# Master's degree thesis

**LOG950 Logistics**

**The Fleet Size and Mix Vehicle Routing Problem with Pickups and Deliveries**

Gurpiar Singh Cheema

Number of pages including this page: 55

Molde, 23 May 2016

## Mandatory statement

Each student is responsible for complying with rules and regulations that relate to examinations and to academic work in general. The purpose of the mandatory statement is to make students aware of their responsibility and the consequences of cheating. Failure to complete the statement does not excuse students from their responsibility.

| *Please complete the mandatory statement by placing a mark **in each box** for statements 1-6 below.* | | |
|---|---|---|
| **1.** | **I/we hereby declare that my/our paper/assignment is my/our own work, and that I/we have not used other sources or received other help than mentioned in the paper/assignment.** | ☒ |
| **2.** | **I/we hereby declare that this paper**<br>1. Has not been used in any other exam at another department/university/university college<br>2. Is not referring to the work of others without acknowledgement<br>3. Is not referring to my/our previous work without acknowledgement<br>4. Has acknowledged all sources of literature in the text and in the list of references<br>5. Is not a copy, duplicate or transcript of other work | Mark each box:<br>1. ☒<br>2. ☒<br>3. ☒<br>4. ☒<br>5. ☒ |
| **3.** | I am/we are aware **that any breach of the above will be considered as cheating, and may result in annulment of the examination and exclusion from all universities and university colleges in Norway for up to one year, according to the** Act relating to Norwegian Universities and University Colleges, section 4-7 and 4-8 **and** Examination regulations section 14 and 15. | ☒ |
| **4.** | **I am/we are aware that all papers/assignments may be checked for plagiarism by a software assisted plagiarism check** | ☒ |
| **5.** | **I am/we are aware that Molde University College will handle all cases of suspected cheating according to prevailing guidelines.** | ☒ |
| **6.** | **I/we are aware of the University College's** rules and regulation for using sources | ☒ |

## Publication agreement

**ECTS credits: 30**

**Supervisor: Arild Hoff**

### Agreement on electronic publication of master thesis

**I/we hereby give Molde University College the right to, free of charge, make the thesis available for electronic publication:** [✓ **yes**] **no**

**Is there an agreement of confidentiality?** **yes** [✓ **no**]

(A supplementary confidentiality agreement must be filled in)

- If yes: **Can the thesis be online published when the period of confidentiality is expired?** **yes** **no**

**Date: 23-May-16**

# Preface and Acknowledgement

This master thesis is written as a necessary work to obtain MSc degree in Logistics at Molde University College -Specialized University in Logistics. The dissertation work has been carried out from December 2015 to May 2016. Associate Professor Arild Hoff supervised the thesis work. Also, discussions with Professor Lars Magnus Hvattum turned out to be fruitful in association with the thesis work.

I would be grateful to pay my gratitude to my supervisor Associate Professor Arild Hoff for his sincere guidance towards my work and the valuable opinions given in the context of the thesis. I am heartily thankful to Professor Lars Magnus Hvattum also for his help.

I would also say thanks to the academic staff and the officials at Molde University College for giving me the opportunity to explore a new academic environment and the social life here in Norway.

Gurpiar Singh Cheema
Molde, Norway
May 2016

## Summary

This master degree thesis introduces a new dimension to the Fleet Size and Mix Vehicle Routing by adding Simultaneous Pickups and Deliveries (FSMVRPPD) at the customers. A literature study has been conducted to investigate the previous work on the particular problem. The idea to explore this particular problem has come from the transportation structure of two different local industries. Although, these have no direct or indirect association with this thesis. A mathematical model has been presented to define the problem in full. This particular problem has the nature of combinatorial optimization and combines two NP-hard problems. Therefore, an Iterated Local Search algorithm has been proposed to investigate results conducted on the standard benchmark instances for the Fleet Size and Mix problems. These benchmark instances were modified to fit with the FSMVRPPD. An analysis of the outcomes from the ILS algorithm was conducted to see the change in the objective value and the structure of the solution. The investigation also led to observe the how challenging the results are from the ILS approach.

# Contents

## List of Figures

## List of Tables

# 1.0 Introduction

In the present market, the oil price crises put many companies to revise their transportation costs to be stable in the business. The transportation costs should require the attention of management as it, in general, is considered to account for about 20% of the total cost of a product (Hoff et al. 2010). Emergent economy, mounting consumption, and globalization give an increasing focus on transport solutions. Fierce competition between transporters and producers results in efficiency, better end-user service, on-time delivery and of course the need to cut off the transportation cost. By now, there are several tools, methods, models and software available in the market for improving transport efficiency and reducing the transportation costs, also by considering other aspects of better service such as on-time delivery, customer service, quality service etc.

A very central issue when improving the transport efficiency is finding the optimal way of visiting customers described as the Vehicle Routing Problem (VRP). The general VRP was first defined by (Dantzig and Ramser 1959) as the problem of minimizing the costs of routing a fleet of vehicles to serve a set of customers with a given demand. The demand could be either pickup or delivery of goods, and the vehicles were assumed to have a fixed and homogeneous capacity. Later, several extensions of the VRP is described in the research literature, but real-world problems are in general complex and contains different constraints and requirements which make the standard models insufficient.

In this thesis, we are going to discuss a more realistic aspect of transportation by combining two of the earlier defined extensions of the VRP. The thesis is focusing on selecting a fleet of different vehicles and creating routes where the demand of several customers is going to be fulfilled. The customers can have both a pickup and delivery demand at the same time, which makes it necessary to check the vehicle capacity at any point during the routes. We call this problem the Fleet Size and Mix Vehicle Routing Problem with Pickups and Deliveries (FSMVRPPD), which can be seen as a combination of two earlier defined variants of the VRP. These are the Fleet Size and Mix VRP (FSMVRP) described by (Gheysens, Golden, and Assad 1984) and VRP with Pickup and Delivery (VRPPD) described by (Alfredo Tang Montané and Galvão 2006). The problem is related to the design of distribution networks for businesses. The distribution networks require a significant share of investment; therefore, it is a precarious issue for many companies.

The motivation to study this problem comes in particular from by looking at the general transportation of two local businesses. One is a bakery, and another is a cold drink factory both located in the town of Molde. The bakery distributes products to a smaller region as compared to the cold drinks factory. They both have a diversified fleet of vehicles but in limited numbers. Both of them deliver their products in plastic crates and then collect these containers (empty or sometimes with returned products) from the customers. Depots are located at the same location as the production factory. These aspects in the VRP terminology are described as heterogeneous vehicle fleet, single supply station or distribution center and customers with two types of demand, i.e. Pickup and Delivery, simultaneously. Study of the previous literature revealed that the FSMVRPPD has not been studied significantly in the research literature. The examples mentioned above are two specific situations from the real world industries that are relatively close to the FSMVRPPD. But the FSMVRPPD could be actual for a load of similar companies and also in other variants of industries.

In this thesis, we are going to make decisions on the selection of an appropriate vehicle fleet from the available alternative vehicle types. Also, as this transportation network is going to serve the customers, finding optimal routes with the chosen vehicle fleet is a priority. The demand of the customers is the main focus in this problem solution. Each customer has some delivery demand or some pickup demand or both.

The first aspect of the problem is to choose the best suitable vehicle fleet. A homogeneous vehicle fleet can do the job in more general cases. However, this is usually not the case for real industrialized problems, where adopting a fleet of similar vehicles would not be cost efficient. Therefore, the inclusion of heterogeneous vehicles in the fleet would be more profitable and reflect the real industrialized problems. It is also a fact that the vehicles usually are acquired over a long period; thus, mechanical and technological developments in that time period leads to vehicles of different features and properties. There are three main categories to differentiate aspects of vehicle types (Hoff et al. 2010):

- Physical Dimensions
- Compatibility Constraints
- Costs of Operation

**Physical characteristics** of a vehicle put constraints on the capacity of the vehicle. Not only the physical appearance affects the capacity, but some technical issues also control the load capacity. The speed of a vehicle can also be identified as a physical restriction.

**Compatibility constraints** deal with the different types of commodities to be loaded on the vehicles. Compatibility issues concern most when commodities of the different state are going to be shipped, such as Liquids, Gasses and Solid materials. Solid materials are most compatible to transport while liquids and gasses call for some special kind of tanks and chambers to fill in. Another issue is to deal with items those strictly need different storage environment such as cold storage, pressure tanks, vacuum, etc. The food items which require freezer trucks while transportation is the typical example of this constraint.

The **operational cost** of the vehicles is also an important factor when selecting the fleet. As all efforts of operational research and decision support systems are to come up with a cost-efficient and profit oriented solution. Costs include the acquisition cost, depreciation cost, maintenance cost, and environmental cost.

In this thesis, we have considered the two primary aspects of heterogeneous vehicles, which is deemed to be the loading capacity and the operational cost of vehicles.

In this thesis, a mathematical model is formulated to describe the problem. Smaller instances are solved to optimality by using the exact mathematical model, and a metaheuristic is developed to solve larger instances of the problem. I chose to use a variant of the metaheuristic Iterated Local Search to solve the large problem instances as mentioned by (Subramanian et al. 2012) and (Lourenço, Martin, and Stützle 2010). Results from both the exact method and the metaheuristic are achieved and compared to the solution of standard instances for the FSMVRP to see how the pickup and delivery constraints affect the objective value and composition of fleet and routes.

In the next chapter, a discussion on the background of FSMVRPPD has been written, which also includes a review of various related papers. The FSMVRPPD is described in Chapter 3. In Chapter 4, we present the mathematical model following by the Iterated Local Search metaheuristic in Chapter 5. Chapter 6 will cover the computations, interpretation, and comparison of results using the methods described in Chapters 4 and 5. At last, a conclusion will be presented in Chapter 7 in combination with suggestions for further research on the problem.

# 2.0 Problem Background and Literature study

In this chapter, we will discuss the history of routing problems thoroughly, starting from the first time when this problem was introduced. It means that we are going to address the problem from its basis until the current situation. The focus of this thesis is to define the problem as well as possible. Therefore, a classification scheme of previous VRP work concentrates on the nature, characteristics and application scenarios of the problems.

## *2.1 VRP*

*Vehicle Routing Problems are concerned with the delivery of some commodities from one or more depots to a number of customer locations with known demand. Such problems arise in many physical systems dealing with distribution networks. For example, delivery of commodities such as mail, food, newspapers, etc. The specific problem which arises is dependent upon the type of constraints and management objective.* This definition is given by (Achuthan, Caccetta, and Hill 1997).

(Lin et al. 2014) Categorized the following extensions to VRP in their survey on Green Vehicle Routing Problem. They have reviewed the last 50 years of VRP, which enlightens the way to study the history of VRP and its extensions and to define the current problem presented in this thesis.

## 2.2 Extensions to VRP

i. Capacitated VRP (CVRP)

(Dantzig and Ramser 1959) First introduced the vehicle routing problem by describing a real world problem of gasoline delivery to a large number of small-sized service stations from a big size distribution terminal. As we know, when we have a large number of customers to be served, the number of possible routes are also increased exponentially. This results in difficulties in finding the optimal routing for the problem. They developed an algorithm approach based on a linear mathematical formulation to produce a near to the optimal solution to the problem. In their problem, the delivery trucks had a loading capacity which makes the problem a Capacitated VRP. By analyzing the cost matrix for the travel distances, the CVRP can be categorized into Symmetric CVRP and Asymmetric CVRP depending on the distance between two nodes are the same independent of direction or not (Toth and Vigo 2002), they used the branch and bound algorithms to solve the problems.

ii. Time-dependant VRP (TDVRP)

In this type of vehicle routing problem, the real world traffic effects the traveling time of the vehicles in between various nodes. The increase in traffic density can reduce the vehicle speed and travel time would be increased. Traditionally, in the VRP, Euclidean distances between nodes are predicted as constant, but this is usually not the situation in a real world case. As a consequence, the actual traveling cost can be calculated completely wrong (Polimeni and Vitetta 2013). A method to generate travel times based on the area or location of the vehicle and on the time of day is proposed by (Lecluyse, Sörensen, and Peremans 2013). They found out that previous work was not good enough to generate time-dependent travel times. The method proposed by them is able to create travel time profiles for all the edges in a traveling network where congestion areas can be defined by users. Also (Cooke and Halsey 1966) stated that the constant traveling time between two nodes (vertices) is not true for many real world applications. Therefore, they proposed an iteration scheme to find the shortest path by considering variable travel times between nodes. Another extension to Time-dependant VRP is discussed by (Solomon 1987) as VRP with scheduling and Time Windows constraints. (Figliozzi 2012) solve the time dependent problems with hard time windows. The proposed algorithm can handle both constant and variable speed as well as hard and soft time windows.

iii. Pickup and Delivery VRP (VRPPD)

There is a huge collection of research papers available on pickup and delivery problems. (Lin et al. 2014) presented a classification of these type of problems, which can be seen in

Figure 1. They referred this classification from (Parragh, Doerner, and Hartl 2008a, b). The first time pickup and delivery problems were described by (Wilson and Weissberg 1967)as a dial-a-ride problem.

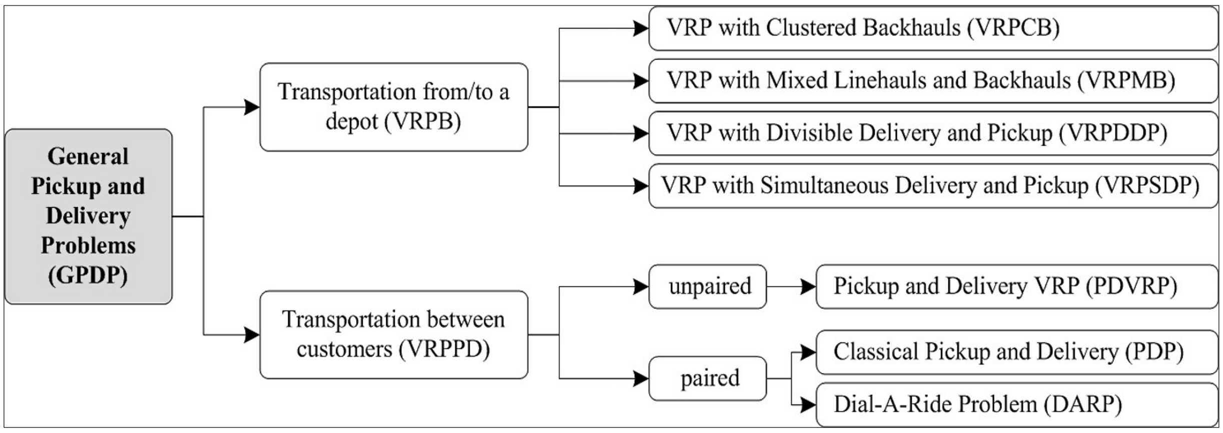
General Pickup and Delivery Problems (GPDP)
Transportation from/to a depot (VRPB)
Transportation between customers (VRPPD)
VRP with Clustered Backhauls (VRPCB)
VRP with Mixed Linehauls and Backhauls (VRPMB)
VRP with Divisible Delivery and Pickup (VRPDDP)
VRP with Simultaneous Delivery and Pickup (VRPSDP)
unpaired
paired
Pickup and Delivery VRP (PDVRP)
Classical Pickup and Delivery (PDP)
Dial-A-Ride Problem (DARP)

*Figure 1. Classification of Pickup and Delivery problem (ref. Lin et al. 2014)*

**Transportation from/to depot.** There are four specific problems specified when transporting goods from or to a depot.

(a.) **VRP with Clustered Backhauls (VRPCB)** states that the group or cluster of linehaul (delivery) customers should be served before starting to visit backhaul (pickup) customers. In VRPCB, the customers can have either delivery demand or pickup demand, but not both.

(b.) **VRP with Mixed Linehauls and Backhauls (VRPMLB)** allow mixed visits to customers. This means vehicles are allowed to visit customers in any sequence, but in one visit only one of the operation performed that could be either delivery or pickup.

(c.) **VRP with Divisible Delivery and Pickup (VRPDDP)** some customers are visited first for only delivering the goods without any pickup. After that, the remaining customers are visited to perform both delivery and pickup operation. On the way back, before the vehicle return to the depot, the remaining pickups, which were left at the start of the route are processed.

(d.) **VRP with Simultaneous Delivery and Pickup (VRPSDP).** Both the delivery and pickup operations must be performed in a single visit to each customer. VRPSDP is the variant considered in the FSMVRPPD.

**Transportation between Customers** is classified into three problems.

(a.) **Pickup and Delivery VRP (PDVRP).** A vehicle traverses across a route to transport the goods, which consists of unpaired pickup and delivery points. Here it is assumed that every unit picked up can be used to satisfy every delivery customer's demand (Parragh, Doerner, and Hartl 2008a).

(b.) **Classical Pickup and Delivery problem (PDP)** this problem can be simplified by considering the example of a bus. A bus driver picks up passengers from different locations and transport them to other locations, where, the driver could have another pickup or not. Means there is no central location/depot of goods. They are distributed over the network of nodes.

(c.) **Dial-a-Ride Problem (DARP)** points to routing and scheduling of vehicles to fulfil the pickup and delivery requests of customers between the start and end point of travel (Cordeau and Laporte 2007). Problems of this kind arise, e.g., in connection with the transportation of handicapped or elderly persons. Another possible application is, the transportation of perishable goods, that also requires maximum ride time limits (Parragh, Doerner, and Hartl 2008a).

iv. Fleet Size and Mix VRP (FSMVRP)

Fleet Size and Mix VRP (FSMVRP) is a situation where one has to decide the fleet composition from the available vehicle fleet and routing of the vehicles simultaneously (Golden et al. 1984). (Baldacci, Battarra, and Vigo 2009) have solved FMSVRP based on two-commodity network flow by proposing a mixed integer programming formulation. (Liu, Huang, and Ma 2009) proposed a genetic algorithm based heuristic to FSMVRP with and without fixed cost. They demonstrate that genetic algorithm approach is as competitive as other, a mathematical programming based and local search based approaches. The FSMVRP with Time Windows has been deeply studied over the past decade.

Some recent studies on FSMVRPTW are done by (Repoussis and Tarantilis 2010) and (Belfiore and Yoshizaki 2013).

v. Multi-depot VRP (MDVRP)

MDVRP is an extension to the CVRP by introducing more than one depot location. Every customer is visited by a vehicle assigned to one of the depots. Origin and destination for a vehicle route must be same depot (Lin et al. 2014).

vi. Stochastic VRP

When one or more components in the VRP are uncertain. SVRP can contain stochastic customers, where probability determines the presence of the customer or not, or the problem can have stochastic demand. Service time and travel time could also be random variables (NEO 2013).

vii. Location Routing Problem (LRP)

Two main decisions have to be taken; one is to select the appropriate depot location and another one is to construct the routes to service the customers. A half century ago (Boventer 1961) introduced the idea of combining location and routing decisions, later, as more rich research known as Location Routing Problem (LRP). Most recent search can be observed from the articles (Koç et al. 2016), (Prodhon and Prins 2014).

viii. Periodic VRP (PVRP)

The objective of PVRP is to find feasible routes such that total routing cost during the time horizon could be minimized. Other important aspects in PVRP are that customers do not need to be visited in every period and the routes can be different in various periods. (Beltrami and Bodin 1974) has proposed algorithms to solve out the municipal waste collection routing problem. They consider time constraint, where locations of waste require a different number of visits and also different combinations of days in a week for a visit. Many real-world situations, such as waste collection, industrial gas distribution, grocery industry and picking up raw materials from suppliers (Alegre, Laguna, and Pacheco 2007), has inspired to study PVRP significantly.

ix. Dynamic VRP

The situation in real world VRP problems is not always constant or static. Over the time, conditions could be different because of uncertainty such as vehicle breakdown, continually getting customer's orders, traffic control etc. (Lin et al. 2014). In a dynamic VRP, the routes can be changed while executed due to such unplanned situations.

x. Inventory Routing Problem

The products are transported from supplier to customer to fulfil the customer's demand over a time period. The customer's demand is deterministic and the time period can be finite or infinite. Transportation is done by a fleet of capacitated vehicles. The inventory cost is applied to customers as well as the supplier. The purpose is to minimize the total routing cost including inventory cost by dealing with time horizon constraint (Bertazzi and Speranza 2012). Typical for these type of problems is that they are not driven by customer orders, but by the customers size of inventory. Thus, it is the supplier who has to make sure that the customers are resupplied before they get out of stock.

xi. Split delivery VRP (SDVRP)

In SDVRP, a fleet of homogeneous vehicles serve the customers, but different from the classical VRP the customers can be visited more than once. This situation occurs where customers may have larger demand than the capacity of the vehicle but not necessarily. Each vehicle must have the same depot as starting and ending point (Archetti and Speranza 2008). The SDVRP was first introduced by (Dror and Trudeau 1989). They showed that one could have savings in cost by splitting the deliveries. (Archetti, Savelsbergh, and Grazia Speranza 2008) wrote in their paper that savings depend on the features of the instance. They found that it could be more beneficial if the average customer demand is slightly more than the half of the vehicle capacity and the variance in customer demand is small.

## 2.3 Solution methods

From the study of literature available in the previous section, it is clear that the standard VRP is a combinatorial optimization type and therefore NP-hard as described by

(Stutzle 1998). The FSMVRPPD combines two variants i.e. VRPPD and FSMVRP which makes the problem even harder. Thus, developing a good solution method is not an easy process. There are numerous papers pointing to various approaches to follow while finding the best solution for these types of problems. In general, the vehicle routing problems can be handled through two main classes of solution methods as shown below:

(i) Exact Methods

(ii) Approximation Methods

- Heuristics
- Meta-heuristics

### 2.3.1 Exact Methods

Finding the exact optimal solution by using some mathematical techniques know as exact methods. In exact methods, there are some functions which follow the mathematical rules for calculations to develop the solutions. For limited size instances, an exact simple method could be to enumerate all possible solutions fully. But, using such type of methods is in practice infeasible for larger instances due to the exponential size of the solution space. The best known exact method for optimization problems is the Branch and Bound algorithm (Stutzle 1998). Researchers have always been challenged to solve NP-hard optimization problems to optimality since computers can be used to solve these problems. Noteworthy progress has been made for solving these problems in recent time span, but still only smaller instances can be solved. *Vehicle routing problems belong to a class of problems that has proved to be difficult to solve. Only moderately sized problems can be solved to optimality consistently (Ropke 2005).* FSMVRPPD is a combinatorial optimization problem that combines the aspects of two other combinatorial problems. Therefore, we cannot expect an instant optimal solution for large or moderate instances of this problem.

Three main classes of exact methods for VRP are proposed by (Toth and Vigo 2002); Branch and Bound, Branch and Cut and Set Covering based algorithm.

**Branch and Bound Algorithms** are based on the idea of divide and conquer. A large solution space is divided into smaller subproblems for solving them separately. A lower bound is found by a linear relaxation of the original problem to determine whether to continue the decomposition of the problem or if optimal solution which satisfies all the constraints, has been reached. (Laporte and Nobert 1987) has done a detailed study on the branch and bound algorithm.

**Branch and Cut Algorithms** are likely to branch and bound algorithm but use a cutting plane algorithm to get linear constraints which are satisfied by all feasible integer values, but excludes present fractional solutions. The inequalities added to the problem yields a less fractional solution. The problem then divided in two and solved again. This process is repeated until a solution satisfying all the integer constraints is found.

**Set Covering based algorithms** are based on the enumeration technique, where all feasible routes in the problem are enumerated and the best possible solution set covering all the customers is selected. Enumeration is a time-consuming process and even enumerating a small instance is not easy. The column generation heuristic can overcome this problem by enumerating only a small subset as the possible routes. In **Column Generation**, the idea is to generate only those routes which have the potential to decrease the total cost. Firstly, a master problem (original problem) is solved to obtain the value for each constraint. A negative reduced cost is obtained from the subproblem which is obtained from the master problem. Then the negative reduced cost is added to the master problem and the master problem is resolved. This process is repeated until the subproblem generates no negative cost. Each column represents a feasible route for a vehicle and the set of routes covering all customers exactly once represent a feasible solution.

### 2.3.2 Heuristics

Solving large combinatorial optimization problems by using exact methods is not easy. So heuristics are commonly used as rules of thumb to solve large instances on these type of problems and produce good solutions in short time. Heuristics can be categorized as:

- Constructive heuristics
- Improvement heuristics

A **constructive heuristic** is initialized by building the solution from a fresh starting point and adding the most acceptable components step by step until a full solution is constructed. The most famous constructive heuristics are the **Sweep algorithm** and **Clarke and Wright Savings algorithm**.

In the **Sweep Algorithm**, a straight line is imagined from the starting point (depot) of a route and then, it is swept across the area around the origin point. The sweeping could be clockwise or anti-clockwise. Nodes are added to the route due to the angle of the sweeping line, and when the capacity of a vehicle is reached a new vehicle is introduced

In **Clarke and Wright Savings Algorithm (Clarke and Wright 1964)**, the idea is to obtain maximum cost savings by combining two routes into a single route as shown in Figure 2.

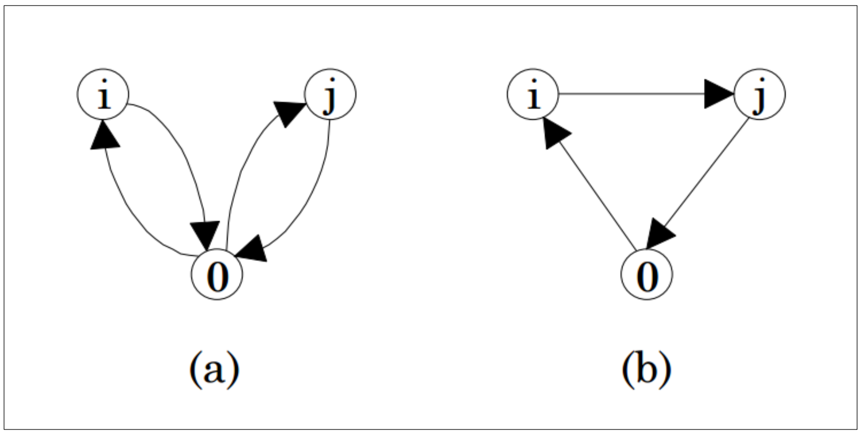

*Figure 2 Clark and Wright Savings Algorithm (Ref. Clarke and Wright (1964)*

In Figure 2(a) two separate routes for nodes *i* and *j* are shown. But *i* and *j* can be visited in single route as shown in Figure 2(b). To combine two routes into one route, the cost savings is calculated for nodes included in the routes. The formula to calculate the savings is:

$$S = c_{i,0} + c_{0,j} - c_{i,j}$$

Where, *c* denotes the cost of traveling to related nodes in the routes.

An **improvement heuristic** focuses on to improve an existing solution by applying changes within a vehicle route or in between different vehicle routes. These changes or modifications could be moving a customer to another route, exchange customers between two routes, change in the visiting sequence in one route, etc. A Local Search implements the modifications to one local solution in expectation of finding improved solution. The Local Search will stop if no improvement solution is found. The Local Search relies on the definition of a neighbourhood which is the possible new solutions that can be reached from a solution by performing one move. Then the resulting solution is defined as a local optimum.

### 2.3.3 Metaheuristics

The idea is to combine two or more heuristics to generate high-quality solutions. The term metaheuristic was introduced by Fred Glover in 1986 and is defined by (Glover and Kochenberger 2003) as:

> “ *Metaheuristics, in their original definition, are the solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust of a solution space.*
>
> *Over time, these methods have also come to include any procedures that employ strategies for overcoming the trap of the local optimality in the complex solution spaces, especially those procedures that utilize one or more neighbourhood strucutures as a means of defining admissible moves to transition from one solution to another, or to build or destroy solutions in constructive and destructive processes.* “

Metaheuristics can be classified into three categories: **Local search based, Constructive** and **Population-based.**

#### 2.3.3.1 Local Search based metaheuristics

These type of heuristics are starting from an initial solution and moving to a neighbouring solution while applying minor changes to the solution until the given number of iterations come to an end. Some good known types of Local search are:

- Iterated Local Search
- Tabu Search
- Simulated Annealing
- Guided Local Search
- Variable Neighbourhood Search

**Iterated Local Search (ILS)**: This metaheuristic extends the local search method to avoid getting stuck in a local optimum, where improving neighbors are not available. ILS modifies the method by repeated calls to the Local Search method. Each time it starts from a new initial solution which usually will end up in a different local optimum.The best of the local optima found is stored and returned when the search is completed.

**Tabu Search:** Tabu search allows the modifications to the solution which not necessary leads to a better solution. Thus, the method can avoid being stuck in a local optimum, by preventing the search from coming back to previously visited solutions. Tabu search uses memory to keep track of already explored solutions and stores components from the most recent moves in a tabu list to prevent them from being part of a new move within a given time frame. The best solution found overall in the search is stored and returned when the search is usually completed after a given number of iterations.

**Simulated Annealing:** This metaheuristic is used to solve the type of problems, where finding a good solution in a short time period is important. Simulated Annealing selects a possible move at random, and uses probability to decide whether to accept a solution or not if the quality is poorer than the previous one. The probability factor decreases during the search, which leads to only good quality solutions will be accepted at the end of the search.

**Guided Local Search (GLS):** GLS uses penalties to help escape a Local Search method from local optima and progress to find global optima. These penalties are calculated while searching the solution space. The idea is to add penalties to the cost function employed by the Local Search method. A given number of times the repetitions are performed on the Local Search by using local optima and the enlarged cost function that directs the search away from local optima.

**Variable Neighborhood Search:** VNS is different from other local search based meta-heuristics because it does not explore one single neighborhood, but alters between several different predefined neighborhoods. This means that it can escape from a local optimum in one neighborhood by introducing another one.

#### 2.3.3.2 Constructive Meta-heuristics

The complete solution is not considered in this type of metaheuristics. The solution is constructed by addition of one element in each iteration. Unlike the constructive heuristics, global memory is used between several runs of the construction process to store information about the quality of the components. The objective value of a solution is not given until the solution is completed, but then it can be used for guiding the next run of the process. These metaheuristics are constructed in a greedy way, and examples are:

- Greedy Randomized Adaptive Search Procedure(GRASP)
- Ant Colony Optimization

**Greedy Randomized Adaptive Search Procedure (GRASP)** is based on the repeated sampling of random greedy solutions and improve those solutions by using a local search approach to find local optima. The sampling function builds a Restricted Candidate List (RCL) that controls the selection of the components of a solution. A threshold on the cost of including the component to the candidate solution defines the greediness of the sampling method.

**Ant Colony Optimization** uses the search history and heuristic information to build the candidate solution and keep the knowledge from solution construction into the history. The solutions are developed separately one at a time by using probability. The selection of component is based on the involvement of component to the overall solution cost and quality of the historic solution from which the component has been included.

#### 2.3.3.3 Population-based Meta-heuristics

These metaheuristics strategy is to look into a group of different solutions. While, other metaheuristics try to improve individual solutions. The objective here is to find high-quality solutions by interchanging the characteristics between existing solution. The idea is that combining the characteristics of two good solutions could lead to an even better solution. Types of these metaheuristics are:

- Genetic algorithm

**Genetic Algorithm** can solve both constrained and unconstrained optimization problems which imitate the way of natural selection. The genetic algorithm randomly selects the component from the population of the solutions and then produce the offspring of the selected components by considering them as parents. As the offsprings are generated over and over, the original population progress towards an optimal solution.

# 3.0 Problem Description

## *3.1 Problem Definition*

The problem (FSMVRPPD) considered in this thesis includes two major decisions. First, the construction of a fleet consisting of different types of vehicles, and then routing the heterogeneous fleet of vehicles such that all customer's demand is fulfilled completely. The storage capacity at customers is not included in the problem. As the fleet includes heterogeneous vehicles, thus each vehicle type has different loading capacity.

Then the question is why a business or company wants networks to design this distribution? The answer is to optimize the transportation costs and to run the business successfully. As the companies have the main goal to achieve the profit from the investments and transportation cost in general accounts for about 1/5th of the total cost.

Therefore, the objective of FSMVRPPD is to minimize the total cost while fulfilling the customers' demand. The cost includes the operational cost of using the heterogeneous fleet of vehicles for traveling between the depot and customers in a route, in addition to a fixed cost of acquiring/owning the vehicles in the fleet.

For a general understanding, we can present the objective function as the cost function below:

$$Minimize\ F_c = F_v + T_r$$

Where, $F_c$ – total cost (sum of both cost factors)

$F_v$ – Fixed dispatch cost of vehicles

$T_r$ – Travelling cost between all nodes

The cost function includes the costs of only those vehicles selected for the fleet, in order to serve the customers and the traveling cost for routing the vehicles between the depot and the customers. There will be some decision criteria to find the optimal results about travelling routes and vehicle fleet.

The total cost is calculated by summing up the cost of each route found in the solution. There are chances that the same type of vehicle can be used on various routes, assuming that the possible number of each vehicle type is unlimited. Therefore, the fixed cost of the selected vehicle type is counted for every route it is assigned to. We can describe the problem by looking at the following rules/constraints:

- There exist a set of heterogeneous vehicle types with different load capacity and fixed cost.
- Only one vehicle can serve a route at a time.
- The fleet can consist of a combination of any number of all vehicle types, and not all vehicle types need to be included in the fleet.
- A customer should be served in only one route.
- All the customers are served from single un-capacitated distribution point/ depot. Each route must start and finish at the depot.
- Customers can have two types of demand; one is delivery, and another is the pickup.
- Some customers can have delivery only and some can have pickup only.
- Both delivery and pickup demand must be fulfilled on a single visit to the respective customer.
- The vehicle capacity cannot be exceeded at any customer in the route.

## *3.2 Example*

In this section, a small example is presented to make an easier explanation of the FSMVRPPD. In this example, there are five customers with both pickup and delivery demands, two vehicle types with the different capacity fleet and a depot. Figure 3 represent the scenario of the problem. A triangle shape denoted with D accounts for the depot node from where the customers are going to be served. The five rectangles represent the five different customers as c1, c2, c3, c4 and c5. Each customer has a delivery demand denoted as ‘d’ and a pickup demand denoted as ‘p’. The values for each demand are shown near to respective customer.

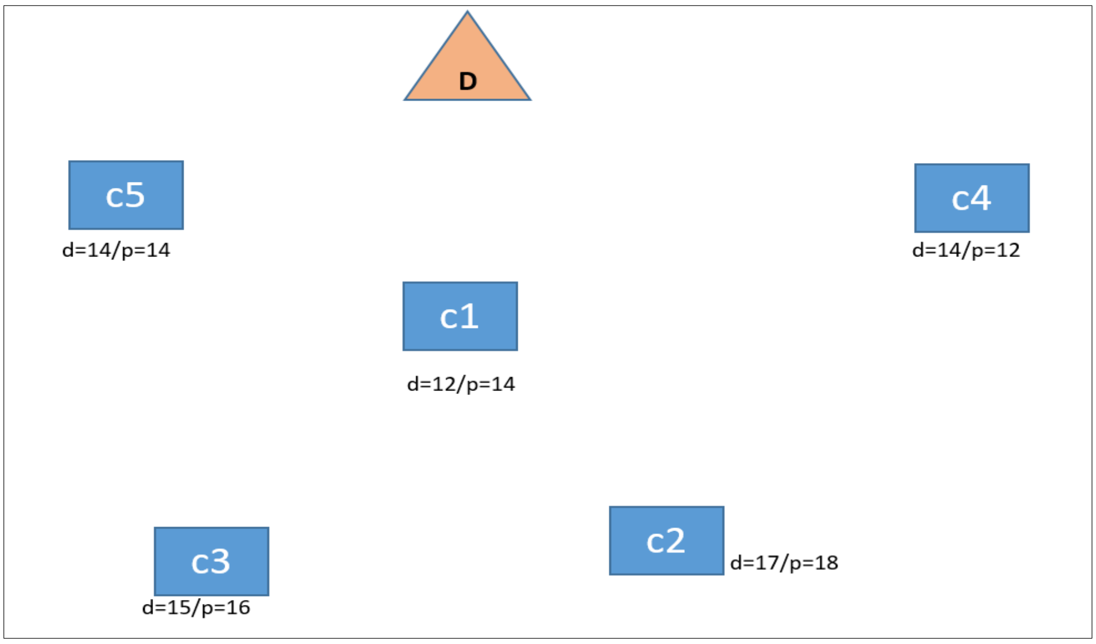

*Figure 3 A VRP problem scenario*

Now we create routes to fulfill the customer demands by using the two types of vehicles V1 and V2, with capacities of 35 and 40 respectively. According to the FSMVRPPD explained above, the solution for this example would be as shown in Figure 4.

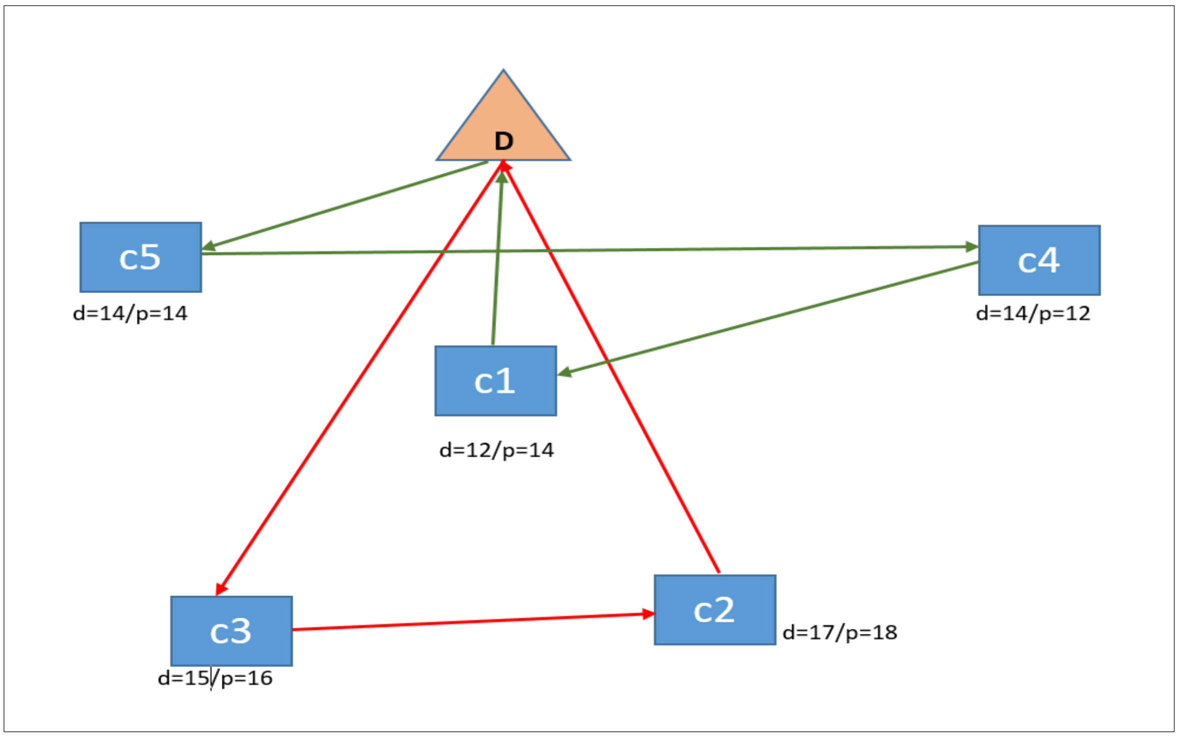

*Figure 4 Vehicle routing solution to example*

The scenario has been solved by using both the vehicles. There are two different routes RED and GREEN, represented by red and green arrows. V1 is assigned to RED and V2 to GREEN. Vehicle V1 serves two customer c2 and c3, while vehicle V2 serves the three remaining customers c1, c4, and c5. On the RED route, the customer has the total delivery demand of 32 units and total pickup demand of 34 units. If vehicle V1 starts from the depot with 32 units, visits c3 where it delivers 15 and picks up 16 units. The load is now 33 units, which still is below the total capacity of 35. Next c2 is visited where 17 units are delivered, and 18 are picked up. The total load is increased to 34 and the vehicle returns to the depot. On the other hand, vehicle V2 serves the GREEN route. Both delivery and pickup demand is the same i.e. 40 units, so the vehicle is fully loaded when leaving as well as when returning to the depot. We can, however, see that c1 cannot be visited as the first customer on the route since the pickup demand is higher than the delivery demand, which would lead to overload on the vehicle. In this example, c5 is chosen as the first customer on the route, and then the only option to avoid overload is to visit c4 next. Thus, the routes would be:

RED: D →c3 →c2 →D; d=32, p =34; vehicle = V1 (capacity 35)

GREEN: D →c5 →c4 →c1 →D; d=40, p=40; vehicle= V2 (capacity 40)

These routes are a feasible solution to the problem and do not violate any of the constraints of the FSMVRPPD. Still better solutions exist for this instance.

## 4.0 Mathematical Model for FSMVRPPD

In this section, the mathematical formulation for the FSMVRPPD is presented. This formulation is an extension of the FSMVRP proposed by (Gheysens, Golden, and Assad 1984). Their FSMVRP model has been modified to meet the current problem situation, and the pickup and delivery demand constraints have been formulated and added into the FSMVRP model. In this formulation, each customer is going to be served for both demands (i.e. pickup and delivery) at the same time, or in other words, both pickup and delivery are going to be performed in single visit to the particular customer.

To implement the pickups and deliveries in FSMVRP the following rules have been taken into account:

*(i)* ***Feasibility****: A solution is feasible if the total quantity assigned to each route does not exceed the capacity of the vehicle which services the route and the vehicle has enough capacity for picking-up the commodities at customers.*

*(ii)* ***Delivery-feasible****: This case means that the total amount of commodities to deliver in a route must not exceed the vehicle's capacity.*

*(iii)* ***Pickup- feasible****: This rule ensures that the vehicle has enough capacity to pick-up the goods of all the customers of the route.*

*(iv)* ***Load feasible****: The vehicle's capacity is not violated at any node of the route. Such a violation can depend on the sequence of the customers even if the route is both Pickup- and Delivery feasible (NEO 2013).*

The following are the notations used in the FSMVRPPD formulation:

n – total number of customers

N – set of the nodes, including depot node 0, N= 0,1,….n

A – set of arcs of traveling possibilities between nodes $(i, j)$, where $i \neq j, \forall\, i \in N, \forall\, j \in N$

C – set of customers, C=1….n

V – set of vehicle types

$Q_v$ – capacity of a vehicle of type $v$, $\forall v \in V$ and $Q_1 < Q_2 < \ldots < Q_v$

$f_v$ – fixed operation cost of vehicle type $v$, $\forall v \in V$ and $f_1 < f_2 < \ldots < f_v$

$d_j$ – delivery demand of customer $j$, $\forall\, j \in C$

$p_j$ – pickup demand of customer $j$, $\forall\, j \in C$

$T_{i,j}$ – cost of traveling from customer $i$ to customer $j$ , $\forall\, (i,j) \in A$

$Y_{i,j}$ - delivery load from customer $i$ to customer $j$ , $\forall\, (i,j) \in A$

$Z_{i,j}$ - pickup load from customer $i$ to customer $j$ , $\forall\, (i,j) \in A$

$X_{i,j,v}$ – 1 if vehicle of type $v$ travels directly from customer $i$ to customer $j$, 0 otherwise

$\forall\, (i,j) \in A$ , $\forall v \in V$

$$Minimize \sum_{v \in V} \sum_{j \in N} f_v\, X_{0,j,v} + \sum_{v \in V} \sum_{(i,j) \in A} T_{i,j}\, X_{i,j,v} \qquad (1)$$

Subject to

$$\sum_{v \in V} \sum_{i \in N} X_{i,j,v} = 1, \quad \forall\, j \in C \qquad (2)$$

$$\sum_{i \in N} X_{i,j,v} - \sum_{i \in N} X_{j,i,v} = 0, \quad \forall\, j \in C, v \in V \qquad (3)$$

$$\sum_{i \in N} Y_{i,j} - \sum_{i \in N} Y_{j,i} = d_j, \quad \forall\, j \in C \qquad (4)$$

$$\sum_{i \in N} Z_{j,i} - \sum_{i \in N} Z_{i,j} = p_j, \quad \forall\, j \in C \qquad (5)$$

$$Y_{0,j} \leq \sum_{v \in V} Q_v\, X_{0,j,v}\,, \quad \forall\, j \in C \qquad (6)$$

$$Z_{j,0} \leq \sum_{v \in V} Q_v\, X_{j,0,v}\,, \quad \forall\, j \in C \qquad (7)$$

$$Y_{i,j} + Z_{i,j} \leq \sum_{v \in V} Q_v \, X_{i,j,v} , \qquad \forall \, (i,j) \in A \tag{8}$$

$$Y_{i,j} \leq \sum_{v \in V} M \, X_{i,j,v} , \qquad \forall \, (i,j) \in A \tag{9}$$

$$Z_{i,j} \leq \sum_{v \in V} M \, X_{i,j,v} , \qquad \forall \, (i,j) \in A \tag{10}$$

$$Y_{i,j} \geq 0, \quad \forall \, (i,j) \in A \tag{11}$$

$$Z_{i,j} \geq 0, \quad \forall \, (i,j) \in A \tag{12}$$

$$X_{i,j,v} \in (0,1), \quad \forall \, v \in V, (i,j) \in A \tag{13}$$

The objective function (1) outputs the total cost of servicing all the customers to fulfill their demand. The total cost consists of fixed operational cost of vehicles used and the variable cost of the routes used in the solution. Constraint (2) guarantees that every customer is serviced only once, in other words, pickup and delivery operations are performed in the single visit. Constraint (3) states that same type of vehicle that reach a customer must leave the same customer, while constraints (4) and (5) are the flow equations for both delivery and pickup demands of customers, respectively. In constraint (6), it is stated that total delivery load must not exceed the vehicle capacity while leaving the depot. On the other hand, constraint (7) shows that when the vehicle arrives at the depot after serving customers, the pickup load should not violate the vehicle's capacity. While Constraint (8) ensures that the vehicle capacity must not be violated during the visit to customers on the route. Constraints (9) and (10) represent, respectively, that no delivery and pickup operation will be performed on arc $i \ to \ j$ if that arc is not served by any of the vehicles. Constraints (11) and (12) state that variables for delivery load and pickup load must be positive number, and in (13), the route decision variable must hold a binary value either 0 or 1.

# 5.0 Iterated Local Search

In this chapter, we will see the implementation of an Iterated Local Search algorithm on our current problem, and analysis/interpretation of results produced by the algorithm.

## *5.1 ILS Algorithm*

ILS is a simple metaheuristic approach for solving combinatorial optimization problems. ILS helps a traditional Local Search method to avoid getting stuck in a local optimum by applying some simple modifications to it. The changes to the Local Search method consists in repetitions of it, every time commencing from a new initial point. The starting points are obtained by applying perturbation to the current solution or eventually choosing a random one. Thus, it does not use information obtained from previous Local Search stages but uses memory about previously found local optima to develop better starting points for a Local Search.

### 5.1.1 General ILS Procedure

This section presents a general procedure for iterated local search as shown in Figure 5:

```
    General Algorithm: Iterated Local Search
1.  Generate: INITIAL_SOLUTION $s_i$
2.  s* ← LOCAL_SEARCH ($s_i$)
        -  Local solution driven from $s_i$
3.  ITERATE until END CONDITION
4.      s' ← MODIFY (s*)
        -  Modifies the solution $s^*$ to obtain a random starting point
5.      s*' ← LOCAL_SEARCH (s')
6.      s*  ← ACCEPTANCE_CRITERION (s*, s*')
7.  EXIT
```

*Figure 5 General Procedure for Iterated Local Search*

In general, this procedure is implemented as a problem-specific optimization algorithm. By the repetitions of the algorithm, one could obtain significantly good results, but it needs improvements in the search space.

Usually, this algorithm is implemented to optimize cost for combinatorial optimization problems. Suppose, we have a candidate solution *s* and the set *S* which contains all possible solutions.

Let *C* be a cost function that is to be minimized. The algorithm takes a solution $s'$ as input to the local search algorithm, then the local search produces the solution $s^{*'}$ with lower or equal cost value than *C(s).* After each performance of the local search, the resulting solution $s^{*'}$ is compared to the previous $s^*$, and if fulfilling the acceptance criterion (i.e. best so far in the search), it replaces the previous as $s^*$. (Lourenço, Martin, and Stützle 2010) presented the basin of attraction in Figure 6.

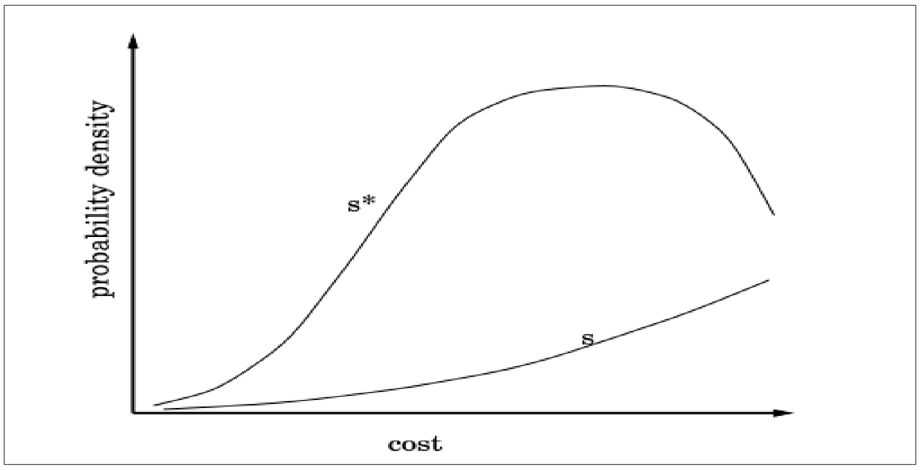

*Figure 6 Probability densities of costs*

*The curve labelled s indicates the left tail of the cost density function for all solutions, while the curve labelled s* indicates the cost density function for the solutions that are local optima (Lourenço, Martin, and Stützle 2010).*

In Iterated Local Search the set *S** of all solutions *s*'* is explained as walking from one *s** to a neighbouring one. It is implemented heuristically as below:

- It applies a change or modification to *s** (current given) which results as *s'*
- LOCAL_SEARCH operates on *s'* and produces a solution *s*'*
- If *s*'* qualifies the acceptance test, it becomes the next solution to be used as a basis for the search, otherwise *s** remains the basis.

This procedure should lead to a large exploration of the solution space if the modifications/perturbations are not too small. In our approach, we are obtaining random solutions by making large modifications to the current solution. On the other hand, smaller modification would often lead the solution back to *s** and explore very limited area of the current solution. Diagrammatically the ILS procedure could look as shown in Figure 7.

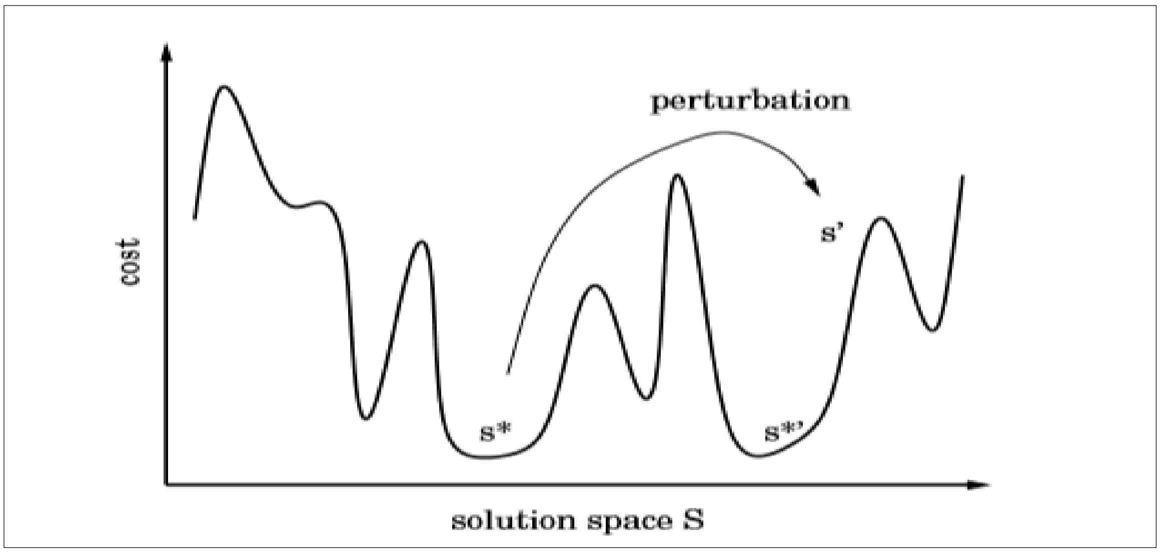

*Figure 7 ILS procedure*

In Figure 7, *Starting with a local minimum s*, we apply a perturbation leading to a solution s'. After applying LOCAL_SEARCH, we find a new local minimum s*' that may be better than s* (Lourenço, Martin, and Stützle 2010).*

As a summary of this section, we can say that ILS is more efficient when it has biased exploration of the set of local optimal solutions. The exploration effectiveness is directly influenced by the modifications to solutions and acceptance conditions, but the resulting solutions can be better by adjusting the ILS building blocks.

## 5.2 ILS for FSMVRPPD

This section describes the simple Iterated Local Search (ILS) algorithm implemented for solving the FSMVRPPD. The ILS algorithm follows a very simple technique to iterate over a Local Search method by taking a random starting point every time. The algorithm follows the major steps from Figure 5 where four important procedures have to be defined. These are: INITIAL_SOLUTION, LOCAL_SEARCH, MODIFY and ACCEPTANCE_CRITERION.

The initial starting point is generated through the procedure INITIAL_SOLUTION by randomly selecting customers to be served in the routes and assigning vehicle types that meet the demand in each route. Afterwards, there is feasibility check on the generated routes to make sure that the solution is feasible and if so the solution value is calculated. Then the LOCAL_SEARCH is applied on the solution generated by INITIAL_SOLUTION to find the local optimum $s*$. Now, the ILS goes under the iteration phase where MODIFY performs perturbation to the local solution $s*$ from LOCAL_SEARCH and outputs a changed solution $s'$ as a basis for the next LOCAL_SEARCH call. Again, LOCAL_SEARCH is applied to the new solution $s'$ and generates another local optimum $s*'$. Afterwards, the procedure ACCEPTANCE_CRITERION compares $s*$ and $s*'$ to take a decision on which solution is going to be explored further and which one is going to be discarded. The repetition is bounded by a given number of iterations.

### 5.2.1 INITIAL_SOLUTION

The ‘init()’ procedure is shown by pseudocode in Figure 8 and starts by initializing the current solution with the total possible number of routes in the given problem. Total maximum routes are calculated by dividing the total delivery demand of all customers by the capacity of the smallest available vehicle, and two extra routes are added to account for differences in delivery and pickup demand. Then the procedure ‘rndSolution()’, described in Figure 9, generates some routes by randomly distributing customers to different vehicle routes until all the customers are assigned to vehicle routes. Afterward, a procedure ‘adjustVehicle()’ in Figure 10 assigns vehicles of the necessary capacities to the routes for meeting the customer’s demands. A solution $S_i$ is obtained from the above process and checked for the feasibility on all feasibility constraints. If $S_i$ violates one of the feasibility constraints, then the procedure ‘isFeasiblePD()’shown in Figure 11, returns that the solution is infeasible. If solution $S_i$ is infeasible, then a large penalty is added to the solution’s objective value to make sure that the solution is not considered as the overall best found during the search. On the contrary, an infeasible solution can still be a good basis for finding good feasible solutions after the Local Search. In Figure 12 the procedure ‘TotalRoutingCost()’ which calculate the total cost of a solution is shown.

| | **Procedure: init ()** |
|---|---|
| **1.** | maxRoutes ← (sum of total delivery / smallest vehicle capacity) + 2 |
| **2.** | SET infeas = TRUE |
| **3.** | CALL rndSolution(CurSol_) |
| **4.** | CALL adjustVehicle(CurSol_) |

```
5.  IF CALL isFeasiblePD(CurSol_) = TRUE THEN
6.          -   SET infeas = FALSE
7.  CALL totalRoutingcost(CurSol_)
8.  SET CurSolVal_ = total cost of solution ‘CurSol_’
9.  IF CALL  isFeasiblePD(CurSol_) = TRUE THEN
10.         -   PASS feasible_ ← TRUE
11. ELSE PASS feasible_ ← FALSE
12. Apply infeasible penalty to ‘CurSolVal_’
13. ADD big amount to cost of solution
14. EXIT.
```

*Figure 8 Initialization procedure*

```
    Procedure: rndSolution(Solution &SOL)
1.  Set vIter
2.  Set noVehicle ← number of routes from SOL -1
3.  Repeat FOR matrix size ‘n()’
4.          -   temp ← dimension value of matrix
5.  END of FOR loop 3.
6.  DO WHLE ‘count’ > 1
7.          -   count ← size of matrix
8.          -   randNo ← random number in range of ‘count’ +1
9.          -   vIter ← point to ‘randNo’
10. IF randNo does not point to depot node
11.         -   vTyp ← random vehicle type +1
12.         -   RouteNo ← random number in range of ‘noVehicle’ +1
13.         -   CALL setVehtypRoute (RouteNo, vTyp)
14.         -   CALL addCusRoute (RouteNo, temp[randNo])
15. END of IF
16. Delete element pointed by ‘vIter’
17. END of DO-WHILE loop 6.
```

*Figure 9 Procedure for creating a random solution*

```
Procedure: adjustVehicle(Solution &SOL)
1.  vsize ← Get number of routes in SOL
2.  Repeat FOR vsize
3.      -   cRoute ← Get number of customers in current route ‘vsize’
4.  Repeat FOR cRoute
5.      -   t_demand ← calculate total delivery
6.      -   t_pickup ← calculate total pickup
7.  END of FOR loop 3.
8.  Get total vehicle type ‘noVtypes()’
9.  Repeat FOR noVtypes
10.     -   IF t_demand & t_pickup <= capacity of current vehicle ‘k’
11.     -   Change vehicle type to ‘k’ in route ‘vsize’
12.     -   BREAK
13.     -   END of IF
14. END of FOR loop 6.
15. END of FOR loop 2.
```

*Figure 10 procedure for adjusting the vehicle to the best fitted for meeting the customer demand on the route*

```
Procedure: isFeasiblePD (Solution &SOL)
1.  vsize ← get number of routes in SOL
2.  IF vsize == 0 THEN return FALSE and exit.
3.  Repeat FOR each route ‘vsize’
4.      -   vType ← number of vehicle type
5.      -   capacity ← capacity of vehicle type ‘vType’
6.      -   cRoute ← number of customers in route ‘vsize’
7.  Repeat FOR each customer ‘cRoute’
8.      -   sumDemand ← calculate total delivery demand in route ‘vsize’
9.      -   sumPickup ← calculate total pickup in route ‘vsize’
10.     -   IF sumDemand OR sumPickup > capacity THEN return FALSE and exit.
11. END of FOR loop 7.
12. INITIALIZE currLoad ← sumDemand
13. Repeat FOR each customer ‘cRoute’
14.     -   Subtract delivery ‘d_’ for each customer from ‘currLoad’
```

```
15.         - Add pickup ‘p_’ for each customer into ‘currLoad’
16.         - IF currLoad > capacity THEN return FALSE and exit.
17.     END of FOR loop 13.
18.     END of FOR loop 3.
19.     Return TRUE.
```

Figure 11 Procedure for checking whether a solution is feasible or not

```
        Procedure: totalRoutingcost (Solution &SOL)
1.      INITIALIZE sum =0
2.      Repeat FOR each route ‘vsize’
3.          - cRoute ← number of customers in route ‘vsize’
4.      Repeat FOR each customer in ‘vsize’
5.          - sum ← calculate edge cost
6.      END of FOR loop 4.
7.      typ ← vehicle type in route ‘vsize’
8.      cost ← fixed cost of vehicle type ‘typ’
9.      sum ← sum + cost
10.     END of FOR loop 2.
11.     RETURN sum.
```

Figure 12 Procedure for calculating the total cost of a solution

### 5.2.2 LOCAL SEARCH

The pseudo code for the Local Search is shown in the procedure ‘runSolver ()’ in **Error! Reference source not found.**. The local search procedure ‘runSolver()’ in explores the neighbourhoods until an un-improved solution is found. In the very beginning of ‘runSolver’ procedure, the random initial solution $S_i$ set to be the current best solution $s*$. Then, the repetition starts to perform the local search while the solutions are getting improved. Next step is to collect information from the current solution $s*$ and pass that information to the appropriate variables used in the local search procedure ‘runSolver’. A very large value (i.e. DBL_MAX) is initialized as the best neighbourhood value BN, so that the solutions with large values could be explored. Next, the repetition on the total number of routes in the current solution $s*$ begins to explore the neighbourhoods inside the current solution space. Inside the loop over the routes, an another loop is used to get the customers from the current route. Further inside the loop over number of customers, the number of total routes are repeated to get two different neighbourhoods for exploration. The information on the vehicle's capacity for the current route is collected. Then, a conditional statement validates the vehicle capacity constraint. The vehicle capacity constraint should not be violated. If the conditional statement mentioned in later sentence outputs a Boolean TRUE value, then a neighbourhood N1 by adding the current customer to the current route will be used. Otherwise, a neighbourhood N2 by swapping the customers between two different routes will be used. The optimization procedures from **Error! Reference source not found.** are applied to the neighbour N1 or N2 to obtain the optimality. Afterwards, the feasibility procedure in **Error! Reference source not found.** checks whether the new solution $s*'$ is acceptable or not. If the solution $s*'$ found from the neighbourhood N1 or N2 is feasible and better than the best neighbourhood value BN, then the best neighbourhood value BN will be initialized with the current found solution $s*'$. Otherwise, the infeasible solution $s*'$ would lead to add penalty to objective value to escape from the infeasible solution space. After the completion of the repetitions on all of the routes in the solution $s*$, the best neighbourhood value BN is compared with value of the current solution $s*'$. If the best neighbourhood value BN is better than the value of the solution $s*'$ and if the solution $s*'$ from the neighbourhood is feasible, then the best neighbourhood value BN will be set as the new current best solution $s*$ value and search will continue to improve the solution further. Otherwise, if no improvement is found from the neighbourhood N1 or N2, the local search will be stopped and output the current best value as the solution to the current search.

| | **Procedure: runSolver()** |
|---|---|
| **1.** | START clock |
| **2.** | noRoutes ← number of routes in 'CurSol_' |
| **3.** | SET BestSol_ = CurSol_ |
| **4.** | SET BestSolVal_ = CurSolVal |
| **5.** | SET BestNeighborVal_ = very large value |
| **6.** | WHILE improving = TRUE |
| **7.** | SET improving = FALSE |
| **8.** |     - Previous solution ← Current solution |
| **9.** |     - Get maximum value for neighbour |
| **10.** | Repeat FOR each route I in 'noRoutes' |

```
11.         -   Current solution ← Previous solution
12.         -   noCust1 ← number of customers in route ‘noRoutes’
13.     Repeat FOR each customer J in ‘noCust1’ in route ‘noRoutes’
14.         -   SET emptyRoute = FALSE
15.         -   currCust1 ← current customer
16.     Repeat FOR each route K in ‘noRoutes’
17.         -   Current solution ← Previous solution
18.         -   IF number of customers <= 2 (only depot) THEN emptyRoute = TRUE
19.         -   END of IF 18.
20.         -   currVehicle ←type of current vehicle in route K
21.         -   currCapacity ← capacity of vehicle ‘currVehicle’
22.         -   IF delivery at ‘currCust1’ <= ‘currCapacity’ THEN
23.         -   IF  I != K THEN
24.         -   INSERT ‘currCust1’ to route K
25.         -   REMOVE customer J from route K
26.         -   CALL optimizeTwoRoute(CurSol_, I, K)
27.         -   CALL adjustVehicle(CurSol_)
28.         -   CALL optimizeTwoRoute(CurSol_, I, K)
29.         -   CurSolVal_ ← cost of solution ‘ CurSol_’
30.         -   IF CALL  isFeasiblePD(CurSol_) = TRUE THEN
31.         -   PASS feasible_ ← TRUE
32.         -   ELSE PASS feasible_ ← FALSE
33.         -   Apply infeasible penalty to ‘CurSolVal_’
34.         -   END of IF 30.
35.         -   IF CurSolVal_ < BestNeighborVal THEN
36.         -   BestNeighbor_ ← CurSol_
37.         -   BestNeighborVal_ ← CurSolVal_
38.         -   END of IF 35.
39.         -   END of IF 23.
40.         -   END of IF 22.
41.         -   CurSol_ ←PrevSol_
42.         -   CurSolVal_ ← PrevSolVal_
43.         -   noCust2 ← number of customer in route K
```

```
44.      -  Repeat FOR customer L in ‘noCust2’ (start from 2)
45.      -  IF (K > I && !emptyRoute) THEN
46.      -  CurSol_ ← PrevSol_
47.      -  CurSolVal_ ← PrevSolVal_
48.      -  currCust1 ← customer J
49.      -  currCust2 ← customer L
50.      -  SWAP ‘currCust1’ with ‘currCust2’ in route I and K
51.      -  CALL adjustVehicle(CurSol_)
52.      -  CALL optimizeTwoRoute(CurSol_, I, K)
53.      -  CurSolVal_ ← cost of ‘CurSol_’
54.      -  IF CALL  isFeasiblePD(CurSol_) = TRUE THEN
55.      -  PASS feasible_ ← TRUE
56.      -  ELSE PASS feasible_ ← FALSE
57.      -  Apply infeasible penalty to ‘CurSolVal_’
58.      -  END of IF 54.
59.      -  IF CurSolVal_ < BestNeighborVal THEN
60.      -  BestNeighbor_ ← CurSol_
61.      -  BestNeighborVal_ ← CurSolVal_
62.      -  END of IF 59, 45.
63.      -  END of FOR loop 44, 16, 13, 10.
64. CurSol_ ← BestNeighbor_
65. CurSolVal_ ← BestNeighborVal_
66. IF BestNeighborVal_ < BestSolVal_ THEN
67.      -  improving ← TRUE
68.      -  IF CALL isFasible(BestNeighbor_) ← TRUE THEN
69.      -  BestSol_ ← BestNeighbor_
70.      -  BestSolVal_ ← BestNeighborVal_
71.      -  “Best solution found”
72.      -  END of IF 68, 66.
73. END WHILE loop 6.
74. Apply infeasible penalty to ‘CurSolVal_’
75. STOP clock and Calculate time consumed.
76. EXIT.
```

*Figure 13 Local Search procedure*

To make sure that each route has a close to optimal performance, a separate Local Search is performed on each route affected by a move. This search is described by the procedures OptimizeTwoRoutes, and OptimizeOneRoute in Figure 14. This separate Local Search uses the 2-opt neighbourhood (Flood 1956),which changes the position of two nodes in a single route and reverses the routing of the nodes between them.

**Procedure: optimizeTwoRoute (Solution &SOL, route1, route2)**

```
1.  size1 ← number of customers in ‘route1’
2.  size2 ← number of customers in ‘route2’
3.  avector ← all customers I from ‘size1’
4.  avector ← CALL optimizeOneRoute(avector)
5.  REPLACE customers in ‘route1’
6.  bvector ← all customers I from ‘size2’
7.  bvector ← CALL optimizeOneRoute(bvector)
8.  REPLACE customers in ‘route2’
9.  CLEAR ‘avector’ and ‘bvector’
```

**Procedure: optimizeOneRoute (vector<int> vec)**

```
1.  SET improving = TRUE
2.  orgvector = vec
3.  prevector = vec
4.  orgval = route cost of ‘vec’
5.  preval = orgval
6.  bestvector = vec
7.  bestval = orgval
8.  rSize ← number of nodes(customers) in vector ‘vec’
9.  WHILE improving == TRUE
10.     -   improving ←FALSE
11.     -   Repeat FOR customer I in ‘rSize’-1 (start from node 2)
12.     -   Repeat FOR customer J in ‘rSize’ (next to customer I)
13.     -   tempvector ← REVERT order of all customers from I to J
14.     -   tempval ← route cost of ‘tempvector’
15.     -   IF tempval < bestval THEN
16.     -   improving ← TRUE
```

```
17.         -   bestvector ← tempvector
18.         -   bestval ← tempval
19.         -   END of IF 15.
20.         -   END of FOR loop 12.
21.         -   END of FOR loop 11.
22.   prevector ← bestvector
23.   preval ← bestval
24.   END of WHILE loop 9.
25.   CLEAR prevector, tempvector and orgvector
26.   RETURN bestvector
```

Figure 14 Procedure for optimizing single routes using the 2-opt neighbourhood

### 5.2.3 MODIFY

To avoid the same local optima, the MODIFY procedure should perform perturbations on the solution $s*$. In our implementation of the search, this is done in a simple way, by creating a new solution by random and starting the Local Search from that spot. During the local search the MODIFY procedure takes place as the part of procedure 'runSolver()' in **Error! Reference source not found.**. The modifications could be performed on single route by adding new customers, this results into a neighbourhood for local search. Another way to implement the modifications is by swapping customers in between two different routes, which is rather a neighbourhood for local search.

### 5.2.4 ACCEPTANCE_CRITERION

To validate whether the new solution $s*'$ obtained by LOCAL_SEARCH is better than current best solution $s*$, the ACCEPTANCE_CRITERION is implemented. In this procedure, the new obtained solution $s*'$ value is compared with the previous best solution $s*$ value. If the new solution $s*'$ value is better than $s*$, it is stored as the best solution to our problem.  In our FSMVRPPD the objective is to minimize the total cost, so the acceptance test is represented below (Lourenço, Martin, and Stützle 2010):

$$ACCEPTANCE\_CRITERION(s^*, s^{*\prime}) = \begin{cases} s^{*\prime} & if\ C(s^{*\prime}) < C(s^*) \\ s^* & otherwise \end{cases}$$

### 5.2.5 ILS Algorithm Steps

This section states the abstract from the ILS algorithm process for the FSMVRPPD. The flow of the algorithm is briefed as below:

- A random seed value is passed to get a random starting point
- The problem instance is loaded and read
- The random initial solution $s_i$ is generated by using init() in **Error! Reference source not found.**
- The initial solution is set as the current solution i.e. $s* = s_i$
- WHILE $s*$ is getting improved, REPEAT local search in **Error! Reference source not found.**
- Perform modifications to get the neighbourhoods (i.e. routes)
- Implement separate Local Search to optimize the routes as in **Error! Reference source not found.**
- $s*'$ = solution after route optimization
- IF $s*' < s*$ and $s*'$ is Feasible THEN (if improvement found in solution)
- Set $s* = s*'$ and go to WHILE
- ELSE $s*$ is the best solution so far
- End of WHILE
- Print solution and EXIT.

# 6.0 Computations

This chapter shows the information on the results obtained from the mathematical model and the Iterated Local Search algorithm for the FSMVRPPD.

## *6.1 Test Instances*

20 standard FSMVRP test instances shown in Table 4 in the Appendix are modified to include pickup demand. The pickup demands are created by considering the original delivery demands and multiplying the delivery demands alternately by 0.8 and 1.2. I.e. with an original delivery demand of 10 the pickup demand will be either 8 or 12. If the index number of node is odd then the original demand will be multiplied by 0.8 and otherwise by 1.2. These modifications has earlier been implemented for the VRPPD by (Hoff et al. 2009). The modified test instances have been solved by the ILS algorithm and compared to

solutions to the similar instances without pickup demand (Pasha, Hoff, and Løkketangen 2013).

For the mathematical model for the FSMVRPPD some very small self-defined instances have been used, since solving large instances to optimality in reasonable time is not possible using CPLEX.

## 6.2 *Experimental Results*

### 6.2.1 Mathematical Model: Results

To validate the proposed mathematical model for FSMVRPPD presented in Chapter 4.0, some small data instances were created. The formulation solved the smallest instances to optimality and found the minimum cost in short time, but when trying to extend the data sets by including more customers, the CPU execution time increased exponentially.

| D_scen | #C | #vT | CPU | #R | #vTu | MIP | BnB |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 3 | 0.202 | 2 | 2 | 4781 | 173 |
| 2 | 7 | 3 | 0.968 | 3 | 1 | 28167 | 989 |
| 3 | 10 | 4 | 428.379 | 5 | 1 | 6611600 | 113537 |
| 4 | 10 | 3 | 681.562 | 5 | 2 | 12495963 | 279649 |
| 5 | 10 | 4 | 4108.47 | 3 | 2 | 67238267 | 1162535 |

*Table 1 AMPL results for the FSMVRPPD*

Table 1 shows the optimal results obtained from AMPL model for FSMVRPPD by using CPLEX solver. In the first row of the Table 1, the reference headers to the corresponding columns are presented. The header D_scen denotes the data scenario index number #C is the total number of customers in the instance and #vT denotes the total number of vehicle types in the instance. CPU shows the time (in seconds) used to by the computer to produce the optimal result. #R is the total number of routes and #vTu the number of vehicle types used in the optimal solution. MIP is the total number of MIP simplex iterations performed and BnB the total number of Branch and Bound nodes occurred while solving the AMPL model.

By a glance at Table 1 one can easily understand how complex the FSMVRPPD is. A slight increase in the number of nodes makes the CPU time explode. Here, trying to solve FSMVRPPD instances with 10 customers and 3 and 4 types of vehicle was very time consuming. The variance in both demands also effects the execution time. The instances with a low variance (3) in both types of demands used very short time to produce the optimal results compared to the instance with a high variance (5) even if the number of customers and vehicle types were the same.

### 6.2.2 ILS Heuristic: Results

The ILS solver is coded in C++ by using Microsoft Visual Studio 2012 running on Intel ® Core™ m3-6Y30 (4 CPUs) machine with 0.90 GHz (turbo boost up to 1.5 GHz) processor, 8 GB of RAM and Microsoft Windows 10 operating system. The solver was also executed on the same machine to obtain the results.

For all the 20 benchmark instances the local search were run 20 times with different seed values. The seed value is the input for the random generator, which makes sure that the algorithm achieves a different initial solution for every ILS run. The different initial solutions mean that for all executions of the local search, a different area of the solution space will be explored.

During these 20 different runs, most initial solution constructed were feasible, but sometimes the solver generated an infeasible solution. A huge value of 1,000,000 was added to the objective value on the infeasible solutions to make sure that any feasible solution would be preferred during the search. If the search did not find any feasible solution before the local optimum, the solution was discarded.

The results for the FSMVRPPD were obtained by choosing the best result from all 20 runs of the program. The results are presented in Table 2, and a description of the column's headers is as follows:

- Instance : states the name of the standard benchmark instance (originally taken from Golden et al. (1984)).
- FSMVRPPD ILS: states the best obtained solution from our algorithm for the FSMVRPPD version of the instance.
- Iteration : the number of iterations in the local search which provides the best found solution
- Search time : total time (in seconds) taken by the program to get the best result

| Instance | FSMVRPPD ILS | Iteration | Search time |
|---|---|---|---|
| Golden01 | 662 | 6 | 0 |
| Golden02 | 778 | 5 | 0 |
| Golden03 | 1152 | 6 | 0 |
| Golden04 | 7446 | 7 | 0 |
| Golden05 | 1214 | 7 | 0 |
| Golden06 | 7509 | 10 | 0 |
| Golden07 | 8404 | 13 | 0 |
| Golden08 | 3272 | 19 | 3 |
| Golden09 | 3093 | 10 | 0 |
| Golden10 | 3447 | 9 | 0 |
| Golden11 | 5477 | 14 | 2 |
| Golden12 | 4689 | 11 | 0 |
| Golden13 | 3482 | 4 | 1 |
| Golden14 | 10483 | 7 | 3 |
| Golden15 | 3215 | 11 | 4 |
| Golden16 | 3293 | 23 | 3 |
| Golden17 | 2886 | 13 | 7 |
| Golden18 | 3595 | 16 | 19 |
| Golden19 | 10502 | 17 | 79 |
| Golden20 | 5239 | 27 | 33 |

*Table 2 Results from the ILS solver for the FSMVRPPD*

In the analysis of the results from the Table 2, it can be seen that the solver can generate a good solution for the largest instance with the dimension of 101 nodes in 79 seconds, which is rather reasonable time to solve large instances. Most of the instances with the dimension up to 31 nodes were solved in less than one second for each, except instance Golden08 and Golden11, which took 3 seconds and 2 seconds respectively. The instances with the dimension of 51 nodes and more used minimum search time of 1 second for Golden13 and maximum 79 seconds for one of the largest instance Golden19. The variance of the searching time depends upon the structure of the problem instance. A Local Search on larger routes takes more time than on smaller routes. By considering the two largest instances Golden19 and Golden20, this situation can be understood. Both instance are of dimension of 101 nodes, 3 vehicle types and same demand at the customers. But the capacities of the vehicles in Golden19 are much higher than in Golden20, which lead to larger routes than in the Golden20 test instance.

The number of iterations performed to find the best solution varies for all the instances, and is very dependent on the initial solution. In the Table 2, we can observe that situation. Some of the small instances used more iterations than the larger instances. For example, instance Golden08 with a dimension of 31 nodes used 19 iterations to find the local optimum, while the larger instance Golden19 with 101 nodes used only 17 iterations.

#### 6.2.2.1 Competitiveness of the results obtained from FSMVRPPD algorithm

To observe the competitiveness of the solution after including pickup demand, the results from the FSMVRPPD algorithm were compared with the output from the ILS algorithm for the FSMVRP (without pickup demand) and the results for the FSMVRP from the previous literature. By following the same procedure as in the original FSMVRPPD, the FSMVRP program was also run with 20 different seed values on the same benchmark instances as used in the FSMVRPPD.

| Instance | Best FSMVRP | Br | PHB | FSMVRPPD ILS | FSMVRP-ILS |
|---|---|---|---|---|---|
| Golden01 | 602 | 602 | 602 | 662 | 602 |
| Golden02 | 722 | 722 | 722 | 778 | 724 |
| Golden03 | 961.03 | 961.03 | 971.87 | 1152 | 1019 |
| Golden04 | 6437.33 | 6437.33 | 6437.33 | 7446 | 6446 |
| Golden05 | 1007.05 | 1007.05 | 1008.59 | 1214 | 1108 |
| Golden06 | 6516.47 | 6516.47 | 6516.47 | 7509 | 6519 |
| Golden07 | 7273 | 7273 | 7295 | 8404 | 7542 |
| Golden08 | 2346 | 2347 | 2347 | 3272 | 2622 |
| Golden09 | 2209 | 2209 | 2209 | 3093 | 2410 |
| Golden10 | 2355 | 2355 | 2358 | 3447 | 2459 |
| Golden11 | 4755 | 4755 | 4755 | 5477 | 4901 |
| Golden12 | 4080 | 4080 | 4096 | 4689 | 4309 |
| Golden13 | 2406.36 | 2406.36 | 2468.08 | 3482 | 2748 |
| Golden14 | 9119.03 | 9119.03 | 9154.64 | 10483 | 9124 |
| Golden15 | 2586.37 | 2586.37 | 2601.57 | 3215 | 2766 |
| Golden16 | 2720.43 | 2728.14 | 2783.88 | 3293 | 2903 |
| Golden17 | 1734.53 | 1734.53 | 1745.39 | 2886 | 2189 |
| Golden18 | 2369.65 | 2369.65 | 2428.54 | 3595 | 2845 |
| Golden19 | 8661.81 | 8661.81 | 8850.34 | 10502 | 8739 |
| Golden20 | 4032.81 | 4042.59 | 4137.07 | 5239 | 4336 |
| Average Deviation |  | 1.0003 | 1.0079 | 1.2674 | 1.0684 |

*Table 3 Comparison of results from the FSMVRP literature*

Although, a direct comparison of the results from the FSMVRPPD with the outcomes of the FSMVRP is unfair, because these two problems are different from each other. However, to validate that our implementation of the algorithm is competitive, the results found by other researchers for the FSMVRP on the same instances are presented here. Table 3 shows the results from previous research on the FSMVRP. The column headers of Table 3 are described below:

Br – results from (Brandão 2009) using Tabu Search

PHB – results from (Pasha, Hoff, and Løkketangen 2013)

FSMVRPPD-ILS – the best results from our algorithm on FSMVRPPD instances

FSMVRP-ILS – the best results from our algorithm on FSMVRP instances

The average deviation of 6.84% from the best know solution was achieved when the algorithm was run for the FSMVRP (without pickup demand). Although, a deviation of 6.84% from the best know results is not bad for the simple Local Search algorithm as ours. However, the recommendation is to use some structured or intelligent technique to generate initial solutions rather than the random ones.

By analysing the results from the FSMVRPPD method, the average deviation was 26.74% from the best know solution. Thus, by including a pickup demand of +/-20% on an original delivery routing and fleet composition problem, one can expect an increase of around 25% in the transportation cost.

In the wrap up of this section, it would be appropriate to say that one could achieve less searching time by constructing the initial solution in a more intelligent way. One approach could be to use the search history so that one can produce a better intelligent initial solution for the next run of the Local Search based on the solution found in the previous Local Search. While, the use of a greedy construction heuristic for developing the initial solution could decrease the improvement steps, as a result the Local Search would require less searching time. A way to improve the performance of the proposed ILS algorithm would be to apply small modifications to a local optimum found in one run and use the modified solution as a starting solution for the next run. That would probably be better than using random solutions both with respect to searching time and solution quality.

# 7.0 Conclusion and Further Research

Altogether, this thesis wraps up the study on a new variant of Vehicle Routing Problems by introducing simultaneous Pickups and Deliveries to the Fleet Size and Mix VRP (FSMVRPPD). There are loads of papers available Vehicle Routing Problems and its various variants. However, there was no literature found which can particularly point to the FSMVRPPD. The objective of the FSMVRPPD is to find the best fleet composition and routing of the vehicles in such a way that customer's demand can be fulfilled with minimum transportation cost. When constructing the routes, one need to hold focus on the vehicle load at each visit of a node to make sure that the vehicle's capacity is not exceeded. A mathematical formulation has been presented to define the FSMVRPPD in a more sophisticated way. Whereas, the achievement of optimal results for significant instances by using exact mathematical methods is very time-consuming, some small instances are solved to optimality. Thus, a simple heuristic approach has also been proposed by the use of Iterated Local Search (ILS) metaheuristic. The heuristic has been tested on standard FSMVRP instances and compared to previous research on that problem to show that it gave reasonable results. When running the ILS algorithm on instances for the FSMVRPPD where a pickup of +/-20% of the delivery demand has been included to corresponding the FSMVRP instances, slightly more than 26% increase in the total cost was observed. These results were the average over the whole instance set, but the variation between single instances is in the range of 7% to 66%.

In suggestion to get improved results, a more advanced version of the ILS could be implemented by utilizing the ideas of this method better. Improved solutions could probably be achieved if the initial solutions for Local Search are derived from previous search results, rather than generating random solution each time.

For future research, I would suggest to try solving the FSMVRPPD with other local search based metaheuristics, and more advanced hybrid metaheuristic approaches. Extensions to the FSMVRPPD could be explored by including other aspects from real world problems in the problem definition. The most realistic extension, in my opinion, could be by introducing Location Routing and Time Windows for the customers into the problem.

## References

Achuthan, N. R., L. Caccetta, and S. P. Hill. 1997. "On the vehicle routing problem." *Nonlinear Analysis: Theory, Methods & Applications* 30 (7):4277-4288. doi: http://dx.doi.org/10.1016/S0362-546X(97)00127-2.

Alegre, Jesús, Manuel Laguna, and Joaquín Pacheco. 2007. "Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts." *European Journal of Operational Research* 179 (3):736-746. doi: http://dx.doi.org/10.1016/j.ejor.2005.03.063.

Alfredo Tang Montané, Fermín, and Roberto Diéguez Galvão. 2006. "A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service." *Computers & Operations Research* 33 (3):595-619. doi: http://dx.doi.org/10.1016/j.cor.2004.07.009.

Archetti, Claudia, Martin W. P. Savelsbergh, and M. Grazia Speranza. 2008. "To split or not to split: That is the question." *Transportation Research Part E: Logistics and Transportation Review* 44 (1):114-123. doi: http://dx.doi.org/10.1016/j.tre.2006.04.003.

Archetti, Claudia, and Maria Grazia Speranza. 2008. "The Split Delivery Vehicle Routing Problem: A Survey." In *The Vehicle Routing Problem: Latest Advances and New Challenges*, edited by Bruce Golden, S. Raghavan and Edward Wasil, 103-122. Boston, MA: Springer US.

Baldacci, Roberto, Maria Battarra, and Daniele Vigo. 2009. "Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs." *Networks* 54 (4):178-189. doi: 10.1002/net.20331.

Belfiore, Patrícia, and Hugo T. Y. Yoshizaki. 2013. "Heuristic methods for the fleet size and mix vehicle routing problem with time windows and split deliveries." *Computers & Industrial Engineering* 64 (2):589-601. doi: http://dx.doi.org/10.1016/j.cie.2012.11.007.

Beltrami, E. J., and L. D. Bodin. 1974. "Networks and vehicle routing for municipal waste collection." *Networks* 4 (1):65-94. doi: 10.1002/net.3230040106.

Bertazzi, Luca, and M. Grazia Speranza. 2012. "Inventory routing problems: an introduction." *EURO Journal on Transportation and Logistics* 1 (4):307-326. doi: 10.1007/s13676-012-0016-7.

Boventer, Edwin von. 1961. "THE RELATIONSHIP BETWEEN TRANSPORTATION COSTS AND LOCATION RENT IN TRANSPORTATION PROBLEMS." *Journal of Regional Science* 3:27–40. doi: 10.1111/j.1467-9787.1961.tb01276.x.

Brandão, José. 2009. "A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem." *European Journal of Operational Research* 195 (3):716-728. doi: http://dx.doi.org/10.1016/j.ejor.2007.05.059.

Clarke, G., and J. W. Wright. 1964. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points." *Operations Research* 12 (4):568-581. doi: doi:10.1287/opre.12.4.568.

Cooke, Kenneth L., and Eric Halsey. 1966. "The shortest route through a network with time-dependent internodal transit times." *Journal of Mathematical Analysis and Applications* 14 (3):493-498. doi: http://dx.doi.org/10.1016/0022-247X(66)90009-6.

Cordeau, Jean-François, and Gilbert Laporte. 2007. "The dial-a-ride problem: models and algorithms." *Annals of Operations Research* 153 (1):29-46. doi: 10.1007/s10479-007-0170-8.

Dantzig, G.B., and J.H. Ramser. 1959. "The Truck Dispatching Problem." *Management Science* 6 (1):80-91. doi: doi:10.1287/mnsc.6.1.80.

Dror, Moshe, and Pierre Trudeau. 1989. "Savings by Split Delivery Routing." *Transportation Science* 23 (2):141-145. doi: doi:10.1287/trsc.23.2.141.

Figliozzi, Miguel Andres. 2012. "The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics." *Transportation Research Part E: Logistics and Transportation Review* 48 (3):616-636. doi: http://dx.doi.org/10.1016/j.tre.2011.11.006.

Flood, M. M. . 1956. "The Traveling-Salesman Problem." *Operations Research* 4 (1):61-75. doi: doi:10.1287/opre.4.1.61.

Gheysens, F., B. Golden, and A. Assad. 1984. "A comparison of techniques for solving the fleet size and mix vehicle routing problem." *Operations-Research-Spektrum* 6 (4):207-216. doi: 10.1007/BF01720070.

Glover, Fred W., and Gary A. Kochenberger. 2003. *Handbook of Metaheuristics*, *International Series in Operations Research & Management Science*: Springer US.

Golden, Bruce, Arjang Assad, Larry Levy, and Filip Gheysens. 1984. "The fleet size and mix vehicle routing problem." *Computers & Operations Research* 11 (1):49-66. doi: http://dx.doi.org/10.1016/0305-0548(84)90007-8.

Hoff, Arild, Henrik Andersson, Marielle Christiansen, Geir Hasle, and Arne Løkketangen. 2010. "Industrial aspects and literature survey: Fleet composition and routing." *Computers & Operations Research* 37 (12):2041-2061. doi: http://dx.doi.org/10.1016/j.cor.2010.03.015.

Hoff, Arild, Irina Gribkovskaia, Gilbert Laporte, and Arne Løkketangen. 2009. "Lasso solution strategies for the vehicle routing problem with pickups and deliveries." *European Journal of Operational Research* 192 (3):755-766. doi: http://dx.doi.org/10.1016/j.ejor.2007.10.021.

Koç, Çağrı, Tolga Bektaş, Ola Jabali, and Gilbert Laporte. 2016. "The fleet size and mix location-routing problem with time windows: Formulations and a heuristic algorithm." *European Journal of Operational Research* 248 (1):33-51. doi: http://dx.doi.org/10.1016/j.ejor.2015.06.082.

Laporte, Gilbert, and Yves Nobert. 1987. "Exact Algorithms for the Vehicle Routing Problem*." In *North-Holland Mathematics Studies*, edited by Gilbert Laporte Michel Minoux Silvano Martello and Ribeiro Celso, 147-184. North-Holland.

Lecluyse, Christophe, Kenneth Sörensen, and Herbert Peremans. 2013. "A network-consistent time-dependent travel time layer for routing optimization problems." *European Journal of Operational Research* 226 (3):395-413. doi: http://dx.doi.org/10.1016/j.ejor.2012.11.043.

Lin, Canhong, K. L. Choy, G. T. S. Ho, S. H. Chung, and H. Y. Lam. 2014. "Survey of Green Vehicle Routing Problem: Past and future trends." *Expert Systems with Applications* 41 (4, Part 1):1118-1138. doi: http://dx.doi.org/10.1016/j.eswa.2013.07.107.

Liu, Shuguang, Weilai Huang, and Huiming Ma. 2009. "An effective genetic algorithm for the fleet size and mix vehicle routing problems." *Transportation Research Part E: Logistics and Transportation Review* 45 (3):434-445. doi: http://dx.doi.org/10.1016/j.tre.2008.10.003.

Lourenço, Helena R., Olivier C. Martin, and Thomas Stützle. 2010. "Iterated Local Search: Framework and Applications." In *Handbook of Metaheuristics*, edited by Michel Gendreau and Jean-Yves Potvin, 363-397. Boston, MA: Springer US.

NEO, Research Group. 2013. "VRP with Pick-up and Delivering." Last Modified 7 January, 2013. http://neo.lcc.uma.es/vrp/vrp-flavors/vrp-with-pick-up-and-delivering/.

Parragh, Sophie N., Karl F. Doerner, and Richard F. Hartl. 2008a. "A survey on pickup and delivery problems." *Journal für Betriebswirtschaft* 58 (2):81-117. doi: 10.1007/s11301-008-0036-4.

Parragh, Sophie N., Karl F. Doerner, and Richard F. Hartl. 2008b. "A survey on pickup and delivery problems." *Journal für Betriebswirtschaft* 58 (1):21-51. doi: 10.1007/s11301-008-0033-7.

Pasha, Urooj, Arild Hoff, and Arne Løkketangen. 2013. "The Shrinking and Expanding Heuristic for the Fleet Size and Mix Vehicle Routing Problem. ." *Communications – Scientific Letters of the University of Zilina* 1.

Polimeni, Antonio, and Antonino Vitetta. 2013. "Optimising Waiting at Nodes in Time-Dependent Networks: Cost Functions and Applications." *Journal of Optimization Theory and Applications* 156 (3):805-818. doi: 10.1007/s10957-012-0121-7.

Prodhon, Caroline, and Christian Prins. 2014. "A survey of recent research on location-routing problems." *European Journal of Operational Research* 238 (1):1-17. doi: http://dx.doi.org/10.1016/j.ejor.2014.01.005.

Repoussis, P. P., and C. D. Tarantilis. 2010. "Solving the Fleet Size and Mix Vehicle Routing Problem with Time Windows via Adaptive Memory Programming." *Transportation Research Part C: Emerging Technologies* 18 (5):695-712. doi: http://dx.doi.org/10.1016/j.trc.2009.08.004.

Ropke, Stefan. 2005. "Heuristic and exact algorithms for vehicle routing problems." Doctorate Ph.D., Computer Science, University of Copenhagen.

Solomon, Marius M. 1987. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints." *Operations Research* 35 (2):254-265. doi: 10.1287/opre.35.2.254.

Stutzle, Thomas G. 1998. "Local Search Algorithms for Combinatorial Problems - Analysis, Improvements, and New Applications." Doctrate of Science Phd, Computer Science, Technische Universität Darmstadt.

Subramanian, Anand, Puca Huachi Vaz Penna, Eduardo Uchoa, and Luiz Satoru Ochi. 2012. "A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem." *European Journal of Operational Research* 221 (2):285-295. doi: http://dx.doi.org/10.1016/j.ejor.2012.03.016.

Toth, P., and D. Vigo. 2002. *The Vehicle Routing Problem*, *Discrete Mathematics and Applications*: Society for Industrial and Applied Mathematics. doi:10.1137/1.9780898718515.

Wilson, H., and H. Weissberg. 1967. Advanced dial-a-ride algorithms research project: final report. Cambridge: Massachusetts Institute of Technology.

## Appendix

| Instance name | Dimension | Vehicle Types | Vehicle capacity | Cost of the vehicle |
|---|---|---|---|---|
| Golden01 | 13 | 3 | 15,35,60 | 20,50,100 |
| Golden02 | 13 | 3 | 30,40,110 | 60,90,300 |
| Golden03 | 21 | 5 | 20,30,40,70,120 | 20,35,50,120,225 |
| Golden04 | 21 | 3 | 60,80,150 | 1000,1500,3000 |
| Golden05 | 21 | 5 | 20,30,40,70,120 | 20,35,50,120,225 |
| Golden06 | 21 | 3 | 60,80,150 | 1000,1500,3000 |
| Golden07 | 31 | 5 | 40,100,140,200,300 | 150,500,800,1200,2000 |
| Golden08 | 31 | 4 | 10,50,150,400 | 15,50,200,600 |
| Golden09 | 31 | 5 | 40,100,140,200,300 | 30,100,160,240,400 |
| Golden10 | 31 | 4 | 40,100,140,200 | 30,100,160,240 |
| Golden11 | 31 | 4 | 30,80,200,350 | 60,200,700,1500 |
| Golden12 | 31 | 6 | 30,50,75,120,180,250 | 40,80,150,300,500,800 |
| Golden13 | 51 | 6 | 20,30,40,70,120,200 | 20,35,50,120,225,400 |
| Golden14 | 51 | 3 | 120,160,300 | 1000,1500,3500 |
| Golden15 | 51 | 3 | 50,100,160 | 100,250,450 |
| Golden16 | 51 | 3 | 40,80,140 | 100,200,400 |
| Golden17 | 76 | 4 | 50,120,200,350 | 25,80,150,320 |
| Golden18 | 76 | 6 | 20,50,100,150,250,400 | 10,35,100,180,400,800 |
| Golden19 | 101 | 3 | 100,200,300 | 500,1200,2100 |
| Golden20 | 101 | 3 | 60,140,200 | 100,300,500 |

*Table 4 The benchmark instances from (Golden et al. 1984)*