TRUCK ROUTING PROBLEM IN DISTRIBUTION
OF GASOLINE TO GAS STATIONS

A Thesis in
Industrial Engineering
by
Swagath Janakiraman

© 2010 Swagath Janakiraman

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2010
The thesis of Swagath Janakiraman was reviewed and approved* by the following:

A. Ravi Ravindran  
Professor of Industrial Engineering  
Thesis Advisor

Vittal Prabhu  
Professor of Industrial Engineering

Paul Griffin  
Professor of Industrial Engineering  
Head of the Department of Industrial Engineering

*Signatures are on file in the Graduate School
ABSTRACT

This thesis aims at finding a daily routing plan for a fleet of vehicles delivering gasoline to gas stations for an oil company, satisfying all the system constraints. Main constraints include vehicle capacity constraints, customer demand constraints and distance travel limit. The fleet of vehicles starts from an installation (single depot) and tries to cover the maximum number of deliveries on their way without violating the vehicle capacity constraints. All the customers are visited only once and the vehicles then return back to the installation after delivering the gasoline to the gas stations.

The mathematical problem is a multiple traveling salesman problem, which is computationally hard to solve. In this thesis, the problem is first formulated and solved as a Mixed Linear Integer Programming (MILP) problem. We show that this approach is not practical for larger problems, since the optimal algorithm can take more than 10 hours to solve even a small problem with just 10 customer locations. Hence, we develop an intelligent heuristics approach to solve large problems in few seconds. The heuristics method implemented in MATLAB, not only minimizes the total distance travelled by the fleet of vehicles, but also uses fewer number of trucks in comparison with the optimal method to deliver the gasoline. We compare the solution effectiveness of the heuristics with the optimal solution obtained by MILP. We then apply the heuristic method to an actual case study involving gasoline delivery to 120 gas stations and discuss its solution effectiveness.
# Table of Contents

LIST OF FIGURES .......................................................................................................................... viii

LIST OF TABLES ............................................................................................................................. viii

Chapter - 1: INTRODUCTION ......................................................................................................... 1

1.1. Introduction ............................................................................................................................. 1

1.2. Description of the Current System .......................................................................................... 2

1.2.1. Control capital costs ........................................................................................................... 2

1.2.2. Increase efficiency ............................................................................................................. 3

1.2.3. Reduce labor costs .............................................................................................................. 3

1.3. Disadvantages of outsourcing ................................................................................................. 3

1.3.1. Loss of Managerial Control ................................................................................................. 4

1.3.2. Hidden Costs ....................................................................................................................... 4

1.3.3. Threat to Security and Confidentiality ............................................................................... 4

1.3.4. Quality Problems ................................................................................................................. 5

1.4. Generation of Orders from Gas stations ................................................................................. 5

1.4.1. Company Owned Company Operated Trucks (COCO) ....................................................... 6

1.4.2. Company owned Dealer operated trucks (CODO) ............................................................ 7

1.5. Problem Statement .................................................................................................................. 7

Chapter - 2: LITERATURE REVIEW ............................................................................................... 9

2.1. Integer Programming .............................................................................................................. 9

2.2. Set Covering and Set Partitioning Problems ......................................................................... 10

2.2.1. Set Covering Problem ....................................................................................................... 10

2.2.2. Set Partitioning Problem .................................................................................................. 11

2.3. Traveling Salesmen Problem (TSP) ...................................................................................... 12

2.4. Vehicle Routing Problem (VRP) ........................................................................................... 14

2.4.1. Capacitated Vehicle Routing Problem (CVRP) ............................................................... 16

2.4.2. Heuristics in VRP .............................................................................................................. 17

2.4.3. Survey Papers in VRP ....................................................................................................... 18

2.4.4. Branch and Bound methods in VRP ............................................................................... 18

2.4.5. Vehicle Routing Problem with Backhauls and Time Windows (VRPBTW) .................... 19

Chapter - 3: MODEL DEVELOPMENT ............................................................................................ 22
BIBLIOGRAPHY ................................................................................................................................. 67
Appendix A: Data for the case study .......................................................................................... 72
Appendix B: Heuristic Algorithm in MATLAB for Example 4.3 ............................................. 75
Appendix C: Heuristic Algorithm in MATLAB for the Case study ........................................... 82
LIST OF FIGURES

Figure 1.1: A sample vehicle routing problem.................................................................1

Figure 3.1: Vehicle routing problem..............................................................................23

Figure 3.2: Problem Size (Variables and Constraints) Vs. No. of gas stations..............38

Figure 3.3: Computational time in LINGO......................................................................39

Figure 4.1: Comparison of distance travelled.................................................................55

Figure 4.2: Comparison of truck allotment.....................................................................55

Figure 4.3: Analysis of Optimal Vs. Heuristic approach...............................................56

Figure 4.4: Location points of all gas stations ...............................................................60
LIST OF TABLES

Table 3.1: Location (x,y) and Demand (kl) of each gas station........................................30

Table 3.2: Distance matrix.................................................................................................31

Table 3.3: Optimal routing plan............................................................................................37

Table 3.4: Scalability of the MILP model...............................................................................37

Table 3.5: Computational time...............................................................................................38

Table 4.1: Demand Locations.................................................................................................43

Table 4.2: Feasible combinations............................................................................................44

Table 4.3: Location (x,y) and Demand (kl) of each gas station..............................................47

Table 4.4: Distance matrix.....................................................................................................48

Table 4.5: Optimal Allotment.................................................................................................50

Table 4.6: Comparitive results of optimal and heuristic approach.........................................52

Table 4.7: Average Error results............................................................................................54

Table 4.8: Fill capacity............................................................................................................61

Table 4.9: Optimal Allotment.................................................................................................63
To

Mom, Dad & Smrithi
Chapter-1

INTRODUCTION

1.1. Introduction

The delivery of gasoline to gas stations from an installation, accounts for a significant part of the supply chain cost to an Oil company. Optimal planning of the distribution of trucks is an effective way to obtain substantial savings. The optimization model developed in this research generates a daily routing plan for a fleet of vehicles for Oil Company based on a delivery schedule to the gas stations. These vehicles depart from an installation and must visit a set of customers for delivery operations. The problem considers multiple capacities of vehicles associated with the operations. Thus, the problem consists of delivery to gas stations, transportation between gas stations and return to the installation within the time window.

The following figure shows a feasible routing plan to a small Vehicle Routing Problem with 4 vehicles, one depot and 14 gas stations. The depot is shown in the middle (in black), with the customers distributed around it (in blue).

![Traveling route of trucks](image)

**Figure 1.1:** A sample vehicle routing problem
The deliveries to be made to the gas stations are known and the model will be used on a daily basis to design the optimal route schedule. Each truck has a known capacity. Number of trucks to serve customers is also known. The objective of the problem is to plan the routes such that the overall distance covered by the trucks is minimized. The gas stations are geographically dispersed. The research problem is in fact a Multiple Traveling Salesman Problem, which has been shown to be computationally hard to solve.

1.2. Description of the Current System

Oil companies outsource the business of transporting gasoline from an oil installation to gas stations to a third party logistics (3PL) company. The benefits of outsourcing for the oil companies are described below:

1.2.1. Control capital costs

Cost-cutting may not be the only reason to outsource, but it's certainly a major factor. For oil companies, outsourcing converts fixed costs into variable costs, releases capital for investment elsewhere in their business, and allows them to avoid large expenditures in this phase of business.
1.2.2. Increase efficiency

Oil companies have to account for much higher research, development, marketing, and distribution expenses, all of which must be passed on to customers if they were to handle this business by themselves. A 3PL’s cost structure and economy of scale can give the firm an important competitive advantage in controlling distribution cost.

1.2.3. Reduce labor costs

Hiring and training truck drivers can be very expensive and temporary employees do not always live up to expectations. Outsourcing focuses human resources where they are needed most. Typically, the planning team in the installation accumulates all the incoming orders from the gas stations and sorts them out with respect to the requested time of delivery. All orders are then processed by the evening and are set for delivery the next day. 3PL Company receives the delivery order to the gas stations in the morning and they deliver it by first come first serve basis.

1.3. Disadvantages of outsourcing

Oil companies did not recognize the dis-advantages that arise from outsourcing the logistics business. They have started realizing them only after their sales volumes started going down after outsourcing.
The following are frequent **unforeseen problems** of outsourcing faced by the Oil Companies:

1.3.1. **Loss of Managerial Control**

Since a transport agency (3PL) performs the function of an entire logistics department, the management and control of that function have been turned over to the transport agency. The 3PL company will not be driven by the same standards that drive the oil company and its major concern will be to make a profit.

1.3.2. **Hidden Costs**

Oil companies sign a contract with an outsourcing company that will cover the details of the service that they will be providing. Anything not covered in the contract will be the basis for additional charges. Also, oil companies incur legal fees to review the contacts they sign. Generally, the transport agency will be the one to write the contract and the oil companies will be at a disadvantage when negotiations start.

1.3.3. **Threat to Security and Confidentiality**

The life-blood of any business is the information that is vital to its business. Since the outsourcing function involves sharing proprietary company data (e.g. volume of sales, location of gas stations, variety of products sold, etc.), this becomes an issue. Frequent evaluation of the outsourcing company becomes mandatory to ensure that data is protected. Also, the contract has a penalty clause if confidentiality is not maintained by the 3PL company.
1.3.4. Quality Problems

The outsourcing company will be motivated by profit. Since the contract will fix the price, the only way for them to increase profit have to be to decrease expenses. As long as they meet the conditions of the contract, oil companies have to pay. In addition, 3PL providers may not have the ability to rapidly respond to changes in the business environment. The contract will be very specific and Oil Companies will end up paying extra charges for any changes. This scenario is quite prevalent when there is more demand from the gas stations and more trips between installations to gas station are needed.

1.4. Generation of Orders from Gas stations

There are Automatic Tank Gauging machines installed in the gas stations. Whenever the gas in the underground tanks reach a level below the threshold, the owner of the gas station will be alerted through an alarm. The owner then places an order to the oil company through the back office system installed in the gas station.

There is a misconception by the gas station owners that the gas sold by the oil companies is polluted. Gas station owners think that small percentage of kerosene is added with petrol by these truck drivers en route to the gas station. Also, petroleum products expand/contract with respect to temperature difference. The owners think that the exact quantity of fuel is not being delivered to the gas station and these truck drivers are mishandling the shipments on their way.
Oil companies, in order to overcome this misconception, are developing stringent procedures in dealing with the truck drivers who they believe might be causing this problem. Some companies had planned to install GPS systems in the trucks to monitor the truck drivers en route from installation to gas station. They later realized that it was not cost effective to implement the GPS system. Rather, they have started investigating other cost effective methods to control the theft of oil and also ensure that quality products are delivered to the customers.

Another approach suggested to the oil companies was to buy their own trucks to transport petrol and diesel to the gas stations. This idea was validated and was found worthy of implementation. This will ensure that right quantity and quality of products are delivered to the gas stations. Under this system, oil companies operate their business in two different modes as given below:

1. Company owned Company operated trucks (COCO)
2. Company owned Dealer operated trucks (CODO)

1.4.1. **Company Owned Company Operated Trucks (COCO)**

In this operation, the truck is owned and managed by the oil company. This results in owning the trucks, employing the truck drivers and directing the travel route to the truck drivers by the company.
1.4.2. Company owned Dealer operated trucks (CODO)

In this operation, the truck is owned by the oil company but it is managed by a dealer (3PL Company). The dealer employs the truck drivers and chooses the travel route to the gas stations from the installation. The dealer gets paid for each trip made to the gas station. The reason for using CODO is to reduce labor costs for the oil company.

Since, the truck is owned by the oil companies in both types of business operation, gas station owners believe that right quantity and right quality of product is delivered to them. This is due to the fact that the truck is equipped with an advanced locking system, thereby preventing the truck drivers from stealing or polluting the petroleum products during transit.

The next concern for the oil companies is to improve customer responsiveness, thereby increasing the sales. This will occur only if the cycles of refilling the gas stations occur on time. This could only be achieved through optimal planning of travel routes of travel for these trucks from the installation to the gas stations.

1.5. Problem Statement

This thesis aims at finding a set of optimal routes for the fleet of vehicles to satisfy customer demands keeping in consideration all the system constraints. These constraints include – vehicle capacity constraints (different volume and different products), customer demand constraints, etc. The fleet of vehicles starts from an installation (single depot) and tries to cover the maximum deliveries on their way without violating the vehicle capacity constraints and then come back to
the depot. All the customers are visited only once and these vehicles return back to the installation after deliveries. Thus, the problem is a Multiple Traveling Salesman Problem, which can be solved as an integer programming problem.

Chapter 2 consists of literature review on topics relevant to the research work, such as traveling salesman, vehicle routing, heuristics in vehicle routing problem, set covering problem, etc. Chapter 3 presents the problem description and the model formulation based on the mixed integer programming approach. Chapter 4 illustrates the model with an example problem and analyzes its results. Chapter 5 presents the conclusions and the scope for future work.
Chapter–2

LITERATURE REVIEW

2.1. Integer Programming

An integer programming problem is a mathematical programming problem in which some or all of the variables are restricted to be integers.

Various applications of Integer Programming Problems are as follows:

1. Handling fixed cost
   Ex: Facility location, Supplier Selection problems.
2. Either Or Constraints
   Ex: Capacity decisions.
3. Piecewise linear Cost
   Ex: Quantity discount, Transportation, Inventory, Production.
4. Set Covering and Set Partitioning Problems
   Ex: Facility location, Distribution, Scheduling routes.
5. Traveling Salesman Problem
   Ex: DNA sequencing.
2.2. Set Covering and Set Partitioning Problems

Given a matrix $A$ of size $m$ rows and $n$ columns where

\[
A (m \times n) = \begin{bmatrix}
0 & \cdots & 1 \\
\vdots & \ddots & \vdots \\
1 & \cdots & 1
\end{bmatrix} = \{ a_{ij} \} \quad \text{(Equation 1.1)}
\]

such that $a_{ij} \in (0, 1)$

With the understanding that if $a_{ij} = 1$, we say Column $j$ “covers row $i$”

Else if $a_{ij} = 0$, it does not.

2.2.1. Set Covering Problem

The set covering problem is to find the least number of columns such that every row is “covered” by at least one column.

**Decision Variables:**

\[X_j = 1, \text{ if Column } j \text{ of matrix } A \text{ is selected}\]

\[X_j = 0, \text{ otherwise}\]

**Objective function:**

\[Min \ Z = \sum_{j=1}^{n} x_j\]

**Constraints:**

To cover row $i$ by at least one column, we write

\[\sum_{j=1}^{n} a_{ij} \cdot x_j \geq 1 \quad \text{for } i = 1, 2, \ldots, m\]
Warehouse location problem is an example of Set Covering Problem, where the columns represent warehouse locations and rows represent customer regions.

### 2.2.2. Set Partitioning Problem

The Set Partitioning Problem is an extension of the Set Covering Problem where every row is covered exactly by one column.

**Decision Variables:**

\[ X_j = 1, \text{ if Column } j \text{ of matrix } A \text{ is selected} \]

\[ X_j = 0, \text{ otherwise} \]

**Objective function:**

\[ \text{Min } Z = \sum_{j=1}^{n} X_j \]

**Constraints:**

To cover row \( i \) by exactly one column, we write

\[ \sum_{j=1}^{n} a_{ij} * x_j = 1 \quad \text{for } i = 1, 2, \ldots, m \]

Strictly we can’t have 2 rows covering a column.

Example: Vehicle routing problem
2.3. Traveling Salesmen Problem (TSP)

TSP is a well known and important combinatorial problem. The goal is to find the shortest tour that visits each city in a given list exactly once and then return to the starting city.

In the case of the traveling salesman problem, the mathematical structure is a graph, where each city is denoted by a point (or node) and lines are drawn connecting every two nodes (called edges). Associated with every edge is a distance (or cost). When the salesman can get from every city to every other city directly, then the graph is said to be complete. A round-trip of the cities corresponds to some subset of the edges, and is called a tour. The length of a tour is the sum of the lengths of the edges in the round-trip.

Formally, the TSP can be stated as follows:

The distances between $n$ cities are stored in a distance matrix $D$ with elements $d_{ij}$ where $d_{ij}$ represents the distance for traveling from city $i$ to $j$, for $i = \{1, 2..n\}$ and $j = \{1, 2.. n\}$ with the diagonal elements $d_{ii}$ are zero.

Let $K$ denotes any nonempty proper subset of the cities $1...n$.

Also, $X_{ij} = 1$, if the edge $i \rightarrow j$ is in the tour for $i \neq j$

$X_{ij} = 0$, otherwise

Objective function:

$$\text{Min} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \cdot X_{ij}$$
Subject to

\[
\sum_{j=1}^{n} X_{ij} = 1 \text{ for } i = \{1, 2 \ldots n\} \quad \text{(Equation 2.3.1.)}
\]

\[
\sum_{i=1}^{n} X_{ij} = 1 \text{ for } j = \{1, 2 \ldots n\} \quad \text{(Equation 2.3.2.)}
\]

\[
\sum_{i \in N} \sum_{j \in N} X_{ij} \leq |N| - 1 \text{ for } N \subset \{1, 2 \ldots n\} \quad \text{(Equation 2.3.3.)}
\]

Equations 2.3.1 and 2.3.2 ensure that every node must have only one edge pointing towards it and one pointing away from it. Equation 2.3.3 eliminates the formation of "sub tours", thereby not allowing the disjoint loops to occur. For example, for \(N = \{1, 2, 3\}\), \(X_{12} + X_{23} + X_{31} \leq 2\) avoids a sub tour.

For this minimization task, the tour length of \((n-1)!\) Permutation vectors have to be compared. This results in a problem which is very hard to solve and in fact known to be NP-complete.

However, solving TSPs is an important part of applications in many areas including vehicle routing, computer wiring, machine sequencing and scheduling (1976), frequency assignment in communication networks (1991), applications in statistical data analysis include ordering and clustering objects.
2.4. Vehicle Routing Problem (VRP)

Vehicle Routing refers to the distribution of goods between depots and customers.

Many real-life problems were found to be instances of the VRP, e.g., the delivery of newspapers to retailers, delivery of food and delivery of beverages to grocery stores and the collection of milk products from dairy farmers, delivery of express mails from customers, etc.

Certain inputs that are required to solve VRP problems are vehicles, depots, road network, costs and customer requirements.

Output that has to be satisfied in a VRP involves solving for set of routes where

- Customer requirements are to be fulfilled
- Operational constraints are to be satisfied
- Global transportation costs are to be minimized

The input parameters required about Vehicles in VRP are:

- Capacity.
- Types of goods.
- Fix costs associated for the use of vehicle.
- A-priori partition of customers.

The input parameters required about Road Network in VRP are:

- Travel costs and travel times on the arcs obtained by shortest paths.
- Directed or undirected complete graph.
The output parameters required about Operational Constraints in VRP are:

- Vehicle Capacity.
- Delivery or collection.
- Time windows.
- Working periods of vehicle drivers.

The output parameters required about Customers in VRP are:

- Vertices of the graph.
- Collection or delivery demands.
- Time windows for service.
- Service time.

The objective parameters in a VRP problem are:

- Minimization of global transportation cost (Variable + Fixed Costs).
- Minimization of the number of vehicles.
- Balancing of the routes.
- Minimization of penalties for un-served customers.

Classifications of VRP are as follows:

- Capacity (and Distance Constrained) VRP (CVRP and DCVRP).
- VRP with Time Windows (VRPTW).
- VRP with Backhauls (VRPB).
- VRP with Pickup and Delivery (VRPPD).
- Periodic VRP (PVRP).
- Multiple Depot VRP (MDVRP).
- Split Delivery VRP (SDVRP).
- VRP with Satellite Facilities (VRPSF).
- Site Dependent VRP.
- Open VRP.
- Stochastic VRP (SVRP).

2.4.1. Capacitated Vehicle Routing Problem (CVRP)

The classical CVRP is a hard combinatorial optimization problem, in which customers of known demand are supplied from a single depot with a fleet of \( V (>1) \) vehicles of fixed loading capacity and/or with traveling time (distance) constraint. The problem consists of designing a set of at most \( V \) delivery or collection routes such that

1. Each route starts and ends at the depot
2. Each customer is visited exactly once by exactly one vehicle
3. The total demands of customers in each route does not exceed vehicle capacity,
4. The total time (distance) of each route does not exceed the allowed traveling time (distance) of each vehicle, and
5. The total routing cost (time or distance) is minimized.
2.4.2. Heuristics in VRP

Several families of heuristics have been proposed for the VRP. They can be broadly classified into two main classes: classical heuristics, developed mostly between 1960 and 1990 and Meta-heuristics, whose growth occurred in the last decade as discussed by Laporte et al. (2000). Most standard construction and improvement heuristics belong to the first class. Construction heuristics build routes sequentially or in parallel, whereas improvement heuristics try to apply different notifications to the incumbent solution to improve the objective value. These methods explore a limited search space and produce relatively good quality solutions within modest computing times.

Meta-heuristics can be viewed as enhancements of classical heuristics whose emphasis is on performing a deep exploration of the most promising area of the search space, and where neighborhood moves are guided by supervisory control strategies to avoid getting trapped in local optima. The quality of the solutions produced by these methods is usually better than that obtained using classical heuristics, but at the expense of increased computing time—a typical tradeoff.

Among the classical heuristics stand the famous Clarke and Wright (1964) savings algorithms, sweep algorithms by Gillett and Miller (1974), two-phase heuristic by Bramel and Simchi-Levi (1995) and cluster-first, route-second algorithm by Fisher and Jayakumar (1981). A broad survey of classical and modern heuristics for the VRP is presented in Laporte et al. (2000). In recent years, attempts have been made to solve the vehicle routing problems using ant colony algorithms. Bullnheimer et al. (1999) proposed a hybrid Ant colony system with capacity utilization and saving heuristics.
2.4.3. Survey Papers in VRP

There are several survey papers on the subject of VRPs. A classification scheme was given in Desrochers et al. (1990). Laporte and Nobert (1987) presented an extensive survey that was entirely devoted to exact methods for the VRP, and they gave a complete and detailed analysis of the state of the art up to the late 1980s.

The set-partitioning (SP) formulation of the VRP was originally proposed by Balinski and Quandt (1964). A comprehensive review of the VRP can be found in Bodin et al. (1983) and Ball et al. (1995). Useful techniques for the general VRP are outlined in Golden and Assad (1988) and Aarts and Lenstra (1997). Reeves (1993) covers modern techniques such as simulated annealing, tabu search, and genetic algorithms.

Voudouris and Tsang (1999) develop the GLS (guided local search) meta-heuristic and apply it to the traveling salesman problem. The GLS penalizes solution features (e.g., active arcs) during each iteration based on the value of a utility function. The penalty acts as a disturbance to an augmented objective function which is adjusted during each iteration. This reduces the chance that the solution procedure will get stuck in a local optimum.

2.4.4. Branch and Bound methods in VRP

The branch and bound method has been used in recent decades to solve the Capacitated vehicle routing problem (CVRP) and its main variants. In their extensive survey devoted to exact methods, Laporte and Nobert (1987) gave a complete and detailed analysis of the branch-and-
bound algorithms. When the explicit distinction between Symmetric Capacitated VRP and Asymmetric Capacitated VRP is not needed, we simply use CVRP. CVRP is an extension of well known Traveling Salesman Problem (TSP), with minimum cost visiting a giving set of points exactly once. Therefore, many exact approaches for the CVRP were inherited from the extensive and successful work done for the exact solution of TSP. Until late 1980s, the most effective exact approaches for the CVRP were mainly branch-and-bound algorithms, which used basic combinatorial relaxations, such as the Assignment Problem (AP) and the Shortest Spanning Tree (SST) problem.

Several relaxations based on spanning trees were proposed for SCVRP by extending the well-known 1-tree relaxation proposed by Held and Karp (1971) for the TSP. The earliest branch – and – bound algorithm based on such relaxations, which proved to be able to solve small size instances, was proposed by Christofides and Mingozzi (1989).

2.4.5. Vehicle Routing Problem with Backhauls and Time Windows (VRPBTW)

Several papers deal specifically with the VRPBTW. Gelinas et al. (1995) propose a new branching strategy for branch-and-bound approaches based on column generation. This algorithm finds optimal solutions to different test problems with up to 100 customers. Kontoravdis and Bard (1995) describe a greedy randomized adaptive search procedure (GRASP) for VRPTW. This algorithm is also capable of solving VRPBTW without customer precedence. The authors report experimental results for problems with customers distributed in clusters. Cheung and Hang (2003) develop label matching algorithms for solving
the VRPBTW. Their heuristic approach can handle the addition of complex real-world constraints, such as vehicles of different capacities and penalties for vehicles that arrive early.

The Vehicle Routing Problem with Backhauls and Time Windows (VRPBTW) can be stated as follows: A set of customers with deterministic demand, deterministic time windows, and certain types of service requirements (pick up and/or delivery) must be served by a homogeneous fleet of vehicles with fixed capacity starting from and ending at a central depot, which also has a certain time horizon. For each service requirement (pickup or delivery), each customer must be assigned exactly once to a vehicle. The customers are classified into two groups: line haul customers whose demand needs to be delivered and backhaul customers whose demand needs to be picked up (if a customer requires both pickup and delivery, it is modeled as two separate customers). The VRPBTW has two objectives: minimize the number of routes and minimize the sum total distance of all the routes. The VRPBTW can be classified into two cases. In the VRPBTW with customer precedence, all of the line haul customers must be served before any backhaul customers in each route. In the VRPBTW without customer precedence, line haul and backhaul customers can be interspersed on a given route.

Several routing problems with pick-up and delivery service are reported in the literature. Savelsbergh and Sol (1995) describe a general routing problem in the category, which they call the General pick up and delivery problem, so that the majority of pick-up and delivery problems can be stated as particular cases of this general problem.
A classical problem discussed in the Literature, the Dial-a-Ride problem consists of picking up clients in pre-specified locations and transporting them to known delivery locations, using vehicles based in a given depot. Each pick-up location is associated with a delivery location, forming pairs of locations, with priority given to the pick-up activity. The problem is subject to restrictions on vehicle capacity, maximum travel times for clients and time windows. It is sought to minimize the following objectives in a hierarchical fashion:

(1) Number of vehicles
(2) Total distance travelled
(3) The difference between effective pick-up and delivery times and those desired by clients.

Exact dynamic programming algorithms were developed by Psaraftis (1980, 1983) to solve the static and dynamic versions of the Dial-a-Ride Problem, as well as a variant of the problem with time windows. Desrosiers at al. (1986) proposed a more efficient algorithm for the time windows variant, and Psaraftis (1983a), Sexton and Bodin (1985) and Sexton and Choi (1986) proposed heuristic methods for this problem.

Another routing problem in this category is the problem with backhauls, where deliveries are given priority and all deliveries must be made before any pick-up can be affected. In the case of a single vehicle, the TSP with backhauls is defined. Gendreau et.al (1996) developed several two-phase heuristics for this problem.
Chapter -3

MODEL DEVELOPMENT

3.1. Assumptions

- Customer demands are known—
  Orders are placed immediately when the level of fuel in the underground tank falls below the threshold limit through Automatic Tank Gauging System. Discrete orders are then placed in such a way to refill the underground tank to its full capacity. Thus, the customer demand is known in advance and the planning team accumulates all the orders for the day and processes them for delivery the next day.

- Maximum carrying capacity of the truck is known in advance.

- There is no breakdown of the truck –
  If there exists a problem of breakdown of a truck, the market in real world gives alternate options for renting a truck to deliver the product.

- All trucks start from the depot and come back to the depot.

- There is no stopping of truck in between other than at the gas stations.

- There is no waiting time at the gas station. (Truck stays for a very small time).

- Discrete orders are placed by each customer in line with the maximum capacity of their underground tanks.

- Total distance that a truck is allowed to cover/trip = 240 kms(8 hrs of shift * Average traveling speed of 30kms/hr).

- Every truck can make at most one trip during the shift.
3.2. Model Formulation

The model is formulated as a Mixed Integer Linear Program (MILP). Detailed mathematical formulation of the problem is given below:

3.2.1. Decision Variables

Input data-

n – Number of gas stations

\( D_i \) – Demand at gas station \( i \) (in litres) for \( \{i = 1, 2…n\} \)

\( d_{ij} \) – Distance between node \( i \) and node \( j \) for \( \{i = 0, 1, 2…n\} \) & \( \{j = 0, 1, 2…n\} \) where Node ‘0’ represents the depot and the other nodes represent the gas stations.

C – Capacity of truck (in kilo-litres)

m – Maximum number of trucks in the fleet

Figure 3.1. shows a small Vehicle Routing Problem with 4 vehicles, one depot and 14 gas stations. The depot is shown in the middle (in black), with the customers distributed around it (in blue).

![Vehicle routing problem](image)

**Figure 3.1:** Vehicle routing problem
A **tour** is defined when a truck starts from the depot, visits one or more gas stations and returns to the depot. For example in Figure 3.1 four tours are illustrated.

Tour 1: Depot $\Rightarrow$ Gas station 1 $\Rightarrow$ Gas station 2 $\Rightarrow$ Gas station 3 $\Rightarrow$ Depot.

Tour 2: Depot $\Rightarrow$ Gas station 4 $\Rightarrow$ Gas station 5 $\Rightarrow$ Gas station 6 $\Rightarrow$ Gas station 7 $\Rightarrow$ Depot.

Tour 3: Depot $\Rightarrow$ Gas station 8 $\Rightarrow$ Gas station 9 $\Rightarrow$ Gas station 10 $\Rightarrow$ Depot.

Tour 4: Depot $\Rightarrow$ Gas station 11 $\Rightarrow$ Gas station 12 $\Rightarrow$ Gas station 13 $\Rightarrow$ Gas station 14 $\Rightarrow$ Depot.

A **sub-tour** refers to formation of a tour for a truck starting from any of the gas-stations and visiting one or more gas stations and returning back to the starting node.

For example, in Figure 3.1., if a truck starts from Gas station-1, visits Gas stations 2 and 3 and returns back to Gas station 1, we call that a sub-tour. A sub-tour is not a feasible solution to the MILP problem. Hence, to solve the vehicle routing problem, we use special constraints and continuous variables ($T_{ik}$) to eliminate the sub-tours.

**Continuous Variables:**

$T_{ik}$ – Non-negative continuous variables defined for each node $\{i = 1, 2...n\}$ to eliminate sub-tours for truck $k$, for $k = \{1, 2...m\}$.

Note: Depot is identified as node 0.

**Binary Variables** –

$X_{ijk} = 1$, if truck $k$ travels from node $i$ to node $j$ ($i \neq j$)

$0$, otherwise; $i$ and $j = 0, 1, 2...n$; $k = 1, 2...m$

$S_k = 1$, if truck $k$ is used for delivering gasoline

$0$, otherwise; $k = 1, 2...m$
3.2.2. Objective Function

The objective of this multiple travelling salesman problem is to minimize the total distance travelled by the fleet of trucks without violating the truck capacities and satisfying all the demands at the gas stations.

\[
\text{Minimize } Z = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} d_{ij} \cdot X_{ijk} \text{ for } i \neq j
\]

3.2.3. Constraints

- **Truck Capacity Constraints** – Each truck has a certain capacity associated with it and hence cannot be overloaded. Hence, the total fuel carried by a truck during a trip cannot exceed its capacity.

\[
\sum_{i=0}^{n} \sum_{j=1}^{n} D_j \cdot X_{ijk} \leq C \text{ for } \{k = 1, 2 \ldots m \} \text{ and } i \neq j
\]

- **Delivery travel-limit constraints** – Distance limits imply that a particular delivery should be made within a specified time range. For this, we have assumed that the truck speed is 30 kms/hr. Since the truck can operate for only 8 hrs at one stretch which restricts its total travel to 240 kms for one trip.

\[
\sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} \cdot X_{ijk} \leq 240 \text{ for } k = \{1, \ldots, m \} \text{ and } i \neq j
\]
• When a truck is used for delivering gasoline, it should start from the depot and come back to the depot. All these delivery trucks are needed to start from the depot since the fleet of the trucks is stationed at the depot. Hence the following constraints have to be satisfied for all trucks.

\[
\sum_{j=1}^{n} X_{0jk} = S_k \text{ for } \{k = 1, 2 \ldots m\}
\]

Note:

➢ When truck \(k\) is used (\(S_k = 1\)), the truck has to leave node 0 and go to one of the nodes \(j = 1, 2 \ldots, n\).

➢ When truck \(k\) is not used (\(S_k = 0\)), all the \(X_{0jk} = 0\) for \(\{j = 1, 2 \ldots n\}\).

Also, all the trucks are needed to report back to the depot after satisfying the deliveries at customer’s end.

\[
\sum_{p=1}^{n} X_{p0k} = S_k \text{ for } \{k = 1, 2 \ldots m\}
\]

• **Truck continuity constraints** – Truck that reaches a customer must leave the same customer. This ensures the continuity of the truck that reaches a customer.

\[
\sum_{j=0}^{n} X_{jik} = \sum_{j=0}^{n} X_{ijk} \text{ for } \{i = 0, 1, 2 \ldots n\}, \{k = 1, 2 \ldots m\} \text{ and } j \neq i
\]
**Sub tour elimination constraints** – To eliminate the formation of sub tours, we introduce the following constraints:

\[ T_{ik} - T_{jk} + n \times X_{ijk} \leq n - 1 \text{ for } 1 \leq i \neq j \leq n \text{ and } \{k = 1, 2 \ldots m\} \]

\[ \Rightarrow T_{jk} \geq (T_{ik} + 1) - n \times (1 - X_{ijk}) \text{ for } 1 \leq i \neq j \leq n \text{ and } \{k = 1, 2 \ldots m\} \]

**Case 1:**

When \( X_{ijk} = 1 \), \( T_{jk} \geq T_{ik} + 1 \)

**Case 2:**

When \( X_{ijk} = 0 \), \( T_{jk} \geq (T_{ik} + 1) - n \)

For example, consider a sub tour for a truck starting from Gas station-1, visiting Gas station-2 and then returns to Gas station-1. The following constraints show how the formations of sub-tour between nodes 1 and 2 are eliminated.

Initially, when the truck travels from node 1 to node 2,

\[ X_{12k} = 1 \Rightarrow T_{2k} \geq T_{1k} + 1 \]

(Equation 3.1.)

But also to complete the sub tour, the truck returns to node 1 from node 2,

\[ X_{21k} = 1 \Rightarrow T_{1k} \geq T_{2k} + 1 \]

(Equation 3.2.)

Since, Equations 3.1. and 3.2. contradict each other, the formation of sub tour is eliminated.
Consider another example for a sub tour starting from Gas station-1, visits Gas station-2 and Gas station-3 and then returns to Gas station-1. The following constraints show how the formation of sub-tour between nodes 1, 2 and 3 are eliminated.

Initially, when the truck travels from node 1 to node 2,

\[ X_{12k} = 1 \Rightarrow T_{2k} \geq T_{1k} + 1 \quad \text{(Equation 3.3.)} \]

Then, the truck travels from node 2 to node 3,

\[ X_{23k} = 1 \Rightarrow T_{3k} \geq T_{2k} + 1 \quad \text{(Equation 3.4.)} \]

From Equations 3.3. and 3.4., it is observed that,

\[ T_{3k} \geq T_{1k} + 2 \quad \text{(Equation 3.5.)} \]

But also to complete the sub tour, the truck returns to node 1 from node 3,

\[ X_{31k} = 1 \Rightarrow T_{1k} \geq T_{3k} + 1 \quad \text{(Equation 3.6.)} \]

Since, Equations 3.5. and 3.6. contradict each other, the formation of sub tour is eliminated.

Since, the continuous variable \( T_{ik} \) is defined for \( i = \{1, 2.., n\} \), formation of sub tour is not eliminated at Depot. Thus, we observe formation of 4 different tours in Figure 3.1. starting at the depot.
• **Delivery conformation constraints** – Deliveries should be made for all the gas stations so as to satisfy their demands and each gas station must be visited exactly once.

\[
\sum_{k=1}^{m} \sum_{i=0}^{n} X_{ijk} = 1 \text{ for } \{j = 1, 2 \ldots n\} \text{ and } i \neq j
\]

• **Non-negativity and binary constraints** –

\[
T_{ik} \geq 0
\]

\[
X_{ijk}, S_k \in (0, 1)
\]

### 3.2.4. Problem Size

Given \(m\) – Maximum number of trucks in the fleet and

\(n\) – Number of gas stations,

Total number of Binary variables = \([m \times (n^2+n+1)]\)

Total number of Continuous variables = \([m \times n]\)

Total number of constraints = \([n + m \times (n^2+5)]\)
3.3. Illustrative example

3.3.1. Problem Description

A Vehicle routing problem was considered with trucks leaving the depot to serve the demand at 3 different gas stations.

Maximum number of trucks that are required to satisfy the demand = 3.

Capacity of truck is fixed at 45 kl.

Travel distance/trip is restricted to 240 kms.

Assume that there exist no breakdown of the trucks and negligible waiting time for trucks at the gas station.

Location in (x, y) co-ordinates, representing the distance from the depot (origin) and the demand at each of the gas station, is given in Table 3.1. It is assumed that the depot is at the origin (0,0).

Table 3.1: Location(x, y) and Demand (kl) of each gas station

<table>
<thead>
<tr>
<th>Gas Station No.</th>
<th>Location (X Co-ordinate)</th>
<th>Location (Y Co-ordinate)</th>
<th>Demand (kl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52</td>
<td>60</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>-75</td>
<td>57</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>-49</td>
<td>-20</td>
<td>25</td>
</tr>
</tbody>
</table>
Since, the objective function is to minimize the overall distance travelled by the trucks, shortest
distance between each gas station and the distance of each gas station from the depot is
calculated. They are given in Table 3.2. If the city roads are in a grid pattern, “Manhattan”
distances, instead of Euclidean distances, can be used.

Table 3.2: Distance Matrix

<table>
<thead>
<tr>
<th>Distance(i,j)</th>
<th>Depot</th>
<th>GS-1</th>
<th>GS-2</th>
<th>GS-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depot</td>
<td>0.0</td>
<td>79.4</td>
<td>94.2</td>
<td>52.9</td>
</tr>
<tr>
<td>GS-1</td>
<td>79.4</td>
<td>0.0</td>
<td>127.0</td>
<td>128.8</td>
</tr>
<tr>
<td>GS-2</td>
<td>94.2</td>
<td>127.0</td>
<td>0.0</td>
<td>81.3</td>
</tr>
<tr>
<td>GS-3</td>
<td>52.9</td>
<td>128.8</td>
<td>81.3</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Since, the demand at each gas station is less than the capacity of the truck, we require at most 3
trucks to satisfy the demand at each of these gas station.

3.3.2. Formulation

The problem is formulated for minimizing the overall distance travelled, with constraints
written for Sub-tour elimination, truck continuity, delivery conformation and routing for each of
the trips from depot as discussed in Section 3.2.
Variable definition

\[ X_{ijk} = 1, \text{ if truck k travels from node i to node j} \]

for \( i = \{0,1,2,3\} \), \( j = \{0,1,2,3\} \), \( k = \{1,2,3\} \) and \( i \neq j \)

\[ = 0, \text{ otherwise} \]

\[ S_k = 1, \text{ if truck k is used for delivering gasoline for } k = \{1, 2, 3\} \]

\[ = 0, \text{ otherwise} \]

\[ T_{ik} = \text{Non-negative continuous variable defined for each node } \{i = 1, 2, 3\} \text{ to eliminate subtours for truck k where } \{k = 1, 2, 3\}. \]

Objective Function

\[ \text{MIN}= 79.4 \times X_{011} + 79.4 \times X_{012} + 79.4 \times X_{013} + 94.2 \times X_{021} + 94.2 \times X_{022} + 94.2 \times X_{023} + 52.9 \times X_{031} + 52.9 \times X_{032} + 52.9 \times X_{033} + 79.4 \times X_{101} + 79.4 \times X_{102} + 79.4 \times X_{103} + 127 \times X_{121} + 127 \times X_{122} + 127 \times X_{123} + 128.8 \times X_{131} + 128.8 \times X_{132} + 128.8 \times X_{133} + 94.2 \times X_{201} + 94.2 \times X_{202} + 94.2 \times X_{203} + 127 \times X_{211} + 127 \times X_{212} + 127 \times X_{213} + 81.3 \times X_{231} + 81.3 \times X_{232} + 81.3 \times X_{233} + 52.9 \times X_{301} + 52.9 \times X_{302} + 52.9 \times X_{303} + 128.8 \times X_{311} + 128.8 \times X_{312} + 128.8 \times X_{313} + 81.3 \times X_{321} + 81.3 \times X_{322} + 81.3 \times X_{323}; \]
Truck Capacity Constraints

15 * X_{011} + 20 * X_{021} + 25 * X_{031} + 20 * X_{121} + 25 * X_{131} + 15 * X_{211} + 25 * X_{231} + 15 * X_{311} + 20 * X_{321} \leq 45;

15 * X_{012} + 20 * X_{022} + 25 * X_{032} + 20 * X_{122} + 25 * X_{132} + 15 * X_{212} + 25 * X_{232} + 15 * X_{312} + 20 * X_{322} \leq 45;

15 * X_{013} + 20 * X_{023} + 25 * X_{033} + 20 * X_{123} + 25 * X_{133} + 15 * X_{213} + 25 * X_{233} + 15 * X_{313} + 20 * X_{323} \leq 45;

Delivery Travel Limit Constraints

For truck 1:

79.4 * X_{011} + 94.2 * X_{021} + 52.9 * X_{031} +
79.4 * X_{101} + 127 * X_{121} + 128.8 * X_{131} +
94.2 * X_{201} + 127 * X_{211} + 81.3 * X_{231} +
52.9 * X_{301} + 128.8 * X_{311} + 81.3 * X_{321} \leq 240;

For truck 2:

79.4 * X_{012} + 94.2 * X_{022} + 52.9 * X_{032} +
79.4 * X_{102} + 127 * X_{122} + 128.8 * X_{132} +
94.2 * X_{202} + 127 * X_{212} + 81.3 * X_{232} +
52.9 * X_{302} + 128.8 * X_{312} + 81.3 * X_{322} \leq 240;

For truck 3:

79.4 * X_{013} + 94.2 * X_{023} + 52.9 * X_{033} +
79.4 * X_{103} + 127 * X_{123} + 128.8 * X_{133} +
94.2 * X_{203} + 127 * X_{213} + 81.3 * X_{233} +
52.9 * X_{303} + 128.8 * X_{313} + 81.3 * X_{323} \leq 240;
Constraints to ensure each trip starts and end at depot

\[ X_{011} + X_{021} + X_{031} = S_1; \]
\[ X_{012} + X_{022} + X_{032} = S_2; \]
\[ X_{013} + X_{023} + X_{033} = S_3; \]
\[ X_{101} + X_{201} + X_{301} = S_1; \]
\[ X_{102} + X_{202} + X_{302} = S_2; \]
\[ X_{103} + X_{203} + X_{303} = S_3; \]

**Truck Continuity Constraints**

\[ X_{101} + X_{201} + X_{301} - X_{011} - X_{021} - X_{031} = 0; \]
\[ X_{102} + X_{202} + X_{302} - X_{012} - X_{022} - X_{032} = 0; \]
\[ X_{103} + X_{203} + X_{303} - X_{013} - X_{023} - X_{033} = 0; \]

\[ X_{011} + X_{211} + X_{311} - X_{101} - X_{121} - X_{131} = 0; \]
\[ X_{012} + X_{212} + X_{312} - X_{102} - X_{122} - X_{132} = 0; \]
\[ X_{013} + X_{213} + X_{313} - X_{103} - X_{123} - X_{133} = 0; \]

\[ X_{021} + X_{121} + X_{321} - X_{201} - X_{211} - X_{231} = 0; \]
\[ X_{022} + X_{122} + X_{322} - X_{202} - X_{212} - X_{232} = 0; \]
\[ X_{023} + X_{123} + X_{323} - X_{203} - X_{213} - X_{233} = 0; \]
\[
X_{031} + X_{131} + X_{231} - X_{301} - X_{311} - X_{321} = 0;
\]
\[
X_{032} + X_{132} + X_{232} - X_{302} - X_{312} - X_{322} = 0;
\]
\[
X_{033} + X_{133} + X_{233} - X_{303} - X_{313} - X_{323} = 0;
\]

Sub-Tour Elimination Constraints

For truck 1:
\[
T_{11} - T_{21} + 3 \cdot X_{121} \leq 2;
\]
\[
T_{11} - T_{31} + 3 \cdot X_{131} \leq 2;
\]
\[- T_{11} + T_{21} + 3 \cdot X_{211} \leq 2;
\]
\[
T_{21} - T_{31} + 3 \cdot X_{231} \leq 2;
\]
\[- T_{11} + T_{31} + 3 \cdot X_{311} \leq 2;
\]
\[- T_{21} + T_{31} + 3 \cdot X_{321} \leq 2;
\]

For truck 2:
\[
T_{12} - T_{22} + 3 \cdot X_{122} \leq 2;
\]
\[
T_{12} - T_{32} + 3 \cdot X_{132} \leq 2;
\]
\[- T_{12} + T_{22} + 3 \cdot X_{212} \leq 2;
\]
\[
T_{22} - T_{32} + 3 \cdot X_{232} \leq 2;
\]
\[- T_{12} + T_{32} + 3 \cdot X_{312} \leq 2;
\]
\[- T_{22} + T_{32} + 3 \cdot X_{322} \leq 2;
\]
For truck 3:

\[
\begin{align*}
T_{13} - T_{23} + 3 \times X_{123} & \leq 2; \\
T_{13} - T_{33} + 3 \times X_{133} & \leq 2; \\
-T_{13} + T_{23} + 3 \times X_{213} & \leq 2; \\
T_{23} - T_{33} + 3 \times X_{233} & \leq 2; \\
-T_{13} + T_{33} + 3 \times X_{313} & \leq 2; \\
-T_{23} + T_{33} + 3 \times X_{323} & \leq 2;
\end{align*}
\]

Delivery Confirmation Constraints

For Gas station 1: \( X_{011} + X_{211} + X_{311} + X_{012} + X_{212} + X_{312} + X_{013} + X_{213} + X_{313} = 1; \)

For Gas station 2: \( X_{021} + X_{121} + X_{321} + X_{022} + X_{122} + X_{322} + X_{023} + X_{123} + X_{323} = 1; \)

For Gas station 3: \( X_{031} + X_{131} + X_{231} + X_{032} + X_{132} + X_{232} + X_{033} + X_{133} + X_{233} = 1; \)

3.3.3. Case Study Results

Upon solving the problem as a minimization function, we got the objective value as 387.2 kms.

LINGO software is used to solve this MILP problem. A HP LP1965 model with Windows 7 Enterprise edition, Pentium® Dual Core 2.4 Ghz Processor, 3 GB RAM and 64 bit Operating System is used to run this LINGO software. Computational time in LINGO = 2 seconds.

The problem is solved for 9 continuous variables, 39 binary variables and 45 constraints.

Thus, the routing of trucks required to cater to the demand at each gas station, along with its fill capacity is given in Table 3.3.
Table 3.3: Optimal routing plan

<table>
<thead>
<tr>
<th>Truck</th>
<th>Travel Route</th>
<th>Total travel distance</th>
<th>Fill Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depot-Gas station 3 - Gas station 2 - Depot</td>
<td>228.4 kms</td>
<td>45 kl</td>
</tr>
<tr>
<td>2</td>
<td>Depot - Gas station 1 - Depot</td>
<td>158.8 kms</td>
<td>15 kl</td>
</tr>
</tbody>
</table>

Thus, the formulated problem has been solved for minimizing the overall distance travelled satisfying all the constraints.

3.4. Scalability of the model

When the number of gas stations to be served increases, we observe a huge increase in the number of constraints and variables required to solve the problem by the MILP model. Table 3.4 identifies the required number of binary variables, continuous variables and constraints to solve the Vehicle routing problem for varying number of gas stations.

Table 3.4: Scalability of the MILP model

<table>
<thead>
<tr>
<th>n(No. of gas stations)</th>
<th>m(Max. no. of trucks in the fleet)</th>
<th>Binary Variables</th>
<th>Continuous Variables</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>39</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>84</td>
<td>16</td>
<td>88</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>155</td>
<td>25</td>
<td>155</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>258</td>
<td>36</td>
<td>252</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>399</td>
<td>49</td>
<td>385</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>584</td>
<td>64</td>
<td>560</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>819</td>
<td>81</td>
<td>783</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1110</td>
<td>100</td>
<td>1060</td>
</tr>
</tbody>
</table>
Figure 3.2. indicates the exponential increase in the size of binary variables and constraints that are required to solve larger problem sizes from the data obtained in Table 3.4.

![Graph](image)

**Figure 3.2:** Problem Size (Variables and Constraints) Vs. No. of gas stations

Thus, the attention should be given for the computational time to solve larger problem sizes.

The following table identifies the time (secs) to solve varied problem sizes (No. of gas stations) in LINGO.

**Table 3.5:** Computational time

<table>
<thead>
<tr>
<th>n(No. of gas stations)</th>
<th>Time(secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>27</td>
</tr>
<tr>
<td>6</td>
<td>118</td>
</tr>
<tr>
<td>7</td>
<td>330</td>
</tr>
<tr>
<td>8</td>
<td>3520</td>
</tr>
<tr>
<td>9</td>
<td>14940</td>
</tr>
<tr>
<td>10</td>
<td>37470</td>
</tr>
</tbody>
</table>
Upon drawing a 2-D graph for the values in Table 3.5., we observe an exponential increase in the computational time for the output to generate for different sizes of customers (gas stations).

For example, for the 10 gas station problem, it took 10 hour and 25 minutes to solve the MILP problem. For real world problems, where the number of gas stations served by a depot can exceed 100, the MILP approach is not computationally feasible. Hence, in Chapter 4, we develop a heuristics approach to solve larger problems and compare its solution effectiveness with the optimal solution obtained by MILP.
Chapter -4

HEURISTICS DEVELOPMENT

4.1. Need for heuristic approach

In Chapter 3, we showed that the computational time by the Mixed Integer Linear Programming (MILP) algorithm increased exponentially with respect to the number of gas stations. Even for a small problem with 10 gas stations, it took more than 10 hours to find the optimal solution by MILP. Hence, in this chapter, we develop a heuristic method to solve the Vehicle Routing Problem and compare its effectiveness with the optimal algorithm (MILP).

Thangiah (1995) describes a method called GIDEON that assigns customers to vehicles by partitioning the customers into sectors by genetic algorithm and customers within each formed sector are routed using the cheapest insertion method of Golden and Stewart (1985). To improve the solution quality, these two processes are run iteratively for a finite number of times. The search begins by clustering customers either according to a random process or through polar coordinate angle. A genetic algorithm called GENEROUS was proposed by Potvin and Bengio (1996) that directly applies genetic operators to solutions, thus avoiding the coding issues.

Berger et al. (1998) proposed a method based on the hybridization of a genetic algorithm with well-known construction heuristics. The author omits the coding issues and represent a solution by a set of feasible routes.
Homberger and Gehring (1999) proposed two evolutionary metaheuristics for VRPTW. The proposed algorithms are based on the class of evolutionary algorithms called Evolution strategies.

Also, Braysy et al. (2000) described a two-phase hybrid evolutionary algorithm based on the hybridization of a genetic and evolutionary algorithm consisting of several local search and route construction heuristics.

Though there exist many heuristics approaches to solve complex VRP problem, it has been observed that most of the heuristics have been created to solve a specific case of the VRP problem. This thesis caters to solve a specific problem that exists in the market. There is no heuristics developed in the past that addresses this specific problem and so the need for developing a new one becomes essential.

### 4.2. Development of a Heuristic Approach

To overcome the limitations faced by solving the problem through an optimal algorithm, a heuristic algorithm is developed using MATLAB to generate an acceptable solution to the Vehicle Routing Problem. The heuristic method is based on that of Fisher and Jaikumar (1981) with focus on clustering algorithms. It provides feasible solutions for a VRP that incorporates the same constraints described in Chapter - 3 except for time window restrictions. The heuristic is tested for its effectiveness by solving several problems of different sizes by both the optimal algorithm using MILP and the heuristics method.
4.2.1. Generation of distance matrix

We need to convert the details of location for all the Gas stations whose orders have been placed into numeric distance values. The location for each gas station is available in (X, Y) co-ordinates. The co-ordinates of the depot have been assigned values of (0, 0). Euclidean distance measure is used to calculate the distance values. Distance metrics are calculated between all these locations with the depot and also within each of these locations. Other distance measures, such as Manhattan distances, could be easily incorporated.

Example: If there are 2 gas stations located at distance \((x_1, y_1)\) and \((x_2, y_2)\) from the installation, we find the Euclidean distance between these 2 gas stations by applying the formula

\[
\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.
\]

4.2.2. Creating validation index

To ensure no more than one trip is made to each gas station, we create a binary variable for each of these gas stations, which gets validated from 0 to 1, whenever a truck is assigned to deliver to the gas station. This validation assignment also ensures that each node is covered by exactly one truck.
4.2.3. Clustering locations of similar demand

Each of the discrete demand of the same value is clustered together and stored in a separate array. Example: Consider the following 8 gas stations, whose demands are given in Table 4.1.

**Table 4.1: Demand Locations**

<table>
<thead>
<tr>
<th>Location</th>
<th>Demands (kl)</th>
<th>X Co-ordinate</th>
<th>Y Co-ordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS-1</td>
<td>15</td>
<td>55</td>
<td>80</td>
</tr>
<tr>
<td>GS-2</td>
<td>20</td>
<td>70</td>
<td>48</td>
</tr>
<tr>
<td>GS-3</td>
<td>25</td>
<td>57</td>
<td>25</td>
</tr>
<tr>
<td>GS-4</td>
<td>15</td>
<td>79</td>
<td>34</td>
</tr>
<tr>
<td>GS-5</td>
<td>20</td>
<td>47</td>
<td>82</td>
</tr>
<tr>
<td>GS-6</td>
<td>25</td>
<td>89</td>
<td>29</td>
</tr>
<tr>
<td>GS-7</td>
<td>15</td>
<td>46</td>
<td>59</td>
</tr>
<tr>
<td>GS-8</td>
<td>20</td>
<td>57</td>
<td>70</td>
</tr>
</tbody>
</table>

We observe 3 discrete demand values (15 kl, 20 kl and 25 kl) across the 8 gas stations. Thus, three clusters A15, A20 and A25 are created, where the location and X, Y co-ordinates are stored under each of the respective demands.

Thus, $A15 = \{\text{GS-1 55 80; GS-4 79 34; GS-7 46 59}\}$;

$A20 = \{\text{GS-2 70 48; GS-5 47 82; GS-8 57 70}\}$;

$A25 = \{\text{GS-3 57 25; GS-5 89 29}\}$;

Thus, there is one cluster generated for each distinct demand.
4.2.4. Generating all possible delivery combinations with respect to truck capacity

Here, we generate all possible combinations with the given discrete demands that can be satisfied by the truck capacity. This is achieved by matching each of the discrete demand at different gas stations to the truck’s maximum capacity.

Example: If we have gas stations with discrete capacities of 15 kl and 20 kl and a truck with maximum carrying capacity of 45 kl, a truck can either be filled with 45 kl to satisfy demands at 3 gas stations requiring 15 kl each or can be filled with 40 kl to satisfy demands at 2 stations requiring 20 kl each or with 35 kl to satisfy demands at 15 kl and 20 kl gas stations respectively. Thus, it is possible to generate all possible combinations of routes for the truck. Table 4.2 gives all feasible combinations for a 45 kl truck, filling 3 distinct demands of 15 kl, 20 kl and 25 kl respectively. Note that the possible combinations are listed in the order of unused truck capacity.

**Table 4.2:** Feasible combinations

<table>
<thead>
<tr>
<th>Possible Combinations</th>
<th>Demand (in kl)</th>
<th>Unused truck capacity (in kl)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Combination 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Combination 2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Combination 3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Combination 4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Combination 5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Combination 6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Combination 7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Combination 8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Combination 9</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The algorithm then tries to select a combination with minimal unused truck capacity.
4.2.5. Generating separate feasibility routes for each combination

With all possible combinations formed with each of the demands at different gas stations, we then generate separate feasibility routes for each of these combinations within the array formed in Step 4.2.3.

For example, four gas stations A, B, C, D with demands of 25 kl, 25 kl, 20 kl and 20 kl, there exist 4 possible routes that a truck with maximum capacity of 45 kl can travel with zero unused capacity.

Route 1: Depot ⇨ A ⇨ C ⇨ Depot
Route 2: Depot ⇨ A ⇨ D ⇨ Depot
Route 3: Depot ⇨ B ⇨ C ⇨ Depot
Route 4: Depot ⇨ B ⇨ D ⇨ Depot

We then sort these feasible routes in order of their distances between each of those gas stations and select the best.

4.2.6. Allotment of truck for a minimum distance combination within each array

Since, our objective is to minimize the overall distance travelled, we look for the combination, within the feasibility routes, whose distance is minimal, within the total traveling distance constraint. Then, a truck is allotted to cater to the demands of the gas stations that fall under this combination.
For example, if Route 3 (Depot ⇔ B ⇔ C ⇔ Depot) takes the minimum travelling distance among the 4 possible routes discussed in 4.2.5., we make an assignment for a truck to supply Gas stations B and C in one trip.

4.2.7. Updating of Validation Index

Once we assign a truck to supply some gas stations as illustrated in Section 4.2.6., the validation indices for these allotted gas stations are updated from 0 to 1. In our example above, gas stations B and C will have their indices updated to 1.

4.2.8. General steps of the heuristic algorithm

The following steps indicate the methodology behind the heuristics algorithm used to solve the Vehicle Routing Problem.

Step 1: Generate distance matrix (4.2.1.)

Step 2: Create validation index (4.2.2.)

Step 3: Cluster locations with similar demand (4.2.3.)

Step 4: Generate all possible delivery combinations (4.2.4.)

Step 5: Generate feasible routes for each combination (4.2.5.) and determine the minimal distance.

Step 6: Allot truck for a minimum distance route (4.2.6.)

Step 7: Update the validation index of the gas stations (4.2.7.)

Repeat Steps 5, 6 & 7 until all customer demands have been met.
4.3. Illustrative example and comparison

Consider the same illustrative example discussed in Chapter 3.3. with 3 different gas stations, whose locations in (x, y) co-ordinates are given in Table 4.3.

**Table 4.3:** Location(x, y) and Demand (kl) of each gas station

<table>
<thead>
<tr>
<th>Gas Station No.</th>
<th>Location (X Co-ordinate)</th>
<th>Location (Y Co-ordinate)</th>
<th>Demand (kl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52</td>
<td>60</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>-75</td>
<td>57</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>-49</td>
<td>-20</td>
<td>25</td>
</tr>
</tbody>
</table>

Capacity of truck is fixed at 45 kl.

Travel distance/trip is restricted for 240 kms.

Assumptions are that there exist no breakdown of the trucks and there is no waiting time for trucks at the gas station.

The algorithm was developed in MATLAB to solve this problem and is given in Appendix B.
**Step 1:** Heuristics approach will first generate the distance matrix between all locations. Generated distance matrix is given in Table 4.4.

<table>
<thead>
<tr>
<th>Distance(i,j)</th>
<th>Depot</th>
<th>GS-1</th>
<th>GS-2</th>
<th>GS-3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Depot</strong></td>
<td>0.0</td>
<td>79.4</td>
<td>94.2</td>
<td>52.9</td>
</tr>
<tr>
<td><strong>GS-1</strong></td>
<td>79.4</td>
<td>0.0</td>
<td>127.0</td>
<td>128.8</td>
</tr>
<tr>
<td><strong>GS-2</strong></td>
<td>94.2</td>
<td>127.0</td>
<td>0.0</td>
<td>81.3</td>
</tr>
<tr>
<td><strong>GS-3</strong></td>
<td>52.9</td>
<td>128.8</td>
<td>81.3</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Step 2:** Since, we have 3 locations for this problem, we create 1 x 3 binary validation vector with initial values as 0.

Valid = [0 0 0];

**Step 3:** Cluster locations with similar demands.

Since, all the demand are different, we form 3 arrays, storing the information of each of location number of the 3 gas stations with their (X, Y) Co-ordinate values.

A15 = [GS-1  52   60];
A20 = [GS-2 -75   57];
A25 = [GS-3 -49  -20];

**Step 4:** Generate all possible combinations of the gas station routes with the cumulative gas station demand equivalent to the specified truck carrying capacity.
Since, the truck capacity is 45 kl, all possible combinations of the demands that matches up to the truck capacity are generated and sorted in descending order of maximum truck capacity utilized.

\[
\text{Trip}_A = [20 \ 25]; \\
\text{Trip}_B = [15 \ 15 \ 15]; \\
\text{Trip}_C = [15 \ 25]; \\
\text{Trip}_D = [20 \ 20]; \\
\text{Trip}_E = [15 \ 20]; \\
\text{Trip}_F = [15 \ 15]; \\
\text{Trip}_G = [25]; \\
\text{Trip}_H = [20]; \\
\text{Trip}_I = [15];
\]

**Step 5:** Feasible routes and their distances are calculated for each trip. The route with the least distance is selected for that trip.

For example, for trip A,

\[
\text{Distance}_{\text{Trip A}} = \text{Depot} \rightarrow \text{GS-3} \rightarrow \text{GS-2} \rightarrow \text{Depot} = 52.9 + 81.3 + 94.2 = 228.4 \text{ kms}.
\]

Note: Because the distance matrix is symmetric, \(\text{Depot} \rightarrow \text{GS-2} \rightarrow \text{GS-3} \rightarrow \text{Depot} = 228.4 \text{ kms}\).

**Step 6:** Distance_{Trip A} array is checked to satisfy the minimum distance limit constraint.

Since, the length of the Trip A route (228.4 kms) is less than the allowed travelling distance of 240 kms, a truck is assigned to the route formed by Trip A (\(\text{Depot} \rightarrow \text{GS-3} \rightarrow \text{GS-2} \rightarrow \text{Depot}\)).
**Step 7:** Update the Validation Index.

Since, assignments of truck are made to satisfy the demands at two of the gas stations, the validation vector is updated from 0 to 1 for GS-2 and GS-3 as follows:

$$V = [0, 1, 1].$$

Steps 5 through 7 are repeated for those gas stations that comes under the combination in Step 4 whose validation index is still 0. Thus, truck allotment is made till the validation indices for all gas stations become 1 or till all combinations in Step 4 have been checked.

Thus, for the given illustrative problem, final allotment has been made for 2 trucks that fall under the combination of Trip_A and Trip_I and the final allotment of routes are given in Table 4.5

One truck is sent with full (45 kl) capacity and another with partial (15 kl) capacity. Thus, the average fill capacity of the trucks will be \((45+15)/90 = 67\%\).

<table>
<thead>
<tr>
<th>Truck</th>
<th>Travel Route</th>
<th>Total travel distance</th>
<th>Fill Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depot-Gas station 3 – Gas station 2 - Depot</td>
<td>228.4 kms</td>
<td>45 kl</td>
</tr>
<tr>
<td>2</td>
<td>Depot - Gas station 1 - Depot</td>
<td>158.8 kms</td>
<td>15 kl</td>
</tr>
</tbody>
</table>
4.4. Effectiveness of the Heuristic Approach

The main advantage of the heuristics approach is to minimize the computational time for larger size problems. Though we get the same results for the illustrative example by both the optimal and heuristics approach, we need to verify that the algorithm written for heuristic approach is near optimal for other problems also.

To investigate the effectiveness of the heuristic, we used the heuristic algorithm for different problem sizes and compared the computational time and objective value with the optimal algorithm using the MILP model. The problem sizes varied from 3 to 10 gas stations. The location co-ordinates are generated at random for different problem sizes and the demands for each of the location are also randomly generated as integer values ranging between 10 kl and 30 kl.

For each problem, the fixed capacity of the truck is randomly generated as an integer value between 40 kl and 60 kl. Travel distance per trip is restricted to 240 kms. Five problems are solved for each problem size. The results are shown in Table 4.6. For each problem, the number of trucks used and the total distance travelled by all trucks are given by both methods. Percent Error refers to the fractional deviation on the total distance by the heuristic method compared to the minimum distance obtained by the optimal algorithm.
Table 4.6: Comparative results of optimal and heuristic approach

<table>
<thead>
<tr>
<th>Problem Size (3 x 3)</th>
<th>No. of trucks used</th>
<th>Distance Value(kms)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimal</td>
<td>Heuristic</td>
<td>Optimal</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>254.20</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>350.12</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>351.07</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>279.55</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>211.59</td>
</tr>
<tr>
<td>% Average Error</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem Size (4 x 4)</th>
<th>No. of trucks used</th>
<th>Distance Value(kms)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimal</td>
<td>Heuristic</td>
<td>Optimal</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>382.64</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>227.75</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>322.76</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>415.8</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>221.85</td>
</tr>
<tr>
<td>% Average Error</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem Size (5 x 5)</th>
<th>No. of trucks used</th>
<th>Distance Value(kms)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimal</td>
<td>Heuristic</td>
<td>Optimal</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>556.24</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>365.86</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>753.93</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>556.24</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>576.03</td>
</tr>
<tr>
<td>% Average Error</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem Size (6 x 6)</th>
<th>No. of trucks used</th>
<th>Distance Value(kms)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimal</td>
<td>Heuristic</td>
<td>Optimal</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>455.17</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>291.79</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>288.39</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>463.84</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
<td>583.9</td>
</tr>
<tr>
<td>% Average Error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem Size(7 x 7)</td>
<td>No. of trucks used</td>
<td>Distance Value(kms)</td>
<td>% Error</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------</td>
<td>---------------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optimal</td>
<td>Heuristic</td>
</tr>
<tr>
<td>Problem</td>
<td>Optimal</td>
<td>Heuristic</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>% Average Error</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem Size(8 x 8)</th>
<th>No. of trucks used</th>
<th>Distance Value(kms)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Optimal</td>
<td>Heuristic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Heuristic</td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Optimal</td>
<td>Heuristic</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>% Average Error</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem Size (9 x 9)</th>
<th>No. of trucks used</th>
<th>Distance Value(kms)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Optimal</td>
<td>Heuristic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Heuristic</td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Optimal</td>
<td>Heuristic</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>% Average Error</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem Size(10 x 10)</th>
<th>No. of trucks used</th>
<th>Distance Value(kms)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Optimal</td>
<td>Heuristic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Heuristic</td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Optimal</td>
<td>Heuristic</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>% Average Error</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
It can be seen from Table 4.6, that the heuristics method obtained the exact optimal solution for 15 out of 40 problems. In problems where the solution deviated from the optimal, the heuristics used fewer trucks than that used by the optimal solution. This would result in significant savings in the fixed cost of the running the trucks.

Table 4.7 gives the average computation time by both methods and the average (%) distance error with respect to the minimum distance for a batch of 5 problems under different problem sizes.

**Table 4.7: Average Error results**

<table>
<thead>
<tr>
<th>Problem Size</th>
<th>Average Computation Time</th>
<th>% Average Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 x 3</td>
<td>1 sec</td>
<td>10.77</td>
</tr>
<tr>
<td>4 x 4</td>
<td>3 secs</td>
<td>6.92</td>
</tr>
<tr>
<td>5 x 5</td>
<td>27 secs</td>
<td>9.36</td>
</tr>
<tr>
<td>6 x 6</td>
<td>1 min 58 secs</td>
<td>13.46</td>
</tr>
<tr>
<td>7 x 7</td>
<td>5 min 30 secs</td>
<td>11.77</td>
</tr>
<tr>
<td>8 x 8</td>
<td>58 min 40 secs</td>
<td>9.04</td>
</tr>
<tr>
<td>9 x 9</td>
<td>4 hrs 9 mins</td>
<td>6.92</td>
</tr>
<tr>
<td>10 x 10</td>
<td>10 hrs 25 mins</td>
<td>7.42</td>
</tr>
</tbody>
</table>

The results show that algorithm developed through heuristic approach performed very well compared to the optimal algorithm with respect to computational time, particularly as the problem size increased. The approximate solutions were no more than 15% away from the true minimum.
Figure 4.1 compares the total travelling distance (kms) between the optimal and heuristic approach for 10 different problems discussed in Table 4.6.

![Total distance travelled by both methods](image1)

**Figure 4.1:** Comparison of distance travelled

Figure 4.2 compares the truck allotment by both methods. We observe that the heuristic method always uses the same or fewer number of trucks for delivery.

![Truck allotment by both methods](image2)

**Figure 4.2:** Comparison of truck allotment
Figure 4.3 combines Figure 4.1 and 4.2 to illustrate the tradeoff between the number of trucks allotted and total distance travelled for the 10 problems. For two of the ten problems, the heuristics achieved the same optimal allotment of trucks and the minimum distance. For two other problems, the heuristic allotted the same number of trucks as the optimal method but the total distance travelled was slightly larger (≤ 11 %). For the remaining 6 problems, the increase in the total distance travelled in the heuristic approach is compensated by a fewer allotment of trucks. For example, in problem P1 (Figure 4.3), the optimal solution allots 6 trucks with the travel distance of 868.17 kms to solve the problem, whereas the heuristic solution allots 4 trucks with the travel distance of 949.1 kms. Thus, we observe that the increase of 9.3 % in the travel distance for the heuristic approach is compensated by a decrease of 33.3 % in the number of trucks.

![Comparison of Optimal vs Heuristic approach](figure4.3.jpg)

**Figure 4.3:** Analysis of Optimal vs. Heuristic approach
In summary, the following conclusions can be given based on the experiments:

- The computational time is significantly lower. Thus, very large problems can be solved in negligible times.
- Allotment of number of trucks through the heuristic approach is always less than or equal to the allotment of number of trucks through the optimal approach.
- When the total distance for the heuristic approach deviates from the optimal value, it is compensated by efficient filling capacity or by using fewer trucks to satisfy the demands.
- The average deviation from the minimum distance is less than 15%.

4.5. Case Study

4.5.1. Introduction

The company involved in this case study was a leading Oil refining and marketing company which operates its own installation (depot) in India. The depot receives orders from the gas stations (customers) through Automatic Tank Gauging Systems, when the level of gas in the underground tank falls below the threshold limit, and then delivers the ordered quantity the next day.

Customer orders were supplied from the company depot either:

- by a delivery vehicle operated by a 3PL service provider; or
- by a delivery vehicle operated by the company.
A large proportion of deliveries was made by third party trucks (from a number of depots) and it was this portion of the company's operation that we examined. For the purposes of our study a single depot in India (with a delivery target of 120 gas stations) was chosen as a target for closer analysis.

4.5.2. Assumptions

- There exists no breakdown of the trucks.
- There is no stopping of truck in between, other than at the gas stations.
- There is no waiting time at the gas station. (Truck stays for a very small time).
- Discrete orders are placed by each customer in line with the maximum capacity of their underground tanks.
- Every truck can make at most one trip during the shift.

4.5.3. Vehicles

All vehicles were identical and for each inter-customer link (i, j), the distance for driving that link was established similar to the distance for driving from the depot to each customer. These distances were calculated by using a commercial computer-based roadmap for Indian roadways. Limit on the total travelling distance for each trip was established based on the permitted travel hours per trip (8 hrs) together with company standards for vehicle speeds (30 kms per hour) on the various categories of roads represented in that roadmap.
Thus, total travel distance per trip = 8 * 30 = 240 kms/trip.

Traveling distance limit for each vehicle along with the amount that they could carry was the key capacity constraint here. The physical size of the vehicles was not a constraint for the company's operations.

4.5.4. Data collection

Given the nature of the business, demand among customers (gas stations) was highly correlated. Data collected for this case study referred to a single (peak) day's customer demands. The objective was to design routes that minimized the overall distance travelled by the trucks that would be able to accommodate such peak demand.

Capacity of truck is fixed at 45 kl and the travel distance/trip is restricted to 240 kms.

The (x, y) co-ordinates for each of the 120 demand locations and their demands are given in Appendix A.
4.5.5. Customer locations

For the company, each customer (gas station) was located at a (x, y) position, based upon the Indian National Grid coordinates. This position was calculated by finding the (demand) weighted centre of gravity of the location with respect to the depot. For each customer (gas station), we also identified those customers who were adjacent to it. While we could have collected this information directly from a map, we found it more effective to generate adjacency information using the (x, y) co-ordinates for each of the customer location. Figure 4.4 indicates the distinct locations of all the gas stations scattered across all directions, from the depot (0, 0) within a radius of 100 kms.

**Figure 4.4:** Location points of all gas stations
4.5.6. Results and Analysis

The case study problem was solved using the heuristic approach because of its main advantage of negligible computational time which can be effectively implemented in the future for daily usage. The MATLAB algorithm to solve this problem is given in Appendix C. The problem was solved taking into account truck continuity, sub-tour elimination constraints, and capacity restrictions with all trips starting and ending at the depot. Upon solving the problem by the heuristic approach, we got the objective value as 8090.8 kms. The computational time was one second.

From the demand data (Appendix A), we observe that the demand requirements are 15 kl at 45 gas stations, 20 kl at 47 gas stations and 25 kl at 28 gas stations. To supply the 120 gas stations, 55 trucks were used.

Table 4.8 indicates the required number of trucks with its fill capacity to cater to the overall demand at all gas stations.

<table>
<thead>
<tr>
<th>No. of trucks</th>
<th>Fill capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>45 kl</td>
</tr>
<tr>
<td>9</td>
<td>40 kl</td>
</tr>
<tr>
<td>1</td>
<td>35 kl</td>
</tr>
<tr>
<td>3</td>
<td>30 kl</td>
</tr>
<tr>
<td>2</td>
<td>15 kl</td>
</tr>
</tbody>
</table>
We can observe from Table 4.8 that atmost 53 out of 55 trucks are efficiently utilized to more than 65% of its total capacity. Also, 40 trucks have been completely utilized to its maximum capacity of 45 kl. This clearly indicates that the heuristic approach optimizes the filling capacity of the trucks, apart from minimizing the overall distance travelled by the trucks to cover all the gas stations.

Table 4.9 indicates the optimal allotment of routes for each truck with its fill capacity and its travel length (kms).

We can observe from Table 4.9 that all the routes are assigned within the limits of total distance a truck can travel, which was set at 240 kms. Overall the results were that the company had the potential to substantially reduce the size of their vehicle fleet (to 55 vehicles) and still maintain customer service. Thus, the vehicle routing problem related to this case study has been solved efficiently with minimal computational time to satisfy all the capacity restrictions, travel limit and the demand constraints.
Table 4.9: Optimal Allotment

<table>
<thead>
<tr>
<th>Truck</th>
<th>Travel Route</th>
<th>Fill capacity</th>
<th>Total travel distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depot - GS 82 - GS 3 - Depot</td>
<td>45 kl</td>
<td>158.80</td>
</tr>
<tr>
<td>2</td>
<td>Depot - GS 29 - GS 53 - Depot</td>
<td>45 kl</td>
<td>159.55</td>
</tr>
<tr>
<td>3</td>
<td>Depot - GS 58 - GS 116 - Depot</td>
<td>45 kl</td>
<td>58.17</td>
</tr>
<tr>
<td>4</td>
<td>Depot - GS 111 - GS 85 - Depot</td>
<td>45 kl</td>
<td>114.27</td>
</tr>
<tr>
<td>5</td>
<td>Depot - GS 41 - GS 88 - Depot</td>
<td>45 kl</td>
<td>108.20</td>
</tr>
<tr>
<td>6</td>
<td>Depot - GS 70 - GS 98 - Depot</td>
<td>45 kl</td>
<td>97.12</td>
</tr>
<tr>
<td>7</td>
<td>Depot - GS 87 - GS 113 - Depot</td>
<td>45 kl</td>
<td>76.29</td>
</tr>
<tr>
<td>8</td>
<td>Depot - GS 45 - GS 119 - Depot</td>
<td>45 kl</td>
<td>145.77</td>
</tr>
<tr>
<td>9</td>
<td>Depot - GS 38 - GS 40 - Depot</td>
<td>45 kl</td>
<td>117.38</td>
</tr>
<tr>
<td>10</td>
<td>Depot - GS 86 - GS 71 - Depot</td>
<td>45 kl</td>
<td>141.65</td>
</tr>
<tr>
<td>11</td>
<td>Depot - GS 17 - GS 101 - Depot</td>
<td>45 kl</td>
<td>142.76</td>
</tr>
<tr>
<td>12</td>
<td>Depot - GS 80 - GS 60 - Depot</td>
<td>45 kl</td>
<td>118.48</td>
</tr>
<tr>
<td>13</td>
<td>Depot - GS 79 - GS 81 - Depot</td>
<td>45 kl</td>
<td>145.67</td>
</tr>
<tr>
<td>14</td>
<td>Depot - GS 4 - GS 12 - Depot</td>
<td>45 kl</td>
<td>191.24</td>
</tr>
<tr>
<td>15</td>
<td>Depot - GS 89 - GS 73 - Depot</td>
<td>45 kl</td>
<td>92.10</td>
</tr>
<tr>
<td>16</td>
<td>Depot - GS 77 - GS 75 - Depot</td>
<td>45 kl</td>
<td>124.05</td>
</tr>
<tr>
<td>17</td>
<td>Depot - GS 48 - GS 34 - Depot</td>
<td>45 kl</td>
<td>150.96</td>
</tr>
<tr>
<td>18</td>
<td>Depot - GS 117 - GS 13 - Depot</td>
<td>45 kl</td>
<td>137.27</td>
</tr>
<tr>
<td>19</td>
<td>Depot - GS 118 - GS 72 - Depot</td>
<td>45 kl</td>
<td>88.30</td>
</tr>
<tr>
<td>20</td>
<td>Depot - GS 76 - GS 104 - Depot</td>
<td>45 kl</td>
<td>139.49</td>
</tr>
<tr>
<td>21</td>
<td>Depot - GS 120 - GS 100 - Depot</td>
<td>45 kl</td>
<td>77.95</td>
</tr>
<tr>
<td>22</td>
<td>Depot - GS 99 - GS 31 - Depot</td>
<td>45 kl</td>
<td>142.29</td>
</tr>
<tr>
<td>23</td>
<td>Depot - GS 39 - GS 107 - Depot</td>
<td>45 kl</td>
<td>85.69</td>
</tr>
<tr>
<td>24</td>
<td>Depot - GS 46 - GS 32 - Depot</td>
<td>45 kl</td>
<td>149.56</td>
</tr>
<tr>
<td>25</td>
<td>Depot - GS 36 - GS 18 - Depot</td>
<td>45 kl</td>
<td>173.33</td>
</tr>
<tr>
<td>26</td>
<td>Depot - GS 35 - GS 25 - Depot</td>
<td>45 kl</td>
<td>172.36</td>
</tr>
<tr>
<td>27</td>
<td>Depot - GS 7 - GS 67 - Depot</td>
<td>45 kl</td>
<td>204.17</td>
</tr>
<tr>
<td>28</td>
<td>Depot - GS 5 - GS 59 - Depot</td>
<td>45 kl</td>
<td>192.15</td>
</tr>
<tr>
<td>29</td>
<td>Depot - GS 62 - GS 84 - GS 90 - Depot</td>
<td>45 kl</td>
<td>158.26</td>
</tr>
<tr>
<td>30</td>
<td>Depot - GS 52 - GS 65 - GS 69 - Depot</td>
<td>45 kl</td>
<td>134.42</td>
</tr>
<tr>
<td>31</td>
<td>Depot - GS 2 - GS 8 - GS 121 - Depot</td>
<td>45 kl</td>
<td>191.00</td>
</tr>
<tr>
<td>32</td>
<td>Depot - GS 24 - GS 93 - GS 109 - Depot</td>
<td>45 kl</td>
<td>154.09</td>
</tr>
<tr>
<td>33</td>
<td>Depot - GS 21 - GS 91 - GS 105 - Depot</td>
<td>45 kl</td>
<td>122.69</td>
</tr>
<tr>
<td>34</td>
<td>Depot - GS 23 - GS 92 - GS 115 - Depot</td>
<td>45 kl</td>
<td>124.57</td>
</tr>
<tr>
<td>35</td>
<td>Depot - GS 33 - GS 43 - GS 51 - Depot</td>
<td>45 kl</td>
<td>143.16</td>
</tr>
<tr>
<td>36</td>
<td>Depot - GS 20 - GS 28 - GS 102 - Depot</td>
<td>45 kl</td>
<td>129.68</td>
</tr>
<tr>
<td>37</td>
<td>Depot - GS 56 - GS 74 - GS 97 - Depot</td>
<td>45 kl</td>
<td>137.48</td>
</tr>
<tr>
<td>38</td>
<td>Depot - GS 50 - GS 64 - GS 96 - Depot</td>
<td>45 kl</td>
<td>199.97</td>
</tr>
<tr>
<td>39</td>
<td>Depot - GS 49 - GS 83 - GS 103 - Depot</td>
<td>45 kl</td>
<td>199.74</td>
</tr>
<tr>
<td>40</td>
<td>Depot - GS 9 - GS 11 - GS 61 - Depot</td>
<td>45 kl</td>
<td>229.32</td>
</tr>
<tr>
<td>41</td>
<td>Depot - GS 37 - GS 57 - Depot</td>
<td>40 kl</td>
<td>100.18</td>
</tr>
<tr>
<td>42</td>
<td>Depot - GS 19 - GS 108 - Depot</td>
<td>40 kl</td>
<td>104.57</td>
</tr>
<tr>
<td>43</td>
<td>Depot - GS 44 - GS 114 - Depot</td>
<td>40 kl</td>
<td>133.55</td>
</tr>
<tr>
<td>44</td>
<td>Depot - GS 54 - GS 95 - Depot</td>
<td>40 kl</td>
<td>126.78</td>
</tr>
<tr>
<td>45</td>
<td>Depot - GS 78 - GS 112 - Depot</td>
<td>40 kl</td>
<td>106.02</td>
</tr>
<tr>
<td>46</td>
<td>Depot - GS 6 - GS 16 - Depot</td>
<td>40 kl</td>
<td>205.47</td>
</tr>
<tr>
<td>47</td>
<td>Depot - GS 30 - GS 94 - Depot</td>
<td>40 kl</td>
<td>163.14</td>
</tr>
<tr>
<td>48</td>
<td>Depot - GS 63 - GS 66 - Depot</td>
<td>40 kl</td>
<td>154.42</td>
</tr>
<tr>
<td>49</td>
<td>Depot - GS 22 - GS 26 - Depot</td>
<td>40 kl</td>
<td>191.37</td>
</tr>
<tr>
<td>50</td>
<td>Depot - GS 47 - GS 55 - Depot</td>
<td>35 kl</td>
<td>220.65</td>
</tr>
<tr>
<td>51</td>
<td>Depot - GS 106 - GS 110 - Depot</td>
<td>30 kl</td>
<td>152.12</td>
</tr>
<tr>
<td>52</td>
<td>Depot - GS 14 - GS 42 - Depot</td>
<td>30 kl</td>
<td>212.91</td>
</tr>
<tr>
<td>53</td>
<td>Depot - GS 10 - GS 68 - Depot</td>
<td>30 kl</td>
<td>219.58</td>
</tr>
<tr>
<td>54</td>
<td>Depot - GS 15 - Depot</td>
<td>15 kl</td>
<td>179.62</td>
</tr>
<tr>
<td>55</td>
<td>Depot - GS 27 - Depot</td>
<td>15 kl</td>
<td>191.05</td>
</tr>
</tbody>
</table>
Chapter -5

CONCLUSION AND FUTURE WORK

In this thesis, we developed an optimization model by generating a daily routing plan for a fleet of vehicles for Oil Company based on a delivery schedule to the gas stations. The problem encompass delivery to gas stations, transportation between gas stations and return to the installation within the time window. The objective of the problem was to plan the routes such that the overall distance covered by the trucks was minimized for all the gas stations, which were geographically dispersed, without violating the truck capacities and satisfying all the demands at the gas stations. We showed that the problem was a multiple traveling salesman problem.

In Chapter 3, we presented the problem description and the model formulation based on the Mixed Integer Linear Programming (MILP) approach. Detailed mathematical formulation for the MILP problem was discussed along with an illustrative example. Upon solving this MILP problem through LINGO software, we observed an exponential increase in the size of binary variables and constraints for solving larger problems. Also, we observed an exponential increase in the computational time for the optimal algorithm as the number of gas stations increased. Even for a small problem with just 10 gas stations, the optimal algorithm took more than 10 hours to solve.
In Chapter 4, we developed an intelligent heuristics approach to solve larger size problems in minimal time and compared its solution effectiveness with the optimal solution obtained by MILP. The heuristic algorithm was written in MATLAB to generate an acceptable solution to the Vehicle Routing Problem. We illustrated the heuristics model with an example problem and analyzed its results in comparison to the optimal algorithm.

The main advantage of heuristics approach is to minimize the computational time for larger size problems. We verified the effectiveness of the heuristic approach by testing it on different problem sizes and comparing the computational time and objective value with the optimal algorithm, using the MILP model. The results showed that the algorithm developed through the heuristic approach was as good as the optimal approach with respect to the total distance travelled by the fleet of trucks, with the average error within 15% of the optimal value. Moreover, the heuristic algorithm used the same or fewer number of trucks for supplying the gasoline, thereby saving in fixed cost of operating the vehicles in several cases. Also, we observed that the computational time is negligible for the heuristic approach. Thus, very large problems can be solved in realistic times. Also, we observed for the cases, when the total distance for the heuristic approach deviates from the optimal value, it was compensated by efficient filling capacity or by using fewer trucks to satisfy the customer demands.

We then applied the heuristic approach to solve an actual case study in India, with a problem size involving demand points of 120 gas stations being served by a single depot. The demand requirements were 15 kl at 45 gas stations, 20 kl at 47 gas stations and 25 kl at 28 gas stations. The capacity of truck was fixed at 45 kl. With the constraint limits on total travel distance,
carrying capacity for each truck, demand at each gas stations, elimination of sub-tour, etc., the problem was solved in MATLAB in just one second. Overall the results were that the company had the potential to substantially reduce the size of their vehicle fleet (to 55 vehicles) and still maintain customer service. Of the 55 vehicles used to deliver the gasoline to the 120 gas stations, 53 were efficiently utilized at more than 65% of the vehicle capacity. Infact, 40 trucks were completely utilized to its maximum capacity of 45 kl. This clearly indicated that the heuristic approach optimized the filling capacity of the trucks, apart from minimizing the overall distance travelled to deliver the gasoline to all the customers.

For future research, the heuristic method can be expanded to accommodate stochastic demand variations at the gas stations. Also, the method can be extended to multiple depots and multiple time windows. Allotment of several trips for the same truck within the stipulated time can also be included. With the possibility of design changes in the capacity of the truck in the near future to accommodate different fuel types within each truck, the model can be extended further to handle trucks with multiple capacities.
BIBLIOGRAPHY


Appendix A: Data for the case study

<table>
<thead>
<tr>
<th>Gas station</th>
<th>X Co-ordinate</th>
<th>Y Co-ordinate</th>
<th>Demand (in kl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52</td>
<td>57</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>52</td>
<td>-60</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>65</td>
<td>70</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>65</td>
<td>-70</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>75</td>
<td>57</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>75</td>
<td>-57</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>49</td>
<td>65</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>49</td>
<td>-55</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>86</td>
<td>47</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>86</td>
<td>-43</td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>60</td>
<td>62</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>-55</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>22</td>
<td>85</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>29</td>
<td>-85</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>85</td>
<td>47</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>-58</td>
<td>-40</td>
<td>25</td>
</tr>
<tr>
<td>17</td>
<td>-58</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>18</td>
<td>-49</td>
<td>-12</td>
<td>20</td>
</tr>
<tr>
<td>19</td>
<td>-46</td>
<td>67</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>-59</td>
<td>-27</td>
<td>15</td>
</tr>
<tr>
<td>21</td>
<td>-20</td>
<td>27</td>
<td>20</td>
</tr>
<tr>
<td>22</td>
<td>25</td>
<td>-49</td>
<td>15</td>
</tr>
<tr>
<td>23</td>
<td>-20</td>
<td>-16</td>
<td>15</td>
</tr>
<tr>
<td>24</td>
<td>-17</td>
<td>-57</td>
<td>20</td>
</tr>
<tr>
<td>25</td>
<td>-65</td>
<td>70</td>
<td>20</td>
</tr>
<tr>
<td>26</td>
<td>-65</td>
<td>-70</td>
<td>15</td>
</tr>
<tr>
<td>27</td>
<td>-52</td>
<td>60</td>
<td>15</td>
</tr>
<tr>
<td>28</td>
<td>-52</td>
<td>-60</td>
<td>25</td>
</tr>
<tr>
<td>29</td>
<td>52</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>-47</td>
<td>-32</td>
<td>20</td>
</tr>
<tr>
<td>31</td>
<td>17</td>
<td>72</td>
<td>20</td>
</tr>
<tr>
<td>32</td>
<td>-26</td>
<td>-68</td>
<td>15</td>
</tr>
<tr>
<td>33</td>
<td>34</td>
<td>59</td>
<td>20</td>
</tr>
<tr>
<td>34</td>
<td>-54</td>
<td>-53</td>
<td>25</td>
</tr>
<tr>
<td>35</td>
<td>-36</td>
<td>62</td>
<td>25</td>
</tr>
<tr>
<td>36</td>
<td>-8</td>
<td>-47</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>79</td>
<td>47</td>
<td>27</td>
<td>25</td>
</tr>
<tr>
<td>80</td>
<td>-41</td>
<td>-52</td>
<td>20</td>
</tr>
<tr>
<td>81</td>
<td>50</td>
<td>-58</td>
<td>25</td>
</tr>
<tr>
<td>82</td>
<td>-12</td>
<td>-37</td>
<td>15</td>
</tr>
<tr>
<td>83</td>
<td>48</td>
<td>-35</td>
<td>15</td>
</tr>
<tr>
<td>84</td>
<td>-35</td>
<td>-43</td>
<td>20</td>
</tr>
<tr>
<td>85</td>
<td>64</td>
<td>-29</td>
<td>25</td>
</tr>
<tr>
<td>86</td>
<td>-16</td>
<td>-34</td>
<td>25</td>
</tr>
<tr>
<td>87</td>
<td>-16</td>
<td>-47</td>
<td>20</td>
</tr>
<tr>
<td>88</td>
<td>-27</td>
<td>-29</td>
<td>25</td>
</tr>
<tr>
<td>89</td>
<td>49</td>
<td>-40</td>
<td>15</td>
</tr>
<tr>
<td>90</td>
<td>-45</td>
<td>-32</td>
<td>15</td>
</tr>
<tr>
<td>91</td>
<td>16</td>
<td>-43</td>
<td>15</td>
</tr>
<tr>
<td>92</td>
<td>-29</td>
<td>-17</td>
<td>15</td>
</tr>
<tr>
<td>93</td>
<td>51</td>
<td>43</td>
<td>20</td>
</tr>
<tr>
<td>94</td>
<td>-57</td>
<td>27</td>
<td>20</td>
</tr>
<tr>
<td>95</td>
<td>19</td>
<td>22</td>
<td>15</td>
</tr>
<tr>
<td>96</td>
<td>-54</td>
<td>26</td>
<td>15</td>
</tr>
<tr>
<td>97</td>
<td>24</td>
<td>37</td>
<td>20</td>
</tr>
<tr>
<td>98</td>
<td>-63</td>
<td>-28</td>
<td>25</td>
</tr>
<tr>
<td>99</td>
<td>27</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>100</td>
<td>-52</td>
<td>-41</td>
<td>20</td>
</tr>
<tr>
<td>101</td>
<td>-58</td>
<td>50</td>
<td>15</td>
</tr>
<tr>
<td>102</td>
<td>-37</td>
<td>-12</td>
<td>15</td>
</tr>
<tr>
<td>103</td>
<td>-35</td>
<td>48</td>
<td>20</td>
</tr>
<tr>
<td>104</td>
<td>-43</td>
<td>-35</td>
<td>15</td>
</tr>
<tr>
<td>105</td>
<td>-29</td>
<td>64</td>
<td>15</td>
</tr>
<tr>
<td>106</td>
<td>-34</td>
<td>-16</td>
<td>20</td>
</tr>
<tr>
<td>107</td>
<td>-47</td>
<td>-16</td>
<td>20</td>
</tr>
<tr>
<td>108</td>
<td>-29</td>
<td>-27</td>
<td>15</td>
</tr>
<tr>
<td>109</td>
<td>-40</td>
<td>49</td>
<td>15</td>
</tr>
<tr>
<td>110</td>
<td>-32</td>
<td>-45</td>
<td>25</td>
</tr>
<tr>
<td>111</td>
<td>-43</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>112</td>
<td>-17</td>
<td>-29</td>
<td>20</td>
</tr>
<tr>
<td>113</td>
<td>43</td>
<td>51</td>
<td>20</td>
</tr>
<tr>
<td>114</td>
<td>27</td>
<td>-57</td>
<td>15</td>
</tr>
<tr>
<td>115</td>
<td>22</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>116</td>
<td>26</td>
<td>-54</td>
<td>25</td>
</tr>
<tr>
<td>117</td>
<td>37</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>118</td>
<td>-28</td>
<td>-63</td>
<td>20</td>
</tr>
<tr>
<td>119</td>
<td>17</td>
<td>27</td>
<td>25</td>
</tr>
<tr>
<td>120</td>
<td>47</td>
<td>52</td>
<td>15</td>
</tr>
</tbody>
</table>
Appendix B: Heuristic Algorithm in MATLAB for Example 4.3.

**Heuristic inputs**

\[ n=4; \]
\[
\text{loc} = \begin{bmatrix}
0 & 0 & 0 & 1; \\
52 & 60 & 15 & 2; \\
-75 & 57 & 20 & 3; \\
-49 & -20 & 25 & 4;
\end{bmatrix};
\]

**Variable Definition**

\[ \text{Dist\_Matrix} = \text{Distance Matrix generated between all locations from the location co-ordinates given as inputs;} \]

\[ \text{Valid\_Index} = \text{Matrix containing Validation Indices for each given location;} \]

\[ \text{A25} = \text{Cluster containing location co-ordinates for gas stations Having demand is 25 kl;} \]

\[ \text{A20} = \text{Cluster containing location co-ordinates for gas stations Having demand is 20 kl;} \]

\[ \text{A15} = \text{Cluster containing location co-ordinates for gas stations Having demand is 15 kl;} \]

\[ \text{Dem\_Combine} = \text{Matrix containing all possible combinations of demand that Matches upto truck capacity;} \]

\[ \text{Tot\_Dist} = \text{Total minimum distance travelled by all the trucks to serve the demand at all gas stations;} \]

\[ \text{Trip\_Assign} = \text{Matrix containing final allotment of routing of trucks to cater to demand of all gas stations;} \]
Step 1: Generating Distance matrix

\[
\text{Dist\_Matrix} = 0; \\
\text{for } j=1:n \\
\quad \text{for } i=1:n \\
\quad \quad a = [(\text{loc}(j)-\text{loc}(i))^2]; \\
\quad \quad b = [(\text{loc}(j+n)-\text{loc}(i+n))^2]; \\
\quad \quad c = a + b; \\
\quad \quad \text{Dist\_Matrix}(i,j) = \sqrt{c}; \\
\quad \quad a = 0; \\
\quad \quad b = 0; \\
\quad \text{end} \\
\text{end} \\
\text{end}
\]

Step 2: Creating Validation index

\[
\text{Valid\_Index} = []; \\
\text{for } i=1:n \\
\quad \text{Valid\_Index}(1,i) = 0; \\
\quad \text{Valid\_Index} = [\text{Valid\_Index}]; \\
\text{end}
\]

Step 3: Clustering locations with similar demand

\[
\text{count}5 = 0; \\
\text{count}6 = 0; \\
\text{count}7 = 0; \\
\text{count}1 = n; \\
\text{A}25 = \text{zeros}(n,4); \\
\text{A}20 = \text{zeros}(n,4); \\
\text{A}15 = \text{zeros}(n,4); \\
\text{for } i=1:\text{count}1 \\
\quad \text{if } \text{loc}(i,3) == 25 \\
\quad \quad \text{count}5 = \text{count}5 + 1; \\
\quad \quad \text{A}25(\text{count}5,1) = \text{loc}(i,1); \\
\quad \quad \text{A}25(\text{count}5,2) = \text{loc}(i,2); \\
\quad \quad \text{A}25(\text{count}5,3) = \text{loc}(i,3); \\
\quad \quad \text{A}25(\text{count}5,4) = \text{loc}(i,4); \\
\quad \quad \text{A}25 = [\text{A}25]; \\
\text{end}
\]
else if loc(i,3)==20
    count6 = count6+1;
    A20(count6,1)=loc(i,1);
    A20(count6,2)=loc(i,2);
    A20(count6,3)=loc(i,3);
    A20(count6,4)=loc(i,4);
    A20=[A20];

else if loc(i,3)==15
    count7 = count7+1;
    A15(count7,1)=loc(i,1);
    A15(count7,2)=loc(i,2);
    A15(count7,3)=loc(i,3);
    A15(count7,4)=loc(i,4);
    A15=[A15];

end
end
end
end

Step 4: Generating all possible delivery combinations with respect to truck capacity

A=[15,20,25];
B=[];
Capacity=45;
D=[];
E=[];
len=length(A);

for i=1:len
    c=floor(Capacity/A(i));
    B=[B;A(i) c];

    for j=1:c
        D=[D A(i)];
    end
end
len_D=length(A);
perm=[];
perm2=[];

for i=1:1:3
    perm=nchoosek(D,i);
    [xdim ydim] = size(perm);
    for j=1:xdim
        if(ydim ~= len_D)
            perm1(j,:)=perm(j,:),zeros(1,len_D-ydim)];
        else
            perm1=perm;
        end
    end
    perm2=[perm2;perm1];
perm2=unique(perm2,'rows');
end

[xdim ydim]=size(perm2);
Dem_Combine=[];

for j=1:xdim
    if(sum(perm2(j,:))<= Capacity)
        Dem_Combine=[Dem_Combine;perm2(j,:)];
    end
end

[xdim ydim]=size(Dem_Combine);

for i = 1:xdim-1
    for j = (i + 1):1:xdim
        if (sum(Dem_Combine(i,:)) < sum(Dem_Combine(j,:)))
            temp=Dem_Combine (i,:);
            Dem_Combine(i,:)=Dem_Combine(j,:);
            Dem_Combine(j,:)=temp;
        elseif (sum(Dem_Combine(i,:)) == sum(Dem_Combine(j,:)))
            if (Dem_Combine (i,1) < Dem_Combine(j,1))
                temp=Dem_Combine (i,:);
                Dem_Combine(i,:)=Dem_Combine(j,:);
                Dem_Combine(j,:)=temp;
            end
        end
    end
end
**Step 5 & 6: Generation of separate feasibility route and assignment of trucks for each possible combination**

**Truck Assignment (Depot ⇒ 25 kl location ⇒ 20 kl location ⇒ Depot):**

dst=Dist_Matrix;
Trip_Assign=[];
truck2520=[];
temp=10000;
total_count5 = count5;
total_count6 = count6;
buffer=0;
flag=0;
d2520=0;

while(count5>0 && count6>0)
    temp = 10000;
    for i=1:total_count5
        for j=1:total_count6
            if(Valid_Index(A25(i,4))==0)
                if(Valid_Index(A20(j,4))==0)
                    if ((dst(A25(i,4),A20(j,4))< temp))
                        temp = dst(A25(i,4),A20(j,4));
                        dst25=sqrt((A25(i,1)^2 + A25(i,2)^2));
                        dst20=sqrt((A20(j,1)^2 + A20(j,2)^2));
                        d2520=temp+dst25+dst20;
                        if (d2520<240)
                            small25=A25(i,4);
                            small20=A20(j,4);
                            flag=1;
                            truck2520=[small25 small20 buffer d2520];
                        end
                    end
                end
            end
        end
    end
end

end
Step 7: Updating Valid Indexation index

```matlab
if (flag==1)
    if (Valid_Index(small25)==0)
        Valid_Index(small25)=1;
        Trip_Assign=[Trip_Assign;truck2520];
    end
end
end
```

Truck Assignment (Depot ⇨ 15 kl location ⇨ Depot):

```matlab
buffera=0;
bufferb=0;
truck15=[];
count7=total_count7;
total_count7=total_count7;
flag=0;
d15=0;

while(count7>0)
    temp = 10000;
    for i=1:total_count7
        if(Valid_Index(A15(i,4))==0)
            if (sqrt(A15(i,1)^2+A15(i,2)^2)< temp)
                temp = sqrt(A15(i,1)^2+A15(i,2)^2);
                d15=temp+temp;
                if (d15<240)
                    small15=A15(i,4);
                    flag=1;
                    truck15=[small15 buffera bufferb d15];
                end
            end
        end
    end
end
```
Step 7: Updating Valid Indexation index

```matlab
if (flag==1);
    if (Valid_Index(small15)==0)
        Valid_Index(small15)=1;
        flag=0;
        Trip_Assign=[Trip_Assign;truck15];
        dist=[dist;d15];
    end
end
```

Calculating total distance

```matlab
Tot_Dist=0;

for i=1: length (dist)
    Tot_Dist=Tot_Dist+dist(i);
end
```
Appendix C: Heuristic Algorithm in MATLAB for the Case study

**Heuristic inputs**

\[ n = 120; \]

Location points and demand values are taken as input from Appendix I.

**Step1: Generating Distance matrix**

\[ \text{Dist\_Matrix}=0; \]

\[ \text{for } j=1:n \]

\[ \quad \text{for } i=1:n \]

\[ \quad \quad a=([\text{loc}(j)-\text{loc}(i)]^2); \]
\[ \quad \quad b=([\text{loc}(j+n)-\text{loc}(i+n)]^2); \]
\[ \quad \quad c=a+b; \]
\[ \quad \quad \text{Dist\_Matrix}(i,j)=\sqrt{c}; \]
\[ \quad \quad a=0; \]
\[ \quad \quad b=0; \]

\[ \quad \text{end} \]

\[ \text{end} \]

**Step2: Creating Valid Indexation index**

\[ \text{Valid\_Index}=[]; \]

\[ \text{for } i=1:n \]

\[ \quad \text{Valid\_Index}(1,i)=0; \]
\[ \quad \text{Valid\_Index}=[\text{Valid\_Index}]; \]

\[ \text{end} \]

**Step3: Clustering locations with similar demand**

\[ \text{count5}=0; \]
\[ \text{count6}=0; \]
\[ \text{count7}=0; \]

\[ \text{count1} = n; \]
\[ A25=\text{zeros}(n,4); \]
\[ A20=\text{zeros}(n,4); \]
\[ A15=\text{zeros}(n,4); \]
for i=1:count1
    if loc(i,3)==25
        count5 = count5+1;
        A25(count5,1)=loc(i,1);
        A25(count5,2)=loc(i,2);
        A25(count5,3)=loc(i,3);
        A25(count5,4)=loc(i,4);
        A25=[A25];

    else if loc(i,3)==20
        count6 = count6+1;
        A20(count6,1)=loc(i,1);
        A20(count6,2)=loc(i,2);
        A20(count6,3)=loc(i,3);
        A20(count6,4)=loc(i,4);
        A20=[A20];

    else if loc(i,3)==15
        count7 = count7+1;
        A15(count7,1)=loc(i,1);
        A15(count7,2)=loc(i,2);
        A15(count7,3)=loc(i,3);
        A15(count7,4)=loc(i,4);
        A15=[A15];
        end
    end
end
end

Step 4: Generating all possible delivery combinations with respect to truck capacity

A=[15,20,25];
B=[];
Capacity=45;
D=[];
E=[];
len=length(A);

for i=1:len
    c=floor(Capacity/A(i));
    B=[B;A(i) c];

    for j=1:c
        D=[D A(i)];
    end
end

len_D=length(A);
perm=[];
perm2=[];

for i=1:1:3
    perm=nchoosek(D,i);
    [xdim ydim] = size(perm);

    for j=1:xdim
        if(ydim ~= len_D)
            perm1(j,:)=[perm(j,:),zeros(1,len_D-ydim)];
        else
            perm1=perm;
        end
    end
    perm2=[perm2;perm1];
    perm2=unique(perm2,'rows');
end

[xdim ydim]=size(perm2);
Dem_Combine =[];

for j=1:xdim
    if(sum(perm2(j,:))<= Capacity)
        Dem_Combine =[Dem_Combine ;perm2(j,:)];
    end
end

[xdim ydim]=size(Dem_Combine );
for i = 1:xdim-1
    for j = (i + 1):1:xdim
        if (sum(Dem_Combine (i,:)) < sum(Dem_Combine (j,:)))
            temp=Dem_Combine (i,:);
            Dem_Combine (i,:)=Dem_Combine (j,:);
            Dem_Combine (j,:)=temp;
        elseif (sum(Dem_Combine (i,:)) == sum(Dem_Combine (j,:)))
            Dem_Combine (i,:);
            if (Dem_Combine (i,1) < Dem_Combine (j,1))
                temp=Dem_Combine (i,:);
                Dem_Combine (i,:)=Dem_Combine (j,:);
                Dem_Combine (j,:)=temp;
            end
        end
    end
end
Step 5 & 6: Generation of separate feasibility route and assignment of trucks for each possible combination

Truck Assignment (Depot $\rightarrow$ 25 kl location $\rightarrow$ 20 kl location $\rightarrow$ Depot):

Trip_Assign=[];
dst=Dist_Matrix;
truck2520=[];
temp=10000;
total_count5 = count5;
total_count6 = count6;
buffer=0;
flag=0;
d2520=0;
dist=[];

while(count5>0 && count6>0)
    for i=1:total_count5
        for j=1:total_count6
            if(Valid_Index(A25(i,4))==0)
                if(Valid_Index(A20(j,4))==0)
                    if((dst(A25(i,4),A20(j,4))< temp))
                        temp = dst(A25(i,4),A20(j,4));
                        dst25=sqrt((A25(i,1)^2 + A25(i,2)^2));
                        dst20=sqrt((A20(j,1)^2 + A20(j,2)^2));
                        d2520=temp+dst25+dst20;
                        if(d2520<240)
                            small25=A25(i,4);
                            small20=A20(j,4);
                            flag=1;
                            truck2520=[small25 small20 buffer d2520];
                        end
                    end
                end
            end
        end
    end
end
Updating Valid Indexation Index

```plaintext
if (flag==1)
    if (Valid_Index(small25)==0)
        if (Valid_Index(small20)==0)
            Valid_Index(small25)=1;
            Valid_Index(small20)=1;
            Trip_Assign=[Trip_Assign;truck2520];
            dist=[dist;d2520];
            flag=0;
        end
    end
end

count5=count5-1;
count6=count6-1;
end

Truck Assignment (Depot ⇒ 15 kl location ⇒ 15 kl location ⇒ 15 kl location ⇒ Depot):

Trip_truck15cube=[];
temp=10000;
total_count7 = count7;
flag=0;
d151515=0;

while(count7>=3)
    for i=1:total_count7-2
        for j=i+1:total_count7-1
            for k=j+1:total_count7
                if(Valid_Index(A15(i,4))==0)
                    if(Valid_Index(A15(j,4))==0)
                        if(Valid_Index(A15(k,4))==0)
                            if ((dst(A15(i,4),A15(j,4)))+
                                (dst(A15(i,4),A15(k,4)))+
                                (dst(A15(j,4),A15(k,4))))< temp)
                                temp = ((dst(A15(i,4),A15(j,4)))
                                + (dst(A15(i,4),A15(k,4)))
                                + (dst(A15(j,4),A15(k,4))));
                                dst15a=sqrt((A25(i,1)^2 + A25(i,2)^2));
                                dst15b=sqrt((A20(k,1)^2 + A20(k,2)^2));
                                d151515=temp+dst15a+dst15b;
```
if (d151515<240)
    small15cube1=A15(i,4);
    small15cube2=A15(j,4);
    small15cube3=A15(k,4);
    flag=1;
    truck15cube=[small15cube1 small15cube2
    small15cube3 d151515];
end
end
end
end
end
end
end

Updating Valid Indexation index

if (flag==1);
    if (Valid_Index(small15cubel)==0)
        if (Valid_Index(small15cube2)==0)
            if (Valid_Index(small15cube3)==0)

                Valid_Index(small15cube1)=1;
                Valid_Index(small15cube2)=1;
                Valid_Index(small15cube3)=1;
                flag=0;
                Trip_Assign=[Trip_Assign;truck15cube];
                dist=[dist;d151515];

            end
        end
    end
end
end

end
count5=count5-1;
count6=count6-1;
count7=count7-1;
end
Truck Assignment (Depot ⇔ 25 kl location ⇔ 15 kl location ⇔ Depot):

buffer=0;
truck2515=[];
temp=10000;
count5=total_count5;
count7=total_count7;
total_count5 = count5;
total_count7 = count7;
flag=0;
d2515=0;
while(count5>0 && count7>0)

    for i=1:total_count5
        for j=1:total_count7
            if(Valid_Index(A25(i,4))==0)
                if(Valid_Index(A15(j,4))==0)
                    if((dst(A25(i,4),A15(j,4))< temp))
                        temp = dst(A25(i,4),A15(j,4));
                        dst25=sqrt((A25(i,1)^2 + A25(i,2)^2));
                        dst15=sqrt((A15(j,1)^2 + A15(j,2)^2));
                        d2515=temp+dst25+dst15;

                        if(d2515<240)
                            small25=A25(i,4);
                            small15=A15(j,4);
                            flag=1;
                            truck2515=[small25 small15 buffer d2515];
                        end
                    end
                end
            end
        end
    end

    Updating Valid_Indexation index

        if (flag==1)
            if (Valid_Index(small25)==0)
                if (Valid_Index(small15)==0)

                    Valid_Index(small25)=1;
                    Valid_Index(small15)=1;
                    Trip_Assign=[Trip_Assign;truck2515];
                    dist=[dist;d2515];
                    flag=0;
                end
            end
        end
    end
    count5=count5-1;
    count7=count7-1;
end
Truck Assignment (Depot ⇨ 20 kl location ⇨ 20 kl location ⇨ Depot):

buffer=0;
truck2020=[];
temp=10000;
count6=total_count6;
total_count6 = count6;
flag=0;
d2020=0;

while(count6>=2)
    temp = 10000;
    for i=1:total_count6-1
        for j=i+1:total_count6
            if(Valid_Index(A20(i,4))==0)
                if(Valid_Index(A20(j,4))==0)
                    if ((dst(A20(i,4),A20(j,4)))< temp)
                        temp = ((dst(A20(i,4),A20(j,4))));
                        dst20A=sqrt((A20(i,1)^2 + A20(i,2)^2));
                        dst20b=sqrt((A20(j,1)^2 + A20(j,2)^2));
                        d2020=temp+dst20A+dst20b;
                        
                        if (d2020<240)
                            small20A=A20(i,4);
                            small20b=A20(j,4);
                            flag=1;
                            truck2020=[small20A small20b buffer d2020];
                        end
                        end
                    end
                end
            end
        end
    end
    count6=count6-1;
end

Updating Valid Indexation index

if (flag==1)
    if (Valid_Index(small20A)==0)
        if (Valid_Index(small20b)==0)
            Valid_Index(small20A)=1;
            Valid_Index(small20b)=1;
            Trip_Assign=[Trip_Assign;truck2020];
            dist=[dist;d2020];
            flag=0;
        end
    end
end

count6=count6-1;
end
Truck Assignment (Depot ⇔ 20 kl location ⇔ 15 kl location ⇔ Depot):

buffer=0;
truck2015=[];
temp=10000;
count6=total_count6;
count7=total_count7;
total_count6 = count6;
total_count7 = count7;
flag=0;
d2015=0;

while(count6>0 && count7>0)
temp = 10000;
for i=1:total_count6
    for j=1:total_count7
        if (Valid_Index(A20(i,4))==0)
            if (Valid_Index(A15(j,4))==0)
                if ((dst(A20(i,4),A15(j,4))< temp))
                    temp = dst(A20(i,4),A15(j,4));
                    dst20=sqrt((A20(i,1)^2 + A20(i,2)^2));
                    dst15=sqrt((A15(j,1)^2 + A15(j,2)^2));
                    d2015=temp+dst20+dst15;
                    if (d2015<240)
                        small20=A20(i,4);
                        small15=A15(j,4);
                        flag=1;
                        truck2015=[small20 small15 buffer d2015];
            end
        end
    end
end

Updating Valid Indexation index

if (flag==1)
    if (Valid_Index(small20)==0)
        if (Valid_Index(small15)==0)
            Valid_Index(small20)=1;
            Valid_Index(small15)=1;
            Trip_Assign=[Trip_Assign;truck2015];
            dist=[dist;d2015];
            flag=0;
        end
    end
end
count6=count6-1;
count7=count7-1;
end
Truck Assignment (Depot \(\Rightarrow\) 15 kl location \(\Rightarrow\) 15 kl location \(\Rightarrow\) Depot):

```
buffer=0;
truck1515=[];
season=10000;
count7=total_count7;
total_count7=count7;
flag=0;
d1515=0;

while(count7>=2)
    temp=10000;
    for i=1:total_count7-1
        for j=i+1:total_count7
            if (Valid_Index(A15(i,4))==0)
                if (Valid_Index(A15(j,4))==0)
                    if (((dst(A15(i,4),A15(j,4)))< temp)
                        temp = ((dst(A15(i,4),A15(j,4))));
                        dst15A=sqrt((A15(i,1)^2 + A15(i,2)^2));
                        dst15b=sqrt((A15(j,1)^2 + A15(j,2)^2));
                        d1515=temp+dst15A+dst15b;
                        if (d1515<240)
                            small15A=A15(i,4);
                            small15b=A15(j,4);
                            flag = 1;
                            truck1515=[small15A small15b buffer d1515];
                    end
                end
            end
        end
    end
    count7=count7-1;
end
```

Updating Valid Indexation index

```
if(flag==1)
    if (Valid_Index(small15A)==0)
        if (Valid_Index(small15b)==0)
            Valid_Index(small15A)=1;
            Valid_Index(small15b)=1;
            Trip_Assign=[Trip_Assign;truck1515];
            dist=[dist;d1515];
            flag = 0;
        end
    end
end
count7=count7-1;
```
Truck Assignment (Depot ⇔ 25 kl location ⇔ Depot):

buffera=0;
bufferb=0;
truck25=[];
temp=10000;
count5=total_count5;
total_count5=count5;
flag=0;
d25=0;

while(count5>0)
    temp = 10000;
    for i=1:total_count5
        if(Valid_Index(A25(i,4))==0)
            if (sqrt(A25(i,1)^2+A25(i,2)^2)< temp)
                temp = sqrt(A25(i,1)^2+A25(i,2)^2);
                d25=temp+temp;

                if (d25<240)
                    flag=1;
                    small25=A25(i,4);
                    truck25=[small25 buffera bufferb d25];
                end
            end
        end
    end
    count5=count5-1;
end

Updating Valid Indexation index

if (flag==1);
    if (Valid_Index(small25)==0)
        Valid_Index(small25)=1;
        flag=0;
        Trip_Assign=[Trip_Assign;truck25];
        dist=[dist;d25];
    end
end

92
Truck Assignment (Depot ⇐ 20 kl location ⇐ Depot):

buffera=0;
bufferb=0;
truck20=[];
temp=10000;
count6=total_count6;
total_count6=total_count6;
flag=0;
d20=0;

while(count6>0)
    temp = 10000;
    for i=1:total_count6
        if(Valid_Index(A20(i,4))==0)
            if (sqrt(A20(i,1)^2+A20(i,2)^2)< temp)
                temp = sqrt(A20(i,1)^2+A20(i,2)^2);
                d20=temp+temp;

                if (d20<240)
                    small20=A20(i,4);
                    flag=1;
                    truck20=[small20 buffera bufferb d20];
                end
            end
        end
    end
end

ing Valid Indexation index

if (flag==1);
    if (Valid_Index(small20)==0)

        Valid_Index(small20)=1;
        flag=0;
        Trip_Assign=[Trip_Assign;truck20];
        dist=[dist;d20];

    end
end

count6=count6-1;
end
Truck Assignment (Depot ⇛ 15 kl location ⇛ Depot):

buffera=0;
bufferb=0;
Trip_Assign5=[];
temp=10000;
count7=total_count7;
total_count7=count7;
flag=0;
d15=0;
while(count7>0)
    temp = 10000;
    for i=1:total_count7
        if (Valid_Index(A15(i,4))==0)
            if (sqrt(A15(i,1)^2+A15(i,2)^2)< temp)
                temp = sqrt(A15(i,1)^2+A15(i,2)^2);
                truck15=temp+temp;
                if (d15<240)
                    small15=A15(i,4);
                    flag=1;
                    d15=[small15 buffera bufferb d15];
                    end
            end
        end
    end
end

Updating Valid Indexation index

if (flag==1);
    if (Valid_Index(small15)==0)
        Valid_Index(small15)=1;
        flag=0;
        Trip_Assign=[Trip_Assign;truck15];
        dist=[dist;d15];
end
end

Calculating total Distance

Tot_Dist=0;
for i=1:length(dist)
    Tot_Dist=Tot_Dist+dist(i);
end