SOLVING XPRESSBEES LOGISTICS PROBLEM USING EXACT AND HEURISTIC METHOD

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF SCIENCE
IN
[MATHEMATICS]

Submitted by :

[Swati Malhotra](2K20/MSCMAT/31)
[Mitali Khandelwal](2K20/MSCMAT/42)

Under the supervision of

[Prof.L N Das]

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Bawana Road,Delhi -110042

7 MAY 2022

CANDIDATE'S DECLARATION

We (Swati Malhotra),2K20/MSCMAT/31 and (Mitali Khandelwal),2K20/MSCMAT/42 of M.Sc(Mthematics),hereby declare that the project Dissertation titled " Xpressbees Logistics Problem using Exact and Heuristic Method" which is submitted by us to the Department of Applied Mathematics, Delhi Technological University ,Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science ,is original and not copied from any source without proper citation.This work has not previously formed the basis for the award of any Degree,Diploma Associateship ,Fellowship or any other similar title or recognition.

Place: Delhi
Date:

Swati Malhotra

Mitali Khandelwal]

CERTIFICATE

We hereby certify that the project Dissertation titles ”Solving Xpressbees Logistics Problem using Exact and Heuristic Method ” which is submitted by Swati Malhotra (2K20/MSCMAT/31) and Mitali Khandelwal(2K20/MSCMAT/42) to the department of Applied Mathematics ,this document submitted to Delhi Technological University as part of the requirements for the Master of Science degree ,is the record of project work, completed by students under my supervision.This thesis has not been applied in part or full for any degree or diploma at the university or elsewhere, to the best of my knowledge.

Place :Delhi
Date:

Prof. L N Das
Department of Applied Mathematics
Delhi Technological University

ACKNOWLEDGEMENT

Throughout the writing of this dissertation, We received a lot of help and encouragement.
First and foremost,we want to express our gratitude to Professor Mr. L.N Das, who was very helpful in developing the research topics and methodology. His informative remarks encouraged us to improve our thoughts and raise the quality of our work.He provided us with the tools that we needed to choose the right direction and successfully complete our dissertation.
In addition, We would like to thank God and our parents for their wise counsel and sympathetic ear.

ABSTRACT

Finding a route with shortest distance travelled according to various conditions can help travellers to make better route choice decisions.The main purpose of this paper is to solve the routing problem for the company(Xpressbees) fleet vehicle in order to find the optimal route subject to various constraints such that the transportation cost is improved using real data.
It applies the following methodology.First it gives a brief introduction to the formulation of our problem then it characterizes into process of delivering products in a logistic company. As a result, it identifies the principal scientific challenges that need to be addressed other than finding shortest distance like time dependency , traffic conditions and various constraints like vehicle capacity and customer demands.
This paper includes exact and heuristic methods to find optimal solution. In the initial part the branch and bound penalty exact method is used after that it includes Dijkstra's algorithm with signalized intersections. In this part,It includes the lost time occur at the intersections of the different signals calculated using Highway Capacity Manual.
After that this paper incorporates three scenarios Truck Route Optimization , Last Mile Delivery and Hyper Local Delivery and in all of the scenarios the heuristic approach is used. In Truck Route optimization dynamic programming is used using backward recursion approach. In Last Mile Delivery vehicle routing problem with capacity constraint(VRPC) is used which is solved using clarks and wright saving based algorithm and in then hyperlocal delivery the vehicle routing problem with capacity and time windows(VRPCTW) constraint is used which is solved using Holmes and Parker heuristic method.In the last we can also observed that Holmes and Parker heuristic is better than clarks and wright saving based algorithm and gives better solution.

List of Figures

List of Tables

Contents

1 Chapter 1

1.1 INTRODUCTION

Globalization is a fact that has resulted in significant economic developments, with the demand for goods on our globe increasing by the day. By development, this fact creates a significant benefit in Supply Chain Management. Supply chain management is the most important focus of competitive advantage in the business industry. In logistics, it is the primary activity, which emphasises the importance of focusing on how to manage it. Organizations, customers, activities, and information are the basic components of the supply chain, and there is interaction between them. SCM is separated into three categories: purchasing, production, and transportation. This is one of the most significant characteristics. It can be described as a cycle that begins with the sort of material to be used, then moves on to the firm and customers, and finally to the mode of delivery.The focus is on how and when raw resources become items and are transferred to end users. International trade became prevalent as a result of the high demand, with a rapidly growing market share. Because of the competition among logistics companies, transportation and prices are rising as the number of costs grows. This means that the high production of goods cannot be consumed in a short period of time, necessitating the necessity for goods transportation. This reality gives rise to a plethora of transportation businesses and modes. Demand and customers are also increasing. In such a setting, issues such as transportation cost, mode of transportation, and transportation efficiency arise.
The Routing Problem provides solutions to different types of difficulties . The basic goal of this task is to construct a route with the lowest delivery cost, starting with a depot that serves a set of customers. As a result, it's an optimization problem that necessitates finding the best solution for a number of routes served by a fleet of vehicles with a set of customers. This issue contains a number of constraints, including the requirement that each customer be visited exactly once by one route.
TSP and VRP are prevalent issue for businesses that deal with the transportation of goods. As a result, there are various algorithms that provide a solution to this problem, such as the Branch and Bound Penalty Method , Dijkstra's Algorithm , Dynamic Programming ,Clark and Wright saving based algorithm and Homes and Parker Heuristic. These algorithm returns the best option, which is the cheapest route.

1.2 LITERATURE SURVEY

1.2.1 Logistics

In a business setting, logistics refers to the management of commodities movement from point of origin to point of consumption to meet the needs of customers or businesses. In logistics, physical goods such as food, materials, equipment, and liquids, as well as abstract items such as time and information, are all managed. Physical item logistics includes information flow, material handling, production, packing, transit, and warehousing.One approach to think about firm logistics is ”having the right item in the right amount at the right time at the right place for the right price in the right condition to the right consumer.” A logistics operations manager will be involved in shipping, warehousing, and the planning and organisation of these activities. Internal (inbound logistics) or exterior (outbound logistics) logistics may be the focus (outbound logistics).

1.2.2 Inbound Logistics

An inbound logistics manager is in charge of all aspects of the company’s incoming flow of resources needed to make its goods or services. Contacts with suppliers will be managed, raw materials will be obtained, material costs will be negotiated, and faster delivery will be provided.

1.2.3 Outbound Logistics

Storage and transit are two concerns that outbound logistics managers are concerned about. He or she will use warehousing techniques to keep the finished goods safe and accessible. Proper planning is required because the products may need to be delivered to a consumer at any time. Because stored products do not produce income, it is often preferable to have as little stock as possible on hand. As a result, the outbound logistics manager must often strike a compromise between firm cost savings and consumer demand. The transportation aspect in outbound logistics is by far the most complicated. Without transportation, there is no logistics.As a result, it’s critical to be able to move items from one location to another as quickly, cost-effectively, and efficiently as possible. Issues such as delays and fluctuations in fuel costs must be examined in order to cover all possible scenarios that could affect the effective flow of products.

1.3 Supply Chain Management

The management of a product’s or service’s entire manufacturing flow, from raw materials to delivery of the finished product to the client, is referred to as supply chain management. To transport the product from raw material suppliers to organisations that engage directly with clients, a company builds a network of suppliers (”links” in the chain).
Effective supply chain management systems decrease cost, waste, and time during the manufacturing process. The industry norm is a just-in-time supply chain, in which retail sales instantly convey replenishment orders to manufacturers. Following that, store shelves can be restocked almost as quickly as things are sold. One strategy for improving this process is to analyse data from supply chain partners to see where more adjustments can be made.

1.3.1 Importance of SCM

Detecting potential issues
The buyer may be unsatisfied with the service if the client asks more items than the producer can deliver. Through data analysis, manufacturers may be able to forecast a scarcity before the buyer is dissatisfied.
Dynamic pricing optimization
Seasonal products have a finite shelf life. Toward the conclusion of the season, these things are frequently dumped or sold at a significant discount. Airlines, hotels, and other businesses selling perishable items adjust tariffs on the fly to meet demand. Using analytic tools, similar forecasting strategies can enhance profits even for hard commodities.
Improving the distribution of inventory that is "available to promise"
Based on sales estimates, actual orders, and promised raw material delivery, analytical software solutions aid in the dynamic allocation of resources and scheduling of work. Manufacturers can confirm a product delivery date when an order is placed, reducing the amount of orders that are filled incorrectly.

1.4 Process in delivering products in SCM

1. The First Step is to collect orders from the sellers(vendors) to the distribution centres.The Logistics Companies have various Distribution Centres for collecting the orders from their sellers.

2. After that,Each Products has moved to a Common center Known as 'HUB' through a vehicle.Hub comprises of products comes from all the distribution centres where the products is arranged into sacks depends upon the pin code.

3. Once the Order reached its hub of their respective city,They are sorted depends upon the pin code,and then Delivery Boys comes and pick it up and deliver to the customers.

1.5 Travelling Salesman Problem

The TSP is an algorithmic problem in which the goal is to find the shortest path between a set of points and places. Cities that a salesman might visit are represented by the points in the problem statement. The salesperson's goal is to keep travel expenses and distance travelled to a minimum. TSP is a computer science algorithm that focuses on determining the most effective data transmission path between nodes.

Salesman on the Move Last-mile delivery agents are faced with a problem. It's an attempt to discover the fastest route between numerous cities/destinations and back to your starting point. Given the multiple delivery-related constraints such as traffic and so on, it is currently a complicated issue.

By overcoming the TSP barrier, supply chains can become more efficient and logistics costs can be reduced. TSP is a simple problem to define but a challenging challenge to solve.

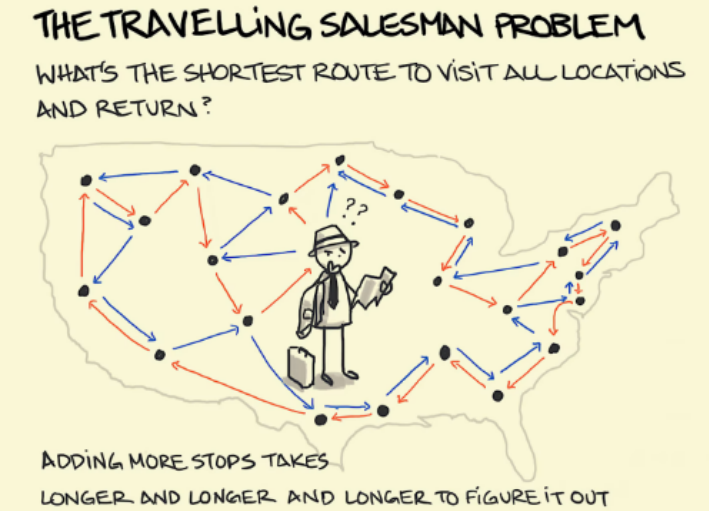


Figure 1: Travelling Salesman Problem

1.6 Vehicle Routing Problem

The Traveling Salesman Problem is a subset of the Vehicle Routing Problem. It's a combinatorial and integer linear programming problem that aims to find the optimal path for a fleet of trucks to deliver to a set of consumers. All consumers who are visited by a vehicle form a tour. Each customer can only be served once, and each tour must start and end at the depot.

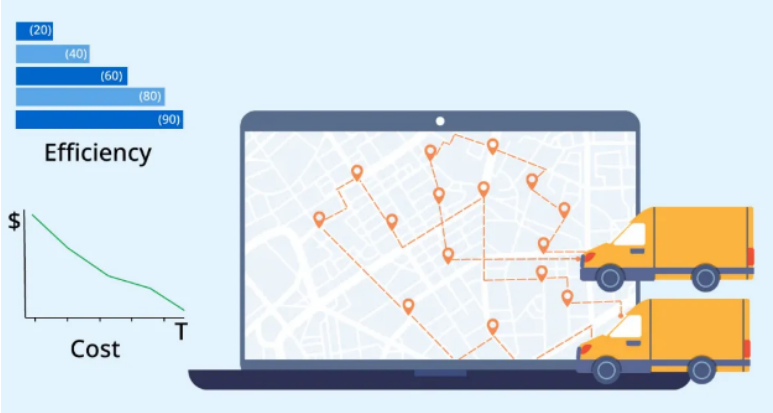


Figure 2: Vehicle Routing Problem

The Vehicle Routing Problem is about determining the optimum routes for a fleet of vehicles given operational constraints like time window, route length, and so on. It aids fleet managers in route planning to maximise fleet efficiency while minimising last-mile delivery expenses.

1.7 Capacitated vehicle routing problem

The capacitated vehicle routing problem (CVRP) is a vehicle routing problem in which vehicles with restricted carrying capacity must pick up or deliver things at many locations. The things have a quantity, such as their weight or volume, while the vehicles

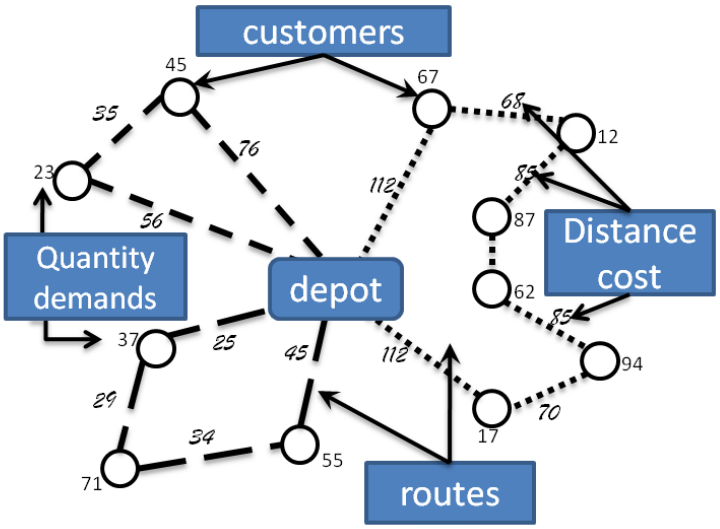


Figure 3: CVRP

have a carrying capacity. The challenge is to pick up or deliver the products for the least amount of money while never exceeding the vehicles' capacity.

1.8 Capacitated Vehicle Routing Problem with Time-Windows

In the capacitated vehicle routing problem with time-windows, a fleet of delivery trucks with uniform capacity must serve clients with known demand and opening hours for a certain commodity (CVRPTW). The vehicles all start and finish their journeys at the same station. Each consumer may only be served by one car. The objectives are to reduce fleet size and assign a sequence of customers to each truck in the fleet while reducing total distance travelled, ensuring that all customers are served and that each vehicle's capacity is not exceeded.

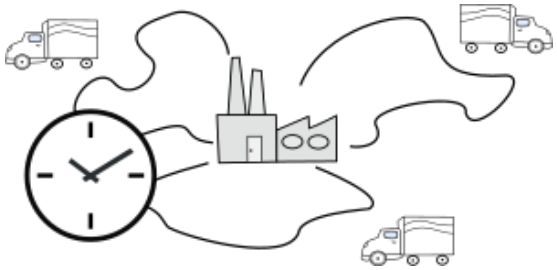


Figure 4: CVRPTW

1.9 Methods to solve Routing Problem

- Branch and Bound Penalty Method
  The travelling salesman problem is solved using a "branch and bound with penalty" technique. A process known as branching divides the set of all tours (possible solutions) into increasingly smaller subsets. A lower bound on the length of the

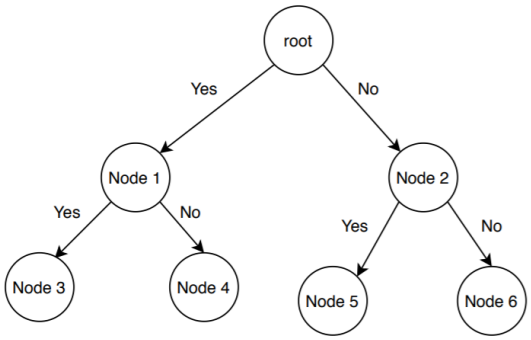


Figure 5: Branch and Bound Pen Method

tours in each subset is calculated. Eventually, a subset is discovered that has a single tour that is shorter than or equal to some lower constraint for each tour. The branching motivation and computation of the bottom bounds are based on concepts that are often utilised in assignment problems. In terms of computation, the technique increases the amount of problem that may be tackled without resorting to problem-specific approaches.

- Dijkstra's Shortest Path Algorithm
  The Dijkstra algorithm's main premise is to discover the shortest path from the source point (marked as s) to the outside progressively. Assign a number to each point (the label of this point) that represents the weight of the shortest path from s to this point (the P label) or the upper bound of the weight of the shortest path from s to this point (the W label) (named as T label).We can determine the shortest path from s to each point in n-1 steps by changing the point with T label to a point with P label and increasing the number of vertex with P label in graph G by one (n is the number of vertexes in graph G).

Figure 6: Dijkstra's Shortest Path Algorithm

- Shortest Path with Dynamic Programming

Dynamic Programming exploits the optimal sub-structure of a problem. The problem has an optimal sub-structure if the optimum answer to the problem contains ideal answers to smaller sub-problems. In the shortest path problem, there is an optimal sub-structure.

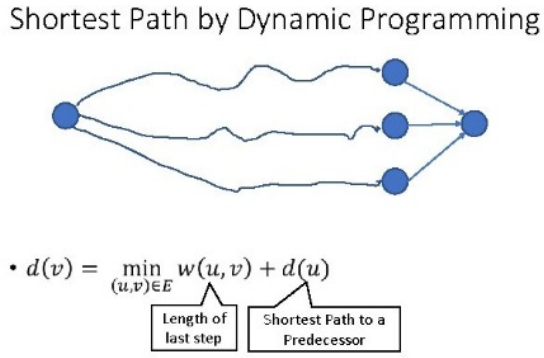


Figure 7: Shortest Path with Dynamic Programming

- Clarke and Wright Savings based Algorithm
  The most well-known approach to the VRP problem is Clarke and Wright's "savings" algorithm. Its core idea is fairly obvious. Consider the following scenario: a depot with D demand points and n demand points. Assume that the first solution to the VRP is dispatching one vehicle to each of the n demand sites, with n vehicles in total.
  If we now employ a single vehicle to serve two points, say I and j, on a single journey, the total distance travelled is cut in half.

$$S_{i,j}=d_{0,i}+d_{0,j}-d_{i,j}$$

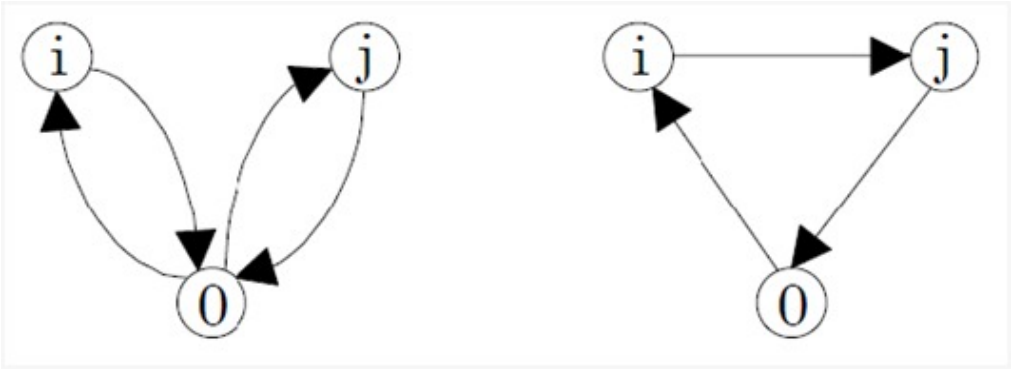


Figure 8: Clarke and Wright Savings based Algorithm

The quantity s(i, j) is known as the "savings" resulting from combining points i and j into a single tour.

- Holmes and Parker Heuristic
  The Clarke and Wright algorithm has several drawbacks of its own. It is possible that it will not always present us with the optimal solution. People try harder to

work harder and come up with better ideas than Clarke and Wright. A better result can be obtained using the Holmes and Parker algorithm, which is superior to the Clarke and Wright solution.

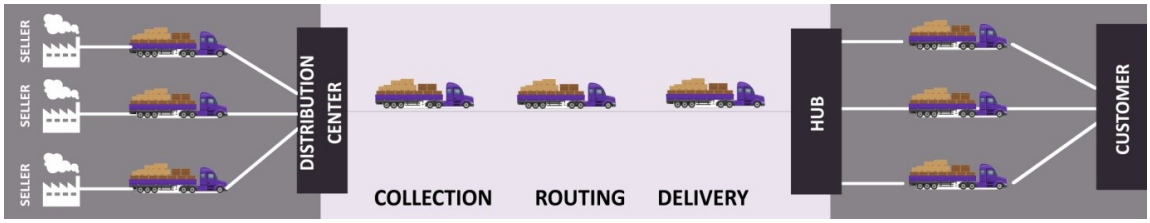
2 Chapter 2

2.1 PROBLEM FORMULATION

A logistic company Xpreessbees situated in banglore is liable for delivering items within a country and in hyperlocal region .Products are sent through established couriers.The Company has its own warehouses ,distributions centres, hubs and motherhubs to deliver the items safely.The Bengaluru based warehouse once handled all orders in South India but In the most recent couple of years, XpressBees has opened various such distribution centers everywhere, in a bid to enormously extend its store network organization.Each item is bundled perfectly and stacked at its predetermined area on the warehouses. Huge machines are put on the racks and moved around using forklifts.Before the order makes its way to the dispatch section, it is wrapped to prevent damage during transportation.Trucks prepared to ship items across the India and when the orders are prepared and recorded by a software, they are loaded onto the waiting trucks and dispatched.
As previously said, Xpressbees is a well-known logistic firm with a large number of customers; therefore, it is their job to deliver the merchandise on time. Although the organisation operates smoothly, the delivery person does not have a set path. As a result, the delivery person has complete control over the sequence in which he or she must travel to each seller/customer to pick up/deliver the items. They pass past the same spot twice in multiple instances. Inefficient route design could result in higher costs for the company.
The Company's primary goal is to improve the efficiency of the routes so that each delivery person drives between their specific routes, which is accomplished by solving a well-known optimization problem, the Traveling Salesman Problem Vehicle Routing Problem, with various exact and heuristic methods.

2.2 Procedure of Delivering Items in a Logistics Company

The procedure of delivering items in a logistics company is shown in Figure .



2.3 About XpressBees

As it refers previously, Xpressbees has numerous hubs almost everywhere and it delivers items all over the india which means that for educational reason it is hard to utilize data of all urban communities.
For our research analysis,we are considering Delhi. The sellers and customers are Delhi-based.There are numerous sellers and numerous customers on a marketplace and almost everyday alot of orders have placed by customers. Xpressbees delivers orders from seller to customer securely. After communication with the company,logistic manager provide the data from one year delivery which implies that the data are considerably more. Thus, we choose to restricted this data and we use information just in delhi .Xpressbees, even however it is a huge organization don't use a particular algorithm to address the routing problem. Thus, with this research analysis we will give an outcome such that they utilize more functional technique.

2.4 Application of Exact Heuristic method in taking care of logistic issue of Xpressbees

2.4.1 Picking the orders from sellers to the Distribution centres

In Delhi distribution centres are selected in different places.We are considering Rohinisector24 as the distribution center. For futher research,10 sellers are picked for this distribution centre from where a delivery person go and pickup the order.
The Distribution centre (Rohini,Sector 24) situtated in North region.
The selected sellers destination point are as follows:
1)Gtb Nagar, 2) Kamla Nagar, 3) Badli, 4) Rithala , 5) Chandni Chowk , 6) Civil Lines,7) TimarPur, 8) Shakti Nagar, 9) Roop Nagar, 10) Malka Ganj
In the Microsoft Access database environment, a database is created with appropriate distances between and distribution centres and seller areas.

Serial_Number	Places	Distance
1	Gtb Nagar	18.1
2	KamlaNagar	23.7
3	Badli	8.2
4	Rithala	3.2
5	Chandni Chowk	25.4
6	Civil Lines	22.5
7	Timarpur	19.4
8	Shakti Nagar	17
9	Roop Nagar	19.6
10	Malka Ganj	23.3

Figure 9: DataBase of distances between North DC and Sellers

Assortment of orders from sellers to Rohini,sector-24(Distribution centre)

For the convenience of further calculations,we are denoting Rohini,Sector-24(depot) as 0 and areas to be visit Gtb Nagar as 1, Kamla Nagar as 2,Badli as 3,Rithala as 4,Chandni Chowk as 5,CivilLines as 6,TimarPur as 7,Shakti Nagar as 8,Roop Nagar as 9 and Malka Ganj as 10.

The main idea is to find an optimal solution to avoid passing the same point twice like in TSP using exact method(Branch and Bound)and It should be noted that this program has solved manually. A set of location pairs, covering as many areas as possible, is selected.For each pair of areas,the distance for that pair is figured by google maps.For all pairs, the Google Maps distance,is set up in a table.

Table to entry the data for further computations is created,namely the distance matrix.

The Process of of calculating the traveling salesman problem manually incorporates the following steps:

Step-1: Find the smallest element in each row and subtract it from that row.

So with the help of Table 1 we find the row minimum and proceed with the step 1.

So, row minimum will be 28.8. (3.2+1.9+1+7.1+3.3+4.8+2.2+2.1+1.2+1+1=28.8)

Step-2: Find the smallest element in each column and subtract it from that column.

So with the help of Table 2 we find the column minimum and proceed with the step 2.

Distance	0	1	2	3	4	5	6	7	8	9	10
0	-	18.1	23.7	8.2	3.2	25.4	22.5	19.4	17	19.6	23.3
1	15	-	2.7	12	14	8.5	2.8	2.4	2.7	1.9	3.1
14 2	13	3	-	29	16	6.7	4.4	3.7	1.2	1	1
3	7.1	10	13	-	11	18	14	11	10	10	13
4	3.3	15	18	5.5	-	17	19	14	12	13	13
5	18	7.8	6.5	17	26	-	4.8	7	7.3	7.6	5.2
6	19	3	3.9	14	22	5.7	-	2.2	4.9	4.4	3.4
7	17	2.1	4.1	11	19	7.9	2.3	-	4	3.5	3.6
8	12	3.1	1.6	9.1	15	6.5	5	4.3	-	1.2	1.9
9	13	2.1	1.1	11	15	8.3	4.5	3.8	1	-	1.6
10	14	3.6	1	16	14	6.8	3.9	3.2	1.7	1.6	-

Table 1: Distance Matrix

Distance	0	1	2	3	4	5	6	7	8	9	10	RowMinimum
0	-	14.9	20.5	5	0	22.2	19.3	16.2	13.8	16.4	20.1	3.2
1	13.1	-	0.8	10.1	12.1	6.6	0.9	0.5	0.8	0	1.2	1.9
2	12	2	-	28	15	5.7	3.4	2.7	0.2	0	0	-1
3	0	2.9	5.9	-	3.9	10.9	6.9	3.9	2.9	2.9	5.9	7.1
4	0	11.7	14.7	2.2	-	13.7	15.7	10.7	8.7	9.7	9.7	3.3
5	13.2	3	1.7	12.2	21.2	-	0	2.2	2.5	2.8	0.4	4.8
6	16.8	0.8	1.7	11.8	19.8	3.5	-	0	2.7	2.2	1.2	2.2
7	14.9	0	2	8.9	16.9	5.8	0.2	-	1.9	1.4	1.5	2.1
8	10.8	1.9	0.4	7.9	13.8	5.3	3.8	3.1	-	0	0.7	1.2
9	12	1.1	0.1	10	14	7.3	3.5	2.8	0	-	0.6	1
7 10	13	2.6	0	15	13	5.8	2.9	2.2	0.7	0.6	-	1

Table 2: Distance Matrix 2

Distance	0	1	2	3	4	5	6	7	8	9	10
0	-	14.9	20.5	2.8	0	18.7	19.3	16.2	13.8	16.4	20.1
1	13.1	-	0.8	7.9	12.1	3.1	0.9	0.5	0.8	0	1.2
2	12	2	-	25.8	15	2.2	3.4	2.7	0.2	0	0
3	0	2.9	5.9	-	3.9	7.4	6.9	3.9	2.9	2.9	5.9
4	0	11.7	14.7	0	-	10.2	15.7	10.7	8.7	9.7	9.7
5	13.2	3	1.7	10	21.2	-	0	2.2	2.5	2.8	0.4
6	16.8	0.8	1.7	9.6	19.8	0	-	0	2.7	2.2	1.2
7	14.9	0	2	6.7	16.9	2.3	0.2	-	1.9	1.4	1.5
8	10.8	1.9	0.4	5.7	13.8	1.8	3.8	3.1	-	0	0.7
9	12	1.1	0.1	7.8	14	3.8	3.5	2.8	0	-	0.6
10	13	2.6	0	12.8	13	2.3	2.9	2.2	0.7	0.6	-
Col	0	0	0	2.2	0	3.5	0	0	0	0	0

Table 3: Distance Matrix 3

So, column minimum will be 5.7. (0+0+0+2.2+0+3.5+0+0+0+0+0=5.7)
Since we know that sum of row minimum and column minimum will give

us the lower bound.we get lower bound as 28.8 + 5.7 = 34.5

Calculate the penalty of all 0's in Table 3

It is important to mentioned that,as in assignment problem we would like to make assignments where there are zeros such that the delivery person don't travel extra distance either to arrive at that hub or to leave that hub.This is the main idea in calculating penalties such that no delivery person will travel additional distance.

There are two penalties one is row penalty and other one is column penalty.Row penalty and column penalty are both the difference between zero and the next highest element and the Total penalties is the sum of row penalty and column penalty.

Distance	0	1	2	3	4	5	6	7	8	9	10
0	-	14.9	20.5	2.8	0(6.7)	18.7	19.3	16.2	13.8	16.4	20.1
1	13.1	-	0.8	7.9	12.1	3.1	0.9	0.5	0.8	0(0.5)	1.2
2	12	2	-	25.8	15	2.2	3.4	2.7	0.2	0(0)	0(0.4)
3	0(2.9)	2.9	5.9	-	3.9	7.4	6.9	3.9	2.9	2.9	5.9
4	0(0)	11.7	14.7	0(2.8)	-	10.2	15.7	10.7	8.7	9.7	9.7
5	13.2	3	1.7	10	21.2	-	0(0.6)	2.2	2.5	2.8	0.4
6	16.8	0.8	1.7	9.6	19.8	0(1.8)	-	0(0.5)	2.7	2.2	1.2
7	14.9	0(1)	2	6.7	16.9	2.3	0.2	-	1.9	1.4	1.5
8	10.8	1.9	0.4	5.7	13.8	1.8	3.8	3.1	-	0(0.4)	0.7
9	12	1.1	0.1	7.8	14	3.8	3.5	2.8	0(0.3)	-	0.6
10	13	2.6	0(0.7)	12.8	13	2.3	2.9	2.2	0.7	0.6	-

Table 4: Distance Matrix 4

Here maximum penalty is 6.7, occur at X0,4 , so we choose X0,4 to begin branch

There are two branches:

1. If $X_{0,4} = 0$, then we have an additional cost of 6.7 and the lower bound becomes 34.5+6.7=41.2
2. If $X_{0,4} = 1$,

we can go 0→4

So we can't go 4→0, so set it to dash(-).

Now we leave row 0 and column 4, so reduced matrix is

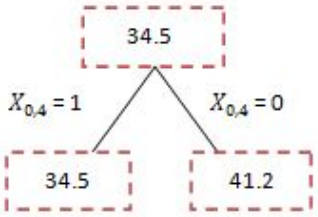
Here we have 0 in every row and column. So, the lower bound remains the same i.e, 34.5+0=34.5

Calculate the penalty of all 0's in reduced matrix(Table5)

Here maximum penalty is 14.4, occur at $X_{4,3}$, so we choose X4,3 to begin branch

Distance	0	1	2	3	5	6	7	8	9	10
1	13.1	-	0.8	7.9	3.1	0.9	0.5	0.8	0	1.2
2	12	2	-	25.8	2.2	3.4	2.7	0.2	0	0
3	0	2.9	5.9	-	7.4	6.9	3.9	2.9	2.9	5.9
4	-	11.7	14.7	0	10.2	15.7	10.7	8.7	9.7	9.7
5	13.2	3	1.7	10	-	0	2.2	2.5	2.8	0.4
6	16.8	0.8	1.7	9.6	0	-	0	2.7	2.2	1.2
7	14.9	0	2	6.7	2.3	0.2	-	1.9	1.4	1.5
8	10.8	1.9	0.4	5.7	1.8	3.8	3.1	-	0	0.7
9	12	1.1	0.1	7.8	3.8	3.5	2.8	0	-	0.6
10	13	2.6	0	12.8	2.3	2.9	2.2	0.7	0.6	-

Table 5: Distance Matrix 5



Distance	0	1	2	3	5	6	7	8	9	10
1	13.1	-	0.8	7.9	3.1	0.9	0.5	0.8	0(0.5)	1.2
2	12	2	-	25.8	2.2	3.4	2.7	0.2	0(0)	0(0.4)
3	0(13.7)	2.9	5.9	-	7.4	6.9	3.9	2.9	2.9	5.9
4	0	11.7	14.7	0(14.4)	10.2	15.7	10.7	8.7	9.7	9.7
5	13.2	3	1.7	10	-	0(0.6)	2.2	2.5	2.8	0.4
6	16.8	0.8	1.7	9.6	0(1.8)	-	0(0.5)	2.7	2.2	1.2
7	14.9	0(1)	2	6.7	2.3	0.2	-	1.9	1.4	1.5
8	10.8	1.9	0.4	5.7	1.8	3.8	3.1	-	0(0.4)	0.7
9	12	1.1	0.1	7.8	3.8	3.5	2.8	0(0.3)	-	0.6
10	13	2.6	0(0.7)	12.8	2.3	2.9	2.2	0.7	0.6	-

Table 6: Distance Matrix 6

There are two branches.

1. If $X_{4,3} = 0$, then we have an additional cost of 14.4 and the lower bound becomes 34.5+14.4=48.9

2. If $X_{4,3} = 1$,

we can go 4→3
Here till now we traversed 0→4→3
So we can't go 3 → 0, so set it to dash(-).
Now we leave row 4 and column 3, so reduced matrix is

Distance	0	1	2	5	6	7	8	9	10
1	13.1	-	0.8	3.1	0.9	0.5	0.8	0	1.2
2	12	2	-	2.2	3.4	2.7	0.2	0	0
3	0	2.9	5.9	7.4	6.9	3.9	2.9	2.9	5.9
5	13.2	3	1.7	-	0	2.2	2.5	2.8	0.4
6	16.8	0.8	1.7	0	-	0	2.7	2.2	1.2
7	14.9	0	2	2.3	0.2	-	1.9	1.4	1.5
8	10.8	1.9	0.4	1.8	3.8	3.1	-	0	0.7
9	12	1.1	0.1	3.8	3.5	2.8	0	-	0.6
10	13	2.6	0	2.3	2.9	2.2	0.7	0.6	-

Table 7: Distance Matrix 7

check each row and column have assignable zeros or not.If they have then lower bound remains same and if they don't have then deduct the row and column mimimum from the corresponding row and column to make assignable zeros.

Step-1: Find the smallest element in each row and subtract it from that row.

So with the help of Table 7 we find the row minimum and proceed with the above step.

Distance	0	1	2	5	6	7	8	9	10	Row Min
1	13.1	-	0.8	3.1	0.9	0.5	0.8	0	1.2	0
2	12	2	-	2.2	3.4	2.7	0.2	0	0	0
3	-	0	3	4.5	4	1	0	0	3	2.9
5	13.2	3	1.7	-	0	2.2	2.5	2.8	0.4	0
6	16.8	0.8	1.7	0	-	0	2.7	2.2	1.2	0
7	14.9	0	2	2.3	0.2	-	1.9	1.4	1.5	0
8	10.8	1.9	0.4	1.8	3.8	3.1	-	0	0.7	0
9	12	1.1	0.1	3.8	3.5	2.8	0	-	0.6	0
10	13	2.6	0	2.3	2.9	2.2	0.7	0.6	-	0

Table 8: Distance Matrix 8

So, row minimum will be 2.9. (0+0+2.9+0+0+0+0+0+0+0=2.9)

Step-2: Find the smallest element in each column and subtract it from that column.

So with the help of Table 8 we find the column minimum and perform the above step.

So, column minimum will be 10.8. (10.8+0+0+0+0+0+0+0+0+0=10.8)

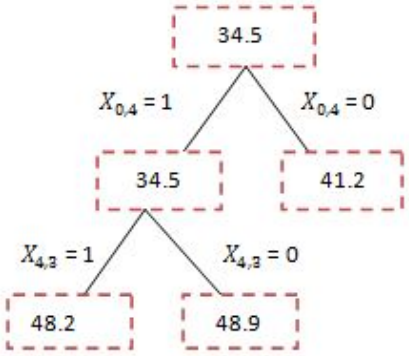
we get the lower bound = 34.5+2.9+10.8=48.2

Distance	0	1	2	5	6	7	8	9	10
1	2.3	-	0.8	3.1	0.9	0.5	0.8	0	1.2
2	1.2	2	-	2.2	3.4	2.7	0.2	0	0
3	-	0	3	4.5	4	1	0	0	3
5	2.4	3	1.7	-	0	2.2	2.5	2.8	0.4
6	6	0.8	1.7	0	-	0	2.7	2.2	1.2
7	4.1	0	2	2.3	0.2	-	1.9	1.4	1.5
8	0	1.9	0.4	1.8	3.8	3.1	-	0	0.7
9	1.2	1.1	0.1	3.8	3.5	2.8	0	-	0.6
10	2.2	2.6	0	2.3	2.9	2.2	0.7	0.6	-
colmin	10.8	0	0	0	0	0	0	0	0

Table 9: Distance Matrix 9

Distance	0	1	2	5	6	7	8	9	10
1	2.3	-	0.8	3.1	0.9	0.5	0.8	0(0.5)	1.2
2	1.2	2	-	2.2	3.4	2.7	0.2	0(0)	0(0.4)
3	-	0(0)	3	4.5	4	1	0(0)	0(0)	3
5	2.4	3	1.7	-	0(0.6)	2.2	2.5	2.8	0.4
6	6	0.8	1.7	0(1.8)	-	0(0.5)	2.7	2.2	1.2
7	4.1	0(0.2)	2	2.3	0.2	-	1.9	1.4	1.5
8	0(1.2)	1.9	0.4	1.8	3.8	3.1	-	0(0)	0.7
9	1.2	1.1	0.1	3.8	3.5	2.8	0(0.1)	-	0.6
10	2.2	2.6	0(0.7)	2.3	2.9	2.2	0.7	0.6	-

Table 10: Distance Matrix 10



Calculate the penalty of all 0's in Table9

Here maximum penalty is 1.8, occur at $X_{6,5}$, so we choose $X_{6,5}$ to begin branch

There are two branches.

1. If $X_{6,5} = 0$, then we have an additional cost of 1.8 and the lower bound becomes 48.2+1.8=50

Distance	0	1	2	6	7	8	9	10
1	2.3	-	0.8	0.9	0.5	0.8	0	1.2
2	1.2	2	-	3.4	2.7	0.2	0	0
3	-	0	3	4	1	0	0	3
5	2.4	3	1.7	-	2.2	2.5	2.8	0.4
7	4.1	0	2	0.2	-	1.9	1.4	1.5
8	0	1.9	0.4	3.8	3.1	-	0	0.7
9	1.2	1.1	0.1	3.5	2.8	0	-	0.6
10	2.2	2.6	0	2.9	2.2	0.7	0.6	-

Table 11: Distance Matrix 11

2. If $X_{6,5} = 1$,

we can go to $6 -> 5$
So we can't go $5 -> 6$, so set it to dash(-).
Now we leave row 6 and column 5, so reduced matrix is
Step-1: Find the smallest element in each row and subtract it from that row.
With the help of Table 11 we find the row minimum and proceed with the above step.

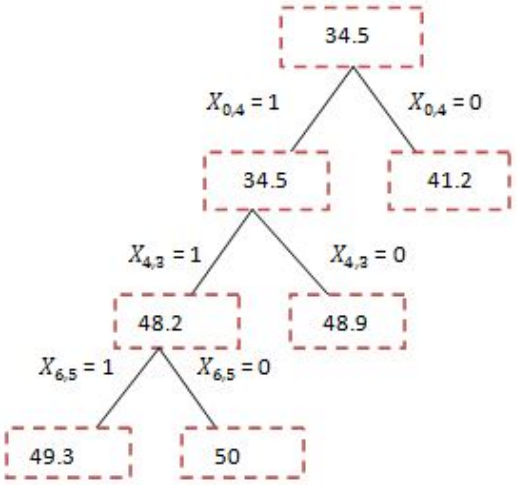
Distance	0	1	2	6	7	8	9	10	rowmin
1	2.3	-	0.8	0.9	0.5	0.8	0	1.2	0
2	1.2	2	-	3.4	2.7	0.2	0	0	0
3	-	0	3	4	1	0	0	3	0
5	2	2.6	1.3	-	1.8	2.1	2.4	0	0.4
7	4.1	0	2	0.2	-	1.9	1.4	1.5	0
8	0	1.9	0.4	3.8	3.1	-	0	0.7	0
9	1.2	1.1	0.1	3.5	2.8	0	-	0.6	0
10	2.2	2.6	0	2.9	2.2	0.7	0.6	-	0

Table 12: Distance Matrix 12

So, row minimum will be 0.4. (0+0+0+0.4+0+0+0+0=0.4)
Step-2: Find the smallest element in each column and subtract it from that column.
Wthe the help of Table 12 we find the column minimum and proceed with the above step.
So, column minimum will be 0.7. (0+0+0+0.2+0.5+0+0+0=0.7)
we get the lower bound = 48.2+0.4+0.7=49.3

Distance	0	1	2	6	7	8	9	10
1	2.3	-	0.8	0.7	0	0.8	0	1.2
2	1.2	2	-	3.2	2.2	0.2	0	0
3	-	0	3	3.8	0.5	0	0	3
5	2	2.6	1.3	-	1.3	2.1	2.4	0
7	4.1	0	2	0.2	0	-	1.4	1.5
8	0	1.9	0.4	3.6	2.6	-	0	0.7
9	1.2	1.1	0.1	3.3	2.3	0	-	0.6
10	2.2	2.6	0	2.7	1.7	0.7	0.6	-
colmin	0	0	0	0.2	0.5	0	0	0

Table 13: Table 13



Calculate the penalty of all 0's in Table 13

Distance	0	1	2	6	7	8	9	10
1	2.3	-	0.8	0.7	0(0.5)	0.8	0(0)	1.2
2	1.2	2	-	3.2	2.2	0.2	0(0)	0(0)
3	-	0(0)	3	3.8	0.5	0(0)	0(0)	3
5	2	2.6	1.3	-	1.3	2.1	2.4	0(1.3)
7	4.1	0(0)	2	0.2	0(0.7)	-	1.4	1.5
8	0(1.2)	1.9	0.4	3.6	2.6	-	0(0)	0.7
9	1.2	1.1	0.1	3.3	2.3	0(0.1)	-	0.6
10	2.2	2.6	0(0.7)	2.7	1.7	0.7	0.6	-

Table 14: Distance Matrix 14

Here maximum penalty is 1.3, occur at $X_{5,10}$, so we choose $X_{5,10}$ to begin branch

There are two branches.

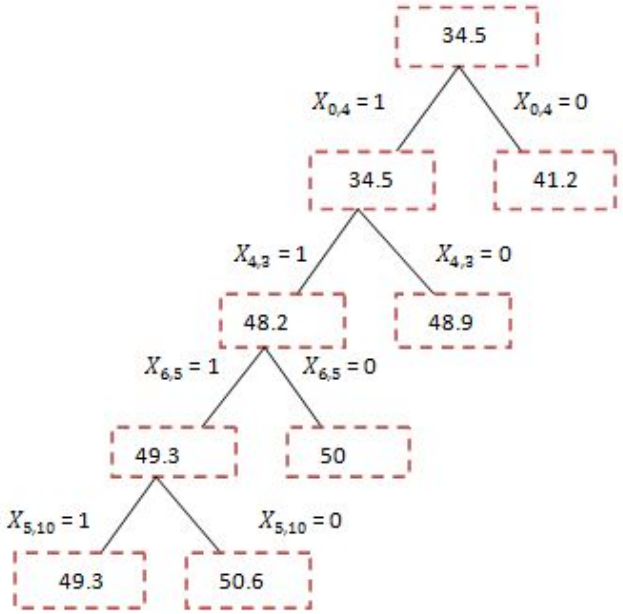
1. If$X_{5,10} = 0$, then we have an additional cost of 1.3 and the lower bound becomes 49.3+1.3=50.6
2. If $X_{5,10} = 1$,

we can go 5→10
Now we leave row 5 and column 10, so reduced matrix is

Distance	0	1	2	6	7	8	9
1	2.3	-	0.8	0.7	0	0.8	0
2	1.2	2	-	3.2	2.2	0.2	0
3	-	0	3	3.8	0.5	0	0
7	4.1	0	2	0.2	0	-	1.4
8	0	1.9	0.4	3.6	2.6	-	0
9	1.2	1.1	0.1	3.3	2.3	0	-
10	2.2	2.6	0	2.7	1.7	0.7	0.6

Table 15: Distance Matrix 15

Here we have 0 in every row and column. So, the lower bound remains the same i.e, 49.3+0=49.3



Calculate the penalty of all 0's in Table15
Here maximum penalty is 1.2, occur at $X_{8,0}$, so we choose $X_{8,0}$ to begin branch

Distance	0	1	2	6	7	8	9
1	2.3	-	0.8	0.7	0(0.5)	0.8	0(0)
2	1.2	2	-	3.2	2.2	0.2	0(0.2)
3	-	0(0)	3	3.8	0.5	0(0)	0(0)
7	4.1	0(0)	2	0.2	0(0.7)	-	1.4
8	0(1.2)	1.9	0.4	3.6	2.6	-	0(0)
9	1.2	1.1	0.1	3.3	2.3	0(0.1)	-
10	2.2	2.6	0(0.7)	2.7	1.7	0.7	0.6

Table 16: Distance Matrix 16

There are two branches.

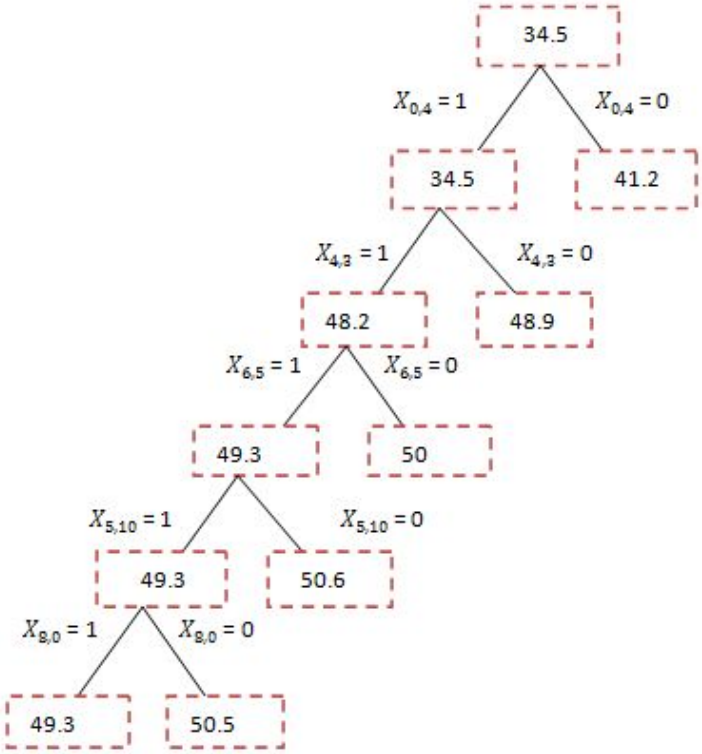
1. If $X_{8,0} = 0$, then we have an additional cost of 1.2 and the lower bound becomes 49.3+1.2=50.5
2. If $X_{8,0} = 1$,

we can go 8→0
Now we leave row 8 and column 0, so reduced matrix is

Distance	1	2	6	7	8	9
1	-	0.8	0.7	0	0.8	0
2	2	-	3.2	2.2	0.2	0
3	0	3	3.8	0.5	0	0
7	0	2	0.2	0	-	1.4
9	1.1	0.1	3.3	2.3	0	-
10	2.6	0	2.7	1.7	0.7	0.6

Table 17: Distance Matrix 17

Here we have 0 in every row and column. So, the lower bound remains the same i.e, 49.3+0=49.3



Calculate the penalty of all 0's in Table17

Distance	1	2	6	7	8	9
1	-	0.8	0.7	0(0.5)	0.8	0(0)
2	2	-	3.2	2.2	0.2	0(0.2)
3	0(0)	3	3.8	0.5	0(0)	0(0)
7	0(0)	2	0.2	0(0.7)	-	1.4
9	1.1	0.1	3.3	2.3	0(0.1)	-
10	2.6	0(0.7)	2.7	1.7	0.7	0.6

Table 18: Distance Matrix 18

Here maximum penalty is 0.7, occur at $X_{7,6}$ or $X_{10,2}$, so we choose $X_{7,6}$ to begin branch
There are two branches.

1. If $X_{7,6} = 0$, then we have an additional cost of 0.7 and the lower bound becomes 49.3+0.7=50

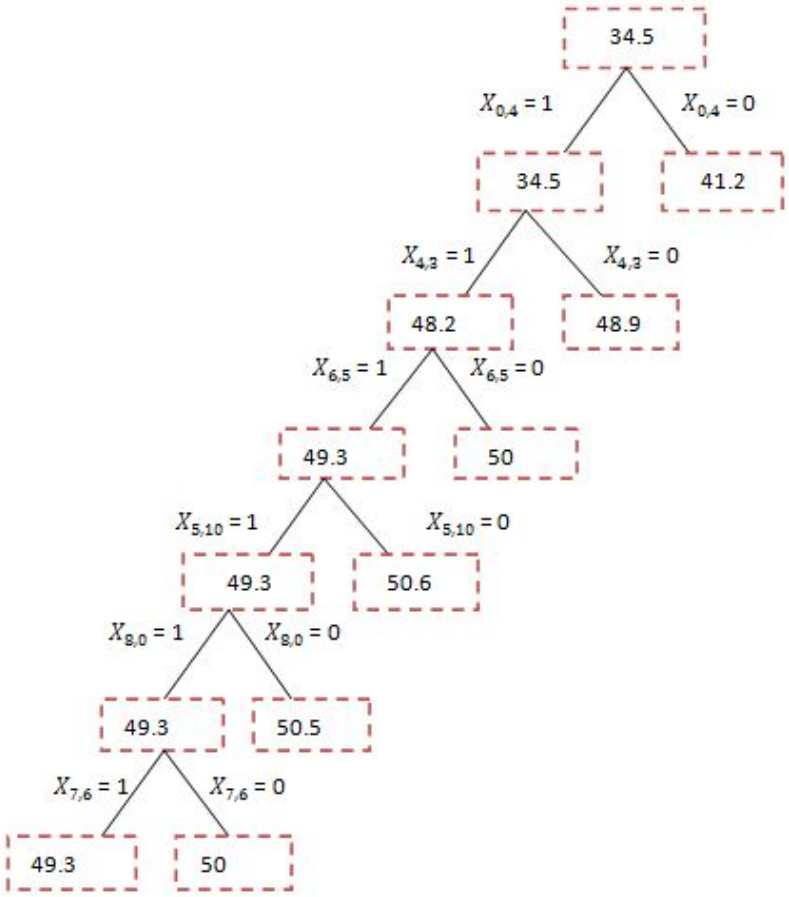
2. If $X_{7,6} = 1$,

we can go 7→6
Now we leave row 7 and column 6, so reduced matrix is

Distance	1	2	7	8	9
1	-	0.8	0	0.8	0
2	2	-	2.2	0.2	0
3	0	3	0.5	0	0
9	1.1	0.1	2.3	0	-
10	2.6	0	1.7	0.7	0.6

Table 19: Distance Matrix 19

Here we have 0 in every row and column. So, the lower bound remains the same i.e, 49.3+0=49.3



Calculate the penalty of all 0's in Table19

Here maximum penalty is 1.1, occur at $X_{3,1}$, so we choose $X_{3,1}$ to begin branch

There are two branches.

1. If $X_{3,1} = 0$, then we have an additional cost of 1.1 and the lower bound becomes 49.3+1.1=50.4

Distance	1	2	7	8	9
1	-	0.8	0(0.5)	0.8	0(0)
2	2	-	2.2	0.2	0(0.2)
3	0(1.1)	3	0.5	0(0)	0(0)
9	1.1	0.1	2.3	0(0.1)	-
10	2.6	0(0.7)	1.7	0.7	0.6

Table 20: Distance Matrix 20

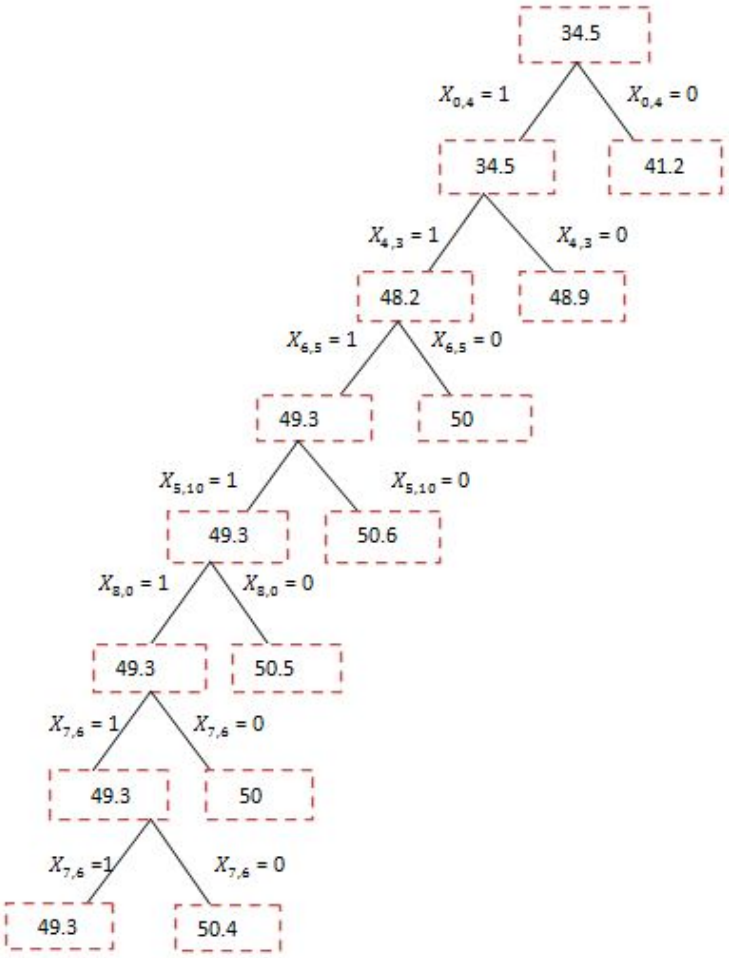
2. If $X_{3,1} = 1$,

we can go 3→1
Now we leave row 3 and column 1, so reduced matrix is

Distance	2	7	8	9
1	0.8	0	0.8	0
2	-	2.2	0.2	0
9	0.1	2.3	0	-
10	0	1.7	0.7	0.6

Table 21: Distance Matrix 21

Here we have 0 in every row and column. So, the lower bound remains the same i.e, 49.3+0=49.3



Calculate the penalty of all 0's in Table21

Distance	2	7	8	9
1	0.8	0(1.7)	0.8	0(0)
2	-	2.2	0.2	0(0.2)
9	0.1	2.3	0(0.3)	-
10	0(0.7)	1.7	0.7	0.6

Table 22: Disatnce Matrix 22

Here maximum penalty is 1.7, occur at $X_{1,7}$, so we choose $X_{1,7}$ to begin branch
There are two branches.

1. If $X_{1,7} = 0$, then we have an additional cost of 1.7 and the lower bound becomes 49.3+1.7=51
2. If $X_{1,7} = 1$,

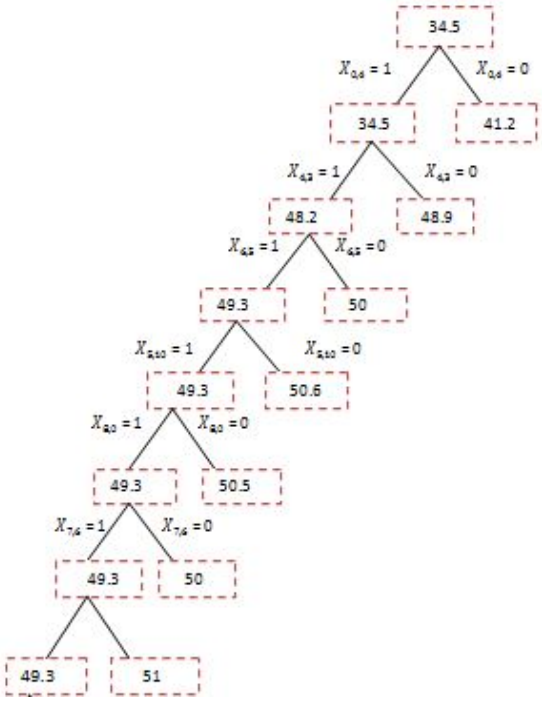
we can go $1 \rightarrow 7$

Here till now we traversed
8→0→4→3→1→7→6→5→10
So we can't go 10→8, so set it to dash(-).
Now we leave row 1 and column 7, so reduced matrix is

Distance	2	8	9
2	-	0.2	0
9	0.1	0	-
10	0	0.7	0.6

Table 23: Distance Matrix 23

Here we have 0 in every row and column. So, the lower bound remains the same i.e, 49.3+0=49.3



Calculate the penalty of all 0's in Table 23

Distance	2	8	9
2	-	0.2	0(0.8)
9	0.1	0(0.3)	-
10	0(0.7)	-	0.6

Table 24: Distance Matrix 24

Here maximum penalty is 0.8, occur at $X_{2,9}$, so we choose $X_{2,9}$ to begin branch// There are two branches.

1. If $X_{2,9} = 0$, then we have an additional cost of 0.8 and the lower bound becomes 49.3+0.8=50.1

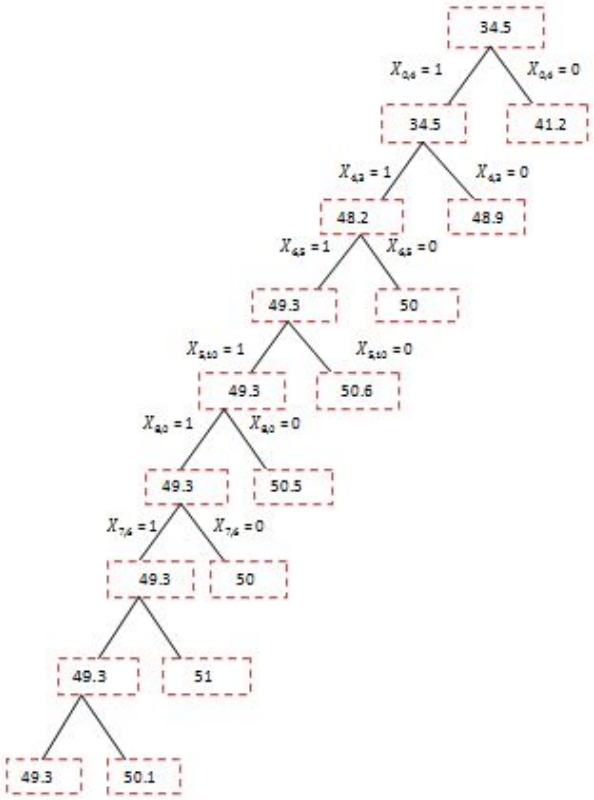
2. If $X_{2,9} = 1$,

we can go 2→9
So we can't go 9→2, so set it to -(dash).
Now we leave row 2 and column 9, so reduced matrix is

Distance	2	8
9	-	0
10	0	-

Table 25: Distance Matrix 25

Here we have 0 in every row and column. So, the lower bound remains the same i.e, 49.3+0=49.3



Calculate the penalty of all 0's in Table25
we can go 9→8
and 10→2

Distance	2	8
9	-	0(0)
10	0(0)	-

Table 26: Distance Matrix 26

. These two are the main tasks which are possible.we realize that in branch and bound strategy at whatever point we have two urban communities left,we consistently try and get the feasible solution out of it. so we have $X_{0,4} = X_{4,3} = X_{6,5} = X_{5,10} = X_{8,0} = X_{7,6} = X_{3,1} = X_{9,8} = X_{10,2} = 1$ so we get feasible arrangement as 0ß4ß3ß1ß7ß6ß5ß10ß2ß9ß8ß0 with $Z = 49.3$ where $Z = 49.2$ is the upperbound,so we can understood lowerbounds which are more than current best upper bound is fathomed and the lower-bound which is equivalent to current best upperbound is also fathomed because when we drop down we get either arrangement of 49.3 or more and we as of now have arrangement of 49.3.so for this situation the feasible solution is the optimal solution

Thus,the branch and bound method identifies the possible route as Rohini,Sector-24(depot) $\rightarrow$ Rithala $\rightarrow$ Badli $\rightarrow$ Gtb Nagar $\rightarrow$TimarPur$\rightarrow$CivilLines $\rightarrow$Chandni Chowk$\rightarrow$Malka Ganj$\rightarrow$Kamla Nagar$\rightarrow$ Roop Nagar$\rightarrow$ Shakti Nagar $\rightarrow$Rohini,Sector-24(depot) with total distance of

$3.2 + 5.5 + 10 + 2.4 + 2.3 + 5.7 + 5.2 + 1 + 1 + 1 + 12 = 49.3Km$

To make the calculation easier, a computer software in the form of a TSP problem-solving application is created. The Branch and Bound Penalty Method is used to solve the TSP issue in general because it can still produce optimal results. The Python Pseudo Code is as follows:

Algorithm 1

```
IMPORT math
SET aj
SET N
SET vis
DEFINE FUNCTION
row_minimum_func(aj, i, vis, minimum):
SET mini TO float('inf')
FOR k IN range(N):
IF i EQUALS k or vis[k] EQUALS True:
    continue
IF mini > aj[i][k]:
SET mini TO aj[i][k]
FOR k IN range(N):
IF i EQUALS k or vis[k] EQUALS
True:
    continue
SET aj[i][k] TO aj[i][k] - mini
minimum[0] += mini
DEFINE FUNCTION
col_minimum_func(aj, i, vis, minimum):
SET mini TO float('inf')
FOR k IN range(N):
IF i EQUALS k or vis[k] EQUALS
True:
    continue
IF mini > aj[k][i]:
    SET mini TO aj[k][i]
FOR k IN range(N):
IF i EQUALS k or vis[k] EQUALS
True:
    continue
SET aj[k][i] TO aj[k][i] - mini
minimum[1] += mini
DEFINE FUNCTION penalty_cal(aj:
object, i:          object, j: object, vis:
object)
    - > object:
SET mini TO float('inf')
SET mini1 TO float('inf')
FOR k IN range(N):
IF aj[i][k] < mini:
    SET mini TO aj[i][k]
FOR k IN range(N):
IF aj[k][j] < mini1:
    SET mini1 TO aj[k][j]
RETURN mini + mini1
DEFINE FUNCTION
my_recursive_function_to_call(aj, lvl,
curr_p, vis, minimum):
OUTPUT(aj)
IF lvl EQUALS N:
IF aj[curr_p[lvl - 1]][curr_p[0]]
    EQUALS 0:
RETURN
ELSE:
SET minimum[0] TO 0
```

```
47.  SET minimum[1] TO 0
48.  FOR i IN range(N):
49.   row_minimum_func(aj, i, vis,
minimum)
50.  FOR i IN range(N):
51.   col_minimum_func(aj, i, vis, minimum)
      minimum[2] += minimum[0] +
minimum[1]
52. RETURN
53. SET flag TO 0
54. SET maxi TO 0
55.  SET index TO -1
56. FOR i IN range(N):
57.  IF aj[curr_p[lvl - 1]][i] EQUALS 0 and
vis[i] EQUALS
False and flag EQUALS 0:
58.        IF maxi <= penalty_cal(aj,
curr_p[lvl - 1], i, vis):
59.    SET maxi TO penalty_cal
   (aj, curr_p[lvl - 1], i, vis)
60.          SET index TO i
61.          SET flag TO 1
62.  IF flag EQUALS 1:
63.    SET curr_p[lvl] TO index
64.   SET vis[index] TO True
65.    my_recursive_function_to_call(aj, lvl +
1, curr_p, vis, minimum)
66.  IF flag EQUALS 0:
67.    SET minimum[0] TO 0
68.    SET minimum[1] TO 0
70.      row_minimum_func(aj, i, vis,
minimum)
71.    FOR i IN range(N):
72.      col_minimum_func(aj, i, vis,
minimum)
    minimum[2] += minimum[0] +
minimum[1]
73.   my_recursive_function_to_call(aj, lvl,
curr_p, vis, minimum)
74. DEFINE FUNCTION main():
75.  SET minimum TO [0, 0, 0]
76.  SET vis TO [False] * (N + 1)
77.  SET vis[0] TO True
78.  FOR i IN range(N):
79.   row_minimum_func(aj, i, vis,
minimum)
80.  FOR i IN range(N):
81.    col_minimum_func(aj, i, vis,
minimum)
  minimum[2] += minimum[0] + minimum[1]
82.  SET curr_p TO [-1] * (N + 1)
83.  SET curr_p[0] TO 0
84.  my_recursive_function_to_call(aj, 1,
     curr_p, vis, minimum)
85.  OUTPUT("Cost : ",minimum[2])
86.  OUTPUT("Path Taken : ", end=' ')
87.  FOR i IN range(N):
88.    OUTPUT(curr_p[i]+1, end=' ')
89.   main()
```

2.4.2 Transportation of items from distribution centers to hubs

The products are classified, transported, and delivered to the hubs in this location. We want the road that takes the least amount of time, so we'll use the Dijkstra Shortest Way Algorithm to determine the path that takes the least amount of time, because we already know that distance is proportional to time.
Dijkstra Algorithm is the best algorithm at present when all the weight number $W_{ij} >= 0$.

2.4.3 Setting up road network model

The initial portion of a specific label is a number, which indicates a sign in front of a point that specifies where it is. The second portion is a number that represents the time from the beginning location to the current position, indicating how long it will take to arrive at the destination. As a result, it can determine the path that takes the least amount of time.

The distribution center is situated in Rohini sector -24 and the hub is situated in Model Town ,New Delhi.
In between the distribution center and the hub the vehicle has to cross few more places in order to reach the hub as soon as possible . These places are Jahangirpuri , Adarsh Nagar,Majlis Park,Azadpur,Udyog Nagar ,Hudsone Lane,Vidhan Sabha ,Golflinks ,Khan Market,Civil Lines and Pratap Nagar.

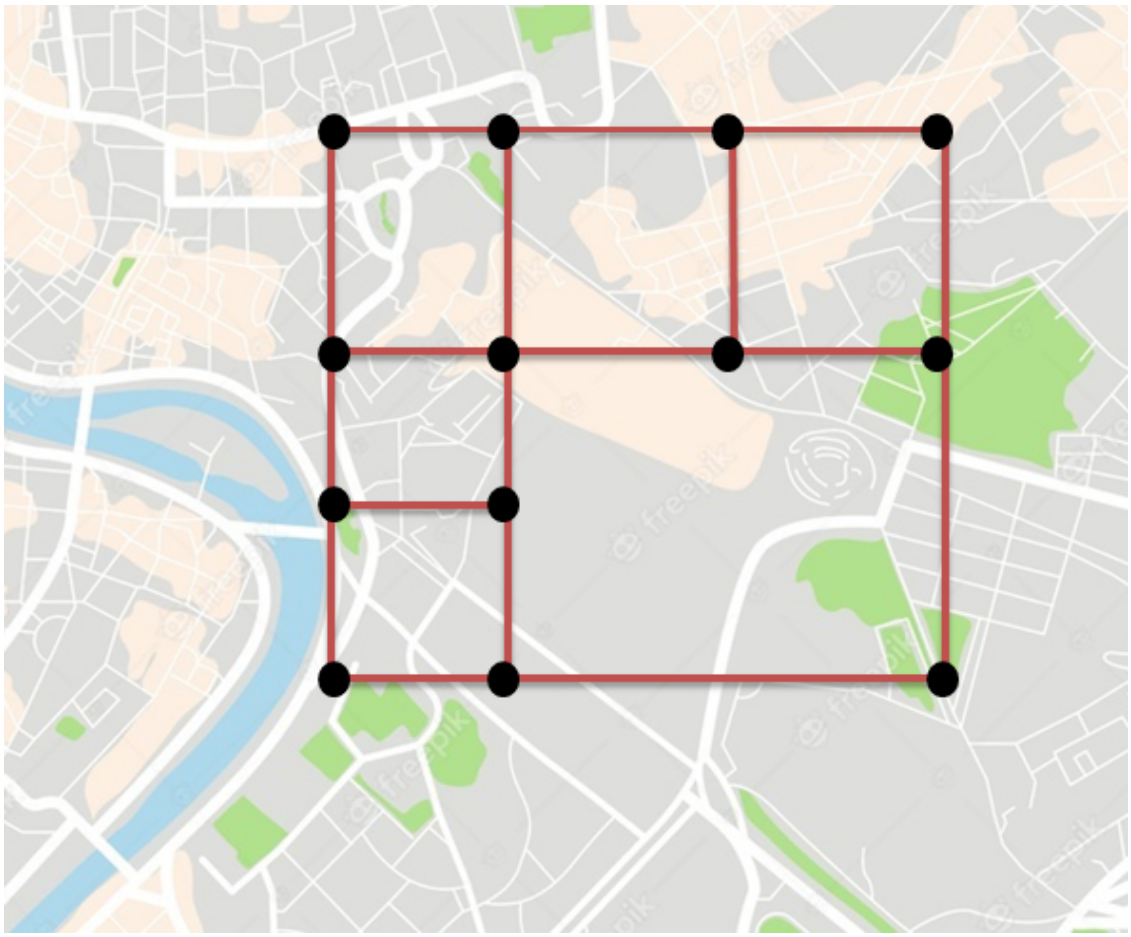


Figure 10: Road Map

So, by representing all of these locations on the map as nodes and connecting them with an edge, we get the following representation, where 1 denotes the Distribution Centre (Rohini Sector 24) and 13 denotes the Depot (Model Town), and 2,3,4,5,6,7,8,9,10,11 denotes various locations between the distribution centres and hub, as shown below.
node 1 = Jahangirpuri
node 2 = Adarsh Nagar

node 3 = Majlis Park
node 4 = Azadpur
node 5 = Udyog Nagar
node 6 = Hudsone Lane
node 7 = Vidhan Sabha
node 8 = Golflinks
node 9 = Khan Market
node 10 = Civil Lines
node 11 = Pratap Nagar

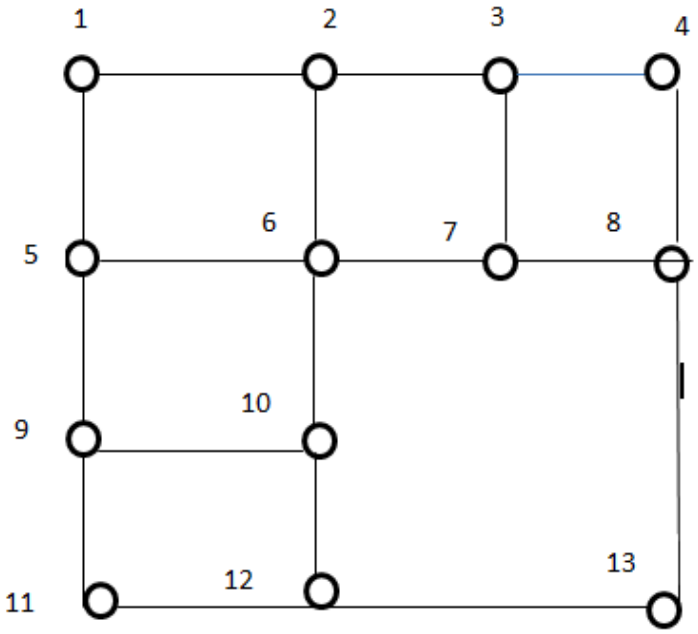


Figure 11: Road network map

When analysing the performance of a signalised intersection, the most essential criterion utilised by transportation professionals is vehicle delay. This is possibly due to the fact that it is directly related to the time lost by a vehicle when crossing a junction.
These models contain assumptions that aid in the reduction of complex traffic circumstances to a measurable model that provides an estimate of the average delay experienced by a vehicle passing an intersection.

Highway Capacity Manual

The delay is given as,

$d = d_1\ PF + d_2 + d_3$

d_1=$\frac{c}{2}\frac{(1-\frac{g}{c})^2}{1-min(1,X)\frac{g}{c}}$

$d_2 = 900T[(X-1)+\sqrt{(X-1)^2+\frac{8klX}{cT}}$

$PF = (\frac{(1-P)}{(1-(g/c)}) f_p$ additional explanation for PF Where,
d = control delay, s/veh, d_1 = uniform delay component, s/veh, PF = progression adjustment factor, d_2 = overflow delay component, s/veh, d3 = delay due to pre-existing queue, s/veh, T = analysis period, h, X = v/c ratio, C = cycle length, s, k = incremental delay factor for actuated controller settings; 0.50 for all pre-timed controllers, l = upstream filtering/metering adjustment factor; 1.0 for all individual intersection analyses, c = capacity, veh/h, P = proportion of vehicles arriving during the green interval and f_p = supplemental adjustment factor for platoon arriving during the green

Saturation Flow Rate

The saturation flow rate is defined as the maximum number of vehicles that can pass the stop line per unit of time in one phase or lane with appropriate traffic demand at the intersection entrance.

Lost Time

It signifies the point at which the intersection is unable to accommodate any movement.

Green Ratio

The ratio of effective green time to cycle length is what this term refers to.

Approach Flow Rate

Uniform Delay

The uniform delay is predicated on the assumption of constant flow and no individual cycle failures.

Random Delay

Because flow is randomly distributed rather than uniform at isolated intersections, random delay occurs in addition to uniform delay.

Control Delay

Delay produced by deceleration, stopped delay, and control delay are all examples of control delay. There is a delay in acceleration.

2.4.4 Calculating using dijkstra algorithm with signalised intersection constraint

Establish a series of N and put starting point s based on the calculation to weight values. When the N point is not null, select a node v_i from the N as the current node.

Calculate all adjacent nodes v_j of the road network node v_i.
If $l(s,v_j) > l(s,v_i + l_{ij})$ where l_{ij} is control delay. So,$l(s,v_j) = l(s,v_i) + l_{ij}$ Add the neighbouring point to the sequence of N, modify it with the shortest distance, and keep it rising in order according to the length of each road section.

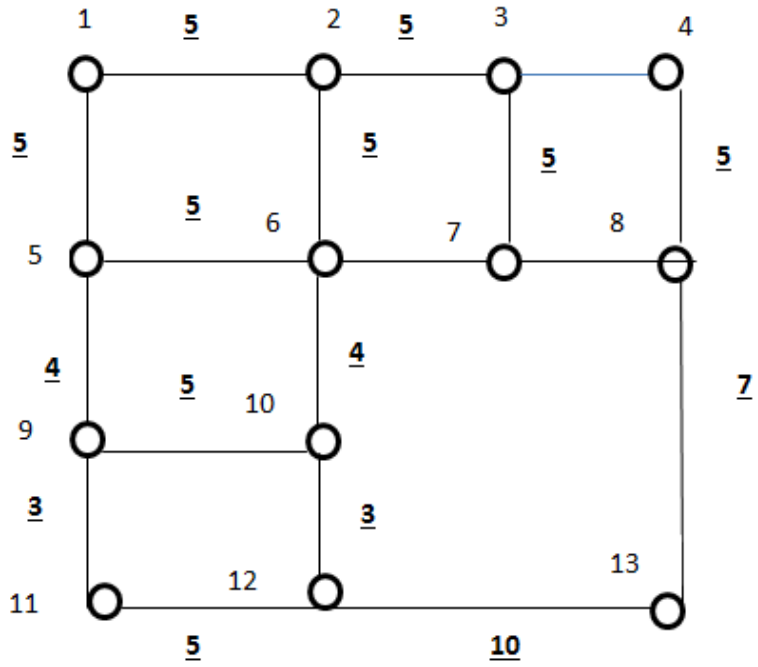


Figure 12: Weighted Network

Edge_num	Saturation_flow_rate	The_signal_cycle_of_intersection	Lost_Time	Green_Ratio
(1,2)	2650	90	10	0.96
(1,5)	2150	75	12	0.84
(2,3)	2179	60	7	0.6
(2,6)	2850	55	11	0.59
(3,4)	3250	75	4	0.64
(3,7)	2690	90	15	0.98
(4,8)	1789	120	18	0.98
(5,6)	1150	45	8	0.54
(5,9)	2954	55	15	0.45
(6,7)	1123	120	20	0.94
(6,10)	3150	55	10	0.84
(7,8)	2341	85	16	0.65
(8,13)	2111	35	14	0.94
(9,10)	3114	95	12	0.96
(9,11)	2170	35	8	0.74
(10,12)	2950	75	12	0.84
(11,12)	2894	50	15	0.95
(12,13)	1568	75	12	0.44

Capacity	Approach_flow_rate	Degree_of_saturation	PAF
2582.4	1700	1.45	1.65
3592.4	1680	1.68	1.95
2772.4	1790	1.89	1.77
2850.7	1.55	1.1	1.59
1134	1.67	1.78	1.56
2346	1.78	1.67	1.44
3567	1.88	1.24	1.78
2345	1.97	1.75	1.89
2364	1.78	1.34	1.67
1245	1.55	1.89	1.09
2311	1.45	1.66	1.76
3111	1.76	1.99	1.65
2364	1.78	1.34	1.67
1245	1.55	1.89	1.09
2311	1.45	1.66	1.76
3111	1.76	1.99	1.65

delay_due_to_pre_existing_queue	Uniform_delay	Overflow_delay	control_delay
12	15.6	0.26	114
10	18.9	0.68	195
27	17.9	1.89	177
28	15.5	1.19	159
11	16.7	1.78	156
23	17.8	1.67	144
35	18.8	1.24	178
25	19.7	1.75	189
24	17.8	1.34	167
12	15.5	1.89	109
11	14.5	1.66	176
31	17.6	1.99	165
21	12.2	1.54	178
15	10.9	1.07	199
12	11	1.4	115
24	19	1.1	177
11	10.9	1.07	199
21	10.5	1.57	177

Figure 13: Road Intersection Parameters

Now using the Dijkstra shortest path algorithm as mentioned above ,we will find the minimum time required to reach the hub.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	**0**	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	-	**6.9**	∞	∞	8.25	∞	∞	∞	∞	∞	∞	∞	∞
3	-	-	14.85	∞	**8.25**	14.55	∞	∞	∞	∞	∞	∞	∞
4	-	-	14.85	∞	-	**14.55**	∞	∞	14.95	∞	∞	∞	∞
5	-	-	**14.85**	∞	-	-	21.35	∞	14.95	21.45	∞	∞	∞
6	-	-	-	24.45	-	-	21.35	∞	**14.95**	21.45	∞	∞	∞
7	-	-	-	24.45	-	-	21.35	∞	-	21.45	**19.85**	∞	∞
8	-	-	-	24.45	-	-	**21.35**	∞	-	21.45	-	28.15	∞
9	-	-	-	24.45	-	-	-	31.1	-	**21.45**	-	28.15	∞
10		-	-	**24.45**	-	-	-	31.1	-	-	-	28.15	∞
11	-	-	-	-	-	-	-	31.1	-	-	-	**28.15**	∞
12	-	-	-	-	-	-	-	**31.1**	-	-	-	-	40.1
13	-	-	-	-	-	-	-	-	-	-	-	-	**40.1**

Shortest time taking path with control delay

Therefore the minimum time required to reach the hub is 40.1 minutes.And the required route is $1->5->11->12->13$.

Algorithm 2

```
1.IMPORT sys
2.from collections IMPORT defaultdict
3.SET adj_matrix
4.SET paf
5.SET uniform delay
6.SET over flow delay
7.SET delay due to pre existing queue
8.SET vertex
9.SET dist TO [sys.maxsize] * (vertex + 1)
10.DEFINE CLASS Graph:
11.  DEFINE FUNCTION init (self, vertices):
12.    SET self.V TO vertices # No. of vertices
13.    SET self.V_org TO vertices
14.    SET self.graph TO defaultdict(list)
  DEFINE FUNCTION addEdge(self, u, v, w):
15.      IF w EQUALS 1:
16.        self.graph[u].append(v)
17.      ELSE:
18.        self.graph[u].append(self.V)
19.        self.graph[self.V].append(v)
20.        SET self.V TO self.V + 1
21.  DEFINE FUNCTION OUTPUTPath
     (self, parent, j):
22.    SET Path_len TO 1
23.    IF parent[j] EQUALS -1
     and j < self.V_org:
24.        OUTPUT(j)
25.        RETURN 0
26.    SET l TO self.OUTPUTPath
     (parent,   parent[j])
27.    SET Path_len TO l + Path_len
28.    IF j < self.V_org:
29.        OUTPUT(j)
30.    RETURN Path_len
31.  DEFINE FUNCTION
     findShortestPath(self, src, dest):
32.    SET visited TO [False] * (self.V)
33.    SET parent TO [-1] * (self.V)
34.    SET queue TO []
35.    queue.append(src)
36.    SET visited[src] TO True
37.    WHILE queue:
38.      SET s TO queue.pop(0)
39.      IF s EQUALS dest:
40.        RETURN self.OUTPUTPath(parent, s)
41.      FOR i IN self.graph[s]:
42.        IF visited[i] EQUALS False:
43.          queue.append(i)
44.          SET visited[i] TO True
45.          SET parent[i] TO s
46.    DEFINE FUNCTION
       cost_calculator(val, i, j):
47.  SET control_delay TO
     uniform_delay[i][j] *
     paf[i][j] +   over_flow_delay[i][j]
     + delay_due_to_pre_existing_queue[i][j]
48.    RETURN control_delay
49.  DEFINE FUNCTION main():
50.  SET g TO Graph(vertex + 1)
51.  FOR i IN range(vertex):
52.    FOR j IN range(vertex):
53.      IF i EQUALS j:
54.        continue
55.      SET adj_matrix[i][j] TO
56.   cost_calculator (adj_matrix[i][j], i, j)
57.      IF adj_matrix[i][j] EQUALS 0:
58.        continue
59.      g.addEdge(i, j, adj_matrix[i][j])
60.  SET src TO 1
61.  SET dest TO 12
62.   OUTPUT("Shortest Path between
63.    %d and %d is " % (src, dest)),
64.  SET l TO g.findShortestPath(src, dest)
65.  OUTPUT("\nShortest Distance between
66.    %d and %d is %d " % (src, dest, l)),
67.    main()
```

2.5 *Scenario 1:Truck loading and routing*

The most important requirements for logistics and transportation in general are to have the correct product at the right time, in the right place, and in the right condition. In the increasingly complex logistic world, a smart logistic system should be developed to match these characteristics. The shipping process is part of the logistics system. As a result, a technology-driven strategy should be used to ensure maximum customer satisfaction and security during the shipping process. Choosing the best manner to pack the goods into containers is one of the most difficult aspects of that procedure. The Bin Packing Problem is the name for this issue.

Xpressbees have had trouble loading diverse things of varying sizes efficiently in the fewest amount of boxes due to excessively extensive computations. The goal is to pack the truck(s) with the optimal combination

of items in order to reduce the number of boxes and, as a result, the shipping cost. As a result, the ability to put more items into a smaller number of vehicles will save money on transportation in the long run.

2.5.1 Proposed Problem

A consumer has placed an order with a vendor for ten pieces of a product. The vendor has enlisted the services of xpressbees' warehouse to safely store his stuff. The product list was sent to the xpressbees logistics team by the seller. The packing and routing will now be handled by the xpressbees team.
The proposed problem is consists of the following methodology firstly The products from the warehouse will be packed in boxes and those boxes will then be loaded in trucks and then an optimized route will be calculated for the truck to reach to the customer on time.
In the first part The number of boxes used for packing the product is calculated. In the second part i.e for loading the products into the trucks a new algorithm is proposed namely ,**Product Loading Algorithm** . The goal of this algorithm is to fit as many boxes as feasible into the smallest number of trucks while ensuring that the weight limit is not exceeded. This algorithm will reduce transportation costs while also ensuring optimal safety.
In the last part i.e the route optimization of trucks, **Dynamic Programming is used**.

2.5.2 Box Packing Problem

In this part the minimum number of boxes used to pack the products is calculated. The dimensions i.e length,width and height , volume and weight of the product,box and the truck are mentioned:

	Product	Box	Truck
Length	2.5 ft	8.4 ft	17 ft
Width	2ft	2.5 ft	5ft
Height	5ft	5.5 ft	6 ft
Volume	25 ft^3	115.5ft^3	510 ft^3
Weight	50 Kg	150Kg	350 Kg

Table 27: Dimensions

The main objective of the problem is to minimise the total number of boxes used . To find the number of boxes required to pack 10 units of the product , lb is calculated . Here firstly the minumum number of product that can fit into one box is calculated . .

In terms of weight :

$$lb_1 = \frac{totalweightofbox}{totalweightofoneproduct}$$

Therefore

$$lb_1 = \frac{150}{50} = 3$$

In terms of volume :

$$lb_2 = \frac{Totalvolumeofbox}{Volumeofoneproduct}$$

Therefore

$$lb_2 = \frac{115.5}{25} = 4$$

$$\text{lb= min } [lb_1, lb_2]$$
$$\text{lb= min [3,4]}$$
$$\text{lb=3}$$

The lb gives an estimate of how many maximum products can be packed into each box. This figure is an estimate that may vary based on the shapes and dimensions of the goods. Just because an item fits into a box by volume does not indicate it will fit in terms of size.
Using mathematical prediction it is obvious to see that minimum 4 boxes are required to pack 10 items of the product i.e 3 items in 3 boxes and 1 item in 1 box.

Box	Total item in the box
1	3
2	3
3	3
4	1

Table 28: Total Boxes and Items

2.5.3 Proposed Algorithm

2.5.4 Product Loading Algorithm

The general idea of the proposed algorithm is to try to place the heaviest box first in the truck and then balance the box's weight.
In addition, several restrictions on the weight, volume, and dimensions of the boxes and trucks were taken into account. The first limitation is that the overall weight of products in a box cannot exceed the weight of the

truck. The second rule states that the overall volume of products placed within a box must not exceed the Truck's volumetric capacity.
The following are the notations used to describe the algorithm:

- w_i represents the weight of the box i, $1 \leq i \leq m$.
- l_i,b_i,h_i represents the length, width and height of the box respectively.
- v_i represents the volume of the box i.
- W_j represents the weight of the truck.
- L_j,B_j,H_j represents the length, width and height of the truck respectively.
- V_j represents the volume of the truck j.

We define a decision variable as follows :

$$x_{ij} = \{\, 1 \; if the box i is places in the truck j$$

0otherwise
(1)
The proposed algorithm will be proceeded as follows:

- Step 1: Arrange the boxes according to their weights in descending order. The heavier box will be positioned near the bottom of the vehicle as a result of this.
- Step 2: Check to see if the box I will fit in the truck j without exceeding the truck's weight limit. This is ensured by

$$w_i \leq \mathrm{W}_j\text{-}\sum_{k=1}^{m} w_k x_{kj}$$

- Step 3:Check to see if the box I will fit into the truck j without exceeding the truck's capacity. This is ensured by

$$v_i \leq \mathrm{V}_j\text{-}\sum_{k=1}^{m} v_k \, x_{kj}$$

- Step 4:Check to see if the box I will fit inside the truck j without exceeding the truck's dimensions in terms of length, breadth, and height.

- Step 5: Place the box I in the truck j if it meets the conditions indicated in steps 2,3,4; otherwise, move to step 6.
- Step 6: Increase the number of trucks by one and return to step 2 if the box I does not meet the conditions mentioned in steps 2,3,4.

Algorithm 3

```
1 IMPORT numpy as np
2 SET boxes
3 SET truck_wei
4 SET truck_vol
5 DEFINE FUNCTION main():
6   SET sortedBoxes TO
boxes[boxes[:,3].argsort()]
7     SET curr_wei TO truck_wei
8   SET curr_vol TO truck_vol
9       SET count TO 1
10      FOR i IN range(len(sortedBoxes)):
11      SET vol TO
sortedBoxes[i][0]*sortedBoxes[i][1]*sortedBoxes[i
][2]
12      IF sortedBoxes[i][3]<= curr_wei and vol <=
curr_vol:
13          curr_wei -= sortedBoxes[i][3]
14          curr_vol -= vol
15      ELSE:
16          count += 1
17          SET curr_wei TO truck_wei
18          SET curr_vol TO truck_vol
19          IF sortedBoxes[i][3]<= curr_wei and vol
<= curr_vol:
20              curr_wei -= sortedBoxes[i][3]
21              curr_vol -= vol
22    OUTPUT(count)
23 main ()
```

2.5.5 Experimental Study

Any express or logistics company can use the presented algorithm to handle the product loading problem.However,data of the xpressbees logistics company is used to examine this proposed alogrithm.
Step 1: Arrange the boxes in descending order of their weights.

Box	Box weight (in Kg)
1	150
2	150
3	150
4	50

Table 29: Boxes and Weights

Step 2: The weight holding capacity of truck is 350 Kg .Therefore 1 truck can hold 2 boxes weighing 150kg each.Also The dimensions and volume of both the boxes are not exceeding the dimensions and volume of the truck.
Step 3: The box 3 cannot be loaded into truck 1 because it is exceeding the truck's weighing capacity. Therefore the number of trucks is increased by 1.So box 3 and 4 are placed in truck 2.
The above is expressed below:

Truck	Box
1	1 and 2
2	3 and 4

Table 30: Truck and box

2.5.6 Truck Route Optimization

In the last step ,the optimal route for both trucks is find using dynamic programming. The products have been loaded into the trucks as previously, and the most efficient path for both trucks has been determined, taking the shortest distance possible.
In Dynamic Programming , The problem is divided into several stages and it is solved at each stage.Here the decisions are made in a particular stage which will affect the decisions at the subsequent stages but they are dependent on one another.The **Backward Recursive Approach** is used in Dynamic Programming to solve the Shortest Path problem. The truck is now at the hub in Modeltown, Delhi and the customer's Location is Rohini east,Delhi.Between the Hub and the Customer, the vehicle must pass through a few more places to come back to the Hub as quickly as possible which are Gtb Nagar , Civil Lines ,Pul Bangash , Shastri Nagar , Inderlok , Kanhiya Nagar, Kohat Enclave , Rohini west.

2.5.7 Network Model

The hub is represented by 1 and the customer location by 9, with 2,3,4,5,6,7,8,10 representing different locations between the hub and the customer on the google map below.
A network model is created in which each node represents a place, an edge connects each node, and the distance between nodes is mentioned in the network diagram.
The Truck is currently in position 1, which is the hub, and it must travel to location 9, which is the customer location.This is the problem where the decision making can easily decomposed into various stages for instance, There is no connectivity between node 1 and node 5 or node 3 and node 8 and so on but it can easily decompose into several stages and the connectivity can be obtained from node of one stage to the next stage.It is the simple shortest path problem with the concept of Dynamic Programming.

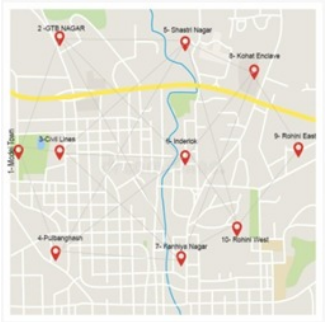


Figure 14: Road Network

Figure 15: Road Network with Weights

2.5.8 Solution Approach

The Backward Recursive Approach is used,In order to reach node 9,The Truck has to either start from node 8 or node 10.It is assumed that the truck is at node 8 or node10 and This stage is called as **"One More Stage To Go"**.
The node or the place where the truck is called is the **"State of the System"**.The State of the system is either at node 8 or node10 and from the state of the system the best decision is taken into account which is **"Where to go next"?**
For n=1 (One more stage to go)
The first stage is represented as follows:

$$f_1(s,X_1)=d_{sX_1}$$

and

$$f_1^*(s)=\min f_1(s,X_1)$$

Here s is the state variable ,X_1 is the decision variable ,$f_1^*(s)$ is the distance function and X_1^* is the best decision variable .
The state variable is either at node 8 or at node 10 i.e the truck can be at any of these two nodes. The decision now is very obvious that the truck

s	X_1	$f_1^*(s)$	X_1^*
8	9	8	9
10	9	9	9

Table 31: First Stage

s	X_2		f_2^*(s)	X_2^*
	node8	node10		
5	6+8 = 14	8+9 = 17	14	8
6	9 + 8 =17	7 + 9 =16	16	10
7	5+8 = 13	7+9 = 17	13	8

Table 32: second Stage

will directly go to the node 9 because it is the last stopping point. From the network diagram shown above it is clear that the distance between the node 8 and node 9 is 8Km and between the node 10 and node 9 is 9 km .The best decision is to go to node 9 via node 8 or node 10.Right now since we have only one alternative for X_1 so it automatically becomes X_1^* which is the best decision.
Similarly it is solved for n=2,3 and 4.
For n=2(Two more stages to go)
The second stage is represented below:

$$f_2(s, X_2 = d_{sX_2} + f_1^*(x_2)$$

and

$$f_2^*(s) = min f_2(s, X_2)$$

For n=3(Three more stages to go)
The third stage is represented below:

$$f_3(s, X_3 = d_{sX_3} + f_2^*(x_3)$$

and

$$f_3^*(s) = min f_3(s, X_3)$$

For n=4(Four more stages to go)
The fourth stage is represented below:

$$f_4(s, X_4 = d_{sX_4} + f_3^*(x_4)$$

and

$$f_4^*(s) = min f_4(s, X_4)$$

s	X_3			$f_3^*(s)$	X_3^*
	node 5	node 6	node 7		
2	4+14 = 18	7+16 = 23	8+13=21	18	5
3	8+14 =22	10 + 16 =26	5+18=18	18	7
4	4+14 = 18	5+16 = 21	7+13=20	18	5

Table 33: Third Stage

s	X_4			$f_4^*(s)$	X_4^*
	node 2	node 3	node 4		
1	5+18 = 23	5+18 = 23	6+18=24	23	2,3

Table 34

Result of final evacuated path

Assigned Vehicle	Evacuated Path		Total Distance
Truck 1	$1->2->5->8->9$	$1->3->7->8->9$	23 km

Table 35

Truck 2 follows the same route as followed by Truck 1 because the destination is the same.
So the Optimal Path of both the trucks is Model Town$->$GTB Nagar$->$ Shastri Nagar$->$Kohat Enclave $->$Rohini east or Model Town$->$Civil Lines$->$Kanhya Nagar$->$Kohat Enclave$->$Rohini east
The reversed path is used to reach back to the hub i.e Model Town , New Delhi.

Algorithm 4

```
1 IMPORT numpy as np
2 IMPORT sys
3 IMPORT math
4 from queue IMPORT Queue
5 SET adj_edges
6 SET vertex
7 SET mp
8 mp[1][2]=1
9 mp[2][1]=1
10 dp =[[-1](vertex+1)](vertex+1)
11 SET final_level
12 DEFINE FUNCTION
shortest_path_with_dp(level,x,y,maxi):
13   IF x > maxi:
14      RETURN 0
15  IF dp[x][y]!=-1:
16     RETURN dp[x][y]
17   SET ans TO sys.maxsize
18   FOR i IN level[x]:
19      SET i TO int(i)
20      IF mp[int(y)][i] EQUALS 0:
21         continue
22      SET ans1 TO
mp[int(i)][y]+shortest_path_with_dp(level,x+1,int(i
),maxi)
23      IF ans1 < ans:
24         SET ans TO ans1
25         final_level[x]=i
26    dp[x][y]=ans
27   RETURN ans
28 DEFINE FUNCTION main():
29   SET dist TO [0]*(vertex+1)
30   edges =[np.array([])]*(vertex+1)
31   SET visit TO [False]*(vertex+1)
32   SET level TO [np.array([])]*(vertex+1)
33   FOR arr IN adj_edges:
34      SET edges[int(arr[0])] TO
np.append(edges[arr[0]],int(arr[1]))
35    SET q TO Queue(vertex)
36   q.put(1)
37   dist[1]=0
38   visit[1]=1
39   WHILE q.empty() EQUALS False:
40      SET x TO int(q.get())
41      visit[x]=1
42      FOR i IN edges[x]:
43         IF visit[int(i)] EQUALS True:
44            continue
45        dist[int(i)]=dist[int(x)]+1
46         q.put(i)
47      SET maxi TO 0
48   FOR i IN range(vertex):
49      SET i TO int(i)
50      SET level[dist[i+1]] TO
np.append(level[dist[i+1]],(i+1))
51      IF dist[i+1]>=maxi:
52         SET maxi TO dist[i+1]
53   final_level[0]=1
54   shortest_path_with_dp(level,0,1,maxi)
55   FOR i IN range(maxi+1):
56      OUTPUT(final_level[i])
57  RETURN
58 main ()
```

2.6 *Scenario 2:Last Mile Delivery*

Last mile delivery refers to the absolute last step of the delivery when a package is moved from a hub point to its last objective which, normally, is an customer home or retail location.

This is the most basic step in the delivery interaction, and the one that companies need to guarantee is just about as speedy and effective as could really be expected. This is to keep up the constantly expanding buyer interest for quick delivery.

For our research analysis,Considering Xpressbees has single hub situated in model town,delhi.we have a set of customers who have various demands and vechile has a restricted limit capacity. using clarks and wright saving based algorithm we are tackling a vechile routing problem with capacity from hub to various customers.

Note:

1. each customer must be served only once (within one route),.
2. the limit of serving vehicles should not be exceded.
3. limitation of the maximum duration, i.e. the length of one route.
4. Each Customer has a specific demand.
5. Objective is to limit the distance under this requirement.

The fleet of vehicles contains two (2) bikes with 15 unit capacity each. In the table below the data of all the customers is mentioned.

Customers	Area	Demand
1	Patel Nagar	4 units
2	Rajouri Garden	6 units
3	Janakpuri	3 units
4	Paschimvihar	2 units
5	Kirti Nagar	3 units
6	Vikaspuri	2 units

Figure 16: Customer data

There is a single depot(Hub) which is denoted as 0 and there are sets of customers which is denoted by 1,2,3,4,5 and 6.Atleast two bikes will move from the hub and will transport goods to meet the requirements of these six customers.
In the next step for six cities create a distance matrix between every customer and the depot using Google Maps.

Distance	0	1	2	3	4	5	6
0	-	20	18	14	16	12	19
1		-	22	18	30	26	28
2			-	32	20	22	21
3				-	20	22	21
4					-	30	32
5						-	26
6							-

Table 34: Distance Matrix

2.6.1 Heuirstic solution using clarks and wright saving based algorithm

Because solving this problem precisely might be difficult, different heuristics have been developed. Clarke and Wright's Savings Algorithm is one of the most conceptually basic heuristics. Its core concept is pretty straightforward. Consider a depot with D demand points and n demand points. Assume that the initial VRP solution involves deploying n cars and dispatching one vehicle to each of the n demand points. This solution's overall tour length is plainly 2 d. (D, i). If we now employ a single vehicle to serve two points, say I and j, in a single trip, the total distance travelled is cut in half. Its core concept is pretty straightforward. Consider a stop D and n demand scenario. Assume that the initial solution for the VRP is using n cars and deploying one vehicle to each of the n demand points.The complete visit length of this trip is, clearly,

$$\sum_{i=1}^{n} 2d(D,i)$$

suppose if we use single vehicle to serve two points, say i and j, on a single trip, the distance will be

$$s(i,j) = 2d(D,i) + 2d(D,j) - [d(D,i) + d(i,j) + d(D,j)]$$
$$s(i,j) = d(D,i) + d(D,j) - d(i,j)$$

Now from $S(i,j) = d(D,i) + d(D,j) - d(i,j)$ we create the Savings matrix table which is the main step for the Clark and Wright solution.
We have $n = 6$,so for saving $S(i,j)$ we have $\binom{n}{2}$ values. In this problem we have $\binom{6}{2}$ = 15 values.
Let us try and compute the solution based on savings algorithm Then, we combine courses routes using the savings sorted in descending order and taking into account the capacity constraints of the vehicles.
For $1-2$ the total customer requirement is 10units and and the vehicle capacity is 15units.so we can put items of customer 1 and customer 2 in same vehicle and move through routes $1-2$. The next saving is from $1-3$ and the requirement of customer 1 ,2 and 3 is $4+6+3 = 13 units$ which is less than the vehicle capacity(15units),so we can put items of customer 3 with customer 1 and customer 2 in same vehicle and move through route $1-3$. The next saving is from $2-6$ and we already have items of customer1,customer2 and customer3 in same vehicle so requirement of customer1 ,customer2,customer3 and customer6 is$4+6+3+6 =$ 19 which exceeds the vehicle capacity so we cannot move through route

Routes	Savings
1→2	16
1→3	16
2→6	16
2→4	14
3→6	12
1→6	11
3→4	10
2→5	8
1→4	6
1→5	6
5→6	5
3→5	4
4→6	3
2→3	0
4→5	-2

Figure 17: Savings Matrix

$2-6$. similarly we cannot use routes $2-4,3-6,1-6,3-4,2-5,1-4$ and $1-5$. Next,we have route $5-6$ which is not travelled by vehicle 1,we can put the items of customer 5 and customer 6 in another vehicle as the customer requirements is less than the vehicle capacity.Vehicle can't travel through route $3-5$ as route 3 lies in one vehicle and route 5 lies in another vehicle.for route $4-5$ we can place the things of customer 4 and customer 5 in second vehicle and travel through route $4-5$ as the customer requirement is less than vehicle capacity.

Routes	Demand	Capacity	Assigned Vehicle
1-2	10	15	Vehicle 1
1-3	13	15	Vehicle 1
~~2-6~~	~~19~~	~~15~~	-
~~2-4~~	~~18~~	~~15~~	-
~~3-6~~	~~19~~	~~15~~	-
~~1-6~~	~~19~~	~~15~~	-
~~3-4~~	~~18~~	~~15~~	-
~~2-5~~	~~16~~	~~15~~	-
~~1-4~~	~~18~~	~~15~~	-
~~1-5~~	~~16~~	~~15~~	
5-6	9	15	Vehicle 2
3-5	-	-	-
4-6	14	15	Vehicle 2

Final Solution

For Vehicle 1:

Route	Savings
1 - 2	16
1 -3	16

For Vehicle 2:

Route	Savings
5 - 6	5
4 - 6	3

Total Savings by Vehicle 1 and Vehicle 2 is 16 + 16 + 5 + 5 = 40
For Vehicle 1 the optimal path is Depot(ModelTown)-Customer2(Rajouri Garden)-Customer1(PatelNagar)-Customer3(Janakpuri)-Depot(ModelTown)
For Vehicle 2 the optimal path is Depot(ModelTown)-Customer5(Kirti Nagar)-Customer6(Vikaspuri)-Customer4(Paschim Vihar)-Depot(ModelTown))

Result of Final Evacuation Path

AssignedVehicle	Evacuationpath	TotalDistance	Savings	Requirement	Capacity
Vehicle1	0->2->1->3->0	72km	32	13	15
Vehicle2	0->5->6->4->0	86km	8	14	15

The Total Distance Travelled by these two vehicles is 72 + 86 = 158 Now if the vehicle go indidually to each one of the customers and comes back then it would have travelled 20 + 18 + 14 + 16 + 12 + 19 = 99 *2 = 198 And by using clark's wright saving based algorithm it actually travelled 158 and the savings is 198 – 158 = 40 , which is the same saving we obtained using saving based algorithm.

Algorithm 5

```
1 SET curr_truck_capacity
2 SET mp
3 SET vertex
4 SET adj
5 DEFINE FUNCTION main():
6   SET curr_trucks TO []
7   SET new_adj_list TO []
8    SET weight_list TO []
9   FOR values IN adj:
10       SET flag TO 0
11       IF mp[values[0]] > 0 and mp[values[1]] >
0:
12          continue
13       FOR i IN range(len(new_adj_list)):
14          SET keys TO new_adj_list[i]
15          IF values[0] IN keys and values[1] IN
keys:
16             SET flag TO 1
17            continue
18          FOR j IN range(len(keys)):
19             IF keys[j] EQUALS values[0] or
keys[j] EQUALS values[1]:
20                OUTPUT(weight_list[i])
21                IF weight_list[i] + values[3] <=
curr_truck_capacity:
22                   weight_list[i] += values[3]
23                   SET mp[values[0]] TO 1
24                   SET mp[values[1]] TO 1
25                   SET flag TO 1
26                   IF keys[j] != values[0]:
27                      keys.append(values[0]
28                      break
29                   ELSE:
30                      keys.append((values[1]))
31                      break
32          IF flag EQUALS 1:
33             break
34       IF flag EQUALS 0:
35          new_adj_list.append([values[0],
values[1]])
36          weight_list.append(values[3])
37    OUTPUT("truck weighted list : ");
38    OUTPUT(weight_list)
39    OUTPUT("Final path IN each truck")
40    OUTPUT(new_adj_list)
41 main ()
```

2.7 *Scenario 3:Hyper-Local Delivery*

Hyper local delivery refers to the most common way of delivering items from a sellers to customers.. The process includes a delivery agent getting items from a seller and delivering them directly to the customer's location. Presently, hyper local delivery progressively liked by brands because of the comfort and usefulness they offer.

In this situation, two depots are incorporated one is seller and other is hub and a set of customer is there. The deliver person picked the items from sellers delivers to the customers and then move to the hub.This is vehicle routing problem with with pickup and delivery,In this vehicle capacity and time window requirement is added.

The objective of the problem is to minimize the total time taken to deliver the goods from seller to customer.Sometimes Customers wants their items to be delivered within a time for instance within 3 to 4 hours of their order placed so in this case vehicle leave the sellers's warehouse and visited each customer exactly once but with in a predefined window and move to the hub.

Let $[(a_i, b_i)]$ be the time constraint on each vertex i where $a_i \leq b_i$.

Assumptions:

1. If the vehicle arrives earlier than the beginning of the time window,it can wait until that time.
2. It cannot provide service to the customers if it arrives later than the end of the time window.

Along with time window we have other constraints as well like each customer must be served only once (within one route), the limit of serving vehicles should not be exceeded,and Each Customer has a specific demand.
The diagram below shows the locations to visit in blue and the depots in black. The time windows are shown above each location.
Information of different customers are mentioned below with their demand and explicit time window they need to received their parcel.

customer	Area	Demand	TimeWindow
1	Azad Nagar	4 units	[9:30,10:00]
2	Geeta Colony	6 units	[9:00,10:00]
3	Dilshad Colony	3 units	[9:00,10:00
4	Dallupura	5 units	[8:30,10:00]
5	jagatpuri	3 units	[9:00,10:00]
6	Anand Vihar Colony	6 units	[8:30,10:00]

Figure 18: Area with demand and time windows

Information of depots are mentioned below with their explicit time window to leave the depot and reached the depot.

Depot	Area	Timewindow
1	Kamla Nagar	[8,8]
2	Model Town	[10,12]

Figure 19: Area with time windows

There is two depot seller's warehouse and a hub which is denoted as 0 and 7 and there are sets of customers which is denoted by 1,2,3,4,5 and 6.vehicle will move from the sellers and will transport goods to meet the requirements of these six customers and then move to the hub.Our objective is to find the minimum time taken such that mimimum number of vehicles are used.
In the next step for six cities create a time matrix between every customer and the depot using Google Maps.

Time(in min)	0	1	2	3	4	5	6	7
0	-	20	18	14	16	12	19	22
1		-	22	18	30	26	28	20
2			-	32	20	22	21	18
3				-	20	22	21	14
4					-	30	32	16
5						-	26	12
6								19
7								-

Table 35: Time Matrix

2.7.1 Heuristic solution using Holmes Parker's method

The Clarke and Wright saving algorithm has its own set of constraints. It is possible that it will not provide us with a good heuristic solution because some of the savings will be unable to be used in fresh attempts by people to work better, which would lead to Holmes' heuristic. The Holmes and Parker heuristic, which is superior to the Clarke and Wright heuristic, can be used to improve the solution.

This method involves following steps

1. This is an expansion of the Savings Algorithm; in the first phase, we will delete the first maximum saving examined in the optimal Savings Algorithm Iteration.
2. Following the elimination, we'll begin iteration with the next highest saving.
3. Using the Clarke and Wright approach, repeat the process.
4. Steps 3 and 4 should be repeated until you get the best answer.
5. If you have an optimal solution, you can stop iterating.

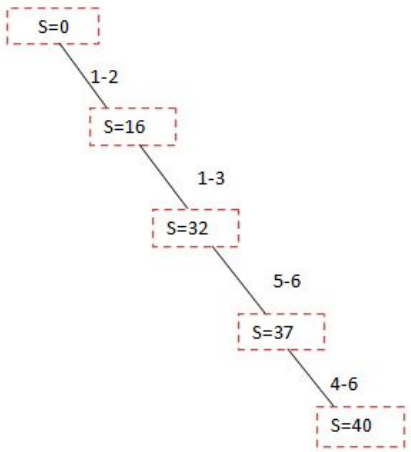
In saving based algorithm,$S(i,j) = d(D,i) + d(D,j) - d(i,j)$ This saving equation utilizes distance between source to objective to process the savings.for our research analysis we are modifying the formula in terms of time taken and savings will compute on the basis the time traveled.

$$S(i,j) = t(D,i) + t(D,j) - t(i,j)$$

Let us try and compute the saving matrix based on clark and wright saving algorithm

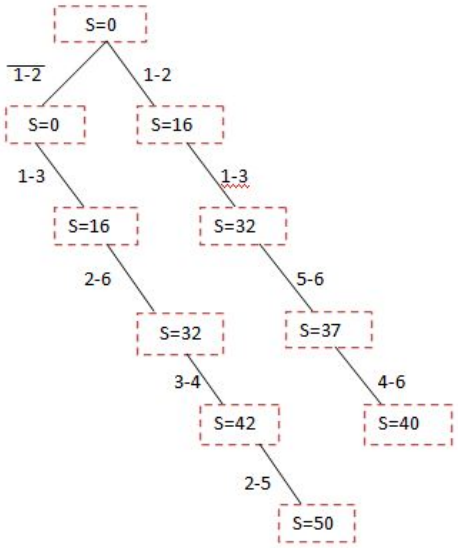
Routes	Savings
1→ 2	16
1→ 3	16
2→ 6	16
2→ 4	14
3→ 6	12
1→ 6	11
3→ 4	10
2→ 5	8
1→ 4	6
1→ 5	6
5→ 6	5
3→ 5	4
4→ 6	3
2→ 3	0
4→ 5	-2

The final solution using clarks and wright algorithm as we did in last scenario is given below:
Let us say that we start with saving s=0,which means there is no saving.The first one from the saving matrix is $1-2$ with savings(s) is equals to 16 min.we can use 1 and 2 together as customer's requirement is less than the vehicle capacity.similarly we can use 1 and 3 and have the saving of 32 min,then we use 5 and 6 together and after that we use 4 and 6 together to get saving(s) = 40min. so The clark and wright saving algorithm gave us this kind of solution with saving (s)=40 so The clark and wright



saving algorithm gave us this kind of solution with saving (s)=40 min.
Now we apply holmes parker method and try to compute the better solution.suppose we proceed from the clark and wright by branching from

s=0,by eliminating 1-2 means not combining 1 and 2 together to obtain the solution and then repeat the same clark and wright method.
we obtained the following solution



The saving we obtained from saving based algorithm was 40 min and using Holmes and Parker heuristic we obtained saving of 50 min.
Result of Final evacuation path

Assigned Vehicle	Evacuationpath	Total Time	Requirement	Capacity
Vehicle1	0->4->3->1->7	74min	12	15
Vehicle2	0->6->2->5->7	74min	15	15

For vehicle 1:
seller warehouse(depot) $->$ Customer4(Dalupura)$->$Customer3(Dilshad Colony)
$->$Customer1(AzadNagar)$->$Hub(depot)
For vehicle 1:
seller warehouse(depot) $->$ Customer6(Anand Vihar Colony)$->$Customer2(Geeta Colony)$->$Customer5(Jagatpuri)$->$Hub(depot)
The Total taken taken by these two vehicles is $74 + 74 = 148$ min Now if the vehicle go individually to each one of the customers from depot (0) and then move to depot(7) then it would have travelled $20 + 18 + 14 + 16 + 12 + 19 + 20 + 18 + 14 + 16 + 12 + 19 = 198$ min And by using

Holmes and Parker heuristic it actually travelled 148 and the savings is 198–148 = 50 , which is the same saving we obtained using Holmes and Parker heuristic.
Also we have added time windows requirement so we need to ensure vehicle came to customer within a predefined time window.
For vehicle 1:
0 = depart from the seller's warehouse at 8:00 am
4 = arrives at 8:16 am and depart at 8:35 am (14 min waiting time + 5 min service time)
3 = arrives at 8:55 am and depart at 9:07 am (5 min waiting time + 7 min service time)
1 = arrives at 9:25 am and leaves at 9:40 am (5 min waiting time + 10 min service time)
7 = arrives at the hub at10:00 am with total duration of 120min(74 min travelling time and 46 min waiting and service time.)
For vehicle 2:
0 = depart from the seller's warehouse at 8:00 am
6 = arrives at 8:19 am and depart at 8:41 am (21 min waiting time + 10 min service time)
2 = arrives at 9:02 am and depart at 9:15 am (15 min service time)
5 = arrives at 9:37 am and leaves at 9:50 am (13 min service time)
7 = arrives at the hub at10:02 am with total duration of 133min(74 min travelling time and 59 min waiting and service time.)
So the Optimal path is 0(seller's warehouse) [8:00,8:00] →Customer4[8:16,8:35] →customer3[8:55,9:07] →1[9:25,9:40] →7[10:00,10:00] for vehicle 1 and 0(seller's warehouse)[8:00,8:00] →Customer6[8:19,8:41] →Customer2[9:02,9:15] →Customer5[9:37,9:50] →Customer7[10:02,10:02] for vehicle2.

Algorithm 6

```
1 SET curr_truck_capacity
2 SET mp
3 SET vertex
4 SET adj
5 DEFINE FUNCTION main():
6   SET curr_trucks
7   SET new_adj_list
8    SET weight_list
9   SET time_list
10   adj.pop(0)
11   FOR k IN range(len(adj)):
12     IF k EQUALS 0 :
13        continue
14      SET values TO adj[k]
15       SET flag TO 0
16       IF mp[values[0]] > 0 and mp[values[1]] >
0:
17          continue
18       FOR i IN range(len(new_adj_list)):
19          SET keys TO new_adj_list[i]
20          IF values[0] IN keys and values[1] IN
keys:
21             SET flag TO 1
22             continue
23          FOR j IN range(len(keys)):
24             IF keys[j] EQUALS values[0] or
keys[j] EQUALS values[1]:
25                OUTPUT(weight_list[i])
26                IF weight_list[i] + values[3] <=
curr_truck_capacity:
27                   weight_list[i] += values[3]
28                   time_list[i] += values[4]
29                    SET mp[values[0]] TO 1
30                   SET mp[values[1]] TO 1
31                    SET flag TO 1
32                    IF keys[j] != values[0]:
33                       keys.append(values[0])
34                        break
35                    ELSE:
36                       keys.append((values[1]))
37                       break
38          IF flag EQUALS 1:
39             break
40       IF flag EQUALS 0:
41          new_adj_list.append([values[0],
values[1]])
42          weight_list.append(values[3])
43          time_list.append(values[4])
44    OUTPUT("time list : ")
45    OUTPUT(time_list)
46    OUTPUT("truck weighted list : ");
47    OUTPUT(weight_list)
48    OUTPUT("Final path IN each truck")
49    OUTPUT(new_adj_list)
50 main()
```

3 Chapter 3

3.1 Results

Experiments with various scenarios were conducted to ensure that the suggested algorithms were effective. The proposed model was also built in the Python programming language, which a logistics company may use to solve its problem.To begin, the Branch and Bound penalty approach generates 18 branches, as illustrated in Figure 8, and the best route is 0→4→3→1→7→6→5→10→2→9→8→0 with total distance of 49.3 km.

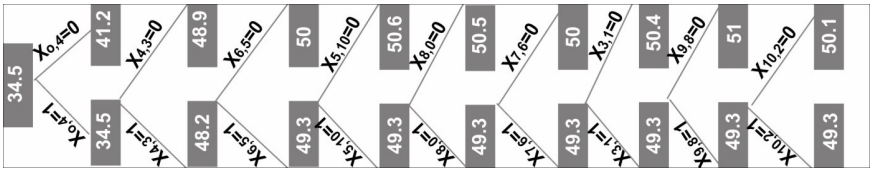


Figure 20: Finding solution through branch and bound penalty method.

In the case of signalised junctions, the best route is 1→ 5→9→11→12→ 13 with a weight of 40.1 minutes. The shortest path becomes the fastest path since the weight used is travel time rather than mileage. As a result, the fastest route is 1→ 5→9→11→ 12→13 (40.1 minutes).
In scenario 1, the least number of trucks required to contain all of the boxes is two, as determined by the product loading algorithm, and the routing of these two trucks was determined by dynamic programming to be either 1→2→5→8 or 1→3→7→8 , with a total distance of 23 km.
In scenerio 2, The results show that only two out of five vehicles were used to deliver the items, and Table 15 shows the optimal route for both vehicles. if the vehicle travelled sequentially to each customer and returned, it would have travelled 20 + 18 + 14 + 16 + 12 + 19 = 99.2 = 198 km, however using Clark's Wright Savings Based Algorithm, it only travelled 158 km, saving 198 – 158= 40 km. As a result, the logistics company saves money by minimizing the distance travelled and saving vehicles.

Vehicles	Optimal Path	Distance(Km)	Savings(km)	Demand	Capacity
Vehicle 1	0→2→1→3→0	72	32	13 units	15 units
Vehicle 2	0→5→6→4→0	86	8	14 units	15 units

Figure 21: Final evacuation path

In scenario 3,The result show that vehicles will leave the seller's warehouse at 08:00 am and meet the needs of all customers within a predeter-

mined window, and then return to the hub. Additionally, by applying the Holmes and Parker heuristic, the vehicle actually travelled 148 minutes, saving 198-148=50 minutes. The final evacuation path with in specified time slots can be observed in Table 16.

Vehicles	Optimal Path	TimeTaken(min)	Demand	Capacity
Vehicle 1	$0[08:00am, 08:00am] \rightarrow 4[08:16am, 08:35am] \rightarrow 3[08:55am, 09:07am] \rightarrow 1[09:25am, 09:40am] \rightarrow 7[10:00am, 10:00am]$	74	12 units	15 units
Vehicle 2	0[08:00am,08:00am]→6[08:19am,08:41am]→2[09:02am,09:15am] →5[09:37am,09:50am]→7[10:02am,10:02am]	74	15 units	15 units

Figure 22: Final evacuation path

3.2 Discussion

The importance, diversity, and uniqueness of vehicle routing in cities can all be shown in the preceding sections. For a big portion of Logistics, vehicle routing is a challenge. There are a variety of items, organizations, and timeframes to consider. Case-study papers cover a wide range of themes, each with its own set of organizational and technological requirements. However, we preserve different qualities for vehicle routing problems based on the study of the systems provided and the studies cited in the previous sections. These characteristics are important for logistics and valuable from an academic standpoint, according to us. The following is a list of them. The time-dependency is the first distinguishing feature. Traffic congestion and peak hours at traffic lights should be considered while planning efficient transportation in urban areas. The Dijkstra algorithm was combined with a highway capacity manual to create a new algorithm. The loading of products into trucks and their routing is a second key topic. Many publications discuss how innovative strategies can help to enhance routing. A third topic concerns last-mile and hyperlocal delivery, which are subject to various constraints such as time limits, client needs, truck capacity, and so on. Finally, the evolution of new communication technologies and the dynamism of the logistics industry stimulate the study of dynamic VRP, in which vehicle routes can be re-optimized based on various forms of data (actual travel times, new requests, unexpected events).

3.3 CONCLUSION

In this paper, a mathematical model was set up on a real-life example of the XpressBees Logistics Company after defining a transport problem where exact and Heuristic methods were applied to solve the problem of routing vehicles in the transport network. The purpose of applying mathematical methods is to reduce transportation costs by optimizing routes and the number of vehicles. Minimizing transportation costs brings savings and increases profits for businesses, so great attention must be taken in organizing transportation. As a result, any logistic company with the same difficulty as the one described above can apply the model to enhance their routing problem and revenues.

4 References

References

[1] Wu, Yenchun Goh, Mark Yuan, Chih-Hung Huang, Shan-Huen. (2017). Logistics management research collaboration in Asia. The International Journal of Logistics Management. DOI:28. 206-223. 10.1108/IJLM-09-2013-0104.

[2] Eldrandaly, Khalid Ahmed, AbdelHadi. (2008). Routing Problems: A Survey The 43rd Annual Conference on Statistics, Computer Sciences and Operations Research.

[3] Sangwan, Shabnam. (2018). Literature Review on Travelling Salesman Problem. International Journal of Research. 5. 1152.

[4] Toro O., E. M., Escobar Z., A. H., Granada E., M. (2015). Literature review on the vehicle routing problem in the green transportation. Luna Azul, (42), 362–387. DOI:10.17151/luaz.2016.42.21

[5] Liong, C.-Y Wan, I. Omar, Khairuddin. (2008). Vehicle routing problem: Models and solutions. Journal of Quality Measurement and Analysis. 4. 205-218.

[6] Braekers, Kris Ramaekers, Katrien Nieuwenhuyse, Inneke. (2015). The Vehicle Routing Problem: State of the Art Classification and Review. Computers Industrial Engineering. 99. DOI:10.1016/j.cie.2015.12.007.

[7] Hrablik, Henrieta Horňáková, Natália Babčanová, Dagmar. (2015). Use of operational research methods in logistics. Carpathian Logistics Congress, Priessnitz Spa, Jesenik, Czech Republic, EU, November4th - 6th 2015(307-312). Proceedings of the conference were published in Web of Science.

[8] Satyananda, D., Wahyuningsih, S. (2019). VND in CVRP, MDVRP, and VRPTW cases. Journal of Physics: Conference Series, (1), 012025. DOI:10.1088/1742-6596/1320/1/012025

[9] M. A. H. Akhand, Zahrul Jannat Peya, T. Sultana and Al-Mahmud, ”Solving Capacitated Vehicle Routing Problem with route optimization using Swarm Intelligence,” 2015 2nd International Conference

on Electrical Information and Communication Technologies (EICT), 2015, pp. 112-117,DOI 10.1109/EICT.2015.7391932.

[10] Gan, X., Wang, Y., Li, S., Niu, B. (2012). Vehicle Routing Problem with Time Windows and Simultaneous Delivery and Pick-Up Service Based on MCPSO. Mathematical Problems in Engineering, 1–11. DOI:10.1155/2012/104279

[11] Ombuki, Beatrice Ross, Brian Hanshar, Franklin. (2006). Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows.Applied Intelligence. DOI:24. 17-30. 10.1007/s10489-006-6926-z.

[12] Sharma, Rahul Patel, Pinakin Umrigar, Nekzad Zala, Dr. (2018). Delay estimation at signalized intersections under heterogeneous traffic conditions DOI:10.13140/RG.2.2.16273.76644

[13] evaluation of delay characteristics at signalized intersections for improvement in level of service. (2018). international journal for traffic and transport engineering, (3), 309– 319. doi:10.7708/ijtte.2018.8(3).05

[14] Alam, M.A. and Faruq, M.O. 2019. Finding Shortest Path for Road Network Using Dijkstra's Algorithm. Bangladesh Journal of Multidisciplinary Scientific Research. 1, 2 (Jul. 2019), 41-45.DOI:10.46281/bjmsr.v1i2.366.

[15] Javaid, Adeel. (2013). Understanding Dijkstra Algorithm. SSRN Electronic Journal.DOI:10.2139/ssrn.2340905

[16] Sitinjak, Anna Pasaribu, Elvina Simarmata, Justin Putra, Tedy Mawengkang, Herman. (2018). The Analysis of Forward and Backward Dynamic Programming for Multistage Graph. IOP Conference Series: Materials Science and Engineering. 300.

[17] Caccetta, L., Alameen, M., Abdul-Niby, M. (2013). An Improved Clarke and Wright Algorithm to Solve the Capacitated Vehicle Routing Problem. Engineering, Technology Applied Science Research, (2), 413–415. DOI:10.48084/etasr.292

[18] Tantikorn Pichpibul, Ruengsak Kawtummachai, "A Heuristic Approach Based on Clarke-Wright Algorithm for Open Vehicle Routing Problem", The Scientific World Journal, vol. 2013, Article ID 874349, 11 pages, 2013. DOI:10.1155/2013/874349

[19] Rajulapudi Bala Sai Shankar,Dr. K. Dharma Reddy,Dr. P. Venkataramaiah.”Solution to a Capacitated Vehicle Routing Problem Using Heuristics and Firefly Algorithm. ”International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 21 (2018) pp. 15247-15254.