Vehicle routing problem with capacity constraint

Dissertation submitted in
partial fulfilment of the requirement
for the award of the degree of

Master of Technology

In

Computer Science and Engineering

By

SACHIN JAKHAR

University Roll No. 2K12/CSE/17

Under the Esteemed Guidance of

Mr. MANOJ SETHI
Computer Engineering Department, DTU

2012-2014

COMPUTER ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY

DELHI – 110042, INDIA

Computer Engineering Department
Delhi Technological University
Delhi-110042
www.dce.edu

CERTIFICATE

This is to certify that the dissertation titled "**Vehicle routing problem with capacity constraint**" is a bonafide record of work done by **Sachin Jakhar, Roll No. 2K12/CSE/17** at **Delhi Technological University** for partial fulfilment of the requirements for the degree of Master of Technology in Computer Science & Engineering. This project was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

(Mr. Manoj sethi)
Department of Computer Engineering
Delhi Technological University

Date: _____

ACKNOWLEDGEMENT

First of all, I would like to express my deep sense of respect and gratitude to my project supervisor Mr. Manoj sethi for providing the opportunity of carrying out this project and being the guiding force behind this work. I am deeply indebted to him for the support, advice and encouragement he provided without which the project could not have been a success.

Secondly, I am grateful to Dr. O.P.Verma, HOD, Computer Engineering Department, DTU for his immense support. I would also like to acknowledge Delhi Technological University for providing the right academic resources and environment for this work to be carried out.

Last but not the least I would like to express sincere gratitude to my parents for constantly encouraging me during the completion of work.

Sachin jakhar
University Roll no: 2K12/CSE/17
M.Tech (Computer Science & Engineering)
Department of Computer Engineering
Delhi Technological University
Delhi – 110042

Date……

ABSTRACT

In wireless network based fourth generation communication system has played vital role for exchanging the information to the user on movement of mobile node.Wireless network may be broadly classified as homogeneous and heterogeneous. Wireless Homogeneous Network set consist of identical network containing same set of attributes , whereas, heterogeneous consist of dissimilar network containing different kinds of attributes. For examples new access technologies are such as IEEE 802.11(Wi-Fi Network), IEEE 802.16 (Wi-Max.), Bluetooth etc. Wireless LANs are changing the way of communication of public wireless access.They are having specific parameters viz. diverse variety of wireless access standards, network availability, coverage and Quality of Service (QoS) may rapidly change depending upon the device mobility and fading suffered by the signal.Initially mobile node is attached to specific cell but whenever mobile node wants to move in another cell while call is in progress. The transferring of mobile node from one specific cell to target cell is called handover problem while call should not be interrupted. The complexity of the challenge goes high with the consideration of random device mobility, resources availability, received signal strength, security and geographical topologies such as environments, buildings etc. One algorithms based model is proposed to minimize the number of handoffs across homogenous and heterogeneous wireless network ensuring the security features.

Also, the behavior of the algorithms is studied with change in number of member wireless nodes varying respective parameters viz. homogenous and heterogeneous, RSS, communication range, cost, velocity of mobile node, bandwidth in the system. It is also shown that this algorithm performs better minimizing the unnecessary handoffs with the proposed heterogeneous algorithm.

The proposed algorithm guarantees much more reliable handover in a multiple-operator environment. The thesis also focuses on the handover across heterogamous networks without a trust relation and focus the handover decision problem. The proposed scheme provides a secure and removes the noise like fading in RSS so unnecessary handoffs can be minimal.

Keywords : Vertical Handoff, Mobile Node, Wi-Fi, Wi-Max,Wireless Mess Network

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This section introduces the reader with some factors that motivated me work on this topic. It also provides information about the prerequisites, to have a clear understanding of the problem and it solution.

1.1 Motivation

With the transition of shop from shopping mall and super markets to internet, the trend of home deliveries is passing through a new environment. Online shopping sites are providing more facilities to attract the customers. They are taking help to discount coupons, referrals; Cash on delivery, try and buy, etc. The delivery to the customer within minimum time is also a point of focus for these companies. The things are quite simple on a small scale, but with the increase in scales it becomes quite complicated for the companies to manage the whole process, from receiving the request to the delivery an taking feedback from the customer. And the process does not stop here, as the companies want the customer to revisit their site. This whole process needs to be systematic, well planed and organize. One of the sub processes of this whole process is the allocation of routes to delivery man. After receiving the request from the customers, when the deliveries are ready to deliver, one big question arises, how to deliver? Answering this question is not as simple as said because companies need to deliver these goods in such a way that they are delivered to the customers on time, while minimizing the total distance travelled by the delivery men, hence reducing the transportation cost for the company.

Easy route is developed as a part of this research work, which make use of various heuristic and meta-heuristic algorithms to solve this problem. We have also proposed a variation of existing meta-heuristic algo that is proved to give better results to its counterpart.

1.2. Prerequisites

As we know, Vehicle Routing Problem (VRP) is combinatorial problem, which is quite hard to solve, has been a part of research since 1960's. The mathematical representation of the

problem requires the in-depth knowledge of mixed integer Programming. Interested readers are advised to go through the well-presented tutorial from J. Cole Smith [2]. VRP is generally represented in the form of graphs (as stated in section 2.3),which makes it necessary to have a good knowledge of graph theory. Some topics from graph theory like representation of graphs in the form of matrices, checking the connectivity of the graphs, traversals of the graphs, finding the neighbors of the nodes and the list goes on, require good understanding of the concepts as they are the basic building blocks of the problem. As this problem is closely co-related to the areas of Operation research, readers need to have clear understanding of this field as well.

If any reader is new to this field, it is advised to read [1], in which Gilbert Laporte has very well classified solutions of VRP into different categories. For more information about different heuristic and meta-heuristic approaches, interested reader can go through [3],[4],[5] and [6].

The source code of EazyRoute is available at [7]. It is written in Java and requires the understating of basic concepts of Java and C/C++. More information about downloading and installing them can be gained from their sites.

Chapter 2

Literature Survey

The Vehicle Routing Problem (VRP) aims at designing optimal delivery routes from a central depot to a set of geographically scattered delivery points, satisfying various constraints, such as vehicle capacity, route length, time windows, precedence relations between customers, etc. VRP is a widely studied combinatorial optimization problem that has many applications. Due to its intrinsic difficulty and the size of problems encountered in practice, most solution methods for the VRP are heuristic in nature and lead to high quality, yet probably not optimal solutions (from [9]). The problem was introduced nearly fifty years ago by Dantzig and Ramser [1] and has since given rise to a rich body of research.

Unlike what happens for several well-known combinatorial optimization problems, there does not exist a single universally accepted definition of the VRP because of the diversity of constraints encountered in practice. Most of the research effort has concentrated on a standardized version of the problem, called the classical VRP, with the understanding that many of the algorithms developed for this case, mostly heuristics, can be adapted to suit the more complicated real-life situations(from [10]).

2.1 VARIANTS OF VRP

The vehicle routing problem can be applied to wide variety of real world applications. As a result there are different variations of this problem; some of them are discussed in following sub-sections.

2.1.1. Capacitated VRP (CPRV)

CVRP is a Vehicle Routing Problem (VRP) in which a fixed fleet of delivery vehicles of uniform capacity must service known customer demands for a single commodity from a common depot at minimum transit cost. That is, CVRP is like VRP with the additional constraint that every vehicle must have uniform capacity of a single commodity.

We can find below a formal description for the CVRP:

- **Objective:** The objective is to minimize the vehicle fleet and the sum of travel time, and the total demand of commodities for each route may not exceed the capacity of the vehicle which serves that route.

- **Feasibility:** A solution is feasible if the total quantity assigned to each route does not exceed the capacity of the vehicle which services the route.

- **Formulation:** Let Q denote the capacity of a vehicle. Mathematically, a solution for the CVRP is the same that VRP's one, but with the additional restriction that the total demand of all customers supplied on a route R_i does not exceed the vehicle capacity Q: $\sum_{i=1}^{m} di \leq Q$.

2.1.2. Multiple Depots VRP (MDVRP)

A company may have several depots from which it can serve its customers. If the customers are clustered around depots, then the distribution problem should be modeled as a set of independent VRPs. However, if the customers and the depots are intermingled then a Multi-Depot Vehicle Routing Problem should be solved.

A MDVRP requires the assignment of customers to depots. A fleet of vehicles is based at each depot. Each vehicle originates from one depot, service the customers assigned to that depot, and return to the same depot.

The objective of the problem is to service all customers while minimizing the number of vehicles and travel distance.

We can find below a formal description for the MDVRP:

- **Objective:** The objective is to minimize the vehicle fleet and the sum of travel time, and the total demand of commodities must be served from several depots.

- **Feasibility:** A solution is feasible if each route satisfies the standard VRP constraints and begins and ends at the same depot.

- **Formulation:** The VRP problem is extended to the case wherein we have multiple depots, so we will note the vertex set like, V= {v_1, v2…v_n} U v_0 where V_0= {v_{01}, v02….. v_{0d}} are the vertex representing the depots. Now, a route ***i*** is defined by R_i= {d, v1… v_m, d}, with d ∈ V_0. The cost of a route is calculated like in the case of the standard VRP.

2.1.3. Periodic VRP (PVRP)

In classical VRPs, typically the planning period is a single day. In the case of the Period Vehicle Routing Problem (PVRP), the classical VRP is generalized by extending the planning period to M days.

We define the problem as follows:

- **Objective:** The objective is to minimize the vehicle fleet and the sum of travel time needed to supply all customers.

- **Feasibility:** A solution is feasible if all constraints of VRP are satisfied. Furthermore a vehicle may not return to the depot in the same day it departs. Over the M-day period, each customer must be visited at least once.

- **Formulation:** Minimize the sum of the cost of all routes. Each customer has a known daily demand that must be completely satisfied in only one visit by exactly one vehicle. If the planning period M = 1, then PVRP becomes an instance of the classical VRP. Each customer in PVRP must be visited k times, where *1<=K<=M*. In the classical model of PVRP, the daily demand of a customer is always fixed. The PVRP can be seen as a problem of generating a group of routes for each day so that the constraints involved are satisfied and the global costs are minimized. PVRP can also be seen as a multi-level combinatorial optimization problem:
    - In the first level, the objective is to generate a group of feasible alternatives (combinations) for each customer. For example, if the planning period has t=3 days {d_1,d_2,d_3} then the possible combinations are: 0→000 ; 1→001 ; 2→010 ; 3→011 ; 4→100 ; 5→101 ; 6→ 110 and 7→111. If a customer requests two visits, then this customer has the following visiting

alternatives: {d_1,d_2}, {d_1,d_3}, and {d_2,d_3} (or the options: 3, 5 and 6 of the Table 2.1).

Customer	Diary Demand	No. of Visits	No. of Combinations	Possible Combinations
1	30	1	3	1, 2, 4
2	20	2	3	3, 5, 6
3	20	2	3	3, 5, 6
4	30	2	3	1, 2, 4
5	10	3	1	7

Table 2.1: Problem Statement for PVRP

- In the second level, we must select one of the alternatives for each customer, so that the daily constraints are satisfied. Thus we must select the customers to be visited in each day.
- In the third level, we solve the vehicle routing problem for each day.

2.1.4. Split Delivery VRP (SDVRP)

SDVRP is a relaxation of the VRP wherein it is allowed that the same customer can be served by different vehicles if it reduces overall costs. This relaxation is very important if the sizes of the customer orders are as big as the capacity of a vehicle.

In [11] it is concluded that it is more difficult to obtain the optimal solution in the SDVRP that in the VRP.

- **Objective:** The objective is to minimize the vehicle fleet and the sum of travel time needed to supply all customers.

- **Feasibility:** A solution is feasible if all constraints of VRP are satisfied except that a customer may be supplied by more than one vehicle.

- **Formulation:** Minimize the sum of the cost of all routes. An easy way to transform a VRP into a SDVRP consists on allowing split deliveries by splitting each customer order into a number of smaller indivisible orders [12].

2.1.5 Stochastic VRP (SVRP)

Stochastic VRP (SVRP) is VRP where one or several components of the problem are random. Three different kinds of SVRP are the next examples:

- Stochastic customers: Each customer v_i is present with probability p_i and absent with probability $(1-p_i)$.
- Stochastic demands: The demand d_i of each customer is a random variable.
- Stochastic times: Service times δi and travel times t_{ij} are random variables.

In SVRP, two stages are made for getting a solution. A first solution is determined before knowing the realizations of the random variables. In a second stage, a recourse or corrective action can be taken when the values of the random variables are known.

- **Objective:** The objective is to minimize the vehicle fleet and the sum of travel time needed to supply all customers with random values on each execution for the customers to be served, their demands and/or the service and travel times.
- **Feasibility:** When some data are random, it is no longer possible to require that all constraints be satisfied for all realizations of the random variables. So the decision maker may either require the satisfaction of some constraints with a given probability, or the incorporation into the model of corrective actions to be taken when a constraint is violated.
- **Formulation:** Minimize $\sum_{i<j} cij\ xij + Q(x)$, where
    - x_{ij} is an integer variable equal to the number of times edge (v_i, v_j) appears in the first stage solution. If i,j>1, then x_{ij} can only take the values 0 or 1, if i=1, x_{ij} be equal to 2 if a vehicle makes a return trip between the depot and v_j.

- Q(x) is the expected second stage recourse function. It is problem dependent and is also related to the particular choice of possible recourse actions. For example, in the capacity constrained SVRP with collections, possible recourse actions are:
  - Return to the depot when the vehicle is full in order to unload, and then resume collections as planned.
  - Return to the depot when the vehicle is full as in the previous case and re-optimize the remaining part of the planned route.
  - Planning a preventive return to the depot even if the vehicle is not full. In such case, this decision could depend on the amount already collected and on the distance separating the vehicle from the depot.

A vehicle not yet full may return to the depot if it is known that going to the next customer would cause its capacity to be exceeded.

2.1.6. VRP with Time Windows (VRPTW)

The VRPTW is the same problem that VRP with the additional restriction that in VRPTW a time window is associated with each customer $v \in V$, defining an interval $[e_v,l_v]$ wherein the customer has to be supplied. The interval $[e_0,l_0]$ at the depot is called the scheduling horizon. Here is a formal description of the problem:

- **Objective:** The objective is to minimize the vehicle fleet and the sum of travel time and waiting time needed to supply all customers in their required hours.

- **Feasibility:** The VRPTW is, regarding to VRP, characterized by the following additional restrictions:
  - A solution becomes infeasible if a customer is supplied after the upper bound of its time window.
  - A vehicle arriving before the lower limit of the time window causes additional waiting time on the route.
  - Each route must start and end within the time window associated with the depot.
  - In the case of soft time widows, a later service does not affect the feasibility of the solution, but is penalized by adding a value to the objective function.

- **Formulation:** Let b_0 denote the beginning of service at customer *v*. Now for a route $Ri=(v_0+v_1\ldots\ldots+v_m+v_{m+1})$ to be feasible it must additionally hold $e_{v\ i}< b_{vi} <l_{vi}$, $1< i< m$, and $b_{vm}+\delta_{vm}+c_{vm,0}< l_0$. Provided that a vehicle travels to the next customer as soon as it has finished service at the current customer, b_{vi} can be recursively computed as $b_{vi}= \max\{e_{vi},b_{vi-1}+ \delta_{vi-1}+c_{vi-1,vi}\}$ with $b_0=e_0$ and δ_0 . Thus, a waiting time $wvi = \max\{0,b_{vi}-b_{vi-1}-\delta_{vi-1}-c_{i-1,i}\}$ may be induced at customer v_i. The cost of route i is now given by :

$$C_{VRPTW}(R_i) = \sum_{i=0}^{m} Ci, i+1 + \sum_{i=1}^{m} \delta i + \sum_{i=0}^{m} wvi.$$

For a solution S with routes $R_1\ldots.R_m$, the cost of S is given by:

$$F_{VRPTW}(S) = \sum_{i=1}^{m} (C_{VRPTW}(R_i) + M),$$

Where, ***M*** is a large constant. ***M*** is added because minimization of the fleet size is considered to be the primary objective of the VRPTW. S is said to be feasible if all routes belonging to S are feasible and its customer is served by exactly one route. As described by Solomon [14], we assume that initially all vehicles leave the depot at the earliest possible time e_0. Having obtained a solution of the VRPTW, we adjust the depot departure time of each vehicle to eliminate any unnecessary waiting time.

2.3. Clark and Wright Algorithm [12]

2.3.1. Problem characteristics

The vehicle routing problem, for which the algorithm has been designed, is characterized as follows. From a depot goods must be delivered in given quantities to given customers. For the transportation of the goods a number of vehicles are available, each with a certain capacity with regard to the quantities. Every vehicle that is applied in the solution must cover a route, starting and ending at the depot, on which goods are delivered to one or more customers. The problem is to determine the allocation of the customers among routes, the sequence in which the customers shall be visited on a route, and which vehicle that shall cover a route. The objective is to find a solution which minimizes the total transportation costs. Furthermore, the solution must satisfy the restrictions that every customer is visited

exactly once, where the demanded quantities are delivered, and the total demand on every route must be within the vehicle's capacity. The transportation costs are specified as the cost of driving from any point to any other point. The costs are not necessarily identical in the two directions between two given points.

2.3.2. The savings algorithm

The savings algorithm is a heuristic algorithm, and therefore it does not provide an optimal solution to the problem with certainty. The method does, however, often yield a relatively good solution. That is, a solution which deviates little from the optimal solution. The basic savings concept expresses the cost savings obtained by joining two routes into one route as illustrated in figure 1, where point 0 represents the depot.

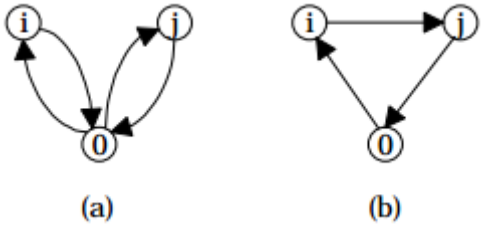

Figure 2.1: Showing saving concept

Initially in figure 1(a) customers i and j are visited on separate routes. An alternative to this is to visit the two customers on the same route, for example in the sequence i-j as illustrated in figure 1(b). Because the transportation costs are given, the savings that result from driving the route in figure 1(b) instead of the two routes in figure 1(a) can be calculated. Denoting the transportation cost between two given points i and j by c_{ij} , the total transportation cost D_a in figure 1(a) is:

$$D_a = c_{0i} + c_{i0} + c_{0j} + c_{j0}$$

Equivalently, the transportation cost Db in figure 1(b) is:

$$D_b = c_{0i} + c_{ij} + c_{j0}$$

By combining the two routes one obtains the savings S_{ij} :

$$S_{ij} = D_a - D_b = c_{i0} + c_{0j} - c_{ij}$$

Relatively large values of Sij indicate that it is attractive, with regard to costs, to visit points i and j on the same route such that point j is visited immediately after point i.

There are two versions of the savings algorithm, a sequential and a parallel version. In the sequential version exactly one route is built at a time (excl. routes with only one customer), while in the parallel version more than one route may be built at a time. In the first step of the savings algorithm the savings for all pairs of customers are calculated, and all pairs of customer points are sorted in descending order of the savings. Subsequently, from the top of the sorted list of point pairs one pair of points is considered at a time.

When a pair of points i-j is considered, the two routes that visit i and j are combined (such that j is visited immediately after i on the resulting route), if this can be done without deleting a previously established direct connection between two customer points, and if the total demand on the resulting route does not exceed the vehicle capacity. In the sequential version one must start anew from the top of the list every time a connection is established between a pair of points (since combinations that were not viable so far now may have become viable), while the parallel version only requires one pass through the list.

The actual way the algorithm works is illustrated in the following by means of a numerical example.

We consider a problem with 5 customers. The transportation costs between all pairs of points are shown in the following table, where 0 represents the depot (the costs are symmetric, and for that reason only the upper half of the table is filled in).

		To					
		0	1	2	3	4	5
From	0	-	28	31	20	25	34
	1		-	21	29	26	20
	2			-	38	20	32
	3				-	30	27
	4					-	25
	5						-

Table 2.2: Representing transportation costs between all pairs of points

The customers' demands that must be delivered from the depot are given in the following table. The vehicle capacity is 100 units.

Customer	Quantity
1	37
2	35
3	30
4	25
5	32

Table 2.3: List of customers and quantity

The savings S_{ij} are calculated to the following values (only the upper half of the table is shown, since the savings are symmetric due to symmetric costs):

		j				
		1	2	3	4	5
i	1	-	38	19	27	42
	2		-	13	36	33
	3			-	15	27
	4				-	34
	5					-

Table 2.4: Evaluation of saving

Now the point pairs are sorted in descending order of the savings. This gives the following sorted list of point pairs:

1-5
1-2
2-4
4-5
2-5
1-4
3-5
1-3
3-4
2-3

2.3.3. The sequential savings algorithm

In the example customers 1 og 5 are considered first. They can be assigned to the same route since their joint demand for 69 units does not exceed the vehicle capacity. Now we establish the connection 1-5, and thereby points 1 and 5 will be neighbors on a route in the final solution.

Next we consider customers 1 and 2. If customers 1 and 2 should be neighbors on a route, this would require the customer sequence 2-1-5 (or 5-1-2) on a route, because we have established already that 1 and 5 must be visited in immediate succession on the same route. The total demand (104) on this route would exceed the vehicle capacity (100). Therefore, customers 1 and 2 are not connected.

If points 2 and 4, which is the next pair in the list, were connected at this stage, we would be building more than one route (1-5 and 2-4). Since the sequential version of the algorithm is limited to making only one route at a time, we disregard the point pair 2 and 4.

The combination of the next pair of points, 4 and 5, results in the route 1-5-4 with a total demand of 94. This combination is feasible, and we establish the connection between 4 and 5 as a part of the solution. Running through the list we find that due to the capacity restriction no more points can be added to the route. Thereby we have formed the route 0-1-5-4-0. In the next pass of the savings list we only find the point pair 2 and 3. These two points can be visited on the same route, and we make the route 0-2-3-0.

The sequential algorithm has constructed a solution with two routes. The transportation costs for the route 0-1-5-4-0 are 98, and for the route 0-2-3-0 the transportation costs are 89. The total transportation costs are, therefore, 187.

2.3.4. The parallel savings algorithm

In the parallel version 1 and 5 are also combined first. Since the parallel algorithm may build more than one route at a time, points 2 and 4 are also combined. Finally, points 3 and 5 are combined. In this way the algorithm constructs the routes 0-1-5-3-0 and 0-2-4-0 with total transportation costs amounting to 171.

It is worth noting that the number of routes may be reduced during the process of the parallel algorithm. For example, the two routes 0-1-2-0 and 0-3-4-0 will be combined into one route if the connection from 2 to 3 is established; in that case the resulting route becomes 0-1-2-3-4-0.

As it also turns out in the present example, the parallel savings algorithm frequently provides better results than the sequential algorithm. Dependent upon the way the algorithms are implemented, the parallel algorithm may also involve more computational work in

connection with the management of several routes at the same time. Therefore, it cannot be stated in general whether the sequential or the parallel algorithm is more appropriate.

2.4. Domain Reduction

Domain reduction is a method that deletes some unneeded values from the domain using a logical based approach [13]. In this paper the domain reduction will be applied by adding a new constraint that deletes some large numbers from the distance matrix and thus forbids the use of certain links. The new restriction is:

$$c_{ij} \leq R \qquad i,j=1,2,\ldots,n$$

where c_{ij} represent the cost between i and j, and R is a threshold that depends on the maximum number in the distance matrix. The domain reduction procedure starts by dividing the distance matrix into four parts based on the distance. In the other hand the demand matrix will be divided into four parts starting from the lowest to the greatest. Customers with lowest demands and high (or very high) costs will be highlighted as critical customers. In the other hand, customers with high (or very high) demands and with lowest (or average) costs will be referred to as leading customers. The domain reduction will delete all the high and very high costs from the cost matrix. For the critical customers we delete the highest 50% of the edges. Taking the obtained solution in consideration, the same procedure should be repeated by applying the domain reduction on the remaining values.

CHAPTER – 3

Related Work

Like conventional GA for the VRP, a chromosome I (n) simply is a sequence (permutation) S of n customer nodes. We check the capacity constraints and distance constraints at the same time from the first gene of chromosome, if do not violate the constraints, considering the next gene; if it violate the constraints in someone gene, we consider to use other vehicle from this gene starting, and repeat the above process, till all of customers were serviced. That is to say the total amount of demands in a route cannot exceed the capacity of the associated vehicle, and the total traveling distance cannot over the maximum distance,

3. IDE Approach for Vehicle Routing Problem

For instance, there are 10 customer nodes, a randomly generated chromosome is 1 3 6 8 9 5 4 10 2 7,1 3 6 8 9 5 4 10 2 7, can be interpreted as r=3 feasible routes: 0-1-3-6-0, 0-8-9-5-0, and 0-4-10-2-7-0. If k>=r, then this chromosome is legal; otherwise, it is illegal. In order to prevent illegal chromosome entering the next generation in great probability, a penalty function is designed. R is the total distance vehicles travelled of the corresponding chromosome, let m= r−k, if r > k >, then m>0 and R= R+M x m where M is a very large integer; if k < r. The fitness function can be expressed as f= 1/(R + M x m). For convenience, capacity of the vehicles is sameness, and the maximum distance that each vehicle can travel are equal, they can be denoted respectively as Q and L [1].

3.1 Improved differential evolution operators

Differential Evolution grew out of Ken Price's attempts to solve the Chebychev Polynomial fitting Problem that had been posed to him by Rainer Storm in 1996. The basic steps are as follows:

- **Initiation population:** We adopted an integer coding as section 3.1, the initial population is generated by random generator and the number of individual is NP, each individual is an N-scaling factor F is a constant from [2].
- **Mutation operation**: Because we adopted an integer code, and each chromosome represents a sequence of the customers, each gene stands for a customer, when the offspring gene oversteps the range, we must consider an auxiliary operator based on

integer order criterion. For the largest gene of offspring gives the largest customer ordinal number, the second evaluated ascba,N1−N, the rest may be deduced by analogy, we can prove that this operator equates to an affine transform, for example, the offspring chromosome is [-7,1,-5,2,-3, 0, 3, 5], the number of the customers is 8, we obtain the offspring chromosome that is [1, 5, 2, 6, 3, 4, 7, 8].

- **Crossover operation:** Following the mutation operation, crossover operation is applied to the population. Crossover operation is employed to generate a trial vector by replacing certain parameters of the target vector by the corresponding parameters of a randomly generated donor vector. That is

$$trial(j)^{G+1} = \begin{cases} v(j)^{G+1}, & rand(j) \le CR \quad or \quad j = randn(i) \\ Chrom(i,j)^{G}, & rand(j) > CR \quad and \quad j \ne randn(i) \end{cases},$$

Where G is the number of current iteration, CR[1,0]∈ is the crossover probability factor. In order to improve the population's diversity and the ability of breaking away from the local optimum, we present a new self-adapting differential evolution algorithm, the key factor is the crossover probability () is time varying, it changes from small to large with iteration number. i.e.

$$CR = CR_{\min} + G * \frac{CR_{\max} - CR_{\min}}{MAXGEN},$$

Where CR_{min} is the proposed minimum crossover probability, and CR_{max} is the maximum crossover probability, G is the number of current iteration; MAXGEN is the number of maximum iteration. In the early stage of evolution, the crossover probability is smaller, which can improve the global searching capability; in the later stage of evolution, the crossover probability is larger, which can improve the local searching capability.

- **Estimation and Selection operation:** The parent is replaced by its offspring if the fitness of the offspring is better than that of its parent. Contrarily, the parent is retained in the next generation if the fitness of the offspring is worse than that of its parent, according to the following equation:

$$Chron(i,:)^{G+1} = \begin{cases} trial^{G+1}, & f(Chron(i,:)^{G+1}) < f(trial^{G}) \\ Chron(i,:)^{G}, & otherwise \end{cases}$$

Usually, the performance of a DE algorithm depends on three variables: the population size NP, the scaling factor F and the crossover probability CR.

3.2 Soft Time Window

In the VRPSTW problem, the number of vehicle is fixed. Our objective is to reduce the total transportation cost (time) needed to serve all the customers exactly once by any of the vehicles. So, for each vehicle, Ti, there will be a route, R_i, that starts at the Depot 0 and after serving some customers ($1 \leq x_i \leq |C|$), again ends at Depot 0, where, |C| is the total number of customers. Now, the steps of our ABC algorithm for solving the VRPSTW problem (ABC_ VRPSTW) are described in the following subsections.

3.2.1 Initialization: This paper presents each solution as a collection of integer- valued vectors. Each vector corresponds to a vehicle and the elements in each vector are the customer served by that vehicle. Therefore, each vector represents a route consisting of sequentially served customer numbers, with Depot 0 fixed at the starting and ending position respectively. The initialization process starts by assigning a randomly generated solution to every employed bee .The steps of generating a single initial solution maintaining all the constraints is listed below:

- For a vehicle Ti, a random customer, i, is chosen and the demand of that customer is recorded.
- Like this, for each vector (vehicle/route), customers are randomly chosen (excluding the one that have been chosen already) and the demand of every one is added to the demand record, until the total demand, Td exceeds the vehicle capacity, T_q, of the vehicle T_i.
- Then, the total number of customers is reduced by total number of customers chosen already.

- The customers for a vector are chosen in a way, so that no customer number is repeated in a solution.
- This process is repeated for each vector in a solution to create a complete solution.

3.2.2 Determination of a food source in the neighbourhood: VRPSTW is a discrete optimization problem. And here, we represent each solution by a set of routes (vectors) for a fixed number of vehicles. Again, each of those routes consists of some integers corresponding to the consecutive customer numbers served by a vehicle. So, there is an ordering between the se numbers, following which, the customers are served. Initially, some random new solutions (maintain all the constraints) are given to some employed bee. Then it is desirable to utilize the information gathered by those employed bee about the food sources while generating a neigh boring solution. This may help in producing a good neighbourhood solution .In our algorithm; we have made the following two modifications on the random solution for employed bee to produce the neighbourhood solutions. Step

- Note that, each customer must be presenting a solution exactly once. We randomly select a customer, say, (m) from of a solution, say, (S m) from which a new neighbourhood solution, say, S_n will be generated. Then, we randomly choose another solution from which we select another random customer, say, (n), where m 6= n. Now, replace m by n. But, n must be already presenting Sum, which is allowed. So, just exchange the position of m and n within S_m which converts S_m into S_n.
- Now, the order of customers being served is important. Varying this order can also vary the travelling cost. We have utilized this observation for generating a new neighbourhood solution S_n from the solution Sm. We select a block of customer numbers from any route of S_m and replace that block by a random permutation of the selected numbers which generates the S_n. Altogether these two steps generate new neighborhood solutions in our problem.

3.2.3 Probability of Selecting a Food Source: In ABC VRPSTW, we are minimizing the travelling cost (time) of the vehicles. And, the probability pi of selecting a Food Source i is determined using the expression, pi = 1=W(i) X_m j=1 1=W(i) , where, W(i) is the cost of the Food Source i and m is the total number of food sources.

$$\frac{1/W(i)}{\sum_{j=1}^{m} 1/w(i)}$$

3.2.4 Cost Evaluation : In ABC VRPSTW, the travelling cost of each solution is the summation of the cost of all the routes with in it, including the penalties(if applicable)and the total number of window breaks is the summation of the number of customers which are not served within the actual(i.e., unrelated) time 635 window. Now, for each possible pair of customers, i and j, the cost to travel between the miss fixed (c_{ij}), which is the Euclidian distance between customer i and j. This process is done for all the routes with in a solution. In this way, cost of initial solution and also for all newly generated neighbourhood solution is evaluated. While evaluating the cost and deciding for selecting/discarding a solution, some important constraints must be considered:

(1) Vehicle Capacity Constraint: Suppose k 2 R_i denote a customer in the route R_i. For any route, R_i the summation of demands of the customers on that route($X \in R_i\ d_k$) must be less than or equal to the capacity of the vehicle (T_q). Otherwise, that solution is then abandoned.

(2) Time Window Constraint: If a vehicle comes at any customer, i, within a_r i to a_i or bi to b_r i, the customer can be served by the vehicle in exchange of penalty (due to the relaxation of time window).But if a vehicle comes before a_r i or after b_r i , the vehicle is not allowed to serve the customer and that solution is abandoned.

3.3 Hybrid Chaotic Particle Swarm Optimization

To deal with the above-mentioned disadvantages of Basic PSO, a new designed HCPSO is put forward in this section. In basic PSO, the particle is represented by decimal numbers as the algorithm is mainly used to solve continuous problems. Firstly, in order to solve discrete problems like CVRP, we introduce two forms of representation of each particle. Secondly, chaotic dynamics is used in PSO initialization to ensure particles distribute evenly across the solution space. Thirdly, searching strategy is applied to enhance the optimization ability. Lastly, several enhanced local searching strategies are also applied to form the proposed HCPSO algorithm eventually.

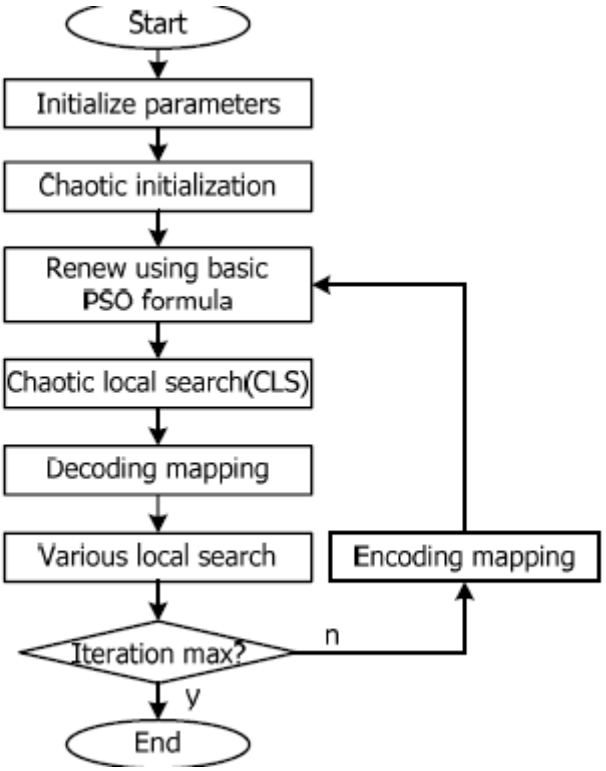

Figure 3.1 : A simplified flow chart for a single run

3.3.1 Particle representation and mutual mapping: in basic PSO, the particle is represented by decimal numbers as the algorithm is mainly used to solve continuous problems. In order to solve discrete problems like CVRP, we introduce two forms of representation of each particle. The first form is the same as the basic PSO, where particles are represented in decimal numbers. In a matrix, we represent a particle by position vector, fitness value, and velocity vector. For instance, in [1.2 2.2 0.5 2.3 162 0.2 0.4 0.1 0.5], the first four values represent the position vector, the value 162 is the fitness value at that particular position, and the last four values represent the velocity vector. Note that the dimensions of both position and velocity vectors are the same, and equal to the number of customers. While the above only represents one particle, in a matrix, there are multiple rows corresponding to different particles. Another form is represented in integers. However, only the position vector and fitness value is recorded. It is also worth mentioning that 0s are inserted in between different integers. For instance, in [2 3 0 1 0 4 162], the first 6 numbers represent the aggregate route (subroute1: 0230, subroute2: 010, subroute3: 040) while the last number 162 is the cost of the aggregate route. In order to utilize the simplicity and

effectiveness of basic PSO, the original updating formula (6) and (7) are still kept. But according to the discussion above, each particle has two forms of representation, it is necessary to update both the decimal form used in basic PSO and integer form after each iteration. However, for some local search strategies manipulating the second set (the integers corresponding to route), it is crucial to map the route back to the continuous numbers. And in this paper, the newly proposed mapping method is capable of mutual mapping. It first generates random numbers, and then ranks all the numbers. And the corresponding ranking position is the customer visited. And 0s have to be inserted between customers to separate routes for different vehicles. However, the disadvantage of this mapping method is that there are more infeasible solutions created because of the violation of capacity constraint. This problem is especially obvious when two 0s are close to each other in the same route, which means some other routes are extremely longer leading to higher probability of infeasibility. For example, it is very likely to come out with a feasible solution 4→0→1→3→6→ 5→1→2 based on this algorithm. It means the second vehicle is not used, which is very likely to cause the third vehicle to overload causing infeasible solution. In order to tackle that, we illustrate the new mapping method using a case of 6 customers and 3 vehicles. Firstly, we generate 6 random numbers from 0 to 3 (the number of vehicles). In a general case, the random number is from 0 to the number of vehicles. Then, we assign the numbers(x) between 0 to 1 to the first route, those between 1 and 2 to the second route, and those between 2 and 3 to the third route. In the same route, ranking is used to determine the sequence visited. And 0s are used to separate different routes. So the solution is 3→4→0→1→5→0→6→2. The first route is 0→3→4→0, second route is 0→1→5→0, and third route is 0→ 6→2→0. Because x is randomly generated from 0 to the number of vehicles, and all the numbers taken for the same sub-route are from interval of size 1, the probability of getting feasible solutions increases. By making the 0s more evenly inserted to solutions, more feasible solutions can be generated.

3.3.2 The chaotic initialization: One of the main factors determining the quality of solution is the initialization process. Ideally after initialization, there should be enough feasible solutions for large scaled problems. In literature [15], two ideal characters of chaotic dynamics, which are sensitivity to initial conditions and periodicity to the entire feasible space, have been shown. Thereby chaotic initialization is utilized to improve the initialization quality. We introduce a piecewise linear function in this paper

$$cx = \begin{cases} cx/0.4, & cx \le 0.4 \\ (1-cx)/0.6, & \text{other} \end{cases}$$

Let initial cx = 0.01 and the iteration number equal to 300. The summary of the procedure of the newly proposed chaos initialization is as follows.

i). Randomize the first chaotic particle using the function random.

ii). Generate the next particle using formula (8), till all the chaotic particles are generated. The number of chaotic particles is 20 more than the number of particles used for searching.

iii). Map all the generated chaotic particles from interval (0, 1) to normal particles from interval (0, vehicle) by multiplying number of vehicles.

iv). Transform the continuous numbers to integer solutions and evaluate all the particles.

v). Sort all the chaotic particles by fitness value. Choose the best (popsize+20) particles used for searching out of (popsize+20) chaotic particles.

3.3.3 The searching strategy: It has been known from section 1 that certain constraints like capacity constraint must be met in CVRP. Those constraints cause a lot of infeasible solutions, which deteriorates the searching ability of optimal solutions. For the feasible solutions, the fitness value is just the total distance travelled by all the vehicles. In terms of dealing with infeasible solutions, a large penalty cost is assigned depending on how many units are exceeded from the capacity. To put it simple, it equals to (a large coefficient) ∗units exceeded capacity + distance of the infeasible route. On this basis, to enhance optimization ability of PSO, a Chaotic Particle Swarm Optimization (CPSO) is put forward in this section by incorporating a Chaotic Local Search (CLS) process.

i). Chaotic Local Search (CLS): Utilizing the characters of Chaos theory, some researchers have incorporated them into Particle Swarm Optimization. For instance, one classical paper was published in 2005, which combined Chaos theory with PSO to solve continuous problems [14]. In this paper, a Chaotic Local Search (CLS) process is studied and is incorporated into basic PSO algorithm. In our proposed CLS, a classical logistics map is used. It was previously found that for the logistic equation.

$$x_{n+1} = \mu * x_n * (1 - x_n), 0 \le x_0 \le 1$$

ii). Chaotic Particle Swarm Optimization (CPSO): After renewing the velocity and position vector using the traditional formulas (6)-(7), the CLS process discussed above will be executed in each iteration of PSO.

- Sort all the particles according to the fitness value, and prepare to pass gbest for the Chaotic Local Search (CLS).
- Start the CLS for best, and return the solution found back to CPSO and use it to substitute a random particle.
- Reserve the top 1/5 particles, and decrease the search space to generate the rest 4/5 particles.
- Generate the new 4/5 particles close to the best particle found after CLS by limiting the boundary to best particle $\pm$ r $*$ (max-min), where r is a constant, say 0.1. At the same time, we must ensure it does not exceed the original min and max boundary.
- Evaluate the newly generated particles, and apply local search strategy to improve them. Reconstruct the whole population using the reserved particles where the best particle has gone through CLS and the newly generated particles which have gone through local search.

iii). enhanced local search strategies: Although CLS has been used, the local search ability is still not strong enough. So we introduce more strategies directly manipulating the routes.

- **The neighbour change strategy:** The first strategy used is the neighbour change strategy. The idea of this adjacent neighbour change is to increase the probability of finding better solutions after each renewing. It is implemented each time after mapping the continuous numbers to integers for each particle. It starts from left to right for each route. For example, at first, it switches the first with the second element and calculates the objective value. If it is better, simply accept the change; if not, still keep the original solution. After that, it switches the second element with the third one and continues for the rest. For instance, an original sequence after the mapping is 7 3 2 1 5 0 4 6 8. After the first switching, it becomes, 3 7 2 1 5 0 4 6 8. And we check whether it is better. Assume it is better, then we accept and continue to consider 7 and 2, and if better, the sequence becomes 3 2 7 1 5 0 4 6 8.

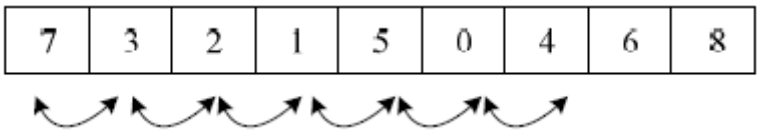

Figure 3.2: Demonstration of neighbour change strategy

- **The exchange strategy between different sub-routes:** The neighbour change strategy is most effective in only dealing with situations where the result can be improved from switching customers near to each other in a route. In particular, the disadvantage is that it cannot switch different customers served by different vehicles effectively. So we introduce an exchange customer strategy to switch customers from different sub-routes. For example, one final route is 7 3 2 1 5 0 4 6 8. By using this strategy, firstly, 7 and 4 are switched, and the route becomes 4 3 2 1 5 0 7 6 8. If the fitness value is better, we accept the change and also update the corresponding continuous solutions. And if not better, we do not change the route. After that, 4 is switched with 6, and it continues for the rest of the route.

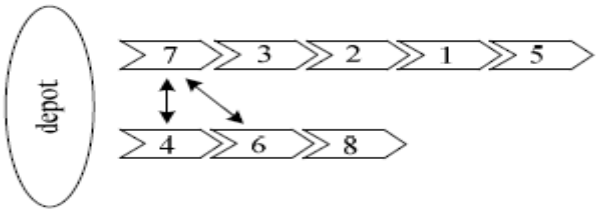

Figure 3.3: Demonstration of exchange strategy

- **The move strategy:** Sometimes, moving one element behind another is more desirable than switching two elements, so moving only one element should be considered. The strategy is implemented for every sub-route to move one single element to reshuffle the route. Similarly, we accept the change if it is better and reject it if not. For example, the route is 7 3 2 1 5 0 4 6 8. We first consider the route for the first vehicle and move 7 after 3 and the route becomes 3 7 2 1 5 0 4 6 8. Actually it is the same as the neighbor change at this step. Assume we accept, then, it moves 3 after 2, so the route becomes 7 2 3 1 5 0 4 6 8. We can see from the following graph that only one element is shifted causing less disruption of the original route compared to exchange strategy.

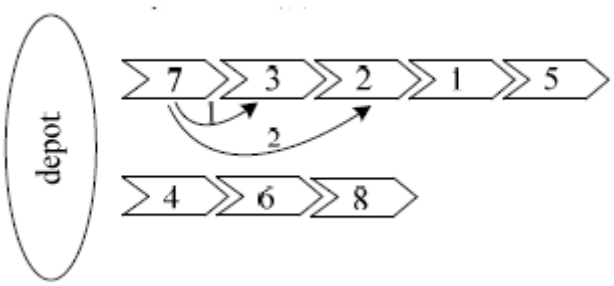

Figure3.4: Demonstration of move strategy

Chapter 4

Proposed Model

Now we look at the problem in hand and try to propose a solution. The solution is formally defined as follows -

The Capacitated Vehicle Routing Problem (CVRP) is to satisfy the demand of a set of customers using a number of vehicles with minimum cost.

Let:

C= {1, 2… n}: the set of customer location.

0: depot location.

$G= (V, E)$: the graph representing the vehicle routing network with $V= \{0, 1… n\}$

j_q : demand of customer j.

Q: common vehicle capacity.

m: number of delivery vehicles.

c_{ij} : distance or associated cost between locations i and j.

L: maximum distance a vehicle can travel.

The vehicle routing problem is characterized as follows. From a depot goods must be delivered in given quantities to given customers. For the transportation of the goods a number of vehicles are available, each with a certain capacity with regard to the quantities. Every vehicle that is applied in the solution must cover a route, starting and ending at the depot, on which goods are delivered to one or more customers. The problem is to determine the allocation of the customers among routes, the sequence in which the customers shall be visited on a route, and which vehicle that shall cover a route. The objective is to find a solution which minimizes the total transportation costs. Furthermore, the solution must satisfy the restrictions that every customer is visited exactly once, where the demanded quantities are delivered, and the total demand on every route must be within the vehicle's capacity. The transportation costs are specified as the cost of driving from any point to any other point.

The constraints are specified as follows –

1. Every delivery must be finished.

2. The vehicles should return to the depot after finishing the way.

3. The vehicle capacity must not be exceeded.

4. Each customer must be visited only once.

5. The vehicles are of uniform capacity delivery.

The solution proposed is as follows –

The Clarke and Wright algorithm used by the older solution was not very productive in most of the cases. The model tries to compute the savings as in Clarke and Wrigth algorithm. But before that it uses domain based reduction to not include the very costly paths. This gives better results as the most of the less gaining paths are discarded.

The model can be describes as follows –

The input are used to in the following form

1. List of distance between the customers.

2. Demand of the customers.

3. Distance of customers from the depot.

4. Number of customers.

The system takes most of these inputs as random however these are defined in the presentation. After taking the input and calculating it the system calculates the paths to be used in the by considering the paths which are less costly than the threshold. This is calculated as follows

Cost[i][j]>threshold - degrade the path by setting the value to a negative number

Cost[i] [j]>threshold - use the path by using general weight calculations.

The distance matrix will now contain special values for costly paths. Now the distance values will be used to calculate the savings. The savings in a path are the less cost that was spend by considering a pair of customers in the route. The saving is calculated as follows

Let the distance of depot d from customers i and j be c_{id} and c_{jd}. The cost between the customers i and j be c_{ij}. Now if we consider each route only serving 1 customer the cost to serve both the will be

Total cost = $2(c_{id} + c_{jd})$

But by using the customer in the same path we would cost

Total cost = $c_{id}+c_{ij}+c_{jd}$

The saving are

$2(c_{id}+c_{jd})-c_{id}+c_{jd}+c_{ij}$

$c_{jd}+c_{id}-c_{ij}$

The saving are now sorted in decreasing order. Now the list containing the following is created

1. Starting customer s1.
2. Ending customer s2.
3. Saving on the path

Now we iterate through the list until all the customers are satisfied. In every iteration a new route is build and the customer satisfying the constraints are only added to that route. The constraints are following –

1. The previous route must not be broken.
2. The customer must not demand the quantity that will exceed the capacity.
3. Each customer must be served only once.

The routes are hence found and the routes assigned to each vehicle. The total cost of the routes is the add of cost of all routes.

The system can be explained by the following figure

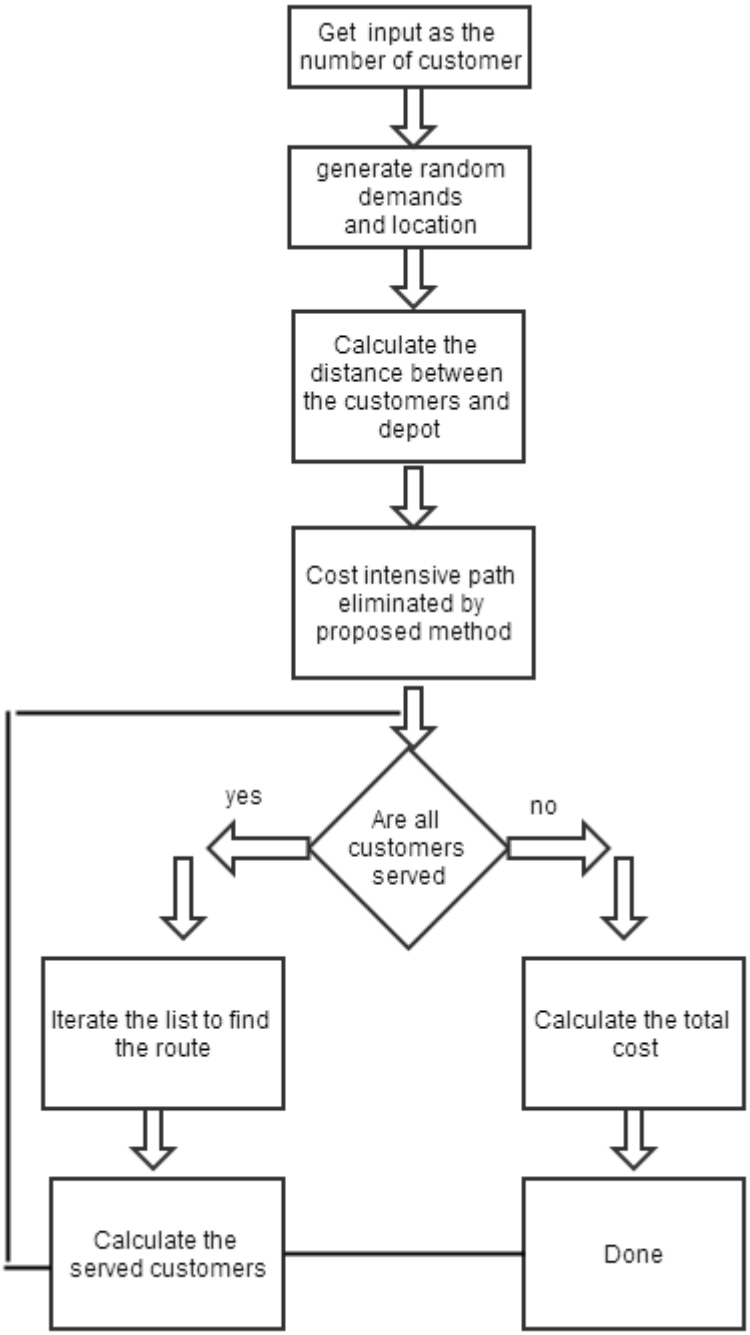

Figure 4.1: Flow chart of algorithm

The algorithm is as shown below

Input – n number of customers

c capacity of vehicles

threshhold to eliminate the costly path

1. Generate the location and demand of customers by random system.

2. Calculate cost of every customers pair and set the values above threshold to null values and eliminate them.

3. Calculate the savings for each customer pair and sort in decreasing order.

4. while(customers_served < number)

a. find starting customers boundry1 and boundry2.

b. set weight=demand[boundry1]+demand[boundry2]

c. partial_solution=null

d. for each item in list i i++

A.if list[i][0]=boundry2 and not visited and capacity not voilated

i. update list[i][1] as boundry2

ii. set customer as served

iii. add customer to partial_solution

iv. increament the weight

B. if list[i][1]=boundry1 and not visited and capacity not voilated

i. update list[i][1] as boundry1

ii. set customer as served

iii. add customer to partial_solution

iv. increament the weight

C. Update the obtained solution.

5. Print statistics and show results.

6. Compare statistics

Chapter 5

Simulation and results

The system was implemented and tested against inputs of different values for measuring performance. The input was run on the system. The simulation was performed under random inputs. The inputs to the system were the number of processes and the capacity of the vehicle, and then the random inputs were generated for the location of the customers in the coordinate plane. The location of the depot was at origin. Also the demand of each customer was randomly generated. The outputs were the routes to be followed and cost of routes.

The following assumptions were made in the simulation model –

1. There are unlimited vehicles available.
2. The joint load of any two customers will be less than the capacity of the vehicle.
3. The distance capacity of vehicles will let it go anywhere.
4. The distance is defined in terms of coordinate system.

The steps in the algorithms were as follows

1. Get input – The system will get the input as number of customers and the capacity of the vehicles. The inputs were then used to generate random values for location and demand.

2. Generate the random values – The inputs were used to generate random values for demand of the customers and the location of the customers.

3. The distance matrix was calculated and the distances were stored in the matrix.

4. The values above the threshold were negated.

5. The savings were stored in the system.

6. The routes were served by taking the routes in the decreasing order of savings.

7. The routes were created.

8. The values for cost were calculated and showed.

9. The algorithm ended here

The system was compared against the standard Clarke Wright algorithm and the results were compared. The following test cases were considered. The threshold value was also seen here.

Case 1:

Number of customers 100		Capacity of vehicles 100
Threshold	Avg. Cost of 5 run	
	According to C-W algo	Acc to proposed model
50	(1201+1796+1678+1321+1587)/5 = 1516.6	(1254+1555+1381+1310+1488)/5 = 1397.6
60	(1772+1609+1425+1411+1548)/5 =1553	(1337+1534+1351+1222+1245)/5 =1337.8
70	(1499+1278+1836+1377+1570)/5 =1512	(1161+1306+1468+1033+1308)/5 =1255.2
80	(1542+1460+1728+1287+1549)/5 =1513.2	(1354+1051+1326+1290+1360)/5 =1276.2

Table No. 5.1 Effective threshold for case.1

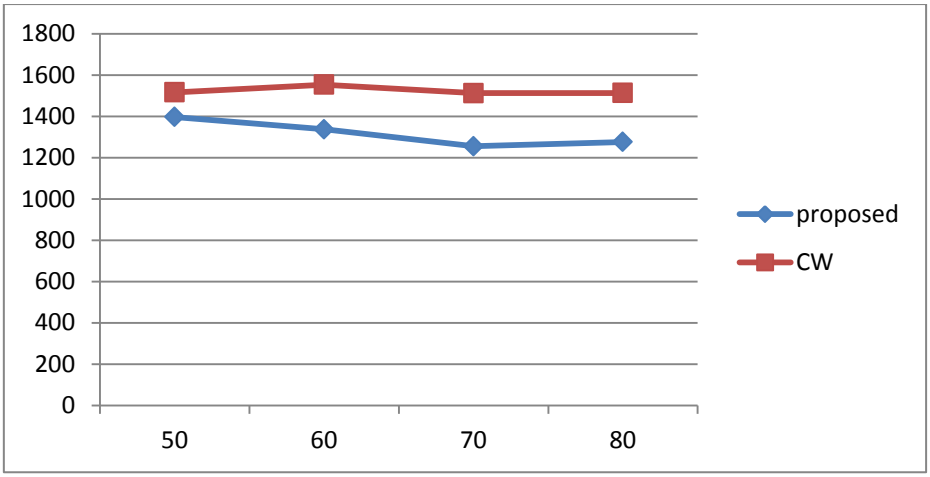

Fig.5.1 Threshold v/s Cost graph for case 1.

Case 2:

Number of customers 150		Capacity of vehicles 150
Threshold	Avg. Cost of 5 run	
	According to C-W algo	Acc to proposed model
50	(2958+2752+2660+2803+2500)/5 =2734.6	(2273+2434+2763+3087+2360)/5 =2583.4
60	(2271+2769+2780+3329+2660)/5 =2861.6	(2502+2562+2304+2312+2200)/5 =2376
70	(2446+2862+2324+2660+2779)/5 =2614.2	(2341+2520+1932+2200+2459)/5 =2286.8
80	(2868+2408+2790+2670+2918)/5 =2720.8	(2282+2970+2402+2369+2330)/5 =2270.6

Table No. 5.2 Effective threshold for case.2

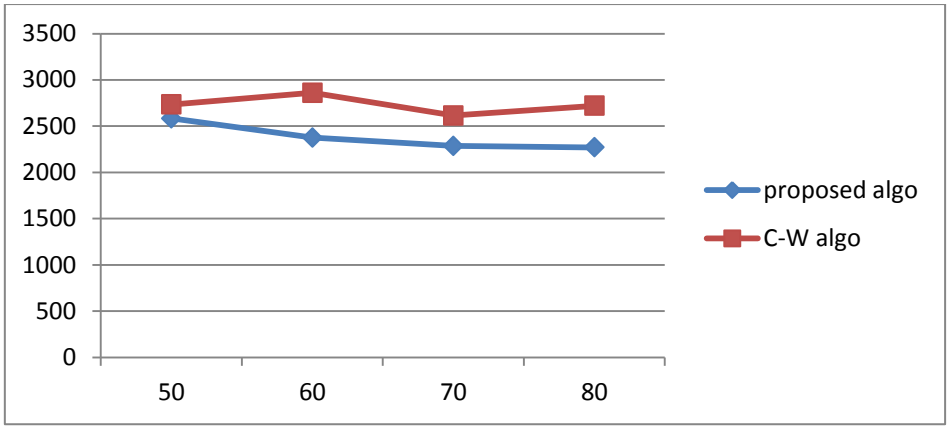

Fig.5..2 Threshold v/s Cost graph for case 2.

Case 3:

Number of customers 200		Capacity of vehicles 200
Threshold	Avg. Cost of 5 run	
	According to C-W algo	Acc to proposed model
50	(3417+3473+3070+3228+2918)/5 =3221.2	(3019+3747+2984+3166+2750)/5 =3133.2
60	(2867+3444+3224+3028+2910)/5 =3094.6	(2987+2833+3099+2918+2768)/5 =2921
70	(3030+3424+4041+3261+3168)/5 =3384.8	(2908+2641+3451+2968+2819)/5 =2957.4
80	(3272+3457+3192+3568+2881)/5 =3274	(2514+2807+2548+2968+2611)/5 =2689.6

Table No. 5.3 Effective threshold for case.3

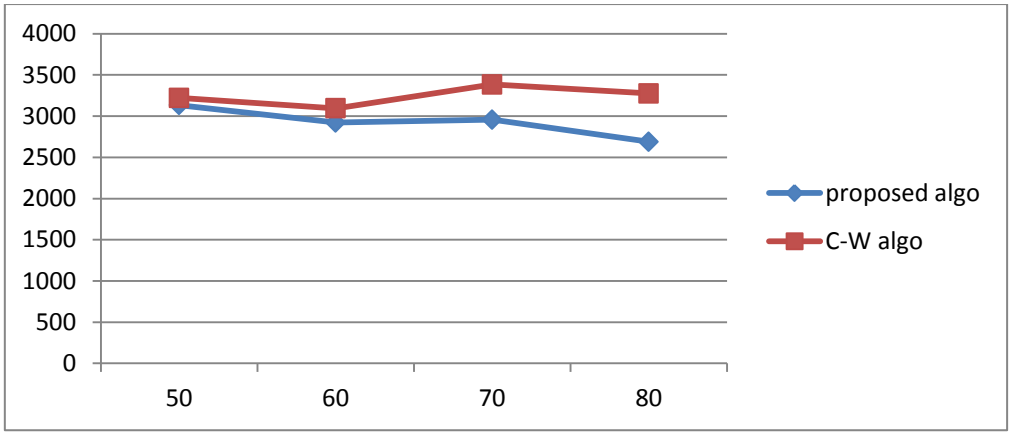

Fig.5.3 Threshold v/s Cost graph for case 3.

Case 4:

Number of customers 250		Capacity of vehicles 250
Threshold	Avg. Cost of 5 run	
	According to C-W algo	Acc to proposed model
50	(3940+4540+3737+3774+3818)/5 =3961.8	(3925+4985+4645+4550+3618)/5 =4344.6
60	(3900+4891+3643+3774+4280)/5 =4097.6	(3829+4316+3323+3670+3829)/5 =3739.4
70	(3624+4191+4825+4425+4168)/5 =4245.8	(3714+4364+4322+4291+3921)/5 =4122.4
80	(4000+4076+3878+3825+3940)/5 =3945.8	(3311+3391+3209+3633+3768)/5 =3462.4

Table No. 5.4 Effective threshold for case.4

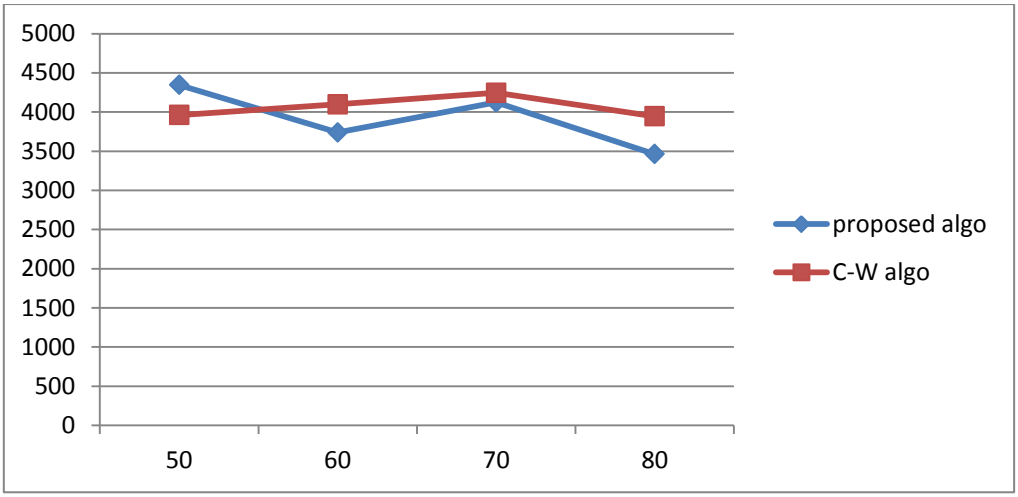

Fig.5.4 Threshold v/s Cost graph for case 4.

Now the algorithm will now be compared will be compared against the Clarke and Wright algorithm and the results were compared against for the cost of work.

Now after finding the efficient values of threshold we will now compare the algorithm. The different values will be considered here.

Parameters 1:

Number of deliveries 100
Capacities of vehicles 100
At effective threshold: 70

	Cost
Proposed Model	1308
Clarke and Wright algorithm	1570

Table 5.5: Cost results for parameter 1

Parameters 2:

Number of deliveries 150
Capacities of vehicles 150
At effective threshold: 80

	Cost
Proposed Model	2282
Clarke and Wright algorithm	2818

Table 5.6: Cost results for parameter 2

Parameters 3:

Number of deliveries 200
Capacities of vehicles 200
At effective threshold: 80

	Cost
Proposed Model	2807
Clarke and Wright algorithm	3457

Table 5.7: Cost results for parameter 3

Parameters 4:

Number of deliveries 250
Capacities of vehicles 250
At effective threshold: 80

	Cost
Proposed Model	3209
Clarke and Wright algorithm	3878

Table 5.8: Cost results for parameter 4

The results show that the results are significantly better than the standard algorithm. The results are better with larger inputs and large bunber of small demands. This show the better quality of the results. The result is different with different load values. The number of inputs significantly shows enhances.

Conclusion and Future Work

In this paper we studied the Vehicle routing problem with capacity constraints. The problem is hard problem. Therefore, there is no efficient solution to the problem. This makes a efficient algorithm for input very difficult. This is an extension of Travelling Salesman problem with multiple salesman and added capacity constraint. The problem is therefore also difficult to solve.

The classical technique to solve the problem was proposed by Clarke and Wright using a heuristic method involving savings. This used the reductions in the path to determine the paths to be taken first. This method is probabilistic. Therefore, it uses techniques to solve the problem efficiently. The technique of saving is able to find an average solution. This is due to the fact that it takes larger paths even though lesser distance paths are available due to more savings on the larger path. In this total context decreases the savings. Therefore there is a need to improve this area.

The model is proposed to improve these problems by using domain based reduction. This technique uses a threshold's value to eliminate the large paths and decrease the unnecessary paths. This way is used to eliminate the large paths and find alternative smaller paths. This technique is use as the pre step to the Standard Algorithm. The model was implemented and the results were generated. The results were seen to be less costly.

The results were particularly good for large inputs. The results were better with taking more vehicles. This due the fact that the vehicles only cover the nearby users. The results were also good with large number of small deliveries taken together.

The model however uses the number of vehicles which are more and hence there are those additional cost involved. Also the results depend on the value of the threshold. The results are also very clustering sensitive.

The future work will be focussed on making the algorithm better by taking the threshold values by a standard method. This would make the process more efficient and cost decreasing. Also the algorithm would be made more quickly and cost cheaper.