weight $w_{pq} \geq 0$ for each arc $(p, q) \in E$. The objective is to find a subset of arcs so that each node is connected to only one other node and the sum of the weights for the subset is maximized.

Note in the t^{th} iteration, there are $|M_t| = n-(t+1)*T$ dummy nodes. A dummy node cannot be matched with any other dummy node. Given the dummy nodes, the

number of real nodes matched with each other in each iteration is $2T$ and the number of matched pairs is T. The basic steps of the parallel saving algorithm of Altinkemer and Gavish (1991) are described below:

Step 0: (Initialization) Assign individual vehicle to each customer and create route R_m = $\{0, \sigma_m(1), 0\}$ for $m = 1,2,\ldots n$. These routes represent the real nodes of a initial graph $G = (V_0, E)$, where, $|V_0| = n$ and E is the set of all arcs (p, q); $p, q \in V_0$. Choose a value for parameter T (see section 6.5.1 for setting the parameter T). Let $t = 1$,

Step 1. Add dummy nodes M_t, such that $|M_t| = n-(t+1)T$. Consider node set $V = V_{t-1} \cup M_t$. For every $p, q \in V$ do

$$S_{pq} = \begin{cases} 0 & p < q \text{ and } p,q \in M_t \\ -\infty & p < q \text{ and } p \in V_{t-1} \text{ and } q \in M_t. \end{cases}$$

For every p and q, $q > p$ and $p, q \in V_{t-1}$ do:

a. If the combined load of routes R_p and R_q is greater than the vehicle capacity, then set $S_{pq} = -\infty$ and go to the next combination of p and q.

b. Otherwise compute S_{pq} according to the following formula:

$$S_{pq} = C(R_p) + C(R_q) - C(R_p \cup R_q)$$

Step 2: If each $S_{pq}; p, q \in V$ is equal to either 0 or $-\infty$, then there are no pairs of existing routes that can be combined in a feasible way. In this case, stop the algorithm. Else, go to Step 3.

Step 3: Solve the maximum weighted matching problem for the graph $G = (V, E)$ where node set $V = V_{t-1} \cup M_t$ and weight $w_{pq} = S_{pq}$ for each $p, q \in V$. For each matching pair (r, s); $r, s \in V_{t-1}$, combine routes R_r and R_s. Let V_t be the set of current routes (including the combined routes), $t = t+1$, and go to step 1.

The calculation of the saving value S_{pq} in step 1b is the saving value obtained when the traveling salesman problem is solved exactly before and after the merging of routes R_p and R_q. Solving the traveling salesman problem exactly makes the PSA algorithm non-polynomial. Altinkemer and Gavish (1991) have proposed the following modification in the algorithm to make it polynomial. They denote the resulting algorithms as PSA2 and PSA3. Let customer $a = \sigma_p(1)$, $b = \sigma_p(n_p)$, $c = \sigma_q(1)$ and $d = \sigma_q(n_q)$. The saving values, S_{pq} is calculated in PSA2 and PSA3 as:

$$S_{pq} = \max \begin{cases} c_{0b} + c_{0c} - c_{bc} \\ c_{0b} + c_{0d} - c_{bd} \\ c_{0a} + c_{0c} - c_{ac} \\ c_{0a} + c_{0d} - c_{ad} \end{cases} \tag{6.26}$$

The above formula estimates the saving value (in terms of the total tour length) when two routes are connected by their end customers. In PSA2, two routes are combined by their end customers in step 2 of the algorithm. In PSA3, instead a TSP is solved exactly for the combined route.

6.4.2 Extended parallel saving algorithms for VRPSPD

The parallel saving algorithm of Altinkemer and Gavish (1991) solves a weighted matching problem with the node set $V = V_{t-1} \cup M_t$ at the t^{th} iteration, where $|M_t| = n-(t+1)T$. Two routes are merged only if they both are not dummy nodes. Altinkemer and Gavish (1991) do not describe the algorithm that they used for solving the maximum weighted matching problem. Although the dummy nodes insure that the weighted matching problem is solved optimally, using them increases the CPU time.

We avoid using the dummy nodes in the extended parallel saving algorithms. In the t^{th} iteration, we solve the maximum weighted matching problem with $V = V_{t-1}$. We use the algorithm proposed by Edmonds (1965) (described in Papadimitriou and Steiglitz (1998)) to solve the maximum weighted matching problem. Edmonds (1965) algorithm is designed to find the $V/2$ matched pairs. However, our goal is to obtain T pairs. Therefore, we stop the algorithm as soon as it finds T matching pairs. To make the algorithm polynomial, we calculate the saving value using equation (6.26). We call the algorithm as the extended parallel saving algorithm (EPSA2). It is similar to PSA2 and is now described below.

Step 0: (Initialization) Assign a vehicle to each customer and create route R_m = {0, $\sigma_m(1)$, 0} for m = 1,2,...,n. Choose a value for parameter T. Let t =1, V_0 = n.

Step 1. For every p and $q \in V_{t-1}$, $p < q$ do:

$$S^1_{pq} = \begin{cases} c_{0b} + c_{0c} - c_{bc} & \text{if equation (6.17) or equation (6.21) is satisfied} \\ 0 & \text{otherwise} \end{cases} \qquad (6.27)$$

$$S_{pq}^2 = \begin{cases} c_{0b} + c_{0d} - c_{bd} & \text{if equation (18) or equation (22) is satisfied} \\ 0 & \text{otherwise} \end{cases} \quad (6.28)$$

$$S_{pq}^3 = \begin{cases} c_{0a} + c_{0c} - c_{ac} & \text{if equation (19) or equation (23) is satisfied} \\ 0 & \text{otherwise} \end{cases} \quad (6.29)$$

$$S_{pq}^4 = \begin{cases} c_{0b} + c_{0c} - c_{bc} & \text{if equation (20) or equation (24) is satisfied} \\ 0 & \text{otherwise} \end{cases} \quad (6.30)$$

and let $S_{pq} = Max\{S_{pq}^1, S_{pq}^2, S_{pq}^3, S_{pq}^3\}$ (6.31)

If $S_{pq} = 0$, combining routes R_p and R_q is not feasible. In this case, we designate the merging of R_p with R_q or merging of R_q with R_p non-admissible.

Step 2: Solve the maximum weighted matching problem with V_{t-1} as its node set using Edmonds (1965) algorithm. Stop the algorithm as soon as T matching pairs are found. For each matching pair (r, s), combine routes R_r and R_s. There are eight different methods for combining routes R_r and R_s. We choose the method M1 to M8 that gives us the maximum value for S_{pq} in equation (6.31)

Step 3: If all mergers are non-admissible then stop, otherwise, let V_t be the set of current routes. Let $t = t + 1$, and go back to step 1.

EPSA3 is similar to PSA3 of Altinkemer and Gavish (1991) and is an improvement over EPSA2. In PSA3, a TSP is solved exactly whenever two existing routes are combined. Similarly, in EPSA3, whenever the two existing routes are combined, the combined route is improved using the 2-opt method of Lin (1965). Unlike PSA3, EPSA3 does not use the exact algorithm to obtain TSP tour for the combined

route. However, the 2-opt algorithm is usually able to find a good solution in less CPU time compare to exact procedures.

6.5. Numerical Analysis

This section presents numerical results for the saving based algorithms proposed in this chapter and compares them with the existing heuristics for VRPSPD and VRPBM.

6.5.1 Numerical Analysis for VRPSPD

The saving based heuristics were coded in C++ and implemented on Intel Xeon with 2.4 GHz computer system. As described before the heuristics are:

SA - Saving Heuristics based on Clarke and Wright (1964) saving heuristic

GSA - Generalized saving heuristic based on Generalized saving of Yellow (1970).

EPSA2 – Extended parallel saving heuristics based on PSA2 heuristics of Altinkemer and Gavish (1991)

EPSA3 – Extended parallel saving heuristics based on PSA3 heuristics of Altinkemer and Gavish (1991).

We compare the above new heuristics with the following existing heuristics for which there are reported results.

LC-INS the heuristic algorithm of Salhi and Nagy (1999).

RCRS the heuristic algorithm of Dethloff (2001),

ALT the heuristic algorithm of Salhi and Nagy (1999).

Benchmark problem instances

The numerical experiment is performed using two sets of problem instances available in the literature for VRPSPD. The first data set is provided by Dethloff (2001) and the second data set is provided by Salhi and Nagy (1999). See section 5.3.2 in chapter 5 for a detailed description of the two data sets.

Altogether, we have used 68 problem instances for VRPSPD: 40 instances from Dethloff (2001) and 28 instances from Salhi and Nagy (1999). Dethloff (2001) instances are referred to as SCA and CON. Salhi and Nagy (1999) instances are referred to as CMTX/Y.

Parameter settings

Our experiment shows that GSA produces a better solution when λ is set between 1 and 2. We generate 5 solutions in GSA using λ of 1, 1.2, 1.4, 1.6 and 1.8 and report the best solution out of these 5 solutions. In our experiment, when we increased λ in steps of 0.1, we got the same solutions most of the times for consecutive λ.

In EPSA2 and EPSA3, parameter T determines the rate of merging process. The larger the value of T, the larger the number of routes that are merged in a given iteration. Note, when $T = 1$, EPSA2 is equivalent to SA. Altinkemer and Gavish (1991) have found that T greater than 13 seems to reduce the solution quality. Hence, they varied T from 2 to 12 in their study and reported the best solution. In our experiment with VRPSPD problem instances, we found that T greater than 6 did not contribute significantly to the solution quality. Hence, we vary T from 2 to 6 in steps of 1 and report the best results out of the 5 solutions generated with EPSA2 and EPSA3.

Computational results and performance analysis of the algorithm

The average costs for the problem instances obtained by different algorithms are presented in Table 6.1 and Table 6.2 for Dethloff (2001) and Salhi and Nagy (1999) problem instances, respectively. The detailed solutions for VRPSPD problem instances using different algorithms are presented in Tables 6.3 and 6.4.

The results reported in Table 6.1 and Table 6.2 indicate that all saving based algorithms designed in this chapter, perform better than the existing insertion based heuristics. Among the four new heuristics designed in chapter, EPSA3 gives the lowest average solution cost. The order in terms of decreasing performance is EPSA3, EPSA2, GSA, SA, ALT, RCRS and LC-INS. As seen from Table 6.1 and Table 6.2, the order of performance is the same for both data sets.

Out of the four heuristics developed in this chapter, the lowest performance (i.e., the highest average cost) is obtained by SA. However, the CPU time for SA is the lowest. Note also that on average, the solution produced by SA is better than the solution produced by the existing heuristics. For Dethloff (2001) problem instances, the average cost for SA is 796.61 while the average cost for RCRS is 842.72. In Salhi and Nagy (1999) problem instances, ALT is the nearest competitor to SA. The average cost for SA is 975.96 whereas the average cost for ALT is 990.

Table 6.1: Average tour length obtained by different heuristics and the corresponding CPU time for 40 VRPSPD instances of Dethloff (2001)

	LC-INS	RCRS	ALT	SA	GSA	EPSA2	EPSA3
Average cost	-	842.72	-	796.61	787.63	785.67	779.00
Average time (sec.)	-	-	-	0.04	0.187	0.218	2.98

Table 6.2: Average tour length obtained by different heuristics and the corresponding CPU time for 28 VRPSPD instances of Salhi and Nagy (1999)

	LC-INS	RCRS	ALT	SA	GSA	EPSA2	EPSA3
Average cost	1087.00	1010.79	990.00	975.96	964.12	959.87	953.24
Average time (sec.)	4.80	-	3.60	0.46	2.27	2.13	2.43

Table 6.3: Total tour length obtained by different heuristics and the corresponding CPU time for 40 VRPSPD instances of Dethloff (2001)

No	Inst.	RCRS	SH		GSH		EPSA2		EPSA3	
		Cost	Cost	Time	Cost	Time	Cost	Time	Cost	Time
1	SCA3-0	689.00	684.72	0.04	658.27	0.19	684.72	0.22	651.55	0.40
2	SCA3-1	765.60	701.98	0.04	701.98	0.19	709.00	0.21	701.86	0.42
3	SCA3-2	742.80	706.83	0.04	675.74	0.19	684.58	0.22	670.62	0.41
4	SCA3-3	737.20	726.47	0.04	709.63	0.19	710.11	0.22	683.11	0.41
5	SCA3-4	747.10	796.99	0.04	711.91	0.19	766.13	0.21	762.74	0.39
6	SCA3-5	784.40	753.62	0.04	709.68	0.19	671.03	0.23	671.03	0.42
7	SCA3-6	720.40	717.45	0.04	689.28	0.19	687.46	0.22	687.46	0.39
8	SCA3-7	707.90	688.55	0.04	675.83	0.19	681.14	0.21	671.34	0.41
9	SCA3-8	807.20	746.35	0.04	746.35	0.19	761.97	0.21	748.39	0.41
10	SCA3-9	764.10	704.94	0.04	704.94	0.19	710.16	0.21	700.98	0.39
11	SCA8-0	1132.90	1017.79	0.04	1017.79	0.18	1017.87	0.21	1015.06	0.32
12	SCA8-1	1150.90	1129.43	0.04	1125.25	0.18	1100.92	0.21	1098.91	0.30
13	SCA8-2	1100.80	1079.30	0.04	1079.30	0.19	1073.11	0.21	1064.54	0.30
14	SCA8-3	1115.60	1049.34	0.04	1047.74	0.18	1036.30	0.22	1021.61	0.32
15	SCA8-4	1235.40	1112.03	0.04	1089.14	0.18	1112.03	0.18	1114.50	0.28
16	SCA8-5	1231.60	1085.68	0.04	1081.79	0.18	1060.60	0.21	1060.43	0.31
17	SCA8-6	1062.50	1003.49	0.04	1003.49	0.19	1004.66	0.21	1004.66	0.32
18	SCA8-7	1217.40	1087.86	0.04	1080.86	0.18	1080.17	0.20	1080.17	0.28
19	SCA8-8	1231.60	1120.17	0.04	1105.30	0.19	1130.23	0.21	1098.02	0.32
20	SCA8-9	1185.60	1135.72	0.04	1135.72	0.19	1126.18	0.20	1123.55	0.28
21	CON3-0	672.40	639.82	0.04	635.53	0.19	639.82	0.24	635.96	0.41
22	CON3-1	570.60	581.44	0.04	577.05	0.19	568.98	0.23	559.02	0.45
23	CON3-2	534.80	534.67	0.04	530.58	0.19	534.24	0.25	526.81	0.43
24	CON3-3	656.90	611.21	0.04	611.21	0.19	604.68	0.22	598.49	0.40
25	CON3-4	640.20	619.94	0.04	613.23	0.19	609.46	0.21	598.37	0.43
26	CON3-5	604.70	608.69	0.04	592.55	0.19	569.52	0.22	565.78	0.41
27	CON3-6	521.30	521.92	0.04	521.92	0.19	520.67	0.22	506.32	0.43
28	CON3-7	602.80	603.41	0.04	599.25	0.19	595.27	0.24	582.49	0.43

29	CON3-8	556.20	603.41	0.04	599.25	0.19	541.82	0.23	540.55	0.42
30	CON3-9	612.80	600.67	0.04	600.64	0.19	595.84	0.22	593.55	0.41
31	CON8-0	967.30	857.17	0.04	857.17	0.19	857.17	0.25	857.17	0.33
32	CON8-1	828.70	762.26	0.04	762.26	0.19	744.87	0.23	742.41	0.31
33	CON8-2	770.20	734.84	0.04	734.84	0.18	716.18	0.22	715.17	0.31
34	CON8-3	906.70	818.72	0.04	818.72	0.18	815.59	0.21	815.59	0.31
35	CON8-4	876.80	802.00	0.04	799.00	0.18	789.13	0.21	789.13	0.31
36	CON8-5	866.90	766.02	0.04	766.02	0.18	758.90	0.22	759.09	0.30
37	CON8-6	749.10	704.06	0.04	704.06	0.19	709.10	0.25	707.83	0.32
38	CON8-7	929.80	824.72	0.04	824.72	0.19	834.64	0.21	834.64	0.32
39	CON8-8	833.10	791.08	0.04	791.08	0.19	792.83	0.21	787.43	0.31
40	CON8-9	877.30	829.92	0.04	816.14	0.19	819.87	0.22	813.84	0.31
Average		842.72	796.62	0.04	787.63	0.19	785.67	0.22	779.00	0.36

Table 6.4: Total tour length obtained by different heuristics and the corresponding CPU time for 28 VRPSPD instances of Salhi and Nagy (1999)

No	Inst.	LC-INS	RCRS	SA		GSA		EPSA2		EPSA3	
		Cost	Cost	Cost	Time	Cost	Time	Cost	Time	Cost	Time
1	CMT1X	601	501	501.22	0.04	493.66	0.20	484.47	0.20	490.74	0.30
2	CMT2X	903	782	732.57	0.10	739.11	0.51	730.32	0.55	726.43	0.69
3	CMT3X	923	847	818.04	0.23	779.79	1.12	769.32	1.09	769.21	1.40
4	CMT4X	1178	1050	965.86	0.70	973.11	3.48	951.73	3.06	947.10	3.62
5	CMT5X	1509	1348	1195.59	1.55	1153.70	7.78	1143.99	6.66	1126.07	7.41
6	CMT6X	594	584	618.39	0.03	597.35	0.18	598.61	0.19	587.48	0.27
7	CMT7X	-	961	953.05	0.10	944.82	0.51	943.00	0.53	939.18	0.63
8	CMT8X	923	928	977.17	0.23	954.02	1.11	963.23	1.09	955.27	1.32
9	CMT9X	1215	1299	1287.64	0.70	1234.45	3.41	1287.64	3.09	1286.92	3.36
10	CMT10X	1573	1571	1509.04	1.55	1483.88	7.68	1490.40	6.78	1481.00	7.27
11	CMT11X	1500	959	937.28	0.37	881.35	1.82	934.85	2.03	917.11	2.57
12	CMT12X	831	804	720.04	0.23	711.59	1.10	712.12	1.22	706.36	1.45
13	CMT13X	1613	1576	1592.26	0.38	1647.20	1.81	1575.40	2.05	1564.67	2.27
14	CMT14X	954	871	853.15	0.23	903.64	1.12	853.15	1.20	847.83	1.44
15	CMT1Y	603	501	501.22	0.04	493.66	0.19	484.47	0.20	490.74	0.31
16	CMT2Y	924	782	732.57	0.11	739.11	0.51	730.32	0.55	726.43	0.68
17	CMT3Y	923	847	818.04	0.23	779.79	1.12	769.32	1.10	769.21	1.39
18	CMT4Y	1178	1050	970.18	0.71	973.11	3.45	951.73	3.10	947.10	3.63
19	CMT5Y	1477	1348	1195.59	1.56	1153.70	7.69	1143.99	6.72	1126.07	7.40
20	CMT6Y	609	584	618.39	0.04	597.35	0.18	598.61	0.20	587.48	0.27
21	CMT7Y	-	961	953.05	0.10	944.82	0.51	943.00	0.53	939.18	0.62
22	CMT8Y	927	936	977.17	0.23	954.02	1.13	963.23	1.09	955.27	1.31
23	CMT9Y	1215	1299	1287.64	0.69	1234.45	3.39	1287.64	3.07	1286.92	3.37
24	CMT10Y	1527	1571	1509.04	1.56	1483.88	7.64	1490.40	6.81	1481.00	7.17
25	CMT11Y	1500	1070	937.28	0.37	881.35	1.86	934.85	2.04	917.11	2.62
26	CMT12Y	873	825	720.04	0.23	711.59	1.12	712.12	1.21	706.36	1.46
27	CMT13Y	1589	1576	1592.26	0.37	1647.20	1.81	1575.40	2.03	1564.67	2.28
28	CMT14Y	920	871	853.15	0.23	903.64	1.13	853.15	1.20	847.83	1.41
Average		1099.31	1010.79	975.96	0.46	964.12	2.27	959.87	2.13	953.24	2.43

6.5.2 Numerical Analysis for VRPBM.

The cumulative net-pickup approach was mainly designed for VRPSPD. Since VRPBM is a special case of VRPSPD, the cumulative net-pickup approach was also applied to VRPBM problem instances. In VRPSPD, each customer is a linehaul as well

as a backhaul customer while in VRPBM a customer is either a linehaul customer or a backhaul customer. The parameter settings for VRPBM problem instances are the same as described in sub-section 6.1.2. We compare our heuristics with the following existing heuristics for which there are reported results.

LC-INS the heuristic algorithm of Salhi and Nagy (1999).

ALT the heuristic algorithm of Nagy and Salhi (2005).

Benchmark problem

The numerical experiment is performed using the data set of Salhi and Nagy (1999) for VRBPM. They generated three sets of problem instances referred as sets T, Q and H using the 14 benchmark problem instances given by Christofides et al. (1979). In set T, every tenth customer is declared as a backhaul customer and is assigned pickup demand equal to the original demand. Hence, in set T, 10% customers are declared backhaul customers. Similarly, in set Q, 25% of customers are declared backhaul customers and in set H, 50% of customers are declared backhaul customers. We refer to the above VRPBM problem instances as CMTT, CMTQ and CMTH, respectively in our comparison.

Computational results and performance analysis of the algorithm

The average results obtained by different algorithms are presented in Table 6.5. The detailed solutions obtained by different algorithms are presented in Table 6.6. The results reported in Tables 6.5 and 6.6 show that the saving based algorithms designed in this chapter work well for VRPBM problem instances. The order of performance is

similar to the one observed for VRPSPD problem instances. EPSA3 gives the lowest average cost followed by EPSA2, GSA, SA, ALT and LC-INS.

The performance of SA is the lowest among the four saving based heuristics designed in this chapter. The average cost produced by SA is 976.03 while the average costs for the existing heuristics ALT and LC-INS are 994.67 and 1037.77, respectively. Note again that SA requires the least CPU time in each case.

Table 6.5: Average tour length obtained by different heuristics and the corresponding CPU time for 42 VRPBM instances of Salhi and Nagy (1999)

% of backhaul customers		LC-INS	ALT	SA	GSA	EPSA 2	EPSA 3
50 %	Average cost	1047.0	991.00	955.80	939.00	942.13	933.64
	Average time (sec.)	3.60	3.90	0.43	2.31	2.12	3.20
25 %	Average cost	1035.0	998.00	976.06	962.97	963.64	957.71
	Average time (sec.)	3.10	3.60	0.36	1.80	1.68	2.51
10 %	Average cost	1011.0	995.00	981.19	996.21	981.13	975.00
	Average time (sec.)	2.80	3.10	0.47	2.30	2.13	3.18

Table 6.6: Total tour length obtained by different heuristics and corresponding CPU time for 42 VRPBM instances of Salhi and Nagy (1999)

No	Inst.	LC-INS	SA		GSA		EPSA2		EPSA3	
		Cost	Cost	Time	Cost	Time	Cost	Time	Cost	Time
1	CMT1H	594	503.02	0.04	473.05	0.19	470.94	0.20	470.94	0.32
2	CMT2H	900	719.73	0.11	709.99	0.53	722.00	0.54	711.19	0.82
3	CMT3H	918	804.31	0.24	751.01	1.13	750.95	1.10	745.60	1.71
4	CMT4H	1164	920.62	0.70	892.32	3.48	897.86	3.09	887.48	4.73
5	CMT5H	1509	1067.28	1.59	1067.28	7.84	1067.28	6.72	1061.56	10.23
6	CMT6H	594	618.39	0.04	594.67	0.18	598.61	0.19	587.48	0.31
7	CMT7H	-	953.05	0.11	937.82	0.52	934.22	0.52	930.40	0.76
8	CMT8H	915	977.17	0.23	967.90	1.14	963.23	1.09	955.27	1.66
9	CMT9H	1164	1287.64	0.71	1236.24	3.46	1287.64	3.05	1286.92	4.55
10	CMT10H	1509	1509.04	1.62	1496.33	7.92	1490.40	6.72	1481.00	9.93
11	CMT11H	1120	891.45	0.38	891.45	1.83	889.90	2.03	870.88	3.25
12	CMT12H	850	684.08	0.23	684.08	1.12	688.22	1.21	669.74	1.86
13	CMT13H	1546	1592.26	0.38	1592.26	1.85	1575.40	2.05	1564.67	2.9
14	CMT14H	866	853.15	0.23	853.15	1.14	853.15	1.21	847.83	1.77
15	CMT1Q	557	547.17	0.04	521.28	0.19	524.32	0.19	517.88	0.33
16	CMT2Q	860	792.18	0.11	778.44	0.53	768.43	0.56	763.78	0.83

17	CMT3Q	918	803.20	0.23	803.20	1.13	794.17	1.09	794.06	1.76
18	CMT4Q	1164	1014.86	0.71	1003.55	3.54	982.96	3.09	982.90	4.71
19	CMT5Q	1477	957.49	0.11	935.53	0.52	955.70	0.54	954.51	0.76
20	CMT6Q	600	618.39	0.04	594.67	0.19	598.61	0.20	587.48	0.32
21	CMT7Q	-	957.49	0.10	935.53	0.53	955.70	0.52	954.51	0.76
22	CMT8Q	918	969.34	0.23	968.74	1.12	963.23	1.11	955.27	1.67
23	CMT9Q	1178	1287.64	0.70	1236.24	3.50	1287.64	3.06	1286.92	4.55
24	CMT10Q	1477	1509.04	1.60	1496.33	7.97	1490.40	6.70	1481.00	10.01
25	CMT11Q	1087	984.97	0.38	984.97	1.83	982.20	2.03	964.72	3.1
26	CMT12Q	853	777.73	0.24	777.73	1.13	759.03	1.21	752.45	1.79
27	CMT13Q	1613	1592.26	0.38	1592.26	1.87	1575.40	2.03	1564.67	2.87
28	CMT14Q	873	853.15	0.23	853.15	1.11	853.15	1.23	847.83	1.73
29	CMT1T	541	547.17	0.04	521.28	0.18	524.32	0.19	517.88	0.32
30	CMT2T	839	792.18	0.11	778.44	0.52	768.43	0.55	763.78	0.82
31	CMT3T	903	803.20	0.23	803.20	1.13	794.17	1.10	794.06	1.75
32	CMT4T	1111	1014.86	0.70	1003.55	3.43	982.96	3.14	982.90	4.68
33	CMT5T	1423	1239.57	1.59	1189.76	7.80	1201.46	6.69	1196.60	10.11
34	CMT6T	571	618.39	0.04	594.67	0.18	598.61	0.20	587.48	0.31
35	CMT7T	-	957.49	0.11	935.53	0.52	955.70	0.54	954.51	0.76
36	CMT8T	911	969.34	0.23	968.74	1.14	963.23	1.09	955.27	1.64
37	CMT9T	1164	1287.64	0.71	1236.24	3.47	1287.64	3.06	1286.92	4.62
38	CMT10T	1418	1509.04	1.59	1496.33	7.92	1490.40	6.69	1481.00	10.01
39	CMT11T	1075	984.97	0.38	984.97	1.88	982.20	2.03	964.72	3.09
40	CMT12T	827	777.73	0.23	777.73	1.13	759.03	1.21	752.45	1.78
41	CMT13T	1600	1592.26	0.38	1592.26	1.83	1575.40	2.05	1564.67	2.85
42	CMT14T	866	853.15	0.23	853.15	1.11	853.15	1.22	847.83	1.76
Average		1037.77	976.03	0.44	961.07	2.14	962.32	1.98	955.45	2.96

6.5.3 Comparisons among saving algorithms

The better performance of EPSA3 in comparison to EPSA2 is attributed to the local search. The use of 2-opt in EPSA3 increase the CPU time. The gap of CPU time between EPSA3 and EPSA2 seems to increase with the problem size. Note that GSA would give better solution than SA for each problem instance since we let $\lambda = 1$. Since we allow five values for λ in GSA, the CPU time for GSA is almost 5 times the CPU time of SA.

The saving algorithms (SA and GSA) and parallel saving algorithms (EPSA2 and EPSA2) are competitive. The performance (in terms of the average cost) of the algorithm from best to worst is in the order EPSA3, EPSA2, GSA and SA. The order however is exactly the opposite in terms of CPU time.

6.5.4 Comparison of saving algorithms with metaheuristics

In comparison to heuristics, metaheuristic invariably use more CPU time to find a better solution. We compare our saving algorithms with metaheuristics mainly to see how close our solution is in comparison to the metaheuristics. We compare the saving algorithm with the following two metaheuristics.

ACS Ant colony system presented in chapter 5.

LNS large neighborhood search by Ropke and Pisinger (2006)

The summary results of different algorithms are presented in Table 6.7, Table 6.8 and Table 6.9. The summary results for Dethloff (2001) VRPSPD problem instances reported in Table 6.7 show that our best algorithm EPSA3 produces a solution that is on average 2.54 % away from the best known solution and it took on average 2.98 seconds of CPU time. The LNS solution is 0.31 % away from the best known solution but it took on average 58.12 seconds. The best results are obtained by ACS, which produces the best known solution for all instances. It took on average 9.29 seconds. It should be noted that the size of all problem instances in Dethloff (2001) data set is only 50.

The summary results for Salhi and Nagy (1999) VRPSPD problem instances reported in Table 6.8 show that the solution produced by EPSA3 is on average 6.82 % away from the best known solution and it took 2.43 seconds to produce it. LNS solution is 3.54 % away from the best known solution but it took 299.31 seconds. ACS solution is 0.01 % away from best known solution but it took 151.54 seconds.

The summary results for Salhi and Nagy (1999) VRPBM problem instances reported in Table 6.9 show that the solution produced by EPSA3 is on average 4.17 %

away from the best known solution and it took 2.96 seconds. LNS solution is just 0.09 % away from the best known solution but it took 55.86 seconds. ACS solution is 0.1 % away from best known solution and it took 151.38 seconds of CPU time. Thus, overall EPSA3 produces a solution that is close to the best known solution with significantly less CPU time.

Table 6.7: Summary results of heuristics and metaheuristics algorithm for 40 VRPSPD instances of Dethloff (2001)

	SA	GSA	EPSA2	EPSA3	LNS	ACS
Average cost	796.6	787.63	785.67	779	761.07	758.64
% deviation from the best known solution	5.15	3.86	3.51	2.54	0.31	0
Average CPU time	0.04	0.187	0.218	2.98	157.93	9.29
Average Scaled CPU time	0.04	0.187	0.218	2.98	58.12	9.29

Table 6.8: Summary results of heuristics and metaheuristics algorithm for 28 VRPSPD instances of Salhi and Nagy (1999)

	SA	GSA	EPSA2	EPSA3	LNS	ACS
Average cost	975.96	964.12	959.87	953.24	932.62	893.22
% deviation from the best known solution	9.52	7.99	7.53	6.82	3.54	0.01
Average CPU time	0.46	2.27	2.13	2.43	813.33	151.54
Average Scaled CPU time	0.46	2.27	2.13	2.43	299.31	151.54

Table 6.9: Summary results of heuristics and metaheuristics algorithm for 42 VRPBM instances of Salhi and Nagy (1999)

	SA	GSA	EPSA2	EPSA3	LNS	ACS
Average cost	976.03	961.07	962.32	955.45	919.59	919.57
% deviation from the best known solution	6.64	4.90	5.02	4.17	0.09	0.1
Average CPU time	0.44	2.14	1.98	2.96	151.48	151.38
Average Scaled CPU time	0.44	2.14	1.98	2.96	55.86	151.38

6.6. Conclusions

The problem of VRPSPD is important given the new emphasis on integrating the forward and reverse flows in supply chains. Although the saving heuristic of Clarke and Wright (1964) is used widely in VRP, no such heuristics exist for VRPSPD. In this chapter, we have filled the gap and designed four saving based heuristics (SA, GSA, EPSA2 and EPSA3) for VRPSPD. All four heuristics involve creating a new route by merging two existing routes. We have also proposed the cumulative net-pickup approach to checking the feasibility of a route that is obtained by merging two existing routes. Computational experiment with benchmark problem instances has shown that all four saving based heuristics perform better than the existing insertion based heuristics. Out of the four heuristics developed in this chapter, SA yields the highest average cost but it also requires the least amount of CPU time. Comparing EPSA3 with metaheuristics, we found that EPSA3 produces a solution that is close to the solutions found by the metaheuristics. However, it did so in significant less CPU time.

CHAPTER 7

Saving based algorithm for multi-depot version of Vehicle routing problem with simultaneous pickup and delivery

7.1 Introduction

In multi-depot vehicle routing problem, goods are delivered from several depots to a set of geographically dispersed customers. Multi-depot vehicle routing problem (MDVRP) is encountered in a large number of real life situations. Some of the real life examples of MDVRP are the delivery of meals (Cassidy and Bennett (1972)) , delivery of packaged food (Pooley (1994)). Other examples are distribution of chemical products (Ball et al. (1983)), distribution of petroleum products (Brown et al. (1987)), distribution of industrial gases (Bell et al. (1983)), distribution of soft drinks (Golden and Wasil (1987)), supply of machines (Fisher et al. (1981; Magnanti (1981)), etc. The classical MDVRP works independently for delivery and pickup of materials from a customer. In the multi-depot version of vehicle routing problem with pickup and delivery (VRPSPD), a customer requires a given shipment to be delivered as well as a given load to be delivered. Complete service (i.e., delivery and pickup) is provided by a vehicle to a customer in a single stop. The vehicle starts from one of the several depots. A multi-depot VRPSPD instance is defined in terms of the following data.

m the number of depots;

n the number of customers;

M the set of depots, numbered 1,...,m;

N the set of customers to be visited, numbered m+1,...,$m + n$;

$G(V, A)$ a complete graph where $V = M \cup N$ is the vertex set and A is the arc set;

p_i non-negative pickup demand for customer i where $i \in N$;

d_i non-negative delivery demand for customer i where $i \in N$;

c_{ij} non-negative cost associated with each arc $(i, j) \in A$ satisfying the triangular inequality (i.e., $c_{ij} \leq c_{ik} + c_{kj}$) for each $k = 1,2,\ldots n$; $k \neq i$; k $\neq$ j.

Q vehicle capacity.

The objective is to determine the minimum cost routes of vehicles so that

- each vehicle starts and ends at the same depot;
- each customer is visited by exactly one vehicle;
- complete service (i.e., delivery and pickup) is provided by a vehicle in a single stop;
- the load of a vehicle never exceeds its capacity Q anywhere in the route.

Multi-depot VRPSPD has not received much attention in literature. Salhi and Nagy (1999) and Nagy and Salhi (2005) have applied the heuristic designed for single-

depot VRPSPD to multi-depot VRPSPD. Min et al. (1992) provide a heuristic for multi-depot vehicle routing with backhauls and mixed load.

The saving based algorithm of Clarke and Wright (1964) has been modified by a number of authors to solve MDVRP (Tillman (1969), Tillman and Hering (1971), Tillman and Cain (1972), Golden et al. (1977), Wren and Holliday (1972)). Despite the successful application of saving based algorithms to MDVRP, no attempts have been made so far to apply them to multi-depot VRPSPD. This chapter describes the saving based algorithms that fill the above gap in the literature. We present four algorithms for the problem. Since multi-depot VRPBM is a special case of multi-depot VRPSPD, we extend the algorithms to multi-depot VRPBM. The algorithms are presented in the next four sections.

7.2 Partition based algorithm (PBA)

This algorithm is based on the idea of borderline customers as used by Salhi and Sari (1997) for the multi-depot vehicle fleet mix problem. Salhi and Nagy (1999) and Nagy and Salhi (2005) also used the idea of borderline customers to extend their heuristics designed for the single depot VRPSPD to multi-depot VRPSPD. In PBA, customers are divided into border and non-borderline customers. Non-borderline customers are first assigned to their nearest depot and the problem is then solved as a single depot VRPSPD for each depot. Borderline customers are then inserted into the vehicle routes. A borderline customer is defined as a customer who is situated approximately halfway between two depots. Consider customer i with his nearest depot p and his second nearest depot q. Let c_{ip} be the distance between customer i and depot p

and c_{iq} be the distance between customer i and depot q, thus, $c_{ip} < c_{iq}$. Customer i is called a borderline customer if $c_{ip}/c_{iq} \geq \rho$, where ρ is a parameter between 0.5 and 1. We use the saving algorithm described in chapter 6 for solving a single depot VRPSPD. The complete algorithm for multi-depot VRPSPD is described below.

1. Divide the customers into borderline and non-borderline customers;

2. Assign the non-borderline customers to their nearest depot;

3. For each depot, solve the single depot VRPSPD problem consisting of non-borderline customers assigned to the depot.

4. Insert the borderline customers into one of the vehicle routes obtained in step 3. Borderline customers are inserted in a route for which the increase in the route length is minimum (Borderline customers are inserted in to each vehicle route obtained in step 3 to check the route for which the increase in the route is minimum).

In the last step of the algorithm, a customer is inserted in one of the routes obtained in step 3. We use cumulative net-pickup approach for checking the feasibility of a route while inserting a customer into the route. We use lemmas 4.9 to 4.14 from chapter 4 to check the feasibility of an insertion operation.

7.3 Nearest depot algorithm

This algorithm is similar to the partition-based algorithm described in the previous section. The algorithm first assigns each customer to his nearest depot. Then a single depot VRPSPD is solved for each depot using the saving algorithm described in chapter 6. The nearest depot algorithm becomes a special case of the partition based algorithm when we divide the customers into borderline and non-borderline customers using $\rho = 1$,. Setting $\rho = 1$, makes all customers non-borderline customers. Thus, all customers are assigned to their nearest depot.

7.4 Saving algorithm

We modify the saving algorithm of Clarke and Wright (1964) and make it applicable to multi depot VRPSPD. The saving algorithm for single depot works on the principle of the maximum saving achieved when two routes are combined and served by a single vehicle instead of two different vehicles. In the case of single depot, the saving value of combining customers b and c is calculated as

$$s_{ab} = c_{0a} + c_{0b} - c_{ab} \tag{7.1}$$

Equation (7.1) is not suitable for Multi-depot vehicle routing problems because customers b and c might be served by vehicles originating from different depots. Thus, we modify the saving value defined above to make it applicable to multi-depot VRPSPD. We use the following notation to describe the saving algorithm for multi-depot VRPSPD.

R_r route of vehicle r

n_r total number of customers served by vehicle r

D_r total demand to be delivered by vehicle r

$\sigma_r(i)$ i^{th} customer in route R_r (i.e., the customer in the i^{th} position in the route of vehicle r)

$d_r(i)$ quantity of product to be delivered to customer $\sigma_r(i)$

$p_r(i)$ quantity of product to be picked up from customer $\sigma_r(i)$

$f_r(i)$ $= p_r(i) - d_r(i)$ net-pickup of customer $\sigma_r(i)$

$F_r(i)$ cumulative net-pickup by vehicle r after visiting customer $\sigma_r(i)$

X_r maximum cumulative net-pickup encountered by vehicle r

Y_r minimum cumulative net-pickup encountered by vehicle r

The complete route of vehicle r is $R_r = \{\sigma_r(0), \sigma_r(1), \ldots, \sigma_r(n_r), \sigma_r(n_r+1)\}$. Here, customers $\sigma_r(0)$ and $\sigma_r(n_r+1)$ both represent the depot. In multi-depot case, each vehicle starts and finishes the tour at the same depot, so that $\sigma_r(0) = \sigma_r(n_r+1)$. Now consider combining routes $R_r = \{\sigma_r(0), \sigma_r(1), \ldots\ldots, \sigma_r(n_r), \sigma_r(n_r+1)\}$ and $R_s = \{\sigma_s(0), \sigma_s(1), \ldots, \sigma_s(n_s), \sigma_s(n_s+1)\}$. With respect to routes R_r and R_s, we use the following additional notation.

u the depot associated with route R_r, i.e., $\sigma_r(0) = \sigma_r(n_r+1) = u$.

v the depot associated with route R_s i.e., $\sigma_s(0) = \sigma_s(n_s+1) = v$.

a first customer to be served by vehicle r, i.e., $\sigma_r(1) = a$.

b last customer to be served by vehicle *r*, i.e., $\sigma_r(n_r) = b$.

c first customer to be served by vehicle *s*, i.e., $\sigma_s(1) = c$.

d last customer to be served by vehicle *s*, i.e., $\sigma_s(n_s) = d$.

When R_r and R_s are combined and served by single vehicle *q*, vehicle *q* can start and finish the tour either from depot *u* or from depot *v*. In addition, depending on the sequence of customers in the combined route, R_r and R_s can be combined in 8 different ways. Hence, routes R_r and R_s can be merged into R_q in the following 16 ways.

M1. $R_q = R_r \cup R_s = \{\sigma_r(0), \sigma_r(1), ..., \sigma_r(n_r), \sigma_s(1), ..., \sigma_s(n_s), \sigma_r(n_r+1)\}$ (7.2)

M2. $R_q = R_r \cup R_s = \{\sigma_s(0), \sigma_r(1), ..., \sigma_r(n_r), \sigma_s(1), ..., \sigma_s(n_s), \sigma_s(n_s+1)\}$ (7.3)

M3. $R_q = R_r \cup \overline{R}_s = \{\sigma_r(0), \sigma_r(1), ..., \sigma_r(n_r), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), \sigma_r(n_r+1)\}$ (7.4)

M4. $R_q = R_r \cup \overline{R}_s = \{\sigma_s(0), \sigma_r(1), ..., \sigma_r(n_r), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), \sigma_s(n_s+1)\}$ (7.5)

M5. $R_q = \overline{R}_r \cup R_s = \{\sigma_r(0), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \sigma_s(1), ..., \sigma_s(n_s), \sigma_r(n_r+1)\}$ (7.6)

M6. $R_q = \overline{R}_r \cup R_s = \{\sigma_s(0), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \sigma_s(1), ..., \sigma_s(n_s), \sigma_s(n_s+1)\}$ (7.7)

M7. $R_q = \overline{R}_r \cup \overline{R}_s = \{\sigma_r(0), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), \sigma_r(n_r+1)\}$ (7.8)

M8. $R_q = \overline{R}_r \cup \overline{R}_s = \{\sigma_s(0), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), \sigma_s(n_s+1)\}$ (7.9)

M9. $R_q = \overline{R}_s \cup \overline{R}_r = \{\sigma_r(0), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \sigma_r(n_r+1)\}$ (7.10)

M10. $R_q = \overline{R}_s \cup \overline{R}_r = \{\sigma_s(0), \overline{\sigma}_s(1), ..., \overline{\sigma}_s(n_s), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \sigma_s(n_s+1)\}$ (7.11)

M11 $R_q = R_s \cup \overline{R}_r = \{\sigma_r(0), \sigma_s(1), ..., \sigma_s(n_s), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \sigma_r(n_r+1)\}$ (7.12)

M12 $R_q = R_s \cup \overline{R}_r = \{\sigma_s(0), \sigma_s(1), ..., \sigma_s(n_s), \overline{\sigma}_r(1), ..., \overline{\sigma}_r(n_r), \sigma_s(n_s+1)\}$ (7.13)

M13. $R_q = \bar{R}_s \cup R_r = \{\sigma_r(0), \bar{\sigma}_s(1), ..., \bar{\sigma}_s(n_s), \sigma_r(1), ..., \sigma_r(n_r), \sigma_r(n_r+1)\}$ (7.14)

M14. $R_q = \bar{R}_s \cup R_r = \{\sigma_s(0), \bar{\sigma}_s(1), ..., \bar{\sigma}_s(n_s), \sigma_r(1), ..., \sigma_r(n_r), \sigma_s(n_s+1)\}$ (7.15)

M15. $R_q = R_s \cup R_r = \{\sigma_r(0), \sigma_s(1), ..., \sigma_s(n_s), \sigma_r(1), ..., \sigma_r(n_r), \sigma_r(n_r+1)\}$ (7.16)

M16. $R_q = R_s \cup R_r = \{\sigma_s(0), \sigma_s(1), ..., \sigma_s(n_s), \sigma_r(1), ..., \sigma_r(n_r), \sigma_s(n_s+1)\}$. (7.17)

Here route R_q obtained by method M9 is the reverse of the route obtained by method M1. We consider routes obtained both by M1 and by M9 because it is possible that one of the two routes is feasible whereas the other is not. Similarly route R_q obtained by methods M10, M10, M12, M13, M14, M15 and M16 are the reverse of the routes obtained by M2, M3, M4, M5, M6, M7 and M8, respectively.

The sequence of customers in R_q obtained by methods M1 and M2 is the same but vehicle q starts and finishes its tour at depot u in M1 and at depot v in M2, respectively. Similarly, the sequence of customers in R_q obtained by method M3, M5, M7, M9, M11, M13 and M15 is same as the sequence of customers in R_q obtained by methods M4, M6, M8, M10, M12, M14 and M16 respectively. The saving values resulting from different methods are summarized below.

M1 and M9: $S_{rs}^1 = c_{bu} + c_{vc} - c_{bc} + c_{dv} - c_{du}$ (7.18)

M2 and M10: $S_{rs}^2 = c_{ua} - c_{va} + c_{bu} + c_{vc} - c_{bc}$ (7.19)

M3 and M11: $S_{rs}^3 = c_{bu} + c_{vd} - c_{bd} + c_{cv} - c_{cu}$ (7.20)

M4 and M12: $S_{rs}^4 = c_{ua} - c_{va} + c_{bu} + c_{vd} - c_{bd}$ (7.21)

M5 and M13: $$S_{rs}^{5} = c_{au} + c_{vc} - c_{ac} + c_{dv} - c_{du} \quad (7.22)$$

M6 and M14: $$S_{rs}^{6} = c_{ub} - c_{vb} + c_{au} + c_{vc} - c_{ac} \quad (7.23)$$

M7 and M15: $$S_{rs}^{7} = c_{au} + c_{vd} - c_{ad} + c_{cv} - c_{cu} \quad (7.24)$$

M8 and M16 $$S_{rs}^{8} = c_{ub} - c_{vb} + c_{au} + c_{vd} - c_{ad} \quad (7.25)$$

The maximum cumulative net-pickup, X_q and the minimum cumulative net-pickup, Y_q of route R_q obtained by above 16 methods can be derived using the lemmas from chapter 6. The expressions for the maximum cumulative net-pickup obtained by 16 different methods are presented below.

M1 and M2. $X_q = Max\{X_r, F_r(n_r) + X_s\}$ (7.26)

M3 and M4. $X_q = Max\{X_r, F_r(n_r) + F_s(n_s) - Y_s\}$ (7.27)

M5 and M6. $X_q = Max\{F_r(n_r) - Y_r,\ F_r(n_r) + X_s\}$ (7.28)

M7 and M8. $X_r = Max\{F_r(n_r) - Y_r,\ F_r(n_r) + F_s(n_s) - Y_s\}$ (7.29)

M9 and M10. $X_r = Max\{F_s(n_s) - Y_s,\ F_s(n_s) + F_r(n_r) - Y_r\}$ (7.30)

M11 and M12. $X_r = Max\{X_s,\ F_s(n_s) + F_r(n_r) - Y_r\}$ (7.31)

M13 and M14. $X_r = Max\{F_s(n_s) - Y_s,\ F_s(n_s) + X_r\}$ (7.32)

M15 and M16. $X_r = Max\{X_s,\ F_s(n_s) + X_r\}$ (7.33)

The expressions for the minimum cumulative net-pickup obtained by 16 different methods are presented below.

M1 and M2. $Y_q = Min\{Y_r, F_r(n_r) + Y_s\}$ (7.34)

M3 and M4. $Y_q = Min\{Y_r, F_r(n_r) + F_s(n_s) - X_s\}$ (7.35)

M5 and M6. $Y_q = Min\{F_r(n_r) - X_r,\ F_r(n_r) + Y_s\}$ (7.36)

M7 and M8. $Y_r = Min\{F_r(n_r) - Y_r,\ F_r(n_r) + F_s(n_s) - Y_s\}$ (7.37)

M9 and M10. $Y_r = Min\{F_s(n_s) - X_s,\ F_s(n_s) + F_r(n_r) - X_r\}$ (7.38)

M11 and M12. $Y_r = Max\{Y_s,\ F_s(n_s) + F_r(n_r) - X_r\}$ (7.39)

M13 and M14 . $Y_r = Min\{F_s(n_s) - X_s,\ F_s(n_s) + Y_r\}$ (7.40)

M15 and M16. $Y_r = Max\{Y_s,\ F_s(n_s) + Y_r\}$ (7.41)

The cumulative net-pickup of the last customer in R_q obtained by any of the above methods is calculated as.

$F_q(n_q) = F_r(n_r) + F_s(n_s).$ (7.42)

The feasibility of route R_q obtained by any of the above combinations is checked using the following conditions.

M1 and M2. $D_r + D_s + Max\{X_r, F_r(n_r) + X_s\} \le Q$ (7.43)

M3 and M4. $D_r + D_s + Max\{X_r, F_r(n_r) + F_s(n_s) - Y_s\} \le Q$ (7.44)

M5 and M6. $D_r + D_s + Max\{F_r(n_r) - Y_r,\ F_r(n_r) + X_s\} \le Q$ (7.45)

M7 and M8. $D_r + D_s + Max\{F_r(n_r) - Y_r,\ F_r(n_r) + F_s(n_s) - Y_s\} \le Q$ (7.46)

M9 and M10. $D_r + D_s + Max\{F_s(n_s) - Y_s,\ F_s(n_s) + F_r(n_r) - Y_r\} \le Q$ (7.47)

M11 and M12. $D_r + D_s + Max\{X_s,\ F_s(n_s) + F_r(n_r) - Y_r\} \le Q$ (7.48)

M13 and M14. $D_r + D_s + Max\{F_s(n_s) - Y_s,\ F_s(n_s) + X_r\} \le Q$ (7.49)

M15 and M16. $D_r + D_s + Max\{X_s,\ F_s(n_s) + X_r\} \le Q$ (7.50)

We use the expressions (7.18) to (7.25) to calculate the saving value, expressions (7.26) to (7.41) to calculate X_q and Y_q and expressions (7.43) to (7.49) to check the feasibility of the combined route. The saving algorithm for multi-depot VRPSPD works as follows

Step 0: (Initialization) Assign each customer to a separate vehicle to create routes R_m= $\{0,\ \sigma_m(1)\ ,\ 0\}$; m = 1,2,…,n. Make all m vehicles as active vehicles.

Step 1. Consider each combination of active vehicles r and s, $r < s$ and calculate the saving as follows:

$$S_{rs} = \max \begin{cases} \max(S_{rs}^1,\ S_{rs}^2, 0) & \text{if equation (7.43) is satisfied} \\ \max(S_{rs}^3,\ S_{rs}^4, 0) & \text{if equation (7.44) is satisfied} \\ \max(S_{rs}^5,\ S_{rs}^6, 0) & \text{if equation (7.45) is satisfied} \\ \max(S_{rs}^7,\ S_{rs}^8, 0) & \text{if equation (7.46) is satisfied} \\ \max(S_{rs}^9,\ S_{rs}^{10}, 0) & \text{if equation (7.47) is satisfied} \\ \max(S_{rs}^{11},\ S_{rs}^{12}, 0) & \text{if equation (7.48) is satisfied} \\ \max(S_{rs}^{13},\ S_{rs}^{14}, 0) & \text{if equation (7.49) is satisfied} \\ \max(S_{rs}^{15},\ S_{rs}^{15}, 0) & \text{if equation (7.50) is satisfied} \\ 0 & \text{otherwise} \end{cases} \quad (7.51)$$

Note that $S_{rs} = 0$ when merging of routes R_r and R_s is either not feasible or not beneficial because doing so does not decrease the total tour length for the problem. Therefore, in this case, we make the merging of route R_r with R_s or merging of route R_r with R_s non- admissible.

Step 2: Suppose $S_{pq} = \max\ (S_{rs})\ \forall r, s$. Combine routes R_p with R_q and assign vehicle q to serve the customers of the combined route. Label vehicle p as an inactive

vehicle, i.e., $R_p = \{\phi\}$. Update X_q, Y_q and $F_q(n_q)$. There are 16 different methods for combining routes R_p with R_q. We choose the method that gives us the same value for S_{pq} as given by equation (7.51) and then update X_q, Y_q and $F_q(n_q)$ using the corresponding formulas.

Step 3: If all mergers are non admissible then stop; else go to step 1.

7.5 Tillman's saving algorithm (TSA)

Tillman (1969) was the first to adopt the saving algorithm for MDVRP. He realized that the saving value calculated using expression (7.1) does not reflect the true saving for MDVRP. He modified the distance between the customers and the depots. Let v be the nearest depot for customer a. Then the distance between customer a and some depot u is modified as

$$\bar{c}_{au} = c_{av} - (c_{au} - c_{av}) \tag{7.52}$$

Using the above expression, Tillman (1969) used the following equation to calculate the saving value between customers a and b with respect to depot u.

$$S^u_{ab} = \bar{c}_{au} + \bar{c}_{bu} - c_{ab} \tag{7.53}$$

Tillman's saving algorithm proposed for multi-depot VRPSPD is similar to single depot VRPSPD described in chapter 6 with few modifications. In case of multi-depot VRPSPD, saving for each combination of customers is calculated for each depot $u \in M$. When two routes are combined and assigned to vehicle q, route R_q starts and finishes the tour from the same depot for which S^u_{ab} is maximum.

7.6 Numerical Analysis

This section presents the numerical results for the saving based algorithms proposed in this chapter and compares them with the existing heuristics for the problem. The algorithms presented in this chapter are mainly designed for multi-depot VRPSPD. We implement the algorithms to multi-depot VRPBM since multi-depot VRPBM is a special case of multi-depot VRPSPD.

7.6.1 Numerical Analysis for multi-depot VRPSPD

The saving based algorithms proposed in this chapter were coded in C++ and implemented on Intel Xeon with 2.4 GHz computer system. The four algorithms are defined as .

PBA - Partition based algorithm

NBA – Nearest depot based algorithm

SA - Saving algorithm based on Clarke and Wright (1964) saving algorithm

TSA - Tillman (1969) algorithm based on saving algorithm for multi-depot VRP

We compare the above new heuristics with the following existing heuristics for which there are reported results.

LC-INS the heuristic algorithm of Salhi and Nagy (1999).

ALT the heuristic algorithm of Nagy and Salhi (2005).

Benchmark problem

The numerical experiment is performed using two sets of problem instances. The first data set for multi-depot VRPSPD is provided by Salhi and Nagy (1999). They

generated instances from 11 benchmark problems of Gillett and Johnson (1976). Let x_a and y_a denote the co-ordinates of customer a and let d_a^{ori} denote the demand for customer a in the original problem. The distance matrix in the new problem is generated using the original co-ordinates. However, d_a^{ori} is split into delivery demand d_a and pickup demand p_a as follows: $d_a = r_a \times d_a^{ori}$ and $p_a = (1 - r_a) \times d_a^{ori}$ where $r_a = \min(x_a / y_a, y_a / x_a)$. In this way, a set of 11 instances referred as set A is generated. Another set of 11 instances referred as set B is generated by exchanging the pickup and delivery demands in problem instances of set A.

We generate the second data-set from the multi-depot VRP data set provided by Chao et al. (1993). This data set contains customers whose x or y co-ordinate is equal to 0. Therefore, the method used by Salhi and Nagy (1999) for splitting the original demand into pickup and delivery demand is not applicable. We use a random number to split the original demand. Consider customer a with original demand d_a^{ori}. For each customer a, generate a uniform random number r_a in the range of [0, 1]. Then the new delivery demand of customer a is calculated as $d_a = r \times d_a^{ori}$ and the new pickup demand, p_a, is calculated as $p_a = (1 - r) \times d_a^{ori}$. In this way a set of 12 instances referred as set X is generated. Another set of 12 instances referred as set Y is generated from set X by exchanging the pickup and delivery demand.

Altogether we use 46 problem instances for VRPSPD: 22 instances from Salhi and Nagy (1999) and 24 instances created from the 12 benchmark problem instances of

Chao et al. (1993). Instances of Salhi and Nagy (1999) are referred to GJX/Y and the instances of Chao et al. (1993) are referred as CGWX/Y.

Computational results and performance analysis of the algorithm

The detailed results obtained by different algorithms for multi-depot VRPSPD problem instances are presented in Table 7.1 for Salhi and Nagy (1999) problem instances and in Table 7.2 for Chao et al. (1993) problem instances. The average solutions for different algorithms for the multi-depot VRPSPD problem instances are presented in Table 7.3 and Table 7.4.

The results reported in Table 7.3 show that the performance of the four saving based algorithms developed in this chapter is better than the performance of existing algorithms. For Salhi and Nagy (1999) problem instances, SA gives the lowest average solution cost followed by NBA, TSA and PBA, respectively. The trend of relative performance of the four algorithms for Chao et al. (1993) problem instances is completely different than the trend for Salhi and Nagy (1999) problem instances. It seems that the change is due to the special characteristics of the data set by Chao et al. (1993). This data set is based upon five concentric squares in each block. A depot is situated at the center of each block and all customers are situated on either the X axis or the Y axis or in the middle of X and Y axis (see Chao et al. (1993) for the figure). It is intuitive that in this case, the partition based algorithms will perform better than the non-partition algorithms. Note that NBA and PBA are partition based algorithms while SA and TSA are construction based algorithm. Thus, the better performance of NBA and PBA over

SA and TSA seen in Table 7.4 for Chao et al. (1993) problem instances supports our intuition.

The comparison of CPU times shows that the CPU time of partition based algorithms PBA and NBA is much less than the CPU time of the construction based algorithms SA and TSA. The less CPU time of the partition based algorithms is due to the partitioning of the problem. In a partition based algorithm, a number of smaller problems with subsets of customers are solved. Solving a small size problem with fewer number of customers reduces the complexity of the problem which in turn reduces the CPU time.

There is a considerable difference in CPU time for SA and TSA. As seen in Tables 7.3 and 7.4, the CPU time taken by SA is much less than the CPU time taken by TSA. In TSA, the saving value is calculated for each pair of customers in combination with each depot. Thus, the size of the saving value matrix is $m \times n \times n$ compared to the size of the saving value matrix of $n \times n$ for SA. The increase in the size of the saving value matrix for TSA increases the CPU time in step 1 of the algorithm where we initialize and store the saving values in non-decreasing order.

Table 7.1: Total tour length obtained by different heuristics and corresponding CPU time for 22 multi-depot VRPSPD instances of Salhi and Nagy (1999)

No	Inst.	LC-INS	PBA		NBA		SA		TSA	
		Cost	Cost	Time	Cost	Time	Cost	Time	Cost	Time
1	GJ1X	674	600.33	0.01	544.21	0.01	541.32	0.08	545.84	0.25

2	GJ2X	569	482.2	0.01	481.88	0.01	491.96	0.08	488.85	0.25
3	GJ3X	734	650.06	0.02	632.41	0.02	637.73	0.26	652.13	1.04
4	GJ4X	1193	1082.78	0.1	931.28	0.11	931.83	0.61	972.77	1.11
5	GJ5X	909	760.33	0.11	777.77	0.11	750.58	0.62	775.11	1.15
6	GJ6X	954	852.59	0.06	860.24	0.07	885.58	0.6	935.97	1.55
7	GJ7X	973	989.16	0.04	833.56	0.05	878.39	0.6	903.24	1.94
8	GJ8X	5326	4458.41	1.25	3797.5	1.53	3750.63	9.56	3793.39	17.67
9	GJ9X	4426	4103.72	0.74	3459.09	0.81	3398.46	9.47	3554.52	24.77
10	GJ10X	4446	4142.56	0.45	3361.87	0.54	3310.98	9.5	3440.19	31.3
11	GJ11X	4323	3545.45	0.38	3264.38	0.42	3262.8	9.42	3460.16	39.2
12	GJ1Y	614	582.36	0.01	544.21	0.01	541.32	0.08	545.84	0.25
13	GJ2Y	519	482.2	0.01	481.88	0.01	491.96	0.08	488.85	0.26
14	GJ3Y	737	651.45	0.02	632.41	0.02	637.73	0.26	652.13	1.03
15	GJ4Y	1162	1075.14	0.1	931.28	0.11	931.83	0.63	972.77	1.14
16	GJ5Y	912	760.33	0.11	777.77	0.11	750.58	0.63	775.11	1.15
17	GJ6Y	1003	852.59	0.06	860.24	0.07	885.58	0.61	935.97	1.68
18	GJ7Y	973	934.98	0.05	833.56	0.05	878.39	0.61	903.24	1.98
19	GJ8Y	4804	4470.82	1.23	3797.5	1.53	3750.63	9.6	3793.39	17.52
20	GJ9Y	4501	4205.61	0.76	3459.09	0.81	3398.46	9.54	3554.52	24.5
21	GJ10Y	4183	3963.82	0.44	3361.87	0.54	3310.98	9.6	3440.19	32.07
22	GJ11Y	4357	3532.57	0.38	3264.38	0.42	3262.8	9.57	3460.16	39.69
Average		2195.09	1965.7	0.29	1722.2	0.33	1712.75	3.73	1774.74	10.98

Table 7.2: Total tour length obtained by different heuristics and corresponding CPU time for 24 multi-depot VRPSPD instances of second data set

No	Inst.	PBA		NBA		SA		TSA	
		Cost	Time	Cost	Time	Cost	Time	Cost	Time
1	CGW1X	1366.84	0.06	1360.83	0.06	1365.01	0.32	1317.57	0.58
2	CGW2X	1414.41	0.06	1365.69	0.06	1417.72	0.32	1547.56	0.59
3	CGW3X	1436.40	0.06	1365.69	0.06	1538.65	0.32	1564.50	0.58
4	CGW4X	2781.21	0.15	2715.81	0.16	2652.55	2.51	2677.75	8.26
5	CGW5X	2754.95	0.15	2731.37	0.16	2808.47	2.52	3219.29	8.49
6	CGW6X	2796.94	0.15	2731.37	0.16	3080.37	2.54	3214.07	8.31
7	CGW7X	4021.79	0.33	4076.64	0.34	3976.49	8.51	4037.29	41.86
8	CGW8X	4083.33	0.33	4097.06	0.34	4149.26	8.46	4897.74	41.98
9	CGW9X	4162.63	0.34	4097.06	0.34	4585.52	8.55	4890.51	41.48
10	CGW10X	6106.75	0.88	6135.46	0.83	5956.60	29.51	6677.16	210.06
11	CGW11X	6085.17	0.87	6145.58	0.83	6168.58	29.30	7450.62	206.56
12	CGW12X	6211.16	0.87	6145.58	0.83	6867.90	29.49	7452.03	204.06
13	CGW1Y	1366.84	0.05	1360.83	0.06	1365.01	0.32	1317.57	0.59
14	CGW2Y	1414.41	0.05	1365.69	0.06	1417.72	0.32	1547.56	0.58
15	CGW3Y	1436.40	0.06	1365.69	0.06	1538.65	0.32	1564.50	0.59
16	CGW4Y	2772.76	0.15	2715.81	0.16	2652.55	2.52	2677.75	8.23
17	CGW5Y	2754.57	0.15	2731.37	0.16	2808.47	2.52	3219.29	8.58
18	CGW6Y	2796.56	0.15	2731.37	0.16	3080.37	2.53	3214.07	8.18
19	CGW7Y	4180.73	0.32	4076.64	0.34	3976.49	8.56	4037.29	41.14

20	CGW8Y	4082.95	0.33	4097.06	0.34	4149.26	8.46	4897.74	41.45
21	CGW9Y	4162.25	0.33	4097.06	0.34	4585.52	8.50	4890.51	41.04
22	CGW10Y	6360.75	0.87	6135.46	0.84	5956.60	29.08	6677.16	206.96
23	CGW11Y	6084.79	0.87	6145.58	0.83	6168.58	29.04	7450.62	218.43
24	CGW12Y	6210.78	0.88	6145.58	0.84	6867.90	29.54	7452.03	210.37
Average		3618.56	0.35	3580.68	0.35	3713.93	10.17	4078.84	64.96

Table 7.3: Average tour length obtained by different heuristics and corresponding CPU time for 22 multi-depot VRPSPD instances of Salhi and Nagy (1999)

	LC-INS	ALT	PBA	NBA	SA	TSA
Average cost	2195.09	1993	1962.7	1722.2	1712.75	1774.74
Average time (sec.)	27.15	12.4	0.29	0.3	3.73	10.98

Table 7.4: Average tour length obtained by different heuristics and corresponding CPU time over 24 multi-depot VRPSPD instances of second data set

	PBA	NBA	SA	TSA
Average cost	3618.56	3580.68	3713.93	4078.84
Average time (sec.)	0.35	0.35	10.17	64.96

7.6.2 Numerical Analysis for multi-depot VRPBM.

Our proposed algorithms are mainly designed for multi-depot VRPSPD. In VRPSPD, each customer is a linehaul as well as backhaul customer while in VRPBM a customer is either a linehaul customer or a backhaul customer. Since multi-depot

VRPBM is a special case of multi-depot VRPSPD, the algorithms can also be applied to multi-depot VRPBM. Thus, we tested PBA, NBA, SA and TA on multi-depot VRPBM problem instances. We compare the four algorithms with the reported results for the existing heuristics listed below.

LC-INS the heuristic algorithm of Salhi and Nagy (1999).

ALT the heuristic algorithm of Nagy and Salhi (2005).

Benchmark problem

Similar to multi-depot VRPSPD, we use two sets of problem instances in the numerical experiment on multi-depot VRPBM. The first data set for multi-depot VRPBM is provided by Salhi and Nagy (1999). They generated three problem instances referred as a T, Q and H from each VRP instance of 14 benchmark problems (Christofides et al. (1979)). In set T, every tenth customers is declared as a backhaul customer and is assigned pickup demand equal to the original demand. Hence, in total 10% customers are declared backhaul. Similarly, for sets Q and H, every fourth and every second customer is declared as a backhaul customer thus making 25% and 50% of customers backhaul customers, respectively. In our comparison, we refer to the above multi-depot VRPBM instances of Salhi and Nagy (1999) by GJT, GJQ and GJH, respectively.

We generate the second data-set for multi-depot VRPBM from the 12 multi-depot VRP problem instances provided by Chao et al. (1993). Similar to the first data set, we generated three problem instances referred as T, Q and H. In set T, every tenth customer is declared as a backhaul customer and is assigned the pickup demand equal to the original demand. In set Q and H, 25% and 50% of customers are declared backhaul

customers, respectively. We refer to the above multi-depot VRPBM instances by CGWT, CGWQ and CGWH, respectively.

Computational results and performance analysis of the algorithm

The detailed results obtained by different algorithms for multi-depot VRPBM problem instances are presented in Table 7.5 for Salhi and Nagy (1999) problem instances and in Table 7.6 for Chao et al. (1993) problem instances. The average solutions for different algorithms for multi-depot VRPSPD problem instances are listed in Table 7.7 and Table 7.8.

The average results reported in Table 7.7 and Table 7.8 show that the algorithms designed in this chapter work equally well for multi-depot VRPBM problem instances. The trend of solution quality for different algorithm is similar to the trend for VRPSPD problem instances. For Salhi and Nagy (1999) problem instances, SA gives the lowest average solution cost followed by NBA, PBA and TSA. For Chao et al. (1993) problem instances, NBA gives the lowest average solution cost followed by PBA, SA and TSA.

Table 7.5: Total tour length obtained by different heuristics and corresponding CPU time for 33 multi-depot VRPBM instances of Salhi and Nagy (1999)

No	Inst.	LC-INS	PBA		NBA		SA		TSA	
		Cost	Cost	Time	Cost	Time	Cost	Time	Cost	Time
1	GJ1H	619	601.074	0.01	615.12	0.01	601.236	0.07	656.519	0.23
2	GJ2H	562	529.005	0.01	515.161	0.01	509.183	0.08	521.345	0.24

3	GJ3H	662	684.142	0.02	661.272	0.02	668.242	0.25	692.859	0.99
4	GJ4H	1055	1098.94	0.09	1041.11	0.1	1041.11	0.59	1094.48	1.06
5	GJ5H	853	892.308	0.1	832.66	0.11	818.659	0.61	812.173	1.12
6	GJ6H	1034	923.405	0.06	912.799	0.06	922.948	0.59	970.504	1.45
7	GJ7H	932	1032.37	0.04	920.443	0.05	920.415	0.57	949.053	1.85
8	GJ8H	5188	5138.23	1.22	4443.77	1.49	4291.12	9.26	4377.19	17.38
9	GJ9H	4087	4466.71	0.73	3984.86	0.79	3933.79	9.21	3977.44	23.99
10	GJ10H	4166	4780.29	0.44	3819.9	0.53	3704.93	9.33	3871.92	31.28
11	GJ11H	3933	4089.48	0.38	3695.99	0.42	3708.16	9.23	3885.18	37.98
12	GJ1Q	666	672.57	0.01	582.399	0.01	564.242	0.07	583.206	0.23
13	GJ2Q	550	518.98	0.01	493.433	0.01	490.091	0.08	507.191	0.25
14	GJ3Q	670	674.241	0.02	648.978	0.02	651.53	0.27	671.949	1.05
15	GJ4Q	1168	971.783	0.09	933.899	0.11	938.918	0.6	957.532	1.1
16	GJ5Q	828	816.187	0.1	795.739	0.11	757.979	0.62	752.698	1.15
17	GJ6Q	988	912.587	0.06	866.8	0.07	855.515	0.59	959.412	1.49
18	GJ7Q	940	909.656	0.05	879.099	0.05	883.844	0.58	929.145	2
19	GJ8Q	4877	4536.12	1.23	4063.33	1.51	4071.01	9.33	4091.4	18.01
20	GJ9Q	4087	4336.96	0.74	3704.63	0.8	3638.17	9.29	3775.45	24.41
21	GJ10Q	3931	4065.83	0.44	3644.01	0.53	3546.96	9.6	3631.3	30.94
22	GJ11Q	3840	3856.29	0.39	3546.95	0.42	3527.19	9.25	3665.43	38.26
23	GJ1T	614	551.288	0.01	537.681	0.01	529.403	0.08	548.892	0.25

24	GJ2T	497	481.264	0.01	481.879	0.01	496.409	0.08	488.852	0.26
25	GJ3T	662	635.798	0.02	631.861	0.02	629.917	0.26	646.699	1.05
26	GJ4T	1055	903.534	0.1	877.313	0.11	867.608	0.62	867.075	1.17
27	GJ5T	794	751.43	0.11	749.325	0.11	741.905	0.63	763.204	1.16
28	GJ6T	914	821.696	0.06	803.237	0.07	817.516	0.61	865.774	1.6
29	GJ7T	992	833.824	0.04	803.564	0.05	824.304	0.6	866.354	2.11
30	GJ8T	4647	4354.74	1.26	3758.58	1.53	3653.97	9.75	3738.27	17.71
31	GJ9T	4087	3674.28	0.74	3307.39	0.81	3206.24	9.61	3505.4	24.81
32	GJ10T	4002	4121.3	0.44	3373.54	0.54	3258.92	9.61	3316.39	31.56
33	GJ11T	3794	3453.45	0.39	3250.37	0.43	3263.79	9.77	3369.08	38.54
Average		2051.33	2033.02	0.29	1823.55	0.33	1798.04	3.69	1857.86	10.81

Table 7.6: Total tour length obtained by different heuristics and corresponding CPU time for 36 multi-depot VRPBM instances of second data set

No	Inst.	PBA		NBA		SA		TSA	
		Cost	Time	Cost	Time	Cost	Time	Cost	Time
1	CGW1H	1399.2	0.06	1365.69	0.06	1421.45	0.32	1534.59	0.58
2	CGW2H	1414.41	0.05	1365.69	0.06	1417.72	0.32	1553.65	0.59
3	CGW3H	1436.4	0.05	1365.69	0.06	1558.65	0.32	1564.5	0.58
4	CGW4H	2717.85	0.15	2731.37	0.16	2755.62	2.53	3250.25	8.21
5	CGW5H	2754.57	0.15	2731.37	0.16	2834.98	2.51	3233.43	8.29

6	CGW6H	2796.56	0.15	2731.37	0.16	3086.22	2.53	3242.36	8.35
7	CGW7H	4029.18	0.32	4097.06	0.34	4108.49	8.51	4960.5	40.88
8	CGW8H	4082.95	0.33	4097.06	0.34	4185.62	8.55	4911.88	40.53
9	CGW9H	4162.25	0.33	4097.06	0.34	4608.95	8.43	4918.79	40.4
10	CGW10H	6005.2	0.87	6145.58	0.83	6155.24	29	7520.22	207.77
11	CGW11H	6084.79	0.87	6145.58	0.83	6204.21	28.9	7476.36	210.95
12	CGW12H	6210.78	0.88	6145.58	0.83	6879.62	29.37	7488.6	210.83
13	CGW1Q	1373.35	0.06	1365.69	0.06	1363.23	0.33	1488.06	0.58
14	CGW2Q	1414.41	0.05	1365.69	0.06	1417.72	0.32	1547.56	0.58
15	CGW3Q	1436.4	0.05	1365.69	0.06	1558.65	0.32	1564.5	0.58
16	CGW4Q	2706.62	0.15	2731.37	0.16	2683.13	2.57	3005.8	8.35
17	CGW5Q	2754.57	0.15	2731.37	0.16	2808.47	2.5	3233.43	8.4
18	CGW6Q	2796.56	0.15	2731.37	0.16	3080.37	2.5	3228.22	8.26
19	CGW7Q	4035	0.33	4097.06	0.34	4009.49	8.49	4525.3	41.27
20	CGW8Q	4082.95	0.32	4097.06	0.34	4149.26	8.43	4911.88	41.69
21	CGW9Q	4162.25	0.34	4097.06	0.34	4585.52	8.43	4904.65	42.83
22	CGW10Q	6011.02	0.88	6145.58	0.83	6005.17	28.85	6980.71	206.95
23	CGW11Q	6084.79	0.86	6145.58	0.84	6180.29	29.08	7476.36	219.7
24	CGW12Q	6210.78	0.87	6145.58	0.83	6867.9	29.49	7460.31	210.27
25	CGW1T	1352.7	0.06	1349.12	0.06	1350.87	0.35	1313.36	0.59
26	CGW2T	1414.41	0.05	1365.69	0.06	1417.72	0.32	1547.56	0.58

27	CGW3T	1436.4	0.05	1365.69	0.06	1538.65	0.32	1564.5	0.58
28	CGW4T	2660.64	0.15	2698.23	0.16	2650.77	2.53	2680.81	8.48
29	CGW5T	2754.57	0.15	2731.37	0.16	2808.47	2.5	3219.29	8.38
30	CGW6T	2796.56	0.15	2731.37	0.16	3080.37	2.54	3214.07	8.45
31	CGW7T	3955.4	0.33	4047.35	0.34	3962.35	8.48	4235.71	41.31
32	CGW8T	4082.95	0.33	4097.06	0.34	4149.26	8.39	4906.02	41.59
33	CGW9T	4162.25	0.33	4097.06	0.34	4585.52	8.46	4890.51	41.46
34	CGW10T	5929.23	0.86	6071.03	0.83	5936.6	29.31	6184.1	208.08
35	CGW11T	6084.79	0.87	6145.58	0.83	6174.34	29.42	7462.22	206.89
36	CGW12T	6210.78	0.88	6145.58	0.83	6867.9	29.14	7452.03	203.53
Average		3584.43	0.35	3580.09	0.35	3734.69	10.12	4184.78	64.93

Table 7.7: Total tour length obtained by different heuristics and corresponding CPU time over 33 multi-depot VRPBM instances of Salhi and Nagy (1999)

% of backhaul customers		LC-INS	ALT	PBA	NBA	SA	TSA
50 %	Average cost	2099.18	1993	2203.27	1949.37	1919.98	1982.61
	Average time (sec.)	13.4	14.80	0.28	0.33	3.62	10.69
25 %	Average cost	2049.55	2007	2049.55	2024.65	1832.66	1811.40
	Average time (sec.)	8.7	8.6	0.29	0.33	3.66	10.81
10 %	Average cost	2005.27	1996	2005.27	1817.15	1688.61	1662.73
	Average time (sec.)	10.9	8.1	0.29	0.34	3.78	10.93

Table 7.8: Average tour length obtained by different heuristics and corresponding CPU time over 36 VRPBM instances of second data set

% of backhaul customers		PBA	NBA	SA	TSA
50 %	Average cost	3591.18	3584.93	3768.06	4304.59
	Average time (sec.)	0.38	0.37	11.00	70.67
25 %	Average cost	3584.93	3584.93	3725.77	4193.90
	Average time (sec.)	0.38	0.37	11.00	71.72
10 %	Average cost	3584.93	3570.43	3710.24	4055.85
	Average time (sec.)	0.38	0.37	11.04	69.94

7.7 Conclusions

The problem of multi-depot VRPSPD is important given the need for integrating forward and reverse flows of material. The saving heuristics of Clarke and Wright (1964) is a the widely used heuristic for solving VRP variants. It is surprising that the saving based algorithms have not yet been developed for multi-depot VRPSPD. In this chapter, we have designed four saving based algorithms for multi-depot VRPSPD.

Computational experiment with benchmark problem instances has shown that the saving based algorithms presented in this chapter perform better than the existing insertion based heuristics for multi-depot VRPSPD. The solutions produced by the four algorithms show that the saving based algorithms have an edge over insertion-based algorithms in multi-depot VRPSPD.

Since multi-depot VRPBM is a special case of multi-depot VRPSPD, we tested the four algorithms on multi-depot VRPBM as well. The four algorithms designed in this chapter works equally well for the multi-depot VRPBM problem instances.

CHAPTER 8

Conclusions

In this thesis, we have focused on the vehicle routing problems with pickup and delivery (VRPPD). This problem is becoming an important problem in reverse logistics given the recent environmental regulations and the increased incentives for returning and reusing products. VRPPD arises in many real life situations whenever the pickup demand and the delivery demand is to be satisfied by the same vehicle. We considered three variants of VRPPD, namely, the vehicle routing problem with backhauls (VRPB), the vehicle routing problem with backhauls and mixed-load (VRPBM) and the vehicle routing problem with simultaneous pickup and delivery (VRPSPD).

The inherent complexity of VRPPD and its variants makes them $\mathcal{NP}$-hard problems. Therefore, the natural choice of methodology for solving such problems is heuristics and/or metaheuristics. In the first part of this thesis, we provided metaheuristic solution procedures for the three variants of VRPPD. In the second part, we provided heuristic solution procedures for solving the single and multi-depot version of VRPSPD.

We use ant colony system (ACS) which is a metaheuristic approach inspired by the foraging behavior of real ants. In the literature on vehicle routing problems (VRP), one finds that Tabu search (TS) has been the widely used metaheuristic approach for solving such problems. The motivation to use ant-colony system (ACS) for VRPPD came from the observation made by Gendreau et al. (2001). The solution quality reported by the existing ant colony systems for VRP was not as good as Tabu search but Gendreau et

al. (2001) considered the performance of ACS encouraging. One of the aims of this thesis was to investigate the potential of ant colony system to solve vehicle routing problems. We have developed ACS algorithms for VRPB, VRPBM and VRPSPD to highlight the potential.

We designed Multi-ant colony system (MACS) for VRPB. Since the number of vehicles is fixed, VRPB can be decomposed into an assignment sub-problem and a sequencing sub-problem. MACS therefore uses two types of ants to solve the two sub-problems. The performance of MACS was tested using benchmark problem instances available in literature. MACS is competitive with existing tabu search heuristics for solving VRPB. An interesting feature of MACS is the control of the solution quality and the CPU time, which can be done by varying the number of ants.

An inherent part of any metaheuristic is the local search. We have improved upon the existing ant colony systems by using better local search schemes and by adding features such as construction rule and the trail updating criteria. Two new multi-route local search schemes -- the customer insertion/interchange multi-route scheme and the sub-path exchange multi-route scheme -- are designed in this thesis. Although these multi-route local search schemes are specially designed for ant colony system, they can also be used in other metaheuristic methods with some modification.

Insuring the feasibility of a route is a major issue in VRPSPD given the fluctuating load on the vehicle. We have proposed the cumulative net-pickup approach for checking the feasibility of a route during local search. An important property of the cumulative net-pickup approach is that it can check the feasibility of the altered route in

constant time. Using an appropriate data structure, one can check the effect of a single move of local search on the feasibility without examining the entire route. We attribute the efficiency of ACS (in terms of the reduced CPU time) to the cumulative net-pickup approach. We have provided several lemmas associated with the cumulative net-pickup approach in chapter 4.

We used benchmark problem instances available in literature to evaluate the proposed ACS algorithms. The computational experiment has shown that on the whole, ACS gives better results than the other metaheuristics. The proposed ACS algorithms have produced 5 new best solutions for VRPB problem instances, 6 new best solutions for VRPBM problem instances and 31 new best solutions for VRPSPD problem instances.

The second part of the thesis was devoted to the development of heuristic solution procedures for VRPSPD. The saving heuristic of Clarke and Wright (1964) is widely used in literature for solving capacitated vehicle routing problem (CVRP). However, no such heuristic exists for VRPSPD or VRPBM. We have designed saving based heuristics for the single depot version of VRPSPD in chapter 6 and for the multi-depot version of VRPSPD in chapter 7. Since VRPBM is a special case of VRPSPD, we have applied the same heuristic to VRPBM without any modification.

The saving heuristics designed in chapters 6 and 7 involve creating a new route by merging two existing routes. We have applied the cumulative net-pickup approach for checking the feasibility of such a route. We have provided several lemmas associated with the cumulative net-pickup approach in chapters 6 and 7.

We used benchmark problem instances available in literature to evaluate the proposed saving based heuristics. We have also created new problem instances for the multi-depot version of VRPSPD and VRPBM. The computational experiment has shown that on the whole, the proposed saving heuristics perform better than the existing insertion based heuristics.

Scope for Future Work

This thesis has illustrated the potential of ant colony system for solving the three version of VRPPD. It will be interesting to examine the performance of ACS in solving other variants of VRPPD. For example, ant colony system can be used to solve the vehicle routing problem with backhauls when demand can be picked up only after a certain percentage of total customer demand is delivered. ACS has potential to produce better solutions for the dynamic version of VRPB, VRPBM and VRPSPD.

Local search is an important part of ACS. Further research can be done to design better local search schemes for ACS. It seems that a hybrid version of local search can make the ant-colony approach more effective for the variants of the vehicle routing problem. The cumulative net-pickup approach designed in this thesis can also be extended to other pickup and delivery problems.

From a practitioner point of view, the heuristics and metaheuristics algorithms developed in this thesis can be embedded into commercial vehicle routing software. We tested the performance of the proposed algorithms using the hypothetical problem

instances used in literature. Studies need to be conducted to see if the proposed algorithms are equally effective in solving real life problems.

BIBLIOGRAPHY

Agarwal, Y., Mathur, K., & Salkin, H. M. (1989). A Set-Partitioning Based Exact Algorithm For The Vehicle-Routing Problem. *Networks*, 19(7), 731-749.

Altinkemer, K. & Gavish, B. (1987). Heuristics for unequal weight delivery problems with a fixed error guarantee. *Operations research letters*, 6(4), 149-158.

Altinkemer, K. & Gavish, B. (1990). Heuristics For Delivery Problems With Constant Error Guarantees. *Transportation Science*, 24(4), 294-297.

Altinkemer, K. & Gavish, B. (1991). Parallel Savings Based Heuristics For The Delivery Problem. *Operations Research*, 39(3), 456-469.

Anily, S. (1996). The vehicle-routing problem with delivery and back-haul options. *Naval Research Logistics*, 43(3), 415-434.

Ball, M. O., Golden, B. L., & Assad, A. (1983). Planning for truck fleet size in the presence of a common-carrier option. *Decision Science*, 14(1), 103-120.

Beasley, J. E. (1983). Route 1st - Cluster 2nd Methods For Vehicle-Routing. *Omega-International Journal of Management Science*, 11(4), 403-408.

Bell, W., Dalberto, L., Fisher, M. L., Greenfield, A., Jaikumar, R., Kedia, P., Mack, R., & Prutzman, P. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*, 13(6), 4-23.

Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1), 1-31.

Bianchessi, N. & Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34(2), 578-594.

Bompadre, A., Dror, M., & Orlin, J. B. (2006). Improved bounds for vehicle routing solutions. *Discrete Optimization*, 3(4), 299-316.

Brandao, J. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, 173(2), 540-555.

Bräysy, O. & Dullaert, W. (2003). A fast evolutionary metaheuristic for the vehicle routing problem with time windows. *International Journal on Artificial Intelligence Tools*, 12, 153–172.

Brown, G. G., Ellis, C. J., Graves, G. W., & Ronen, D. (1987). Real-time, wide area dispatch of Mobil tank trucks. *Interfaces*, 17(1), 107-120.

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89, 319-328.

Casco, D. O., Golden, B., & Wasil, E. A. (1988). Vehicle routing with backhauls: Models, algorithms and case studies. In B. L. Golden & A. A. Assad (Eds.), in *Vehicle Routing: Method and studies.*: 127-147. Amsterdam: Elsevier.

Cassidy, P. J. & Bennett, H. S. (1972). TRAMP--A Multi-Depot Vehicle Scheduling System. *Operational Research Quarterly (1970-1977)*, 23(2), 151-163.

Chao, I. M., Golden, B. L., & Wasil, E. (1993). A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences*, 13(3-4), 371-406.

Chen, J. F. & Wu, T. H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of The Operational Research Society*, 57(5), 579-587.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides & A. Mingozzi & P. Toth & C. Sandi (Eds.), in *Combinatorial Optimization*: 315-318. Chichester: John Wiley.

Christofides, N., Mingozzi, A., & Toth, P. (1981). Exact Algorithms For The Vehicle-Routing Problem, Based On Spanning Tree And Shortest-Path Relaxations. *Mathematical Programming*, 20(3), 255-282.

Clarke, G. & Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4), 568-581.

Cordeau, J. F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2), 105-119.

Crispim, J. & Brandao, J. (2005). Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of The Operational Research Society*, 56(11), 1296-1302.

Deif, I. & Bodin, L. (1984). Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling, in *In proceedings of the Babson Conference on Software uses in Transportation and Logistic Management*: 75-96. Babson Park (USA): Kidder A (ed.).

Dell'Amico, M., Righini, G., & Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2), 235-247.

Desrochers, M. & Verhoog, T. W. (1989). A matching based savings algorithm for the vehicle routing problem. *Technical Report Cahiers du GERAD, G-89-04.*

Dethloff, J. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum*, 23(1), 79-96.

Doerner, K., Gronalt, M., Hartl, R. F., Reimann, M., Strauss, C., & Stummer, M. (2002). Savings Ants for the vehicle routing problem, in *Applications Of Evolutionary Computing, Proceedings*, Vol. 2279: 11-20.

Dongarra, J. J. (2006). Performance of various computers using standard linear equation software. *Computer Science Technical Report, CS-89-85.*

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernatics-Part B*, 26, 29-41.

Dorigo, M., *Ant Colony Optimization*. ed. Vol. MIT Press, pp., 2004

Dueck, G. (1993). New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of computational physics(Print)*, 104(1), 86-92.

Duhamel, C., Potvin, J. Y., & Rousseau, J. M. (1997). A tabu search heuristic for the vehicle routing problem with backhauls and time windows. *Transportation Science*, 31(1), 49-59.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards*, 69, 125-130.

Fisher, M. L. & Jaikumar, R. (1981). A Generalized Assignment Heuristic For Vehicle-Routing. *Networks*, 11(2), 109-124.

Fisher, M. L., Jaikumar, R., & Lester, J. T., *A Computerized Vehicle Routing Application.* ed. Vol. Division of Research, Graduate School of Business Administration, Harvard University, pp., 1981

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P., Reis, M., Uchoa, E., & Werneck, R. F. (2006). Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3), 491-511.

Gambardella, L. M., Taillard, E., & Agazzi, G. (1999). MACS-VRPTW: A Multiple ant colony System for vehicle routing problems with time windows. In D. Corne & M. Dorigo & F. Glover (Eds.), in *New Ideas in Optimization*: 63-76. London, U.K.: McGraw-Hill.

Garey, M. R. & Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness.* ed. Vol. WH Freeman & Co. New York, NY, USA, pp., 1979

Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10), 1276-1290.

Gendreau, M., Laporte, G., & Potvin, J. Y. (2001). Metaheuristics for the capacitated VRP, in *The vehicle routing problem*: 129-154: Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

Gillet, B. & Miller, L. (1976). A heuristic algorithm for vehicle dispatches. *Operations Research*, 24, 340-349.

Gillett, B. E. & Miller, L. R. (1974). A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations Research*, 22(2), 340-349.

Gillett, B. E. & Johnson, J. G. (1976). Multi-terminal vehicle-dispatch algorithm. *Omega*, 4(6), 711-718.

Glover, F. (1986). Future paths for integer programming and artificial intelligence. *Computers & Operations Research*, 13, 533–549.

Goetschalckx, M. & Jacobsblecha, C. (1989). The Vehicle-Routing Problem With Backhauls. *European Journal Of Operational Research*, 42(1), 39-51.

Goetschalckx, M. & Jacobsblecha, C. (1993). The vehicle routing problem with backhauls: properties and solution algorithms. *Technical Report MHRC-TR-88-13.*

Golden, B., Baker, E., Alfaro, J., & Schaffer, J.: The vehicle routing problem with backhauling: two approaches. *In proceedings of the XXI Annual meeting of S.E. Times.*, Martle Beach, Hammesfahr R (ed.), 90-92.1985

Golden, B. L., Magnanti, T. L., & Nguyen, H. Q. (1977). Implementing vehicle routing algorithms. *Networks*, 7, 113-148.

Golden, B. L. & Wasil, E. A. (1987). Computerized Vehicle Routing in the Soft Drink Industry. *Operations Research*, 35(1), 6-17.

Golden, B. L., Wasil, E. A., Kelly, J. P., & Chao, I. M. (1998). Metaheuristics in vehicle routing. *Fleet Management and Logistics*, 33–56.

Haimovich, M. & Kan, A. (1985). Bounds And Heuristics For Capacitated Routing-Problems. *Mathematics of Operations Research*, 10(4), 527-542.

Halse, K., *Modeling and solving complex vehicle routing problems.* ed. Vol. IMSOR, pp., 1992

Hansen, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming. *Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy*, 70-145.

Holland, J. H. (1975). Adaptation in natural and artificial systems, University of Michigan press. *Ann Arbor, MI.*

Irnich, S. (2006). A unified modeling and solution framework for vehicle routing and local search based metaheuristics. In D. P. E. C. o. O. o. D. Networks (Ed.), Secondary.A unified modeling and solution framework for vehicle routing and local search based metaheuristics.: 1-38: RWTH Aachen university, Aachen, Germany.

Irnich, S. (2007). Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, 1-36.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1989). Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning. *Operations Research*, 37(6), 865-892.

Kindervater, G. A. P. & Savelsbergh, M. W. P. (1997). Vehicle routing: handling edge exchanges, EH Aarts, JK Lenstra (eds) Local Search in Combinatorial Optimization, Secondary.Vehicle routing: handling edge exchanges, EH Aarts, JK Lenstra (eds) Local Search in Combinatorial Optimization: 337-360: John Wiley & Sons, Chichester, UK.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598), 671.

Kolen, A. W. J., Kan, A., & Trienekens, H. (1987). Vehicle Routing with Time Windows. *Operations Research*, 35(2), 266-273.

Laporte, G., Nobert, Y., & Arpin, D., *Optimal Solutions to Capacitated Multidepot Vehicle Routing Problems*. ed. Vol. École des hautes études commerciales, pp., 1984

Laporte, G., Nobert, Y., & Desrochers, M. (1985). Optimal Routing Under Capacity And Distance Restrictions. *Operations Research*, 33(5), 1050-1073.

Laporte, G., Mercure, A., & Nobert, Y. (1986). An exact algorithm for the asymmetrical capacitated vehicle-routing problems. *Networks*, 16(1), 33-46.

Laporte, G., Nobert, Y., & Taillefer, S. (1988). Solving A Family Of Multi-Depot Vehicle-Routing And Location-Routing Problems. *Transportation Science*, 22(3), 161-172.

Laporte, G. & Semet, F. (2001). Classical heuristics for the capacitated VRP, in *The vehicle routing problem*: 109-128: Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

Lin, S. (1965). Computer solutions to the travelling salesman problem. *Bell System Technology Journal*, 44, 2245-2269.

Magnanti, T. L. (1981). Combinatorial optimization and vehicle fleet planning: perspectives and prospects, Secondary.Combinatorial optimization and vehicle fleet planning: perspectives and prospects: Massachusetts Institute of Technology, Operations Research Center.

Min, H., Current, J., & Schilling, D. (1992). The Multiple Depot Vehicle Routing Problem with Backhauling. *Journal of Business Logistics*, 13(1), 259-288.

Min, H. K. (1989). The Multiple Vehicle-Routing Problem With Simultaneous Delivery And Pick-Up Points. *Transportation Research Part A-Policy And Practice*, 23(5), 377-386.

Mingozzi, A., Giorgi, S., & Baldacci, R. (1999). An exact method for the vehicle routing problem with backhauls. *Transportation Science*, 33(3), 315-329.

Mole, R. H. & Jameson, S. R. (1976). A sequential routing-building algorithm employing a generalised saving criteria. *Operations Research Quarterly*, 27, 503-511.

Montane, F. A. T. & Galvao, R. D. (2005). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, 33(3), 595-619.

Mosheiov, G. (1998). Vehicle routing with pick-up and delivery: Tour-partitioning heuristics. *Computers & Industrial Engineering*, 34(3), 669-684.

Nagy, G. & Salhi, S. (2005). Heuristic algorithms for single and multiple depot Vehicle Routing Problems with Pickups and Deliveries. *European Journal of Operational Research*, 162(1), 126-141.

Nelson, M. D., Nygard, K. E., Griffin, J. H., & Shreve, W. E. (1985). Implementation Techniques For The Vehicle-Routing Problem. *Computers & Operations Research*, 12(3), 273-283.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41, 421-451.

Osman, I. H. & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63, 513-623.

Osman, I. H. & Wassan, N. A. (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling*, 5(4), 263-285.

Paessens, H. (1988). The Savings Algorithm For The Vehicle-Routing Problem. *European Journal of Operational Research*, 34(3), 336-344.

Papadimitriou, C. H. & Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*. ed. Vol. Dover Publications, pp., 1998

Pooley, J. (1994). Integrated production and distribution facility planning at Ault Foods. *Interfaces*, 24(4), 113-121.

Potvin, J.-Y., Duhamel, C., & Guertin, F. (1996). A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, 6(4), 345-355.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12), 1985.

Raft, O. M. (1982). Modular algorithm for an extended vehicle scheduling problem. *European Journal of Operational Research*, 11(1), 67-76.

Ralphs, T. K. (2003). Parallel branch and cut for capacitated vehicle routing. *Parallel Computing*, 29(5), 607–629.

Rego, C. (2001). Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. *Parallel Computing*, 27(3), 201-222.

Reimann, M., Doerner, K., & Hartl, R. F., *Insertion Based Ants for Vehicle Routing Problems with Backhauls and Time Windows*. ed. Vol. *Lecture Notes in Computer Science*, 135-148 pp., 2002

Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-Ants: Savings Based Ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4), 563-591.

Reimann, M. & Ulrich, H. (2006). Comparing backhauling strategies in vehicle routing using Ant Colony Optimization. *Central European Journal of Operations Research*, 14(2), 105-123.

Renaud, J., Laporte, G., & Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers and Operations Research*, 23(3), 229-235.

Rochat, Y. & Taillard, E. (1995a). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of heuristic*, 1, 147-167.

Rochat, Y. & Taillard, É. (1995b). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1), 147-167.

Ropke, S. & Pisinger, D. (2006). A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, 171(3), 750.

Salhi, S. & Sari, M. (1997). Models for the multi-depot vehicle fleet mix problem. *European Journal of Operational Research*, 103, 95-112.

Salhi, S. & Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50(10), 1034-1042.

Silvano, M. & Paolo, T., *Knapsack problems: algorithms and computer implementations.* ed. Vol. John Wiley & Sons, Inc., 296 pp., 1990

Stewart, J. W. R. & Golden, B. L. (1984). A Lagrangean relaxation heuristic for vehicle routing. *European Journal of Operational Research*, 15(1), 84.

Stuetzle, T. & Hoos, H. (2000). Max-Min ant System. *Future Generation Computer Systems*, 16, 889-914.

Taillard, E. (1993). Parallel Iterative Search Methods For Vehicle-Routing Problems. *Networks*, 23(8), 661-673.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J.-Y. (1997). A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science*, 31, 170-186.

Thangiah, S. (2001). Genetic Clustering: An adaptive heuristic for the multidepot vehicle routing problem. *Applied Artificial Intelligence*, 15(4), 361-383.

Thangiah, S. R., Potvin, J. Y., & Sun, T. (1996). Heuristic approaches to vehicle routing with backhauls and time windows. *Computers & Operations Research*, 23(11), 1043-1058.

Tillman, F. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3(1), 192-204.

Tillman, F. A. & Hering, R. W. (1971). A study of a look-ahead procedure for solving the multi-terminal delivery problem. *Transportation Research*, 5, 225-229.

Tillman, F. A. & Cain, T. M. (1972). An Upperbound Algorithm for the Single and Multiple Terminal Delivery Problem. *Management Science*, 18(11), 664-682.

Toth, P. & Vigo, D. (1996). A heuristic algorithm for the vehicle routing problem with backhauls. In L. Bianco & P. Toth (Eds.), in *Advanced Method in Transportation Analysis*: 585-608: Springer-Verlag, Berlin.

Toth, P. & Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31(4), 372-385.

Toth, P. & Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, 113(3), 528-543.

Toth, P. & Vigo, D. (2001). An overview of vehicle routing problems. In P. Toth & D. Vigo (Eds.), in *The vehicle routing problem*: 1-26: Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

Van Breedam, A. (1995). Improvement heuristics for the Vehicle Routing Problem based on Simulated Annealing. *European Journal of Operational Research*, 86(3), 480-490.

Wade, A. C. & Salhi, S. (2002). An investigation into a new class of vehicle routing problem with backhauls. *Omega*, 30(6), 479-487.

Wade, A. C. & Salhi, S. (2004). An ant system algorithm for the mixed vehicle routing problem with backhauls, in *Metaheuristics: computer decision-making*: 699-719: Kluwer Academic Publishers.

Wassan, N. (2004). Reactive Tabu Adaptive Memory Programming Search for the Vehicle Routing Problem with Backhauls, Secondary.Reactive Tabu Adaptive Memory Programming Search for the Vehicle Routing Problem with Backhauls, Vol. **. Kent, UK: Canterbury Business School, University of Kent.

Wren, A. & Holliday, A. (1972). Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points. *Operational Research Quarterly (1970-1977)*, 23(3), 333-344.

Xu, J. F. & Kelly, J. P. (1996). A network flow-based tabu search heuristic for the Vehicle Routing Problem. *Transportation Science*, 30(4), 379-393.

Yano, C., Chan, T., Richter, L., Cutler, T., Murty, K., & McGettigan, D. (1987). Vehicle Routing at quality Stores. *Interfaces*, 17(2), 52-63.

Yellow, P. C. (1970). A Computational Modification to the Savings Method of Vehicle Scheduling. *Operational Research Quarterly (1970-1977)*, 21(2), 281-283.