Vehicle Routing Problem with Capacity Constraints

A Dissertation submitted in partial fulfillment of the requirement for the

Award of degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

By

PIYUSH KHANDELWAL

Roll No. – 2K11/CSE/08

Under the esteemed guidance of

Mr. MANOJ KUMAR

(ASSOCIATE PROFESSOR)

Department of Computer Engineering

Delhi Technological University

2012-2013

CERTIFICATE

This is to certify that the thesis entitled **"Vehicle Routing Problem with Capacity Constraints"** submitted by **Piyush Khandelwal (2K11/CSE/08)** to the Delhi Technological University, Delhi for the award of the degree of **Master of Technology** is a bona-fide record of research work carried out by him under my supervision.

The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Mr. Manoj Kumar
(Associate Professor)
Department of Computer Engineering
Delhi Technological University, Delhi

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and **Delhi Technological University**. I would like to extend my sincere thanks to all of them.

I would like to express my special gratitude and thanks to **Mrs. Daya Gupta *(Head of Dept.)*** for giving me such an opportunity to work on the project.

I am highly indebted to **Mr. Manoj Kumar *(Project guide)*** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

I would like to express my gratitude towards my **parents** & **staff** of Delhi Technological University for their kind co-operation and encouragement which helped me in completion of this project.

My thanks and appreciations also go to my **friends and colleagues** in developing the project and people who have willingly helped me out with their abilities.

Piyush Khandelwal
Roll No.: 2K11/CSE/08
Dept. of Computer Engineering
Delhi Technological University

ABSTRACT

Today, with the exponential growth of online shopping sites on World Wide Web, there is an increasing demand by the companies, of software solutions that take customers' requests and location as input and automatically design the everyday routes for the delivery men, with the uniform distribution of the load among them. EazyRoute is the desired solution. It provides the user with the ability to generate approximate optimal delivery routes from a central depot to geographically scattered customers depending on given constraints. It creates path in two phases: First it generates an initial solution using heuristic algorithms on the basis of given constraints. Then, the resulting solution is optimized using meta-heuristic algorithms. It provides the detailed description of different routes as well as graphical description of the routes. The main focus of EazyRoute is to generate optimal solutions that minimize the total distance travelled by delivery men while distributing the load uniformly among them. User has the options to customize the routes on the basis of the Number of vehicles, Capacity of vehicle, Maximum route length/vehicle. EazyRoute implements various proved heuristic and meta-heuristic algorithms and provides user with the option to choose different combinations of them to acquire best results. It can also be incorporated with our proposed algorithm New_RTR to generate more optimize solutions.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

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

1.1 Motivation

With the transition of shops from shopping malls and super markets to internet, the trend of home deliveries is passing through a new era. Online shopping sites are providing more and more facilities to attract the customers. They are taking help of discount coupons, promo codes, referrals; Cash on delivery, try and buy, the list goes on. 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 scale it becomes quite complicated for the companies to manage the whole process, from receiving the request to the delivery and 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 planned and organized. One of the sub processes of this whole process is the allocation of routes to the delivery men. After receiving the requests 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.

This is an old combinatorial optimization problem which is very difficult to solve. This problem was first introduced by Dantzig and Ramser [1] in 1959. It has different variants as discussed in section 2.2. Our main concern is the Vehicle routing problem with capacity constraints (VRPCC), described in section 2.3.

EazyRoute 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 algorithm 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 written in C/C++ and requires the understating of basic concepts of C/C++. Two external libraries have been included: VRPHLIB [7] and PLPLOT [8]. More information about downloading and installing them can be gained from their sites.

1.3. Organization of Thesis

This thesis has been comprehensively organized into 5 chapters.

Chapter 1 deals with the brief introduction about the research work, what motivated to work on this topic and also determine various prerequisites with the necessary sources. This chapter is then concluded with the description of the organization of thesis.

Chapter 2 helps the readers to have deep understanding of Vehicle Routing Problem. It describes different variants of VRP. Also, formulate the problem statement of CVRP. Then it presents the various solution approaches adopted by different researcher across the world.

Chapter 3 describes actually how EazyRoute is designed. First it contains the block diagram of EazyRoute with its components. Then each component is described in detail in further subsections. It also explains the use of important data structures used to implement EazyRoute. This chapter is then concluded with our proposed algorithm.

Chapter 4 displays the results of EazyRoute when executed on some standard benchmarks. It also shows the comparison of our proposed algorithm and traditional algorithm.

Chapter 5 concludes our research work with the interpretations from the results and observations.

2.1. INTRODUCTION

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]).

2.2 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.

2.2.1.Capacitated VRP (CVRP)

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: $\sum_{i=1}^{m} di \leq Q$.

2.2.2. 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,v_2,…v_n}U v_0 where V_0={v_{01},v_{02}….. v_{0d}} are the vertex representing the depots. Now, a route ***i*** is defined by $R_i=\{d,v_1,\ldots, v_m,d\}$, with $d \in V_0$. The cost of a route is calculated like in the case of the standard VRP.

2.2.3. 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} thenthe 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).

Table 2.1: Problem Statement for PVRP

Customer	Diary Demand	No. ofVisits	No. of Combinations	PossibleCombinations
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

- 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.

2.2.4. 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].

2.2.5 Stochastic VRP (SVRP)

Stochastic VRP (SVRP) isVRP 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 δ_i 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.

2.2.6. 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 $w_{vi}=\max\{0,b_{vi}-b_{vi-1}-\delta_{vi-1}-c_{i-1,i}\ \}$ 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. PROBLEM STATEMENT

The VRPWCC is described as follows: The problem with one central depot and some customers is considered. Some vehicles are available to carry the delivery for all the customers. The main objective function is to minimize the transport cost. The objective is to determine a viable route schedule which minimizes the distance or the total cost with the following constraints:

(1) Each customer is served exactly once by one vehicle;

(2) Each vehicle starts and ends its route at the depot;

(3) The total demand of the customers served by one vehicle does not exceed the capacity of the vehicle.

Assume define variable as follows:

- **i**: the number of customer (i= 1, 2, ... ,L), and L is the total number of customers; i=0 represents the depot;
- **K**: the number of vehicle (k=1, 2, ... , K), and K is the total number of vehicles;
- $\mathbf{G_i}$: the demand of customer i;
- $\mathbf{C_{ij}}$: the distance of traveling from customer i to customer j;
- **Q** : the capacity of vehicle;
- $\mathbf{Y_{ik}}$,$\mathbf{X_{ijk}}$: binary variable.

Y_{ik}==1 (If the order from customer i is delivered by vehicle k.)

Y_{ik}==0 (Otherwise)

X_{ijk}==1 (If the vehicle k travels directly from customer i to customer j.)

X_{ijk}==0 (Otherwise)

Then the objective function is shown as below:

$$Minimize \sum_{k=1}^{K}\sum_{i=0}^{L}\sum_{j=0}^{L} Cij\ Xijk \qquad (1)$$

Subject to:

$$\sum_{i=1}^{L} GiYik \leq Q \qquad \forall K \qquad (2)$$

$$\sum_{k=1}^{K} Yik = 1 \qquad i = 1,2 \ldots \ldots, L \qquad (3)$$

$$\sum_{i=0}^{L} Xijk = Yik \qquad i = 0,1,2 \ldots \ldots ., L \qquad (4)$$

$$\sum_{j=0}^{L} Xijk = Yik \qquad j = 0,1,2 \ldots \ldots .., L \qquad (5)$$

where equation (1) means the objective function of the problem and aims to minimize the total distance; constraint set (2) regulates that the load of each vehicle should not exceed its capacity; constraint set (3) represents each customer demand must be satisfied and was only fulfilled by one vehicle; constraint sets (4) and (5) ensure that there has only one vehicle which arrival and departure from a customer.

In the classical capacitated Vehicle Routing Problem (VRP), a minimum cost set of routes is constructed for a fleet of identical vehicles. These routes must satisfy the demands of all customers, and the vehicles traversing these routes have a fixed capacity. The VRP is a well-known NP-hard problem and computational experience indicates that the VRP is difficult to solve to optimality. Most real-world vehicle routing problems are solved using heuristic methods.

2.4. SOLUTIONS TO CVRP

Gilbert Laporte in his paper [10] has classified the solutions into three categories: Exact algorithms, Heuristic algorithms and Meta-heuristic algorithms. These are discussed individually in further subsections.

2.4.1.Exact Algorithms

Several families of exact algorithms have been proposed for the VRP with a symmetric cost structure. These are based on integer linear programming (ILP), dynamic programming, and branch-and-bound. For reviews, see Laporte and Nobert[14] and Toth and Vigo [15]. Here we concentrate on three families of ILP based branch-and-cut algorithms which have proved to be the only workable methodology. Unfortunately they all require rather heavy mathematical programming machinery and their success in solving realistic size instances is limited.

2.4.1.1 Two-Index vehicle flow formulations

Two-index vehicle flow formulations for the VRP are rooted in the work of Laporte, Nobert and Desrochers[16] and are extensions of the classical TSP formulation of Dantzig, Fulkerson and Johnson [17]. Let x_{ij} be an integer variable representing the number of times edge [i, j] appears in the optimal solution. If i, j ∈ V \{0}, then x_{ij} is binary; if i = 0, then x_{ij} can be equal to 0, 1 or 2, the latter case corresponding to a return trip between the depot and customer j. The problem is then:

$$Minimize \sum_{[i,j]\in E} CijXij \qquad (1)$$

Subject to:

$$\sum_{j\in V/\{0\}} X0j = 2m \qquad (2)$$

$$\sum_{i<k} Xik + \sum_{j>k} Xkj = 2 \qquad \left(k \in \frac{V}{\{0\}}\right) \qquad (3)$$

$$\sum_{\substack{i\in S, j\in * S \\ or \\ i\in * S, j\in S}} Xij \geq 2b(S) \left(Sc \frac{V}{\{0\}}\right) \qquad (4)$$

$$X_{ij}=0 \text{ or } 1 \qquad (i,j\in V/\{0\}) \qquad (5)$$

$$X_{0j}=0,1,2(j\in V/\{0\}) \qquad (6)$$

In this formulation, the objective function minimizes the total routing cost. Constraint (2) define the degree of vertex 0. Note that the right-hand side can be a constant if m is known a priori, or a variable otherwise. In the latter case, a term fm can be added to the objective function, where f is the vehicle fixed cost. Constraints (3) ensure that two edges are incident to each customer vertex. In constraints (4), b(S) is a lower bound on the number of vehicles required to serve all customers of S. These constraints play a dual role: they prevent the formation of sub-tours by forcing any subset of customers to be connected to the depot, and they ensure that capacity constraints are not violated. In practice, it is common to define b(S) as $\sum_{i \in S} qi/Q$.

Because the number of connectivity constraints is exponential in n, it is common to solve VF by branch-and-cut. Initially the connectivity and integrality constraints are relaxed. Constraints (4) are dynamically generated during the solution process as they are found to be violated, whereas integrality is reached by branching on fractional variables. Several families of valid inequalities can also be used to strengthen the linear relaxation of the problem. These include generalized capacity constraints, frame capacity constraints, and any inequality valid for the TSP, such as comb inequalities, some inequalities combining bin packing and the TSP, as well as inequalities based on the stable set problem. These families of inequalities, and related separation procedures, are described in Naddef and Rinaldi[18].

The best algorithm based on VF is due to Naddef and Rinaldi[18]. Using a branch-and-cut algorithm, these authors have solved at the root node six instances with $22 \leq n \leq 45$, and nine other instances with $51 \leq n \leq 135$ by using some branching.

2.4.1.2. Two-index two-commodity flow formulations

Baldacci, Hadjiconstantinou and Mingozzi[19] have proposed a two-index two-commodity flow formulation for the VRP, based on an earlier similar formulation by Finke, Claus and Gunn [20] for the TSP. The formulation makes use of travel directions on edges and works on an extended graph G = (V,E) where the vertex set V = V ∪ {n + 1} includes a copy n+1 of the depot, and E = E ∪ {[i, n + 1] : i ∈ V }. With this graph representation, a vehicle route is a directed path from 0 to n + 1. Binary variables xij are equal to 1 if and only if edge [i, j] appears in the optimal solution. Binary variables yij and yji represent, respectively, the vehicle load on edge [i, j] and the empty space on the vehicle traveling on edge [i, j], i.e., yij + yji = Q provided xij = 1. The formulation is:

$$Minimize \sum_{[i,j]\in\overline{E}} CijXij \qquad (7)$$

Subject to:

$$\textstyle\sum_{j\in\overline{V}} Yji - Yij = 2qi \qquad (\mathrm{i} \in \mathrm{V}/\{0\}) \qquad (8)$$

$$\sum_{j\in V/\{0\}} Y0j = \sum_{i\in V/\{0\}} qi \qquad (9)$$

$$\sum_{j\in V/\{0\}} Yjo = mQ - \sum_{i\in V/\{0\}} qi \qquad (10)$$

$$\textstyle\sum_{j\in V/\{0\}} Y_{\mathrm{n+1,j}} = \mathrm{mQ} \qquad (11)$$

$$\mathrm{Y_{ij} + Y_{ji} = QX_{ij}} \qquad (12)$$

$$\sum_{i<k} Xik + \sum_{j>k} Xkj = 2 \qquad \left(k \in \frac{V}{\{0\}}\right) \qquad (13)$$

$$\mathrm{Y_{ij} \geq 0,\ Y_{ji} \geq 0\ ([i, j] \in E)} \qquad (14)$$

$$\mathrm{X_{ij} = 0 \text{ or } 1\ ([i, j] \in E).} \qquad (15)$$

In this formulation, constraints (8)–(11) and (14) define consistent flows from vertex 0 to vertex n+1. Constraints (12) ensure that the yij and yji variables take feasible values, while constraints (13) specify the degree of each customer vertex. The authors have solved this formulation using a branch-and-cut algorithm in which valid VRP inequalities expressed in terms of the xij variables are gradually introduced. Several instances taken from the VRP literature, with $16 \leq n \leq 135$, could be solved to optimality. The algorithm can also solve, in a consistent fashion, randomly generated instances with $30 \leq n \leq 60$ and $m = 3$ or 5 and, less consistently, larger instances involving up to 100 customers and eight vehicles.

2.4.1.3. Set partitioning formulations

The VRP can also be formulated as a Set Partitioning Problem as follows (Balinski and Quandt[21]). Let R be the set of all feasible routes, let dr be the cost of route r ∈ R, and let z_r be a binary variable equal to 1 if and only if route r belongs to the optimal solution. In addition, let air be a binary coefficient equal to 1 if and only if customer i belongs to route r. The formulation is then:

$$Minimize \sum_{r\in R} drZr \qquad (16)$$

Subject to:

$$\sum_{r\in R} airZr = 1 \qquad \left(i \in \frac{V}{\{0\}}\right) \qquad (17)$$

$$\sum_{r\in R} Zr = m \qquad (18)$$

$$Z_r = 0 \text{ or } 1 \qquad (19)$$

As such, this formulation is impractical because of the large number of variables and of the difficulty of computing the d_r values (each requires solving a TSP over the vertices of r). Column generation, which is a natural methodology for this type of formulation, has proved mostly unsuccessful because the problem is not sufficiently constrained. However, the optimal solution value z(LSP) of the linear relaxation of SP provides a tight lower bound on the optimal VRP value. In addition, this formulation is rather flexible since it can easily accommodate a variety of side constraints such as time windows or maximal route lengths, hence reducing the cardinality of R.

As shown by Baldacci, Hadjiconstantinou and Mingozzi[19], the vehicle flow formulation can be rewritten in terms of the Z_r variables by using the following identity:

$$X_{ij}\sum_{r\in R}\mu_{ij}^{r}z_r \quad ([\,i,j] \in E)$$

where if r is the route (0, h, 0), then $\mu^r_{0h} = 2$ and $\mu^r_{ij} = 0$ for all $[i, j] \in E\backslash\{[0, h]\}$; if r contains at least two customers, then $\mu^r_{ij} = 1$ for each edge [i, j] of route r, and $\mu^r_{ij} = 0$ otherwise.

The set partitioning formulation can be strengthened through the introduction of valid inequalities. For $S \subset V\backslash\{0\}$, let R(S) be the set of routes containing at least one customer of S. Then the following capacity constraints are valid:

$$\sum_{r\in R(S)} Zr \geq b(S) \qquad\qquad (S \subset V/\{0\})$$

In addition, any valid inequality for the VRP (see Naddef and Rinaldi[18]; Letchford,Eglese and Lysgaard[22]) of the form:

$$\sum_{[i,j]\in E} \alpha ij\ Xij \geq\ \beta$$

can be re-expressed in terms of the zr variables using (20). Finally, inequalities valid forthe Set partitioning Problem (Balas and Padberg[23]; Hoffman and Padberg[24]) can be incorporated into SP. Thus, Baldacci, Christofides and Mingozzi[25] use the clique inequalities. Let H be a graph whose vertices correspond to vehicle routes. Two vertices are in conflict if the corresponding routes share at least one edge. For any clique C of H, the following inequality is valid:

$$\sum_{r\in C} Zr \leq 1$$

Baldacci, ChristofidesandMingozzi[26] work with the augmented SP formulation defined by (16)–(19) and some constraints (21), (22) and (23). Since solving this problem remains intractable in all but trivial cases, they use the dual of its linear relaxation to compute lower bounds of the optimal primal value by means of three different heuristics. These procedures are embedded within a branch-and-cut algorithm to yield optimal VRP solutions. Using this approach the authors have solved to optimality several VRP instances with 37 ≤ n ≤ 121. Their results are probably the best available.

In closing this section, we should mention the existence of another exact algorithm by Fukasawa et al. [26] combining a set partitioning formulation and cutting planes. This method seems to yield results almost as good as those of Baldacci, Christofides and Mingozzi.

2.4.2. Classical Heuristics

Classical heuristics for the VRP are naturally divided into constructive heuristics and improvement heuristics. The adjective "classical" refers to the fact that the improvement steps of these heuristics perform descents, i.e., they always proceed from a solution to a better one in its neighborhood until no further gain is possible. In contrast, meta-heuristics (outlined in Section 4) allow the consideration of non-improving and even infeasible intermediate solutions. It is common to test heuristics on the Christofides, Mingozzi and Toth[27] instances, referred to as the CMT test-bed and consisting of 14 instances with $51 \leq n \leq 199$, and on the Golden et al. [28] instances, referred to as the GWKC test-bed and consisting of 20 larger instances with $200 \leq n \leq 480$. Since optimal solution values are unknown for most of these instances, comparisons are made with the best known values produced by meta-heuristics.

2.4.2.1. Constructive heuristics

The most popular construction heuristic is the Clarke and Wright [29] savings algorithm. Initially n back and forth routes (0, i, 0) (i = 1,…, n) are constructed, all of which are feasible. A general iteration of the algorithm consists of merging a route ending at i with another route starting at j by removing arcs (i, 0) and (0, j), and adding arc (i, j), provided the saving $sij = ci0+c0j - cij$ is positive and the merged route is feasible. The best variant of the algorithm is the parallel version in which the merge yielding the largest saving is implemented at each iteration. The algorithm ends when no feasible and profitable merges are possible. Since the savings do not change during the algorithm, they can be computed and sorted within the initialization phase, the sorting step being the most time consuming component of the algorithm. As described in Laporte and Semet[30], several variants of the savings algorithm have been proposed, namely to speed up computations (Nelson et al. [31]), to optimize route merges based on savings values (Altinkemer and Gavish[32]; Wark and Holt [33]), and to make the savings definition more flexible (Golden, Magnanti and Nguyen [34]). While this algorithm is not the best available in terms of accuracy (on benchmark instances it produces an average deviation of about 7% on the CMT test-bed), it is rather fast and simple to implement, which explains its popularity.

Another important class of constructive heuristics is made up of petal heuristics which consist of first generating a family R′ of feasible routes and then solving the SP formulation over R′ rather than the full set R. The success of the algorithm depends on the quality of the generated routes. The most elementary version of this type of heuristic is the sweep algorithm of Gillett and Miller [35]. Starting with a half-line rooted at the depot, this heuristic gradually constructs feasible routes by rotating another half-line. Customers are gradually incorporated into the current route in increasing order of the angle they make with the initial half-line. The route closes when the inclusion of a further customer becomes infeasible (see Figure 2). This procedure only generates non-intersecting routes and is rather rudimentary. More sophisticated route generation procedures, proposed by Foster and Ryan [36], Ryan, Hjorring and Glover [37], and Renaud, Boctor and Laporte[38], have been described. In particular, the latter authors allow the creation of intersecting and embedded routes. Average deviations obtained with their heuristic on the CMT testbed are 2.38%.

Fisher and Jaikumar[39] have proposed yet a different type of heuristic based on a two-phase decomposition procedure. In the first phase, a seed is located in the region whereeach route is likely to lie and clusters of customers are created through the solution of a Generalized Assignment Problem (GAP), that is, the sum of distances between customers and the seed to which they are allocated is minimized, subject to the constraints that the total demand of each cluster should not exceed Q. A TSP is then solved on each cluster. There are a number of problems with this approach. One is the determination of seeds which is not explicit in the original article, but some procedures are described in Baker and Sheasby[40]. Another difficulty lies in the solution of the GAP which is itself an NP-hard problem.

Given the strength of current improvement heuristics, particularly meta-heuristics, there is little point in fine tuning construction heuristics. A simple scheme, like the savings method, is adequate for most purposes.

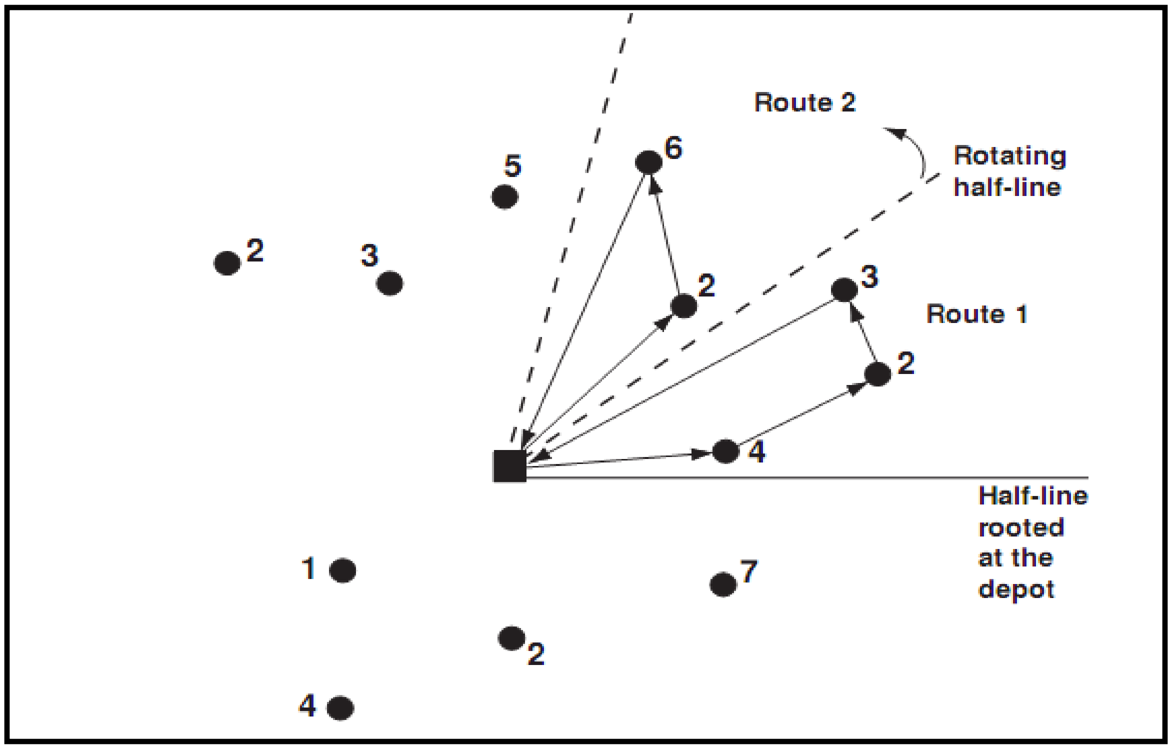


Figure 2.1: Construction of feasible routes in the sweep algorithm. The vehicle capacity is Q = 10. Customer demands are shown next to the vertices.

2.4.2.2. Improvement heuristics

Two types of improvement algorithms can be applied to VRP solutions. Intra-route heuristics post-optimize each route separately by means of a TSP improvement heuristic, e.g., 2-opt or 3-opt. Inter-route heuristics consist of moving vertices to different routes (Laporte and Semet[30]). The most common moves are simple transfers from one route to another, transfers involving several routes, and vertex exchanges between two or more routes. The general frameworks described by Thompson and Psaraftis[41], Van Breedam[42] and Kindervater and Savelsbergh[43] encompass most available inter-route improvement procedures. In particular, Thompson and Psaraftis describe a general b-cyclic, k-transfer scheme in which a circular permutation of b routes is considered, and k vertices from each route are shifted to the next route of the cyclic permutation. The combinations b = 2 or b variable, and k = 1 or 2 yield interesting results. Van Breedam classifies improvement operations as string cross, string exchange, string relocation and string mix, which can be viewed as special cases of 2-cyclic exchanges. The author shows that string exchange moves are the best. Kindervater and Savelsbergh have described similar operations and have performed experiments mostly in the context of the VRP with time windows.

By and large, the performance of classical improvement heuristics is good but not excellent. They are best used as building blocks within meta-heuristics.

2.4.3. Meta-heuristics

Significant progress has been witnessed over the past 15 years in the development of meta-heuristics for the VRP. All allow the exploration of the solution space beyond the first local minimum encountered, and all embed procedures borrowed from classical construction and improvement heuristics. A great variety of schemes have been put forward, but these can be broadly classified into three categories: 1) local search, 2) population search, and 3) learning mechanisms. For a survey and a bibliography on these topics, see Cordeau et al. [44] and Gendreau et al. [45].

2.4.3.1. Local search

A local search heuristic starts from an initial solution s0 (which may be infeasible) and moves at each iteration t from solution S_t of value $f(S_t)$ to another solution located in the neighborhood $N(S_t)$ of S_t. In most cases S_t is the current solution but some multi-start mechanisms allow a re-initiation of the search from a solution that differs from the current one. The neighborhood $N(S_t)$ consists of all solutions that can be reached from S_t by applying a given type of transformation, for example relocating a vertex from its current route into another route. The search ends with the best known solution $s*$ after a stopping criterion has been satisfied, usually a preset number of iterations, or a given number of consecutive iterations without improvement in $s*$.

Within this broad framework, several algorithms can be defined. Here are the local search schemes that have proved the most successful for the VRP. In record-to-record travel (Dueck[46]), a record is $s*$. A solution s is randomly selected in $N(S_t)$, and st+1 := s if $f(s) < \sigma f(s*)$, where σ is a user controlled parameter generally slightly larger than 1. In tabusearch (Glover [47]), st+1 is the best solution in $N(S_t)\backslash T(S_t)$, where $T(S_t)$ is the set of tabu (forbidden) solutions at iteration t. Tabu solutions are necessary to prevent the search from cycling. In the VRP, the tabu mechanism can be implemented as follows (Cordeau, Laporte, and Mercier [48]): A set of attributes $B(st) = \{(i, k) : i \in V \backslash \{0\}, k = 1, 2, \ldots, m\}$ is associated with solution S_t. A neighbor solution consists of removing a pair (i, k) from $B(S_t)$ and of inserting another pair (i, k′) in its place. The pair (i, k) is then declared tabu for θ iterations. In attribute based tabu search, a solution s containing a tabu attribute (i, k) can be accepted if it corresponds to the best known solution containing that attribute. This rule is usually called an aspiration criterion. Other mechanisms are often used in tabu search, such as continuous diversification, a rule aimed at penalizing solutions containing frequently encountered attributes, and intensification, a process aimed at performing a more thorough search around good solutions. It is also now common to allow intermediate infeasible solutions during the search through the use of a penalized objective function and self-adjusting penalties, a mechanism put forward by Gendreau, Hertz and Laporte[49]. In variable neighborhood search (Mladenovi´c and Hansen [50]), several nested neighborhoods are defined. The search is initiated with a given neighborhood. When a local minimum is encountered, it proceeds to the next neighborhood in the nested structure; the search restarts from the first neighborhood whenever a new best solution $s*$ is identified or all neighborhoods have been explored.

In very large scale neighborhood search (Ergun [51]), the size of $N(S_t)$ is very large and an optimization problem may have to be solved to determine the best neighbor of S_t. In adaptive large neighborhood search (Pisinger and Ropke[52]), several insertion and removal heuristics are applied. Their selection is made randomly, by giving a higher weight to heuristics that have performed well in the past.

4.2.3.2. Population search

Genetic algorithms (Holland [53]) evolve a population of solutions encoded as bit strings, or chromosomes, through a crossover and mutation process. The crossover takes two parents from the population and combines them to generate one or two offspring solutions. A mutation (typically random) is applied to each offspring, and the offspring replace the worst elements of the population. In the context of the VRP, encoding solutions as bit strings is not the most natural way to proceed and more natural mechanisms are used instead. For example, Prins[54] transforms the VRP solution into a TSP solution by removing the route delimiters and reconstructs the VRP solution at the end of the process. Also, random mutations are often replaced by the application of a standard improvement heuristic to the offspring, yielding what is commonly known as a memetic algorithm. A principle similar to genetic search, called an adaptive memory procedure, has been put forward by Rochat and Taillard[55] for the VRP. These authors first execute a tabu search algorithm and keep the best solutions in a memory. These solutions are then recombined through a crossover and local search process in the hope of generating even better solutions. For another implementation of this principle, see Tarantilis and Kiranoudis[56].

4.2.3.3. Learning mechanisms

Learning mechanisms include neural networks and ant colony optimization. Early attempts to apply neural networks to the VRP have been rather unsuccessful (see, e.g., Ghaziri[57]; Schumann and Retzko[58]) and this line of research seems to have been abandoned. Ant colony optimization heuristics attempt to mimic the behavior of ants that detect paths containing pheromone and strengthen them with their own pheromone. This leads to the emergence of shortest paths on which pheromone accumulate faster. In meta-heuristics pheromone represents the memory of the system and corresponds to edges appearing often in good solutions. Thus the algorithm remembers good edges and is more likely to include them in a solution.

3.1 Block diagram of Eazy Route

The figure 3.1shows the block diagram of EazyRoute. It has 5 main components, as numbered in the figure. A brief description of these components is given below. Further sections contain the detailed description and block diagrams of each of the components.

Input: Three input parameters are Number of vehicles (m), Capacity of each vehicle (C), Maximum route length (L).

Compute solution: After getting all the three inputs, Compute solution component try to generate an initial feasible solution. It returns false, if no feasible solution exists, otherwise it returns the initial solution for the optimizer.

Change Parameters: When no initial feasible solution is found, EazyRoute provides user with a choice of change in parameters. User can change any of the two parameters: number of vehicles or capacity of each vehicle.

Increase vehicles: Determine the minimum number of vehicles required to generate an initial feasible solution, with fixed vehicle capacity and maximum longest distance.

Increase capacity: Determine the minimum vehicle capacity required to generate an initial feasible solution, with fixed number of vehicles and maximum longest distance.

Optimizer: Optimize the initial solution by applying different meta-heuristic algorithms.

Plotter*:* Plotter helps to graphically display the routes for different vehicles.

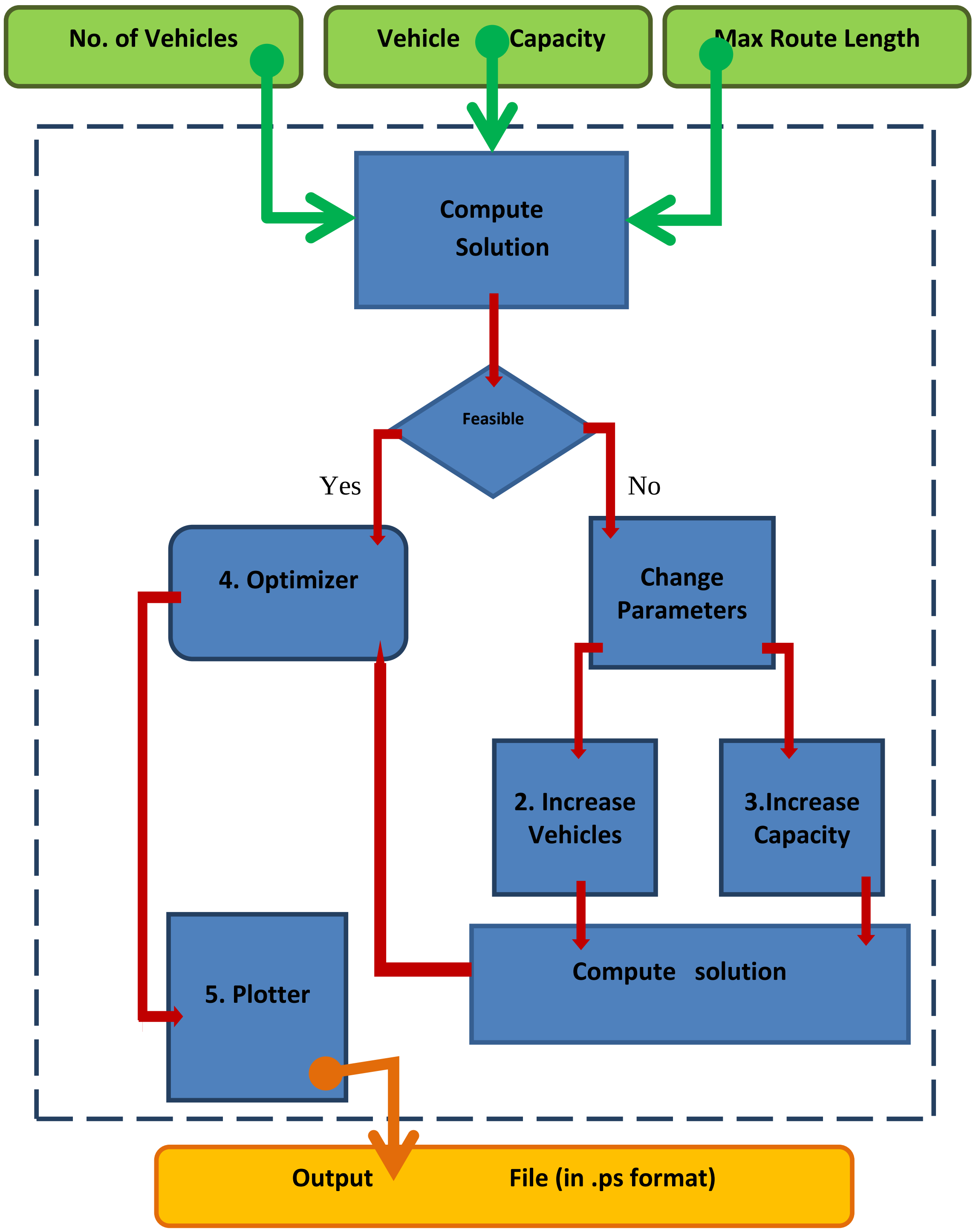


Figure 3.1 : Block diagram of EazyRoute

3.2 Description of Components of EazyRoute:

This section contains the detailed description of all the major components of EazyRoute.

3.2.1 Compute solution

The detailed description of this component is depicted through the block diagram in figure 3.2. It takes three inputs (vehicle capacity, number of vehicles) shown by green arrows.

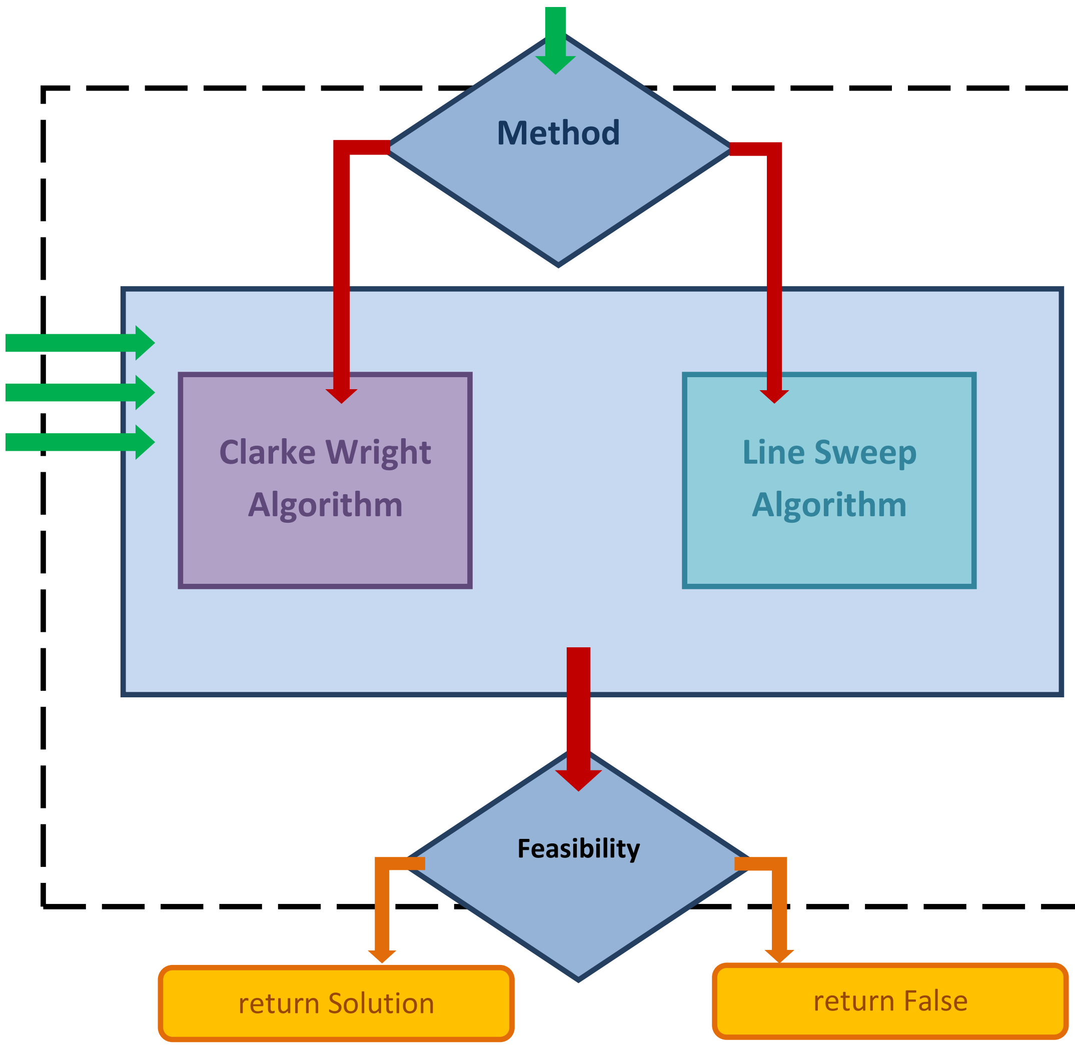


Figure 3.2: Block diagram of Compute solution component of EazyRoute

Then it prompts the user for the choice of heuristic algorithm to be used to compute the feasibility of the initial solution. User has two choices, 0 for Clarke-Wright algorithm and 1 for Line sweep algorithm. These algorithms are described in section 3.5. After choosing the method, the particular heuristic algorithm is applied and feasibility of the solution is checked. If the solution is feasible, then the resulting solution is passed to the optimizer. Otherwise, it returns false.

3.2.2 Increase number of vehicles

The description of this component is depicted through the block diagram in figure 3.3.

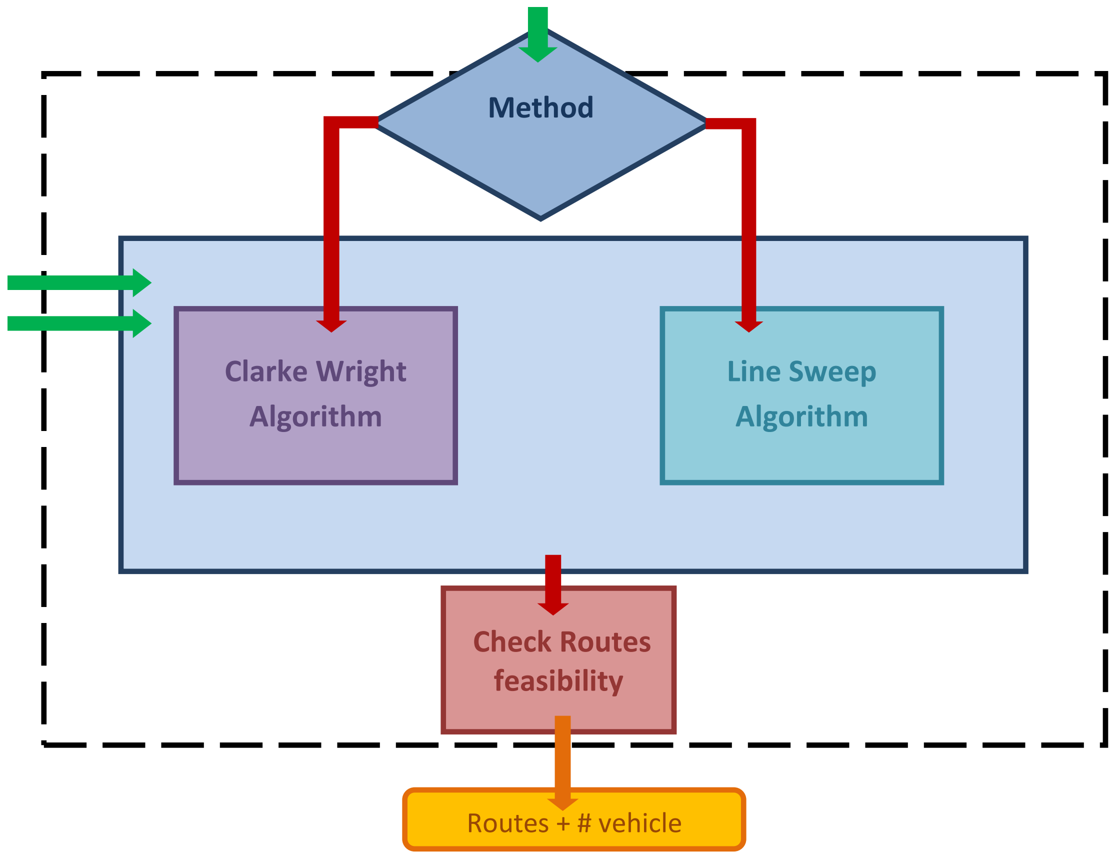


Figure 3.3: Block diagram of Increase Vehicles

This component is responsible to determine the minimum number of vehicles required to generate the initial feasible solution, keeping the vehicle capacity and maximum route length fixed. It takes two inputs (vehicle capacity & max route length), and ask the user for the choice of heuristic algorithm. Then apply chosen algorithm to find minimal number of vehicles. Then it performs a feasibility check on all the routes in the solution, if feasible, returns the no. of vehicles and the feasible routes.

3.2.3 Increase capacity

This module comes to picture, when user wants to increase the capacity of the homogenous vehicles and keeping the other two input parameters constant. It applies chosen heuristic algorithm on the basis maximum route length and initial vehicle capacity, then it calculate the number of vehicles for given initial capacity. If this number of vehicles exceeds the input number of vehicles, capacity is exponentially increased to reduce the corresponding number of vehicles. This task is performed by the sub module Exponential_ Limit_Check(), which provides a limit or we can say a starting point and an ending point within which the required solution exists. After getting the two points, it applies an efficient variation of binary search method through Binary_Interpolation() sub module, that finds the exact solution in log complexity. The Exponential_Limit_check() and Binary_Interpolation() modules make use of heuristic algorithms to reach the final solution. It returns the minimum capacity required while keeping the other two parameters constant. The detail description of this module is depicted in figure 3.4.

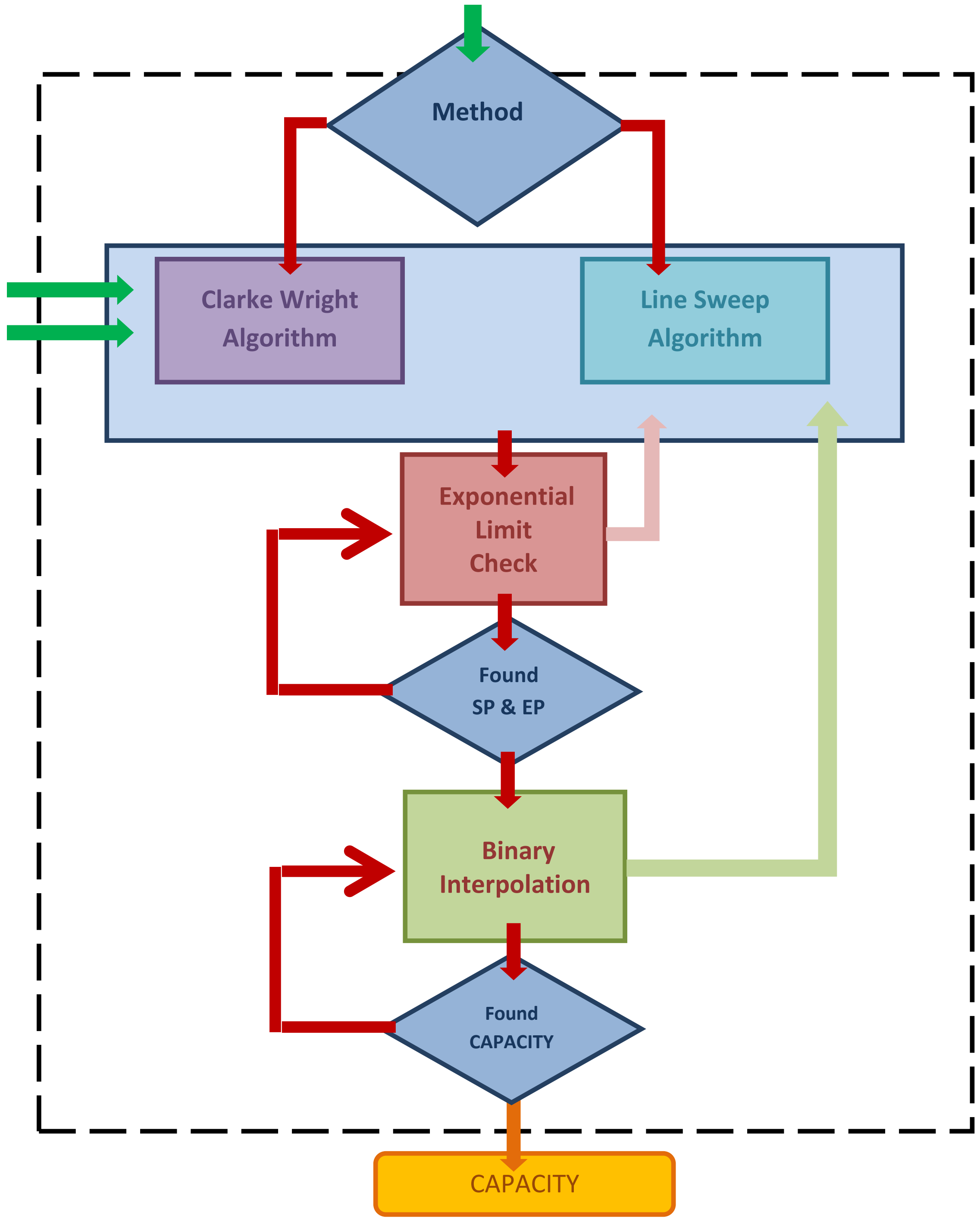


Figure 3.4: Block diagram of Increase Capacity component of EazyRoute

3.2.4 Optimizer

After generating an initial feasible solution using one of the heuristic algorithms, optimizer helps to optimize the routes, with respect to total distance travelled by all the vehicles using any of the three meta-heuristic algorithms. These meta- heuristic algorithms apply various local search operators, described in Chris Groer [9], to generate a new solution. If this new solution is better than the previous best solution, it becomes the best solution. This process is repeated (repetition depends on different parameters), and the final best solution is returned as the output of the optimizer. The three meta-heuristic algorithms differ from each other because of the local search operators. Each algorithm has its own set of operators. The optimized output is then provided to the plotter to plot the results. The block diagram of Optimizer is shown in figure 3.5.

3.2.5 Plotter

This component is responsible for generating output file that contains the graphical display of routes of the solution. It uses a third party library PlPlot to achieve the goal. It generates the output in postscript (.ps) format, and also uses postscript (.ps) to pdf converter to generate output in .pdf format. The block diagram of Plotter is shown in figure 3.6.

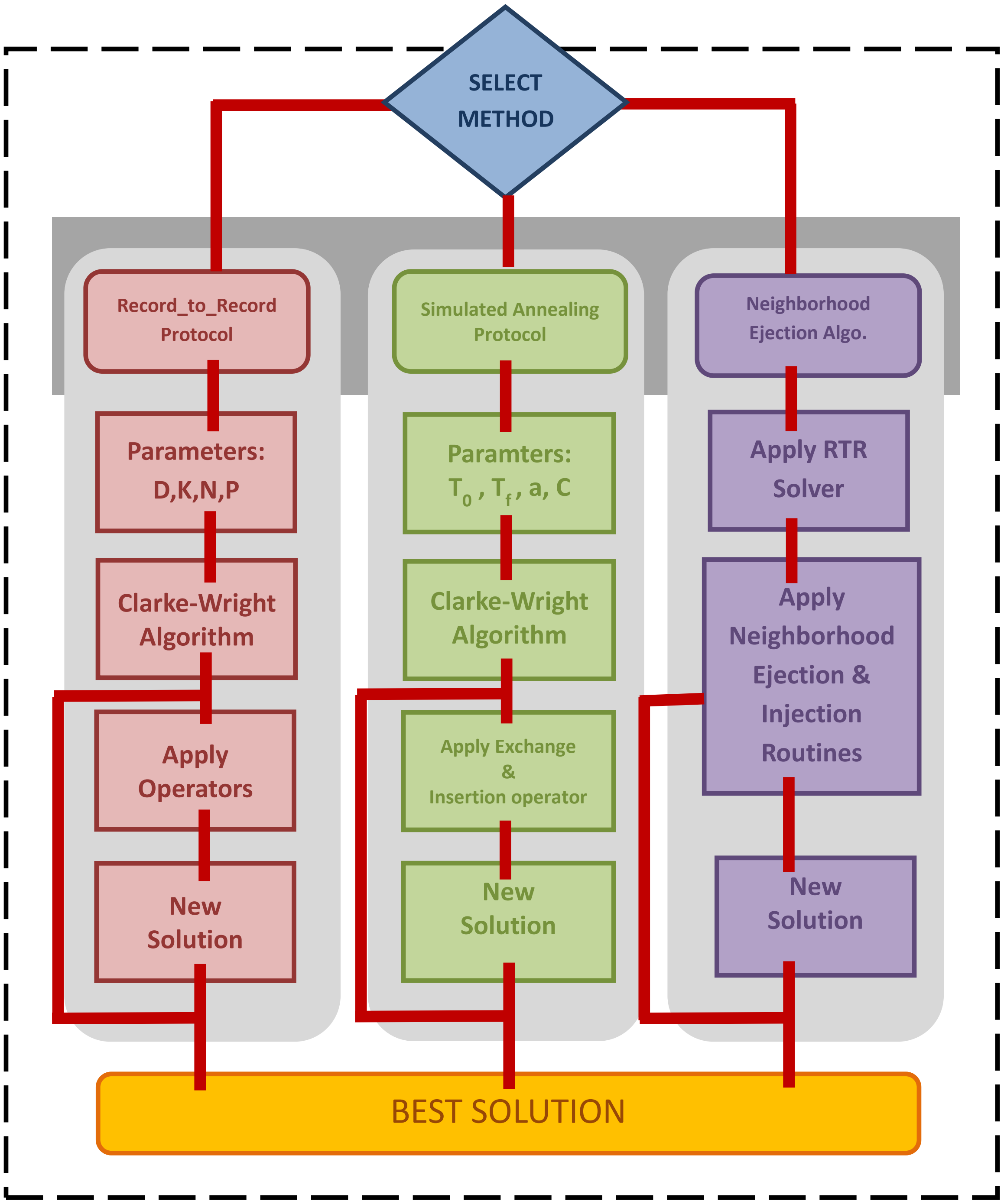

SELECT METHOD
Record_to_Record Protocol
Parameters: D,K,N,P
Clarke-Wright Algorithm
Apply Operators
New Solution
Simulated Annealing Protocol
Paramters: T_0 , T_f, a, C
Clarke-Wright Algorithm
Apply Exchange & Insertion operator
New Solution
Neighborhood Ejection Algo.
Apply RTR Solver
Apply Neighborhood Ejection & Injection Routines
New Solution
BEST SOLUTION

Figure 3.5: Block diagram of Optimizer component of EazyRoute

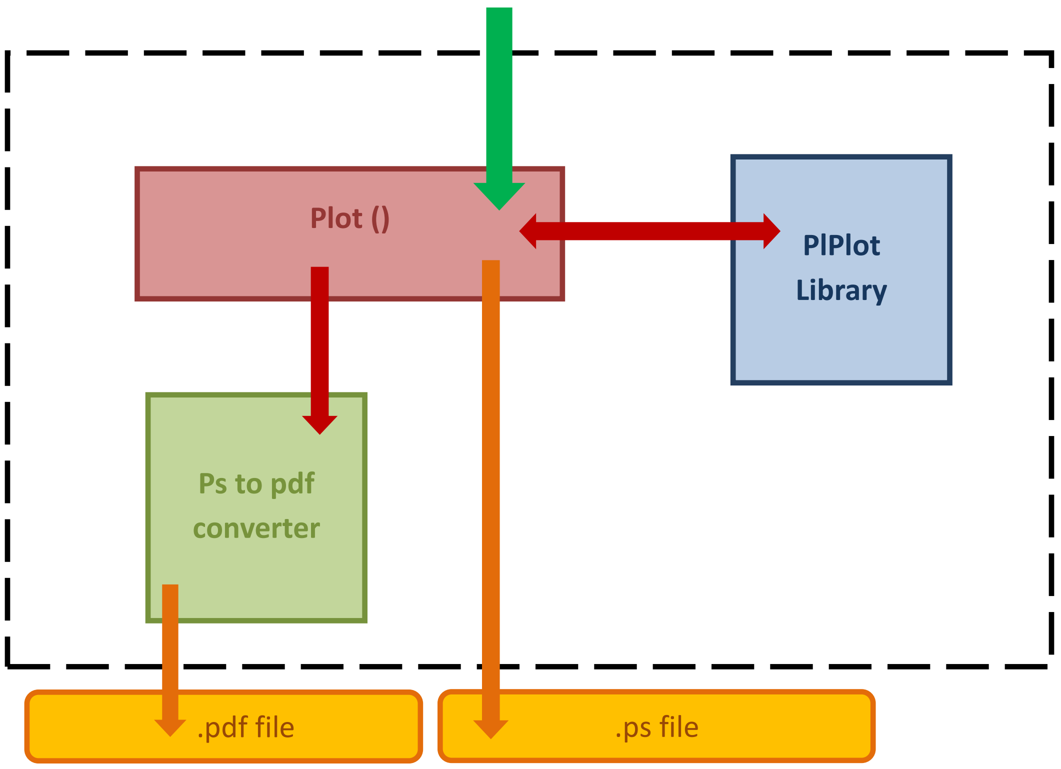


Figure 3.6: Block diagram of Plotter component of EazyRoute.

3.3 Data Structures

This section contains the description of some of the most important data structures used to implementEazyRoute. Class VRP (shown in figure 3.7) is the top level data structure that consists of various built in and derived data types. The figure shows only some of its constituents, to make the diagram simple and easy to understand. This section also throws some light on other important data structures like VRPRoute, VRPNode, DistanceMatrix, SavingList.

3.3.1 Class VRP

The figure 3.7 gives an overview of VRP class. VRP represents the whole problem set. It contains the functions and data structures to read the input parameters and also to write or print the output solutions. It also contains functions and data structures like VRPRoute, VRPSolution, SavingList, Distance Matrix, etc. to compute and store intermediate results. Final solution is stored in doubly linked list represented by two arrays next_array&pred_array.

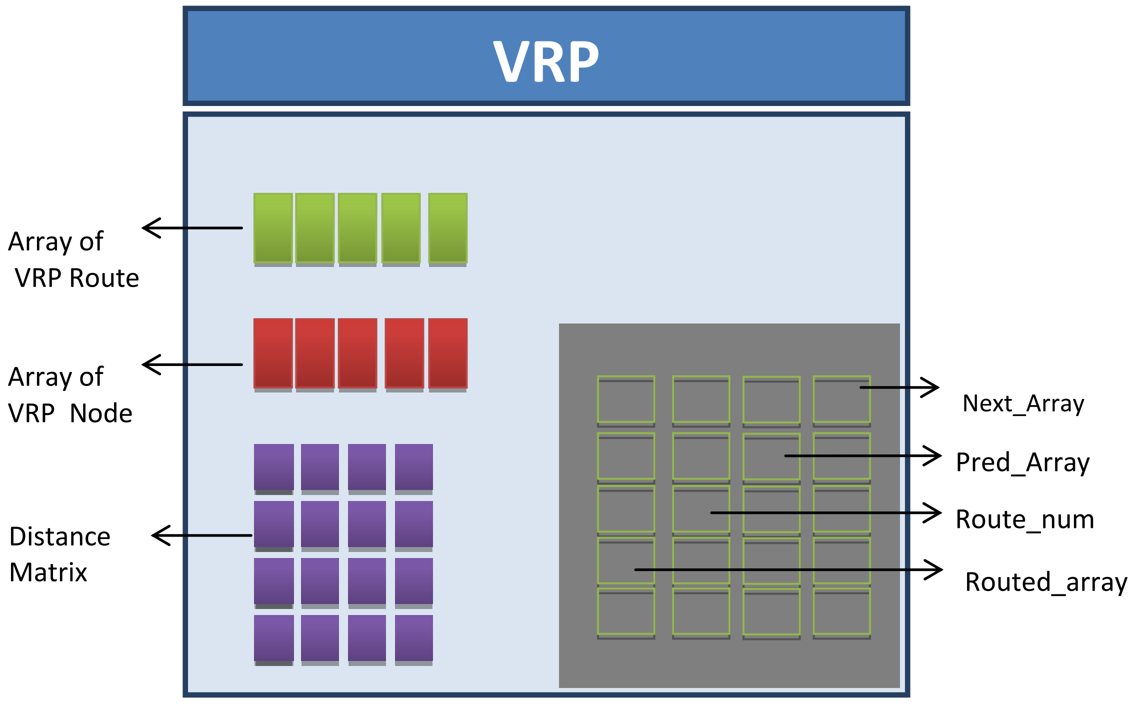


Figure 3.7: Block diagram of VRP Class data structure.

3.3.2 VRP Node

Every customer is represented with the help of this data structure. Each customer has its location, represented by x and y coordinates and its demand. EazyRoute provides a Node_id to easily identify the customer. Every customer has a neighbor list that contains pointers to all the neighbors of the customer. Each element of neighbor list contains the value and position of the neighbor in the list.The size of neighbor list can be controlled by the end user. The block diagram is shown in figure 3.8.

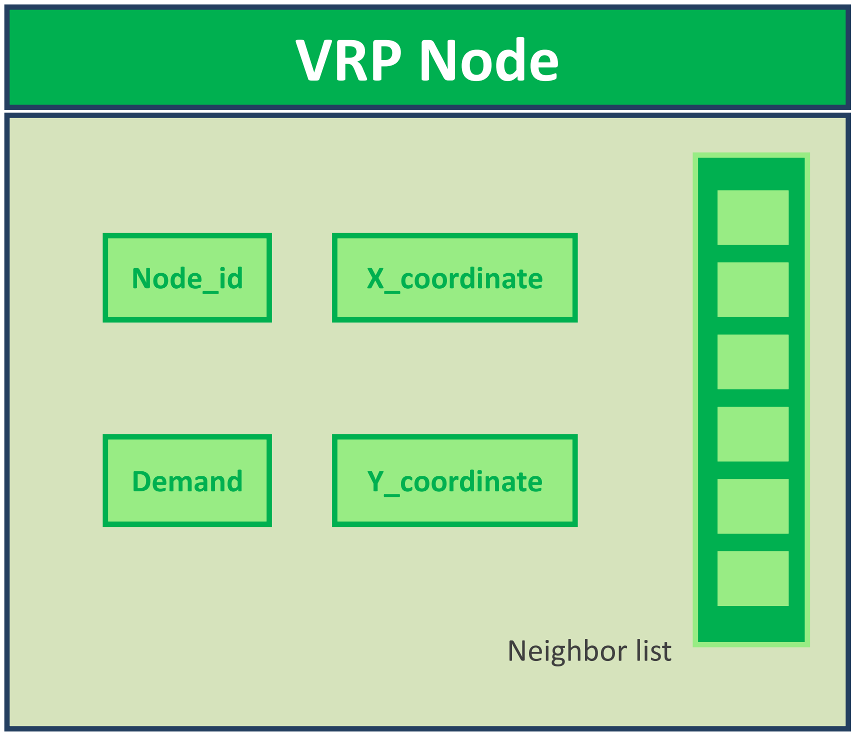


Figure 3.8: Block diagram of VRP Node data structure.

3.3.3. VRP Route

Each route is a class with some attributes like starting of the route, ending point of the route, total length of the route and total load on the route. EazyRoute also provides an id or name to a route to easily identify it. These attributes of the class are shown in the figure 3.9. There is one more attribute called Hash_Value. The Hash_Value is used during the optimization phase, where new routes are generated from the existing ones by applying local search operators. Then hash value for each route is computed& stored in a table and matched with the hash value of newly generated route, to avoid duplication of same route i.e. newly generated route is same as one of the previous routes. The block diagram is shown in figure 3.9.

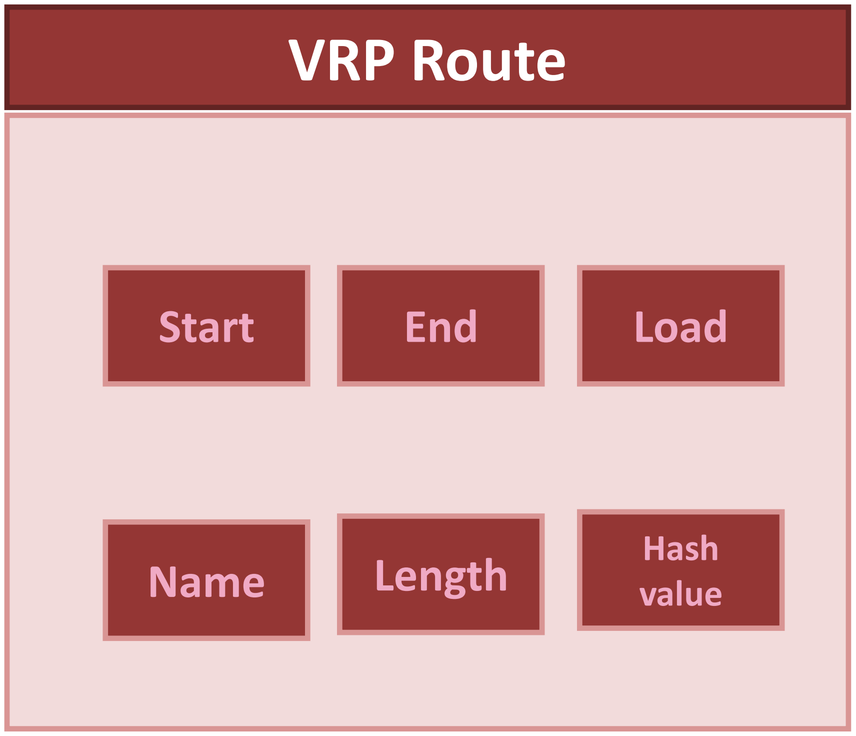


Figure 3.9: Block diagram of VRP Route data structure.

3.3.4 Distance Matrix

It is a (n+1) x (n+1) size matrix, used to store the distance between any two nodes. D_{ij}represents the distance between the i^{th} and j^{th} nodes. It is a symmetric matrix. It also helps to compute the saving matrix or list, an essential resource for implementing Clarke-wright algorithm.

3.3.5 Solution

For an n-node problem, the current solution at any stage in the solution procedure is storedin a doubly linked list contained in two arrays of length (n + 1): next_array and pred_array.This method suggested in [60]where negative indices in these arraysindicate the beginning of a new route. This presents a very compact way of storing the solutioninformation that exists in a contiguous block of memory, as compared to traditional pointer-based linked list implementation.

In any solution, each customer is assigned to a single route. EazyRoute stores this information in the route_num array and also maintains anarray of VRPRoute objects.

3.4 Input- Output formats

The Travelling salesman problem instances are generally represented in TSPLIB format [60] .EazyRoute uses quite similar file format. Some sample problem set can be downloaded from [7]. The input files are with extension .vrp.

EazyRoute writes the solution to files in a simple format during its execution. The format of solution file is as follows: the first entry in the file is the number of non-depot nodes. After this, there is a list of order in which customers are visited. The negative index shows the first node in each route. The file is stored with .sol extension. A simple example of solution storage is shown in Table 3.1.

Route	Ordering
1	0-2-5-8-0
2	0-1-3-4-0
3	0-6-7-9-10-0

Solution
10 -2 5 8 -1 3 4 -6 7 9 10 0

Table 3.1: Example solution of 10 node problem and representation of solution.

3.5 Hashing

EazyRoute provides the facility to store a number of best s solutions. When a new solution is encountered, it need to quickly determine that the new solution already exists or not. This is done by using a hashing technique described in [9]. A h-bit hash table is used for this purpose and each solution is associated with a h-bit integer.

Given an R-route solution to an n-node problem, For each j from 1 to R, we represent thejth route as an ordered list, rj. In particular, we write rj = {$a_{j\ldots\ldots}z_j$} where a_j is the indexof the first customer visited in route j and z_j is the index associated with the last customervisited in route j. Additionally, a list of n random 32-bit integers, Y_0, Y_1…….Ynis stored. Given this notation and this list Y, a positive integer is associated with each solution byperforming the steps given in Algorithm described in [9].

3.6 Algorithms Used

This section gives a brief description of all the algorithms that are used by EazyRoute.

Clarke-Wright algorithm: EazyRoute make use of a variation of Clarke-Wright algorithm described in [61] that includes a shape parameter (lambda) in savings calculation. This algorithm is one of the most popular heuristic algorithms for finding initial solution.

Line-Sweep algorithm: Clarke-wright algorithm uses a saving matrix of size (n x n), to select a link with maximum savings. But this can be easily achieved without any saving matrix. The procedure is explained in [61]. The Line sweep algorithm makes use of this procedure to generate a feasible initial solution.

Simulated Annealing algorithm : Simulated annealing algorithm helps to jump over the local minima. The variation of SA algorithm, suitable for vehicle routing problem with capacity constraints, described in [62] is used to find optimize solutions by EazyRoute.

Neighborhood Ejection algorithm : This algorithm is described in [9]. It helps to find optimize solution by ejecting out some nodes from the route and inserting them in another route to find better solution. The choice of node may be RANDOM or REGRET [63].

Record to Record Travel algorithm : RTR is one of the most popular algorithm in the field of operation research. Here we use the variation of RTR described in [9] and also propose its variation New_RTR to find more optimized solutions from the initial solutions. Section 3.6 describes both the algorithms in detail.

3.7 Proposed Algorithm

The record to record travel (RTR) algorithm is the mostly used algorithm for vehicle routing problem with capacity constraints. In this section we propose a variation of RTR, i.e. New_RTR, which proves to give better results (shown in section 4.2) when applied to same benchmarks.

3.7.1. Overview of RTR algorithm

The traditional RTR algorithm described in [64], alternates between two phases: diversification phase and improvement phase. In diversification phase, we accept some worsening moves and explore new parts of solution space. The parameter δ (deviation) controls the degree to which we are allowed to accept the worsening moves. In the improvement phase, we are allowed to accept only improving moves as we seek for the local minima. The algorithm terminates after it has been unable to escape from local minima after making several attempts.

Parameters used:

a) D: Controls the size of main loop in diversification phase. Default value = 30.
b) δ: Controls the deterioration of objective function. Default value = .01.
c) K: No. of local minima that must be reached before terminating the algorithm. Default K=5.
d) N: Size of neighbor list. Default N= 30.

e) P: No. of times solution is perturbed, once it stuck to local minima. Default P=2.

3.7.2. Proposed New_RTR algorithm

In contrast to two phases of RTR algorithm, the New_RTR algorithm has three phases. Additional phase is the Initialization phase, in which the best value of lambda (shape parameter) is chosen to form a feasible initial solution. Then the diversification phase and improvement phases are repeated for different values of deviation (δ), until and unless the best solution is found. The steps are described more clearly in table 3.2.

Algorithm 1: New_RTR

Initialization Phase:

```
1.  For λ=0.1  to 2 do
2.      Generate feasible solution using Clarke-Wright algorithm.
3.      If this is the better than best solution
4.  Best_λ = λ
5.  End If
6.  End For
7.  For δ =0.01 to 0.1 do
```

Diversification Phase :

```
8.      Generate initial feasible solution using Clarke-Wright algo using Best_ λ.
9.      Select a set of local search operators U.
10.     Set record R= Current total route length, Set threshold T=(1+δ)R, set k=p=0;
11.     While p < P do
12.         Fori =1 to D do
13.             Forall operators u in U do
14.                 Forall Nodes j in the solution do
15.                     Apply operator u to node j using RTR travel
16.                 End For
17.             End For
18.         End For
```

Improvement Phase

```
19.         While improving moves can be founddo
20.             For all operators u in U do
21.                 For all Nodes j in the solution do
22.                     Apply operator u to node j accepting onlyimproving
                        moves.
23.                 End For
24.             End For
25.         End While
26.         If The current solution is a new record then
27.             Update R and T and set k=0
28.         End If
29.         k=k+1
30.         If (k = = K ) then
31.             Perturb the solution
32.             p = p + 1
33.         End IF
34.     End While
35. End For
36. Return the best solution found
```

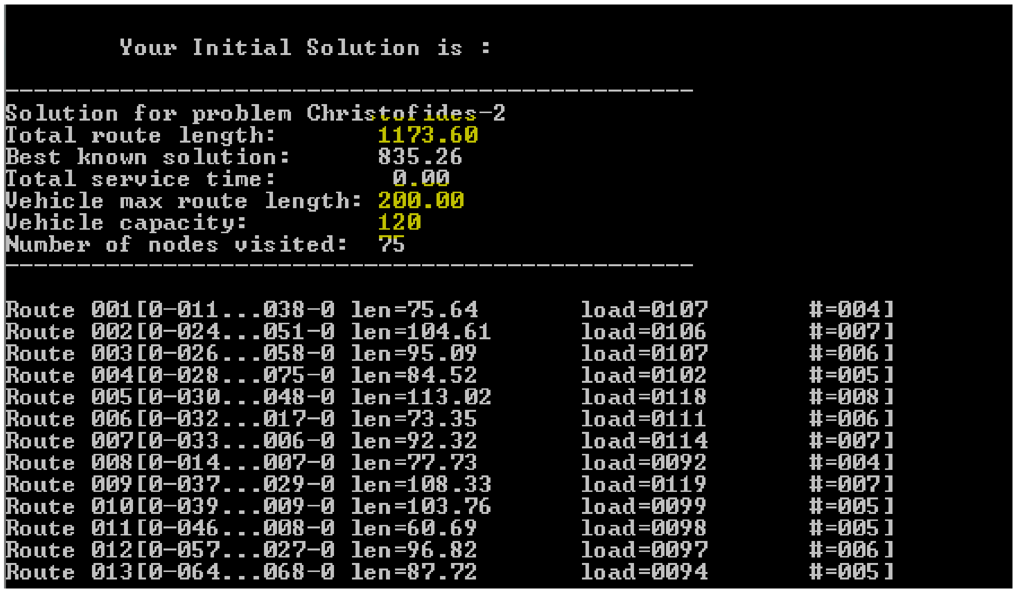
Section 4.1 contains the results of executing the EazyRoute on different standard benchmarks available at https://sites.google.com/site/vrpwcc . They are mainly classified in four categories Christofides et al, Golden et al, Li et al and Taillard et al. Section 4.2 contains the results and observations of the performance of our proposed New_rtr algorithm with respect to general record to record travel algorithm.

4.1 Results:

Figure 4.1 to 4.25 shows the results. Every figure has a table screenshot that indicates the details of the solution, like starting point, ending point, total length, total load, and number of nodes per route. It also tells about the constraints like vehicle maximum route length and vehicle maximum capacity. This screenshot also shows the best known solution that exists for the current benchmark. Part B of every figure shows the graphical display of routes.

For every benchmark, there are five figures. First figure shows the initial solution after applying the Clarke-wright algorithm or Sweep-line algorithm. Second figure shows the optimized solution after applying variation of record to record travel algorithm (see section 3.7.1). Third figure shows the optimized solution after applying the New_RTR algorithm. The fourth figure shows the optimized solution after applying simulated annealing algorithm (see section 3.6) and the fifth figure shows the solution obtained after applying Neighborhood ejection algorithm (see section 3.6) on the initial solution.

4.1.1. Christofides_02.vrp (75 nodes) – Initial Solution.



```
        Your Initial Solution is :

---------------------------------------------
Solution for problem Christofides-2
Total route length:        1173.60
Best known solution:       835.26
Total service time:         0.00
Vehicle max route length:  200.00
Vehicle capacity:          120
Number of nodes visited:   75
---------------------------------------------

Route 001[0-011...038-0 len=75.64       load=0107       #=004]
Route 002[0-024...051-0 len=104.61      load=0106       #=007]
Route 003[0-026...058-0 len=95.09       load=0107       #=006]
Route 004[0-028...075-0 len=84.52       load=0102       #=005]
Route 005[0-030...048-0 len=113.02      load=0118       #=008]
Route 006[0-032...017-0 len=73.35       load=0111       #=006]
Route 007[0-033...006-0 len=92.32       load=0114       #=007]
Route 008[0-014...007-0 len=77.73       load=0092       #=004]
Route 009[0-037...029-0 len=108.33      load=0119       #=007]
Route 010[0-039...009-0 len=103.76      load=0099       #=005]
Route 011[0-046...008-0 len=60.69       load=0098       #=005]
Route 012[0-057...027-0 len=96.82       load=0097       #=006]
Route 013[0-064...068-0 len=87.72       load=0094       #=005]
```

Fig. 4.1(a) Route details for Christofides_02 after applying Clarke-Wright algorithm.

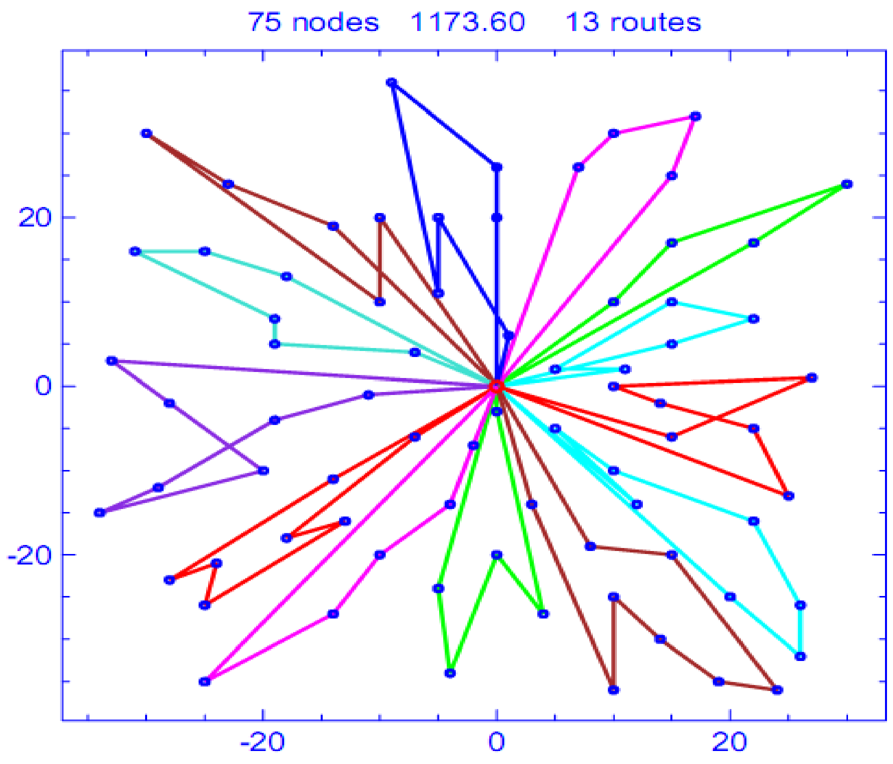
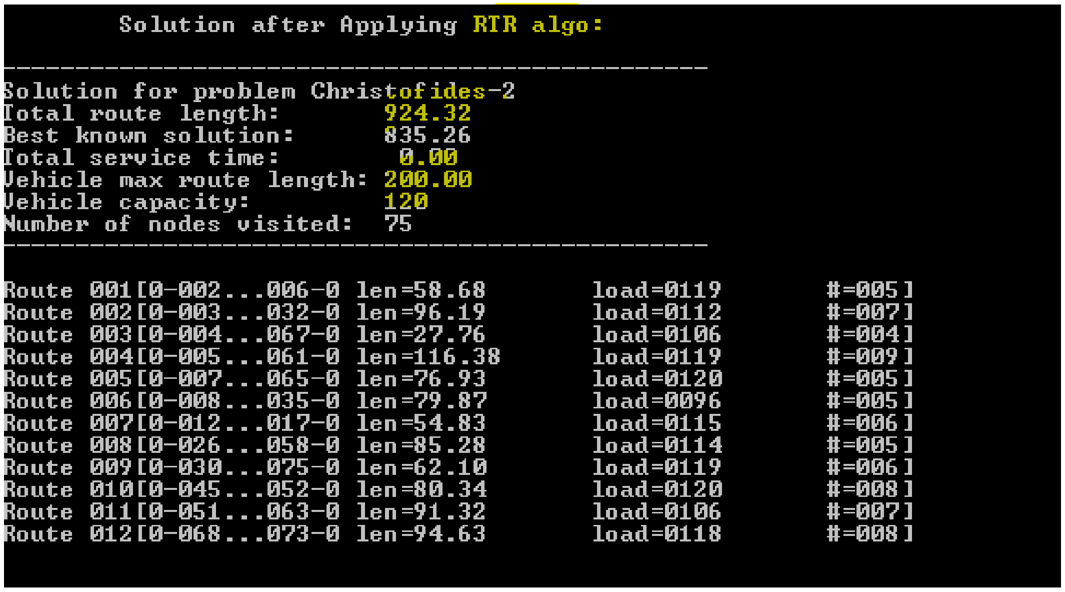


Fig. 4.1(b): Plot of routes for Christofides_02.

4.1.2. Christofides_02.vrp – After applying RTR algorithm.



```
        Solution after Applying RTR algo:

------------------------------------------------
Solution for problem Christofides-2
Total route length:        924.32
Best known solution:       835.26
Total service time:          0.00
Vehicle max route length:  200.00
Vehicle capacity:          120
Number of nodes visited:   75
------------------------------------------------

Route 001[0-002...006-0 len=58.68       load=0119      #=005]
Route 002[0-003...032-0 len=96.19       load=0112      #=007]
Route 003[0-004...067-0 len=27.76       load=0106      #=004]
Route 004[0-005...061-0 len=116.38      load=0119      #=009]
Route 005[0-007...065-0 len=76.93       load=0120      #=005]
Route 006[0-008...035-0 len=79.87       load=0096      #=005]
Route 007[0-012...017-0 len=54.83       load=0115      #=006]
Route 008[0-026...058-0 len=85.28       load=0114      #=005]
Route 009[0-030...075-0 len=62.10       load=0119      #=006]
Route 010[0-045...052-0 len=80.34       load=0120      #=008]
Route 011[0-051...063-0 len=91.32       load=0106      #=007]
Route 012[0-068...073-0 len=94.63       load=0118      #=008]
```

Fig. 4.2(a) Optimized Route details for Christofides_02 after applying RTR algorithm.

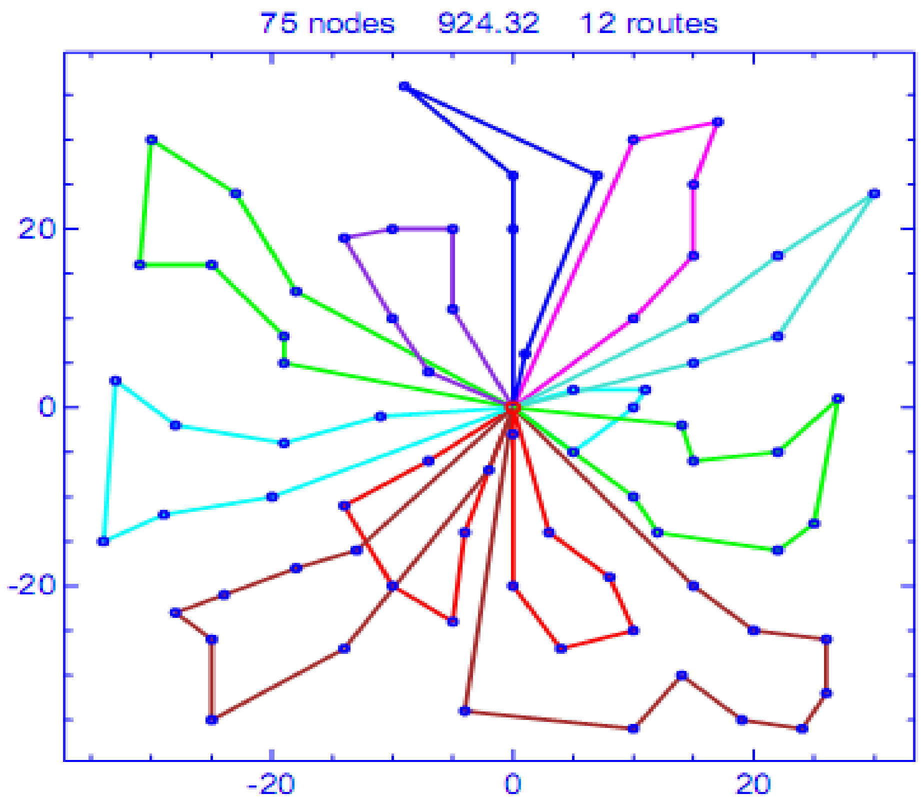
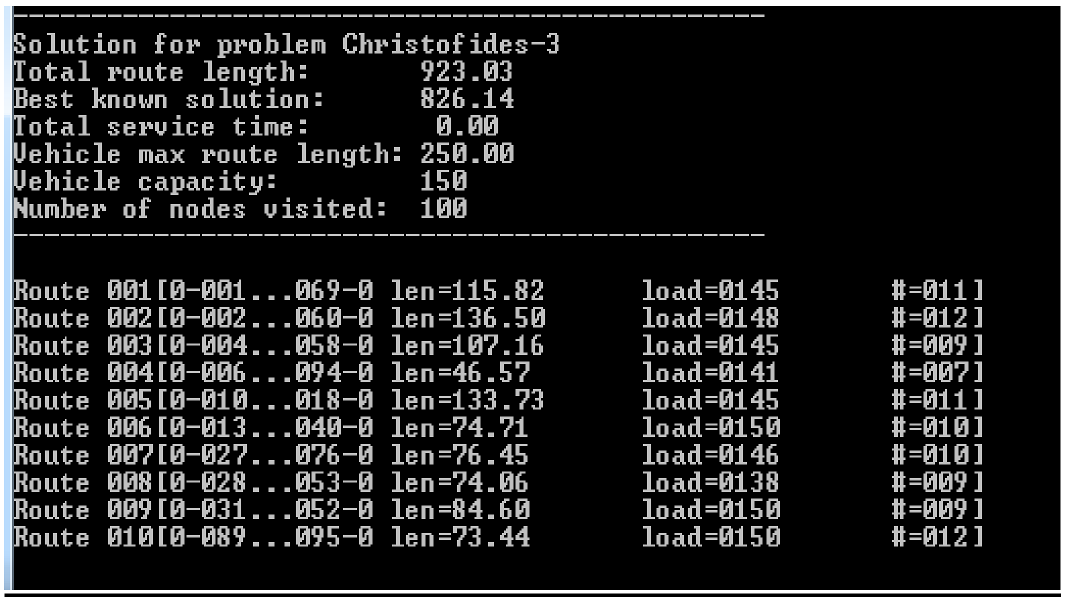


Fig. 4.2(b): Plot of routes for Christofides_02 after optimization by RTR algo.

4.1.3. Christofides_02.vrp – After applying New_RTR algorithm.



```
----------------------------------------------
Solution for problem Christofides-3
Total route length:        923.03
Best known solution:       826.14
Total service time:          0.00
Vehicle max route length:  250.00
Vehicle capacity:          150
Number of nodes visited:   100
----------------------------------------------

Route 001[0-001...069-0 len=115.82      load=0145       #=011]
Route 002[0-002...060-0 len=136.50      load=0148       #=012]
Route 003[0-004...058-0 len=107.16      load=0145       #=009]
Route 004[0-006...094-0 len=46.57       load=0141       #=007]
Route 005[0-010...018-0 len=133.73      load=0145       #=011]
Route 006[0-013...040-0 len=74.71       load=0150       #=010]
Route 007[0-027...076-0 len=76.45       load=0146       #=010]
Route 008[0-028...053-0 len=74.06       load=0138       #=009]
Route 009[0-031...052-0 len=84.60       load=0150       #=009]
Route 010[0-089...095-0 len=73.44       load=0150       #=012]
```

Fig. 4.3(a) Optimized Route details for Christofides_02 after applying New_RTR algorithm.

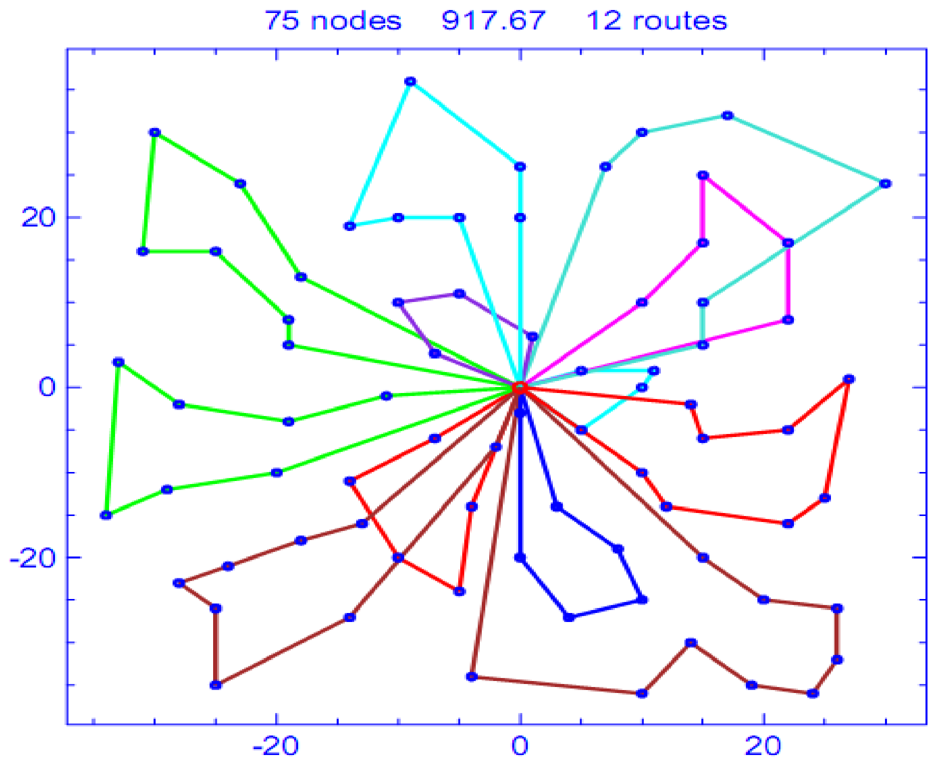
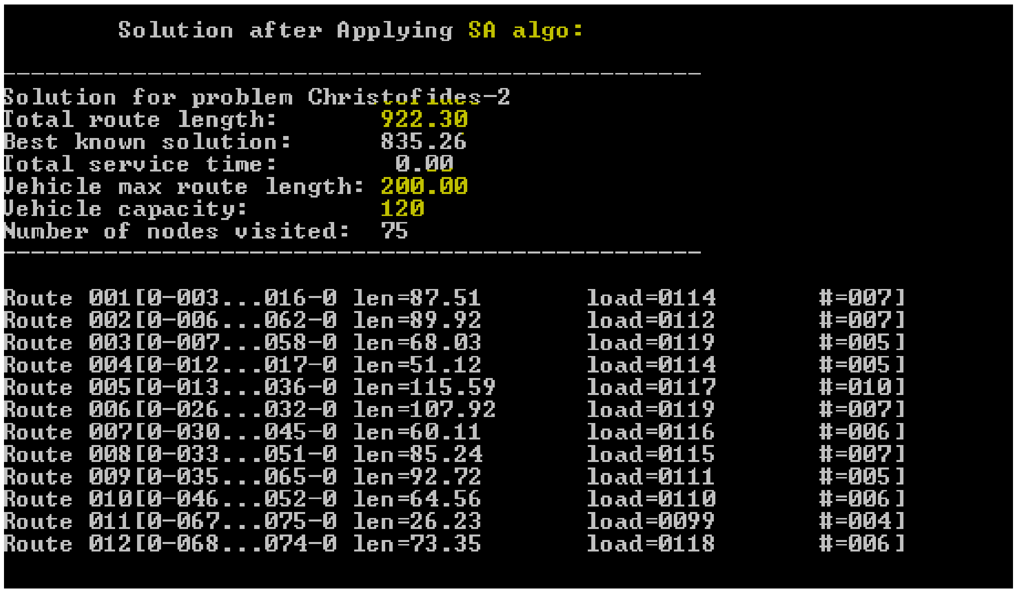


Fig. 4.3(b): Plot of routes for Christofides_02 after optimization by New_RTR algo.

4.1.4. Christofides_02.vrp – After applying SA algorithm.



```
        Solution after Applying SA algo:

------------------------------------------------
Solution for problem Christofides-2
Total route length:         922.30
Best known solution:        835.26
Total service time:          0.00
Vehicle max route length:  200.00
Vehicle capacity:           120
Number of nodes visited:    75
------------------------------------------------

Route 001[0-003...016-0 len=87.51       load=0114       #=007]
Route 002[0-006...062-0 len=89.92       load=0112       #=007]
Route 003[0-007...058-0 len=68.03       load=0119       #=005]
Route 004[0-012...017-0 len=51.12       load=0114       #=005]
Route 005[0-013...036-0 len=115.59      load=0117       #=010]
Route 006[0-026...032-0 len=107.92      load=0119       #=007]
Route 007[0-030...045-0 len=60.11       load=0116       #=006]
Route 008[0-033...051-0 len=85.24       load=0115       #=007]
Route 009[0-035...065-0 len=92.72       load=0111       #=005]
Route 010[0-046...052-0 len=64.56       load=0110       #=006]
Route 011[0-067...075-0 len=26.23       load=0099       #=004]
Route 012[0-068...074-0 len=73.35       load=0118       #=006]
```

Fig. 4.4(a) Optimized Route details for Christofides_02 after applying SA algorithm.

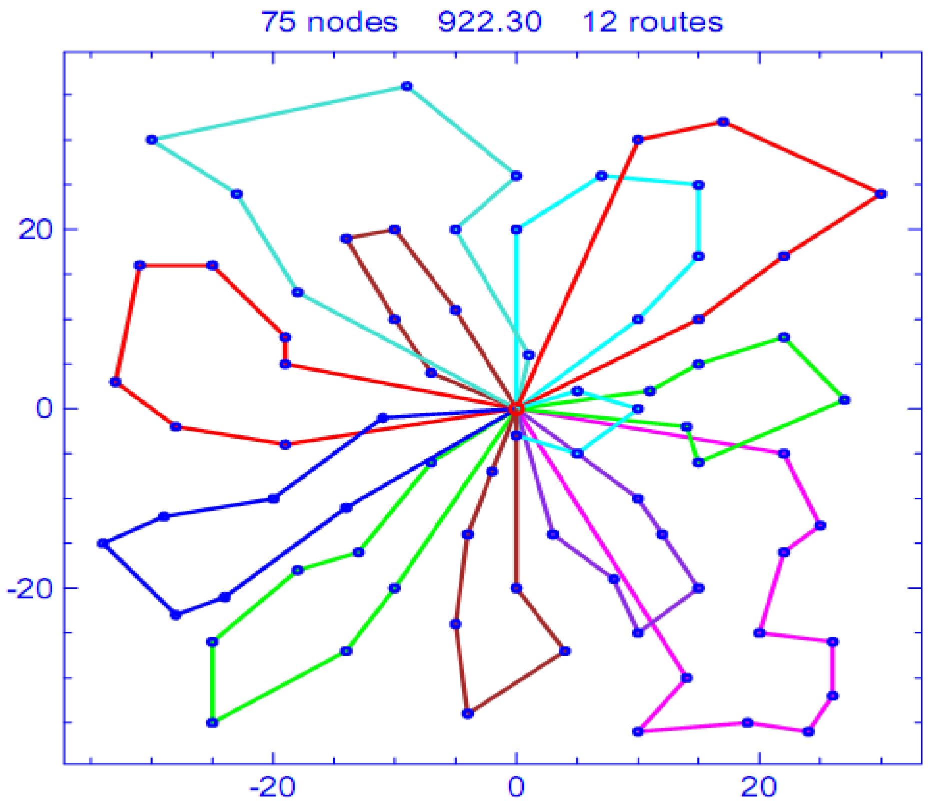
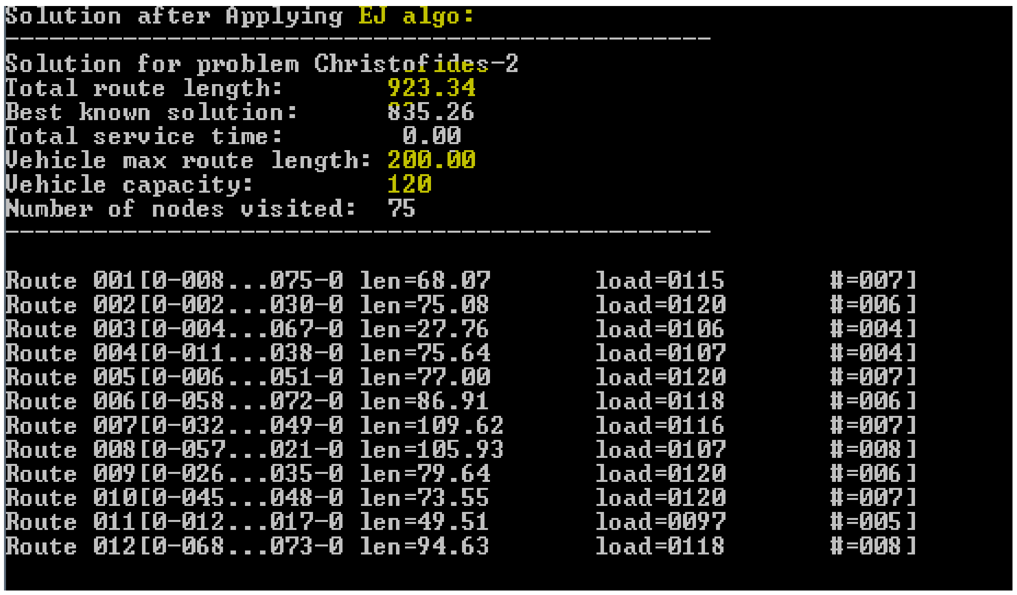


Fig. 4.4(b): Plot of routes for Christofides_02 after optimization by SA algo.

4.1.5. Christofides_02.vrp – After applying EJ algorithm.



```
Solution after Applying EJ algo:
------------------------------------------------
Solution for problem Christofides-2
Total route length:        923.34
Best known solution:       835.26
Total service time:          0.00
Vehicle max route length: 200.00
Vehicle capacity:          120
Number of nodes visited:   75
------------------------------------------------

Route 001[0-008...075-0 len=68.07       load=0115       #=007]
Route 002[0-002...030-0 len=75.08       load=0120       #=006]
Route 003[0-004...067-0 len=27.76       load=0106       #=004]
Route 004[0-011...038-0 len=75.64       load=0107       #=004]
Route 005[0-006...051-0 len=77.00       load=0120       #=007]
Route 006[0-058...072-0 len=86.91       load=0118       #=006]
Route 007[0-032...049-0 len=109.62      load=0116       #=007]
Route 008[0-057...021-0 len=105.93      load=0107       #=008]
Route 009[0-026...035-0 len=79.64       load=0120       #=006]
Route 010[0-045...048-0 len=73.55       load=0120       #=007]
Route 011[0-012...017-0 len=49.51       load=0097       #=005]
Route 012[0-068...073-0 len=94.63       load=0118       #=008]
```

Fig. 4.5(a) Optimized Route details for Christofides_02 after applying EJ algorithm.

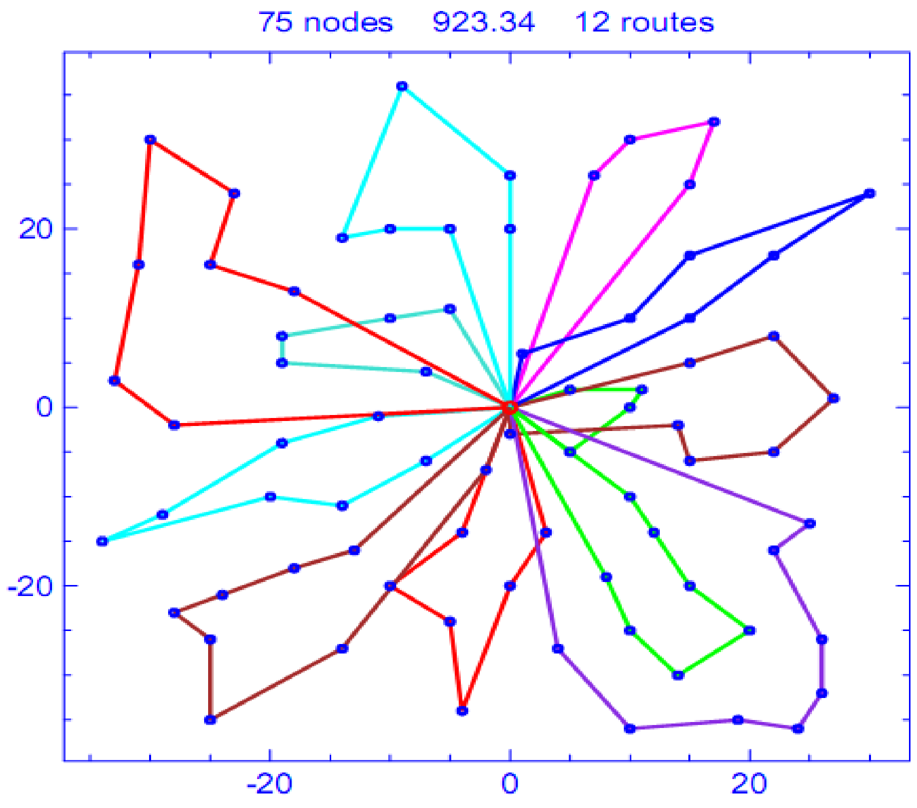
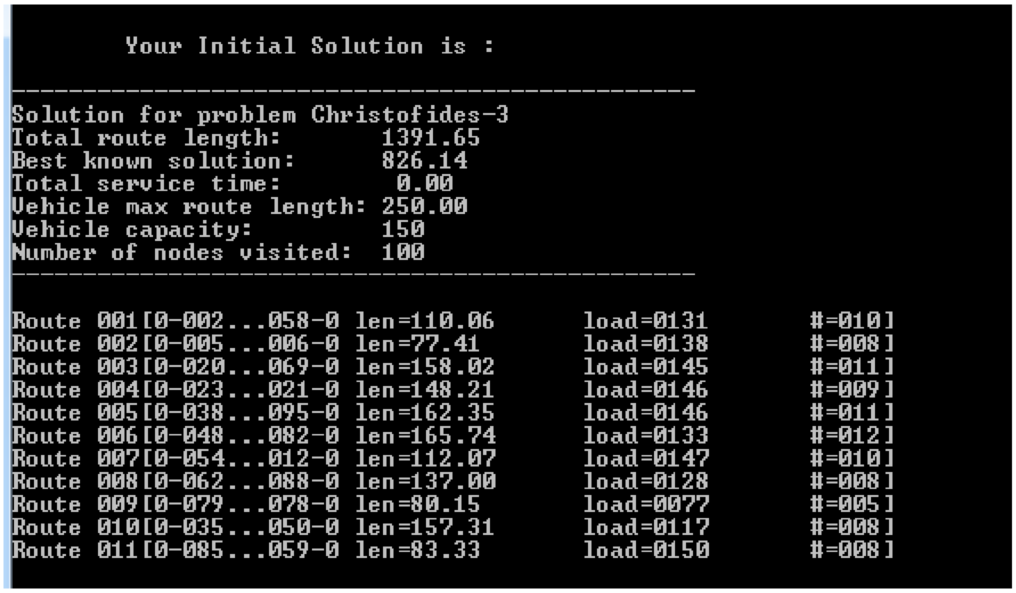


Fig. 4.5(b): Plot of routes for Christofides_02 after optimization by EJ algo.

4.1.6. Christofides_03.vrp- Initial solution.



```
        Your Initial Solution is :

------------------------------------------------
Solution for problem Christofides-3
Total route length:        1391.65
Best known solution:       826.14
Total service time:          0.00
Vehicle max route length: 250.00
Vehicle capacity:          150
Number of nodes visited:   100
------------------------------------------------

Route 001[0-002...058-0 len=110.06      load=0131       #=010]
Route 002[0-005...006-0 len=77.41       load=0138       #=008]
Route 003[0-020...069-0 len=158.02      load=0145       #=011]
Route 004[0-023...021-0 len=148.21      load=0146       #=009]
Route 005[0-038...095-0 len=162.35      load=0146       #=011]
Route 006[0-048...082-0 len=165.74      load=0133       #=012]
Route 007[0-054...012-0 len=112.07      load=0147       #=010]
Route 008[0-062...088-0 len=137.00      load=0128       #=008]
Route 009[0-079...078-0 len=80.15       load=0077       #=005]
Route 010[0-035...050-0 len=157.31      load=0117       #=008]
Route 011[0-085...059-0 len=83.33       load=0150       #=008]
```

Fig. 4.6(a) Route details for Christofides_03 after applying Clarke-Wright algorithm.

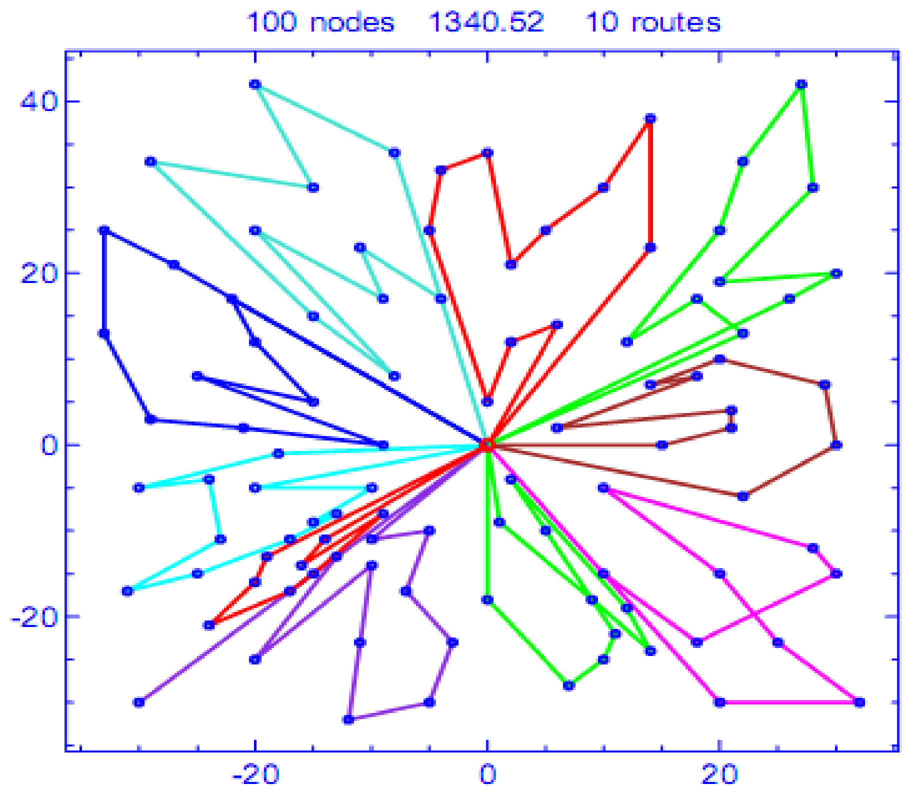
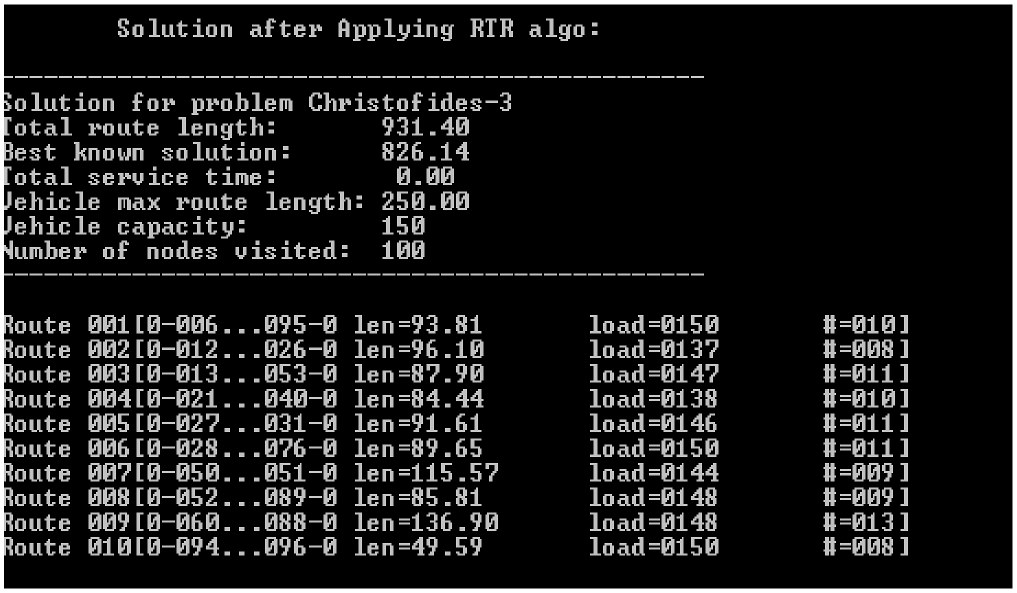


Fig. 4.6(b): Plot of routes for Christofides_03.

4.1.7. Christofides_03.vrp – After applying RTR algorithm.



```
        Solution after Applying RTR algo:

-------------------------------------------------
Solution for problem Christofides-3
Total route length:        931.40
Best known solution:       826.14
Total service time:          0.00
Vehicle max route length: 250.00
Vehicle capacity:          150
Number of nodes visited:   100
-------------------------------------------------

Route 001[0-006...095-0 len=93.81        load=0150        #=010]
Route 002[0-012...026-0 len=96.10        load=0137        #=008]
Route 003[0-013...053-0 len=87.90        load=0147        #=011]
Route 004[0-021...040-0 len=84.44        load=0138        #=010]
Route 005[0-027...031-0 len=91.61        load=0146        #=011]
Route 006[0-028...076-0 len=89.65        load=0150        #=011]
Route 007[0-050...051-0 len=115.57       load=0144        #=009]
Route 008[0-052...089-0 len=85.81        load=0148        #=009]
Route 009[0-060...088-0 len=136.90       load=0148        #=013]
Route 010[0-094...096-0 len=49.59        load=0150        #=008]
```

Fig. 4.7(a) Optimized route details for Christofides_03 after applying RTR algorithm.

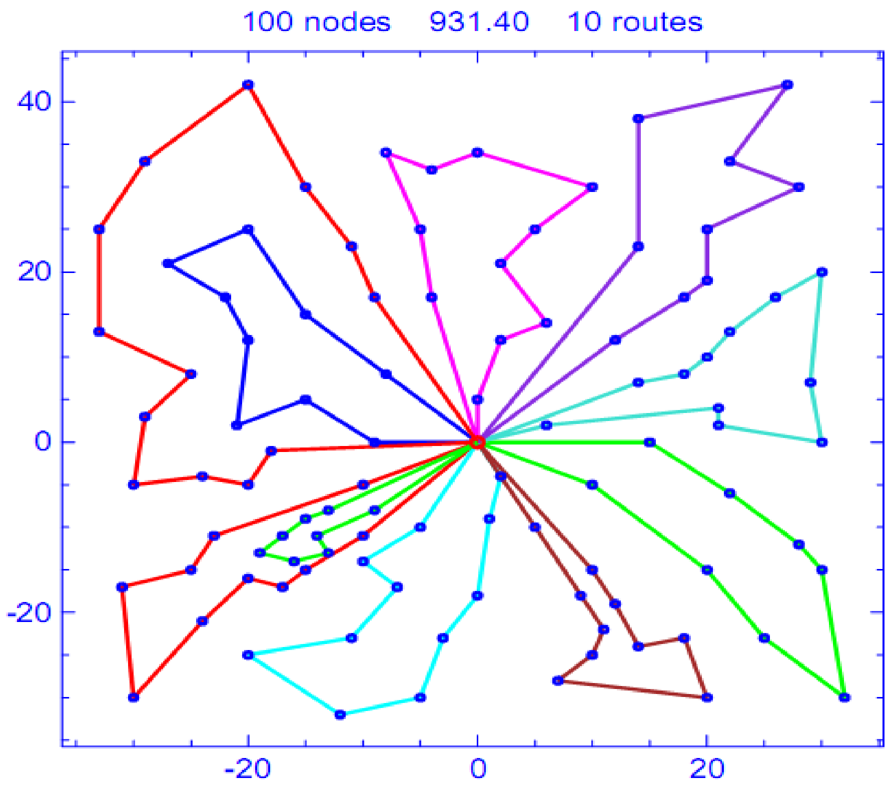
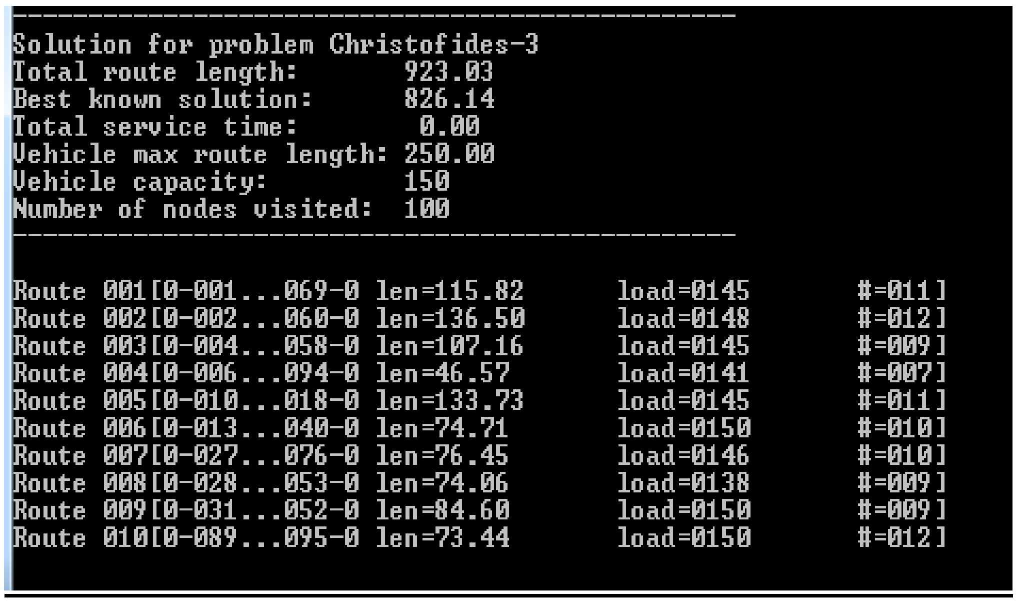


Fig. 4.7(b): Plot of routes for Christofides_03 after optimization by RTR algo.

4.1.8. Christofides_03.vrp – After applying New_RTR algorithm.



```
--------------------------------------------
Solution for problem Christofides-3
Total route length:        923.03
Best known solution:       826.14
Total service time:          0.00
Vehicle max route length: 250.00
Vehicle capacity:          150
Number of nodes visited:   100
--------------------------------------------

Route 001[0-001...069-0 len=115.82      load=0145       #=011]
Route 002[0-002...060-0 len=136.50      load=0148       #=012]
Route 003[0-004...058-0 len=107.16      load=0145       #=009]
Route 004[0-006...094-0 len=46.57       load=0141       #=007]
Route 005[0-010...018-0 len=133.73      load=0145       #=011]
Route 006[0-013...040-0 len=74.71       load=0150       #=010]
Route 007[0-027...076-0 len=76.45       load=0146       #=010]
Route 008[0-028...053-0 len=74.06       load=0138       #=009]
Route 009[0-031...052-0 len=84.60       load=0150       #=009]
Route 010[0-089...095-0 len=73.44       load=0150       #=012]
```

Fig. 4.8(a) Optimized route details for Christofides_03 after applying New_RTR algorithm.

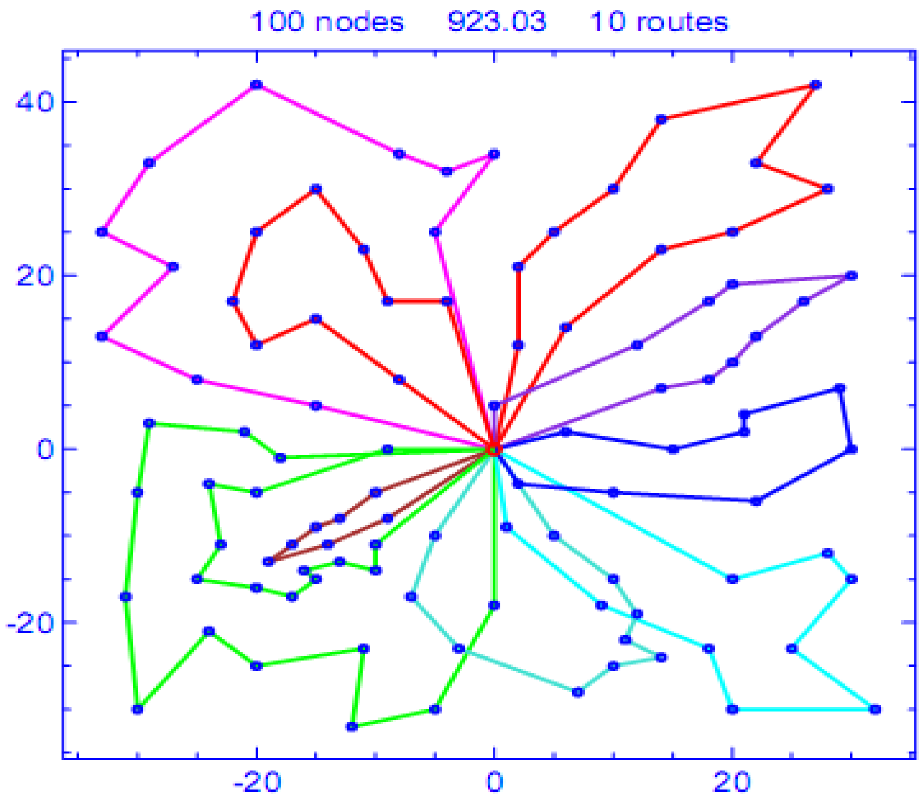
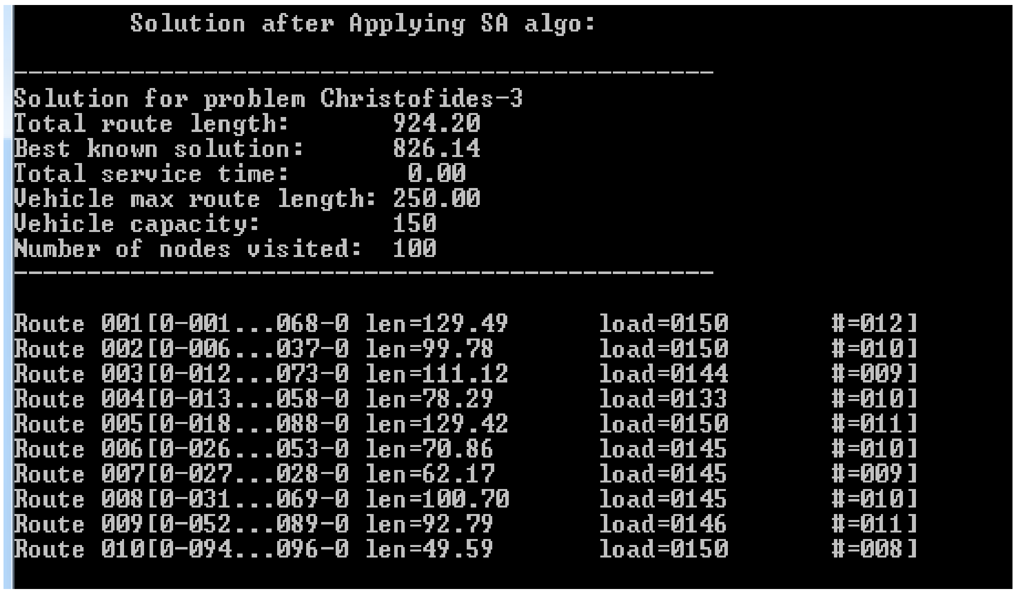


Fig. 4.8(b): Plot of routes for Christofides_03 after optimization by New_RTR algo.

4.1.9. Christofides_03.vrp – After applying SA algorithm.



```
        Solution after Applying SA algo:

------------------------------------------------
Solution for problem Christofides-3
Total route length:       924.20
Best known solution:      826.14
Total service time:         0.00
Vehicle max route length: 250.00
Vehicle capacity:         150
Number of nodes visited:  100
------------------------------------------------

Route 001[0-001...068-0 len=129.49      load=0150       #=012]
Route 002[0-006...037-0 len=99.78       load=0150       #=010]
Route 003[0-012...073-0 len=111.12      load=0144       #=009]
Route 004[0-013...058-0 len=78.29       load=0133       #=010]
Route 005[0-018...088-0 len=129.42      load=0150       #=011]
Route 006[0-026...053-0 len=70.86       load=0145       #=010]
Route 007[0-027...028-0 len=62.17       load=0145       #=009]
Route 008[0-031...069-0 len=100.70      load=0145       #=010]
Route 009[0-052...089-0 len=92.79       load=0146       #=011]
Route 010[0-094...096-0 len=49.59       load=0150       #=008]
```

Fig. 4.9(a) Optimized route details for Christofides_03 after applying SA algorithm.

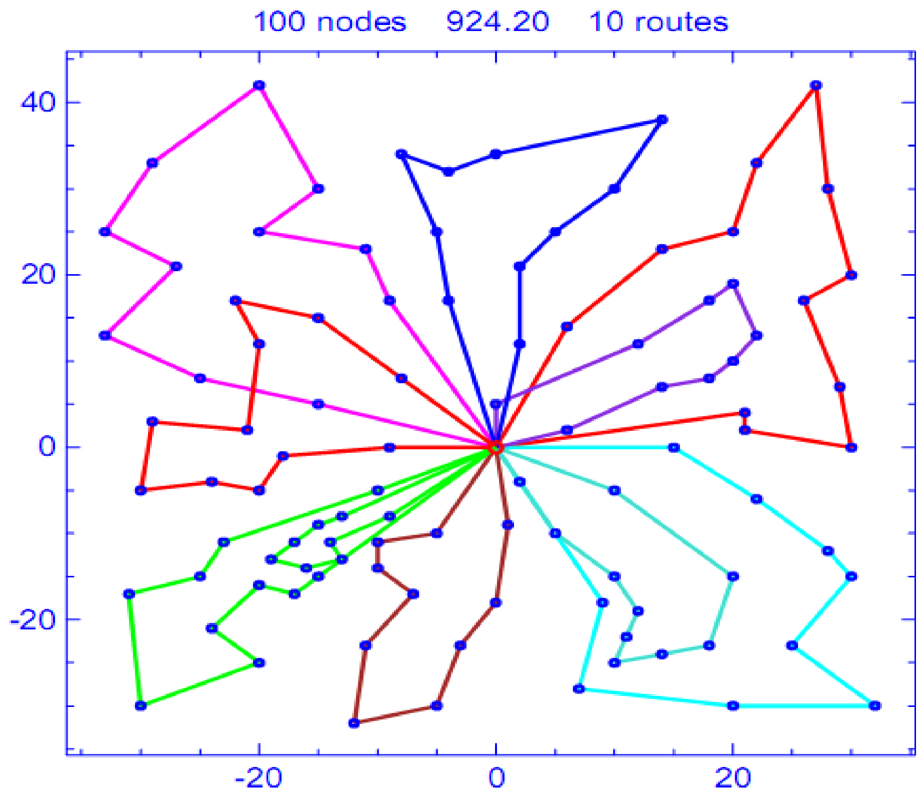
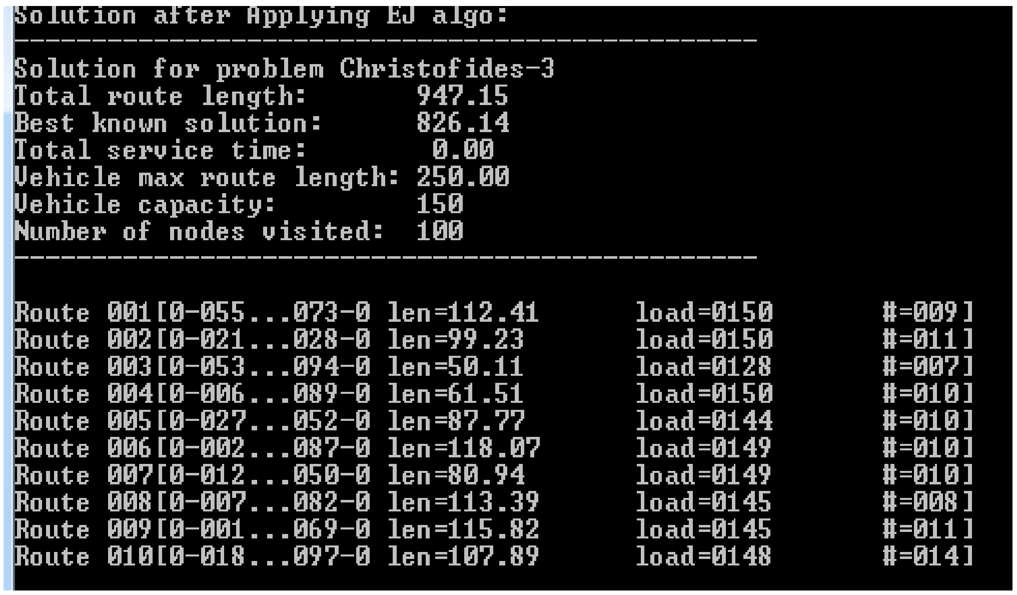


Fig. 4.9(b): Plot of routes for Christofides_03 after optimization by SA algo.

4.1.10. Christofides_03.vrp – After applying EJ algorithm.



```
Solution after Applying EJ algo:
-----------------------------------------------

Solution for problem Christofides-3
Total route length:         947.15
Best known solution:        826.14
Total service time:           0.00
Vehicle max route length: 250.00
Vehicle capacity:           150
Number of nodes visited:   100
-----------------------------------------------

Route 001[0-055...073-0 len=112.41      load=0150       #=009]
Route 002[0-021...028-0 len=99.23       load=0150       #=011]
Route 003[0-053...094-0 len=50.11       load=0128       #=007]
Route 004[0-006...089-0 len=61.51       load=0150       #=010]
Route 005[0-027...052-0 len=87.77       load=0144       #=010]
Route 006[0-002...087-0 len=118.07      load=0149       #=010]
Route 007[0-012...050-0 len=80.94       load=0149       #=010]
Route 008[0-007...082-0 len=113.39      load=0145       #=008]
Route 009[0-001...069-0 len=115.82      load=0145       #=011]
Route 010[0-018...097-0 len=107.89      load=0148       #=014]
```

Fig. 4.10(a) Optimized route details for Christofides_03 after applying EJ algorithm.

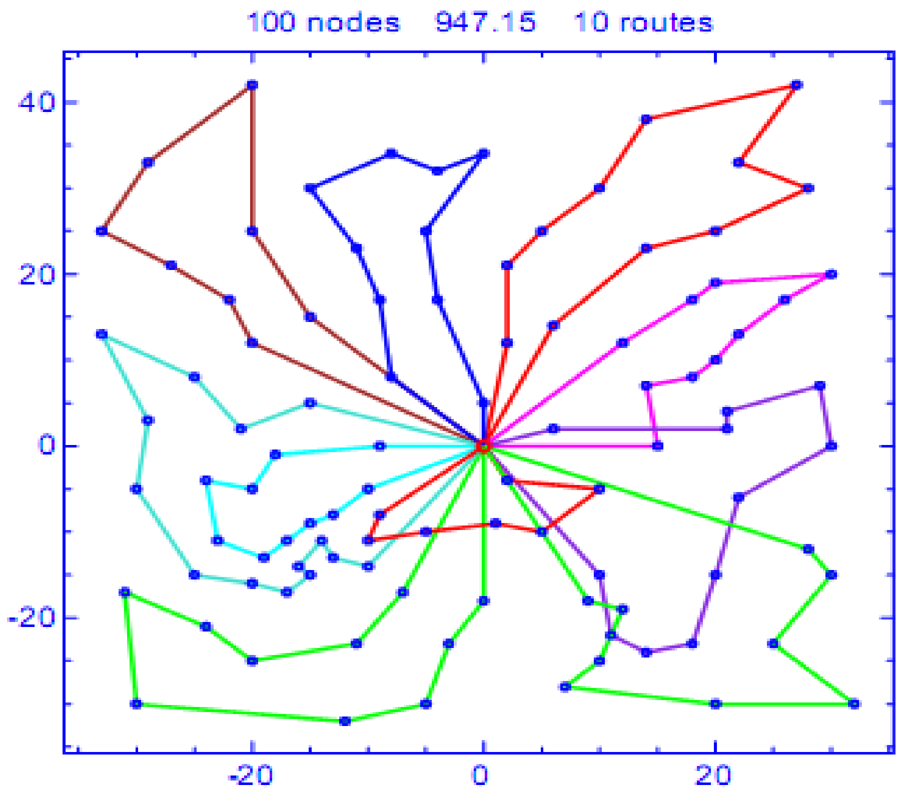
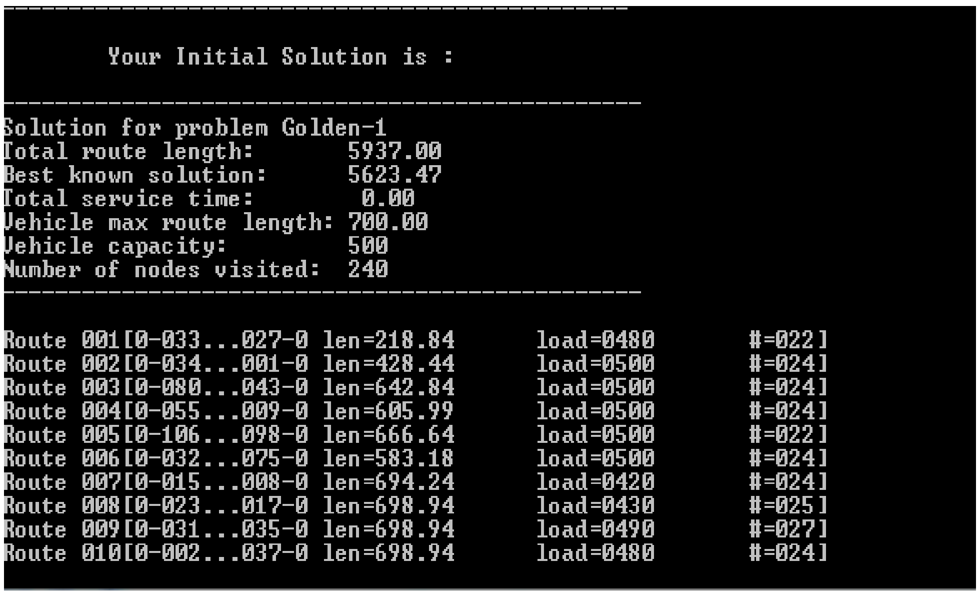


Fig. 4.10(b): Plot of routes for Christofides_03 after optimization by EJ algo.

4.1.11. Golden_01.vrp – Initial Solution.



```
------------------------------------------------

       Your Initial Solution is :

------------------------------------------------
Solution for problem Golden-1
Total route length:       5937.00
Best known solution:      5623.47
Total service time:        0.00
Vehicle max route length: 700.00
Vehicle capacity:         500
Number of nodes visited:  240
------------------------------------------------

Route 001[0-033...027-0 len=218.84      load=0480       #=022]
Route 002[0-034...001-0 len=428.44      load=0500       #=024]
Route 003[0-080...043-0 len=642.84      load=0500       #=024]
Route 004[0-055...009-0 len=605.99      load=0500       #=024]
Route 005[0-106...098-0 len=666.64      load=0500       #=022]
Route 006[0-032...075-0 len=583.18      load=0500       #=024]
Route 007[0-015...008-0 len=694.24      load=0420       #=024]
Route 008[0-023...017-0 len=698.94      load=0430       #=025]
Route 009[0-031...035-0 len=698.94      load=0490       #=027]
Route 010[0-002...037-0 len=698.94      load=0480       #=024]
```

Fig. 4.11(a) Route details for Golden_01 after applying Clarke-Wright algorithm.

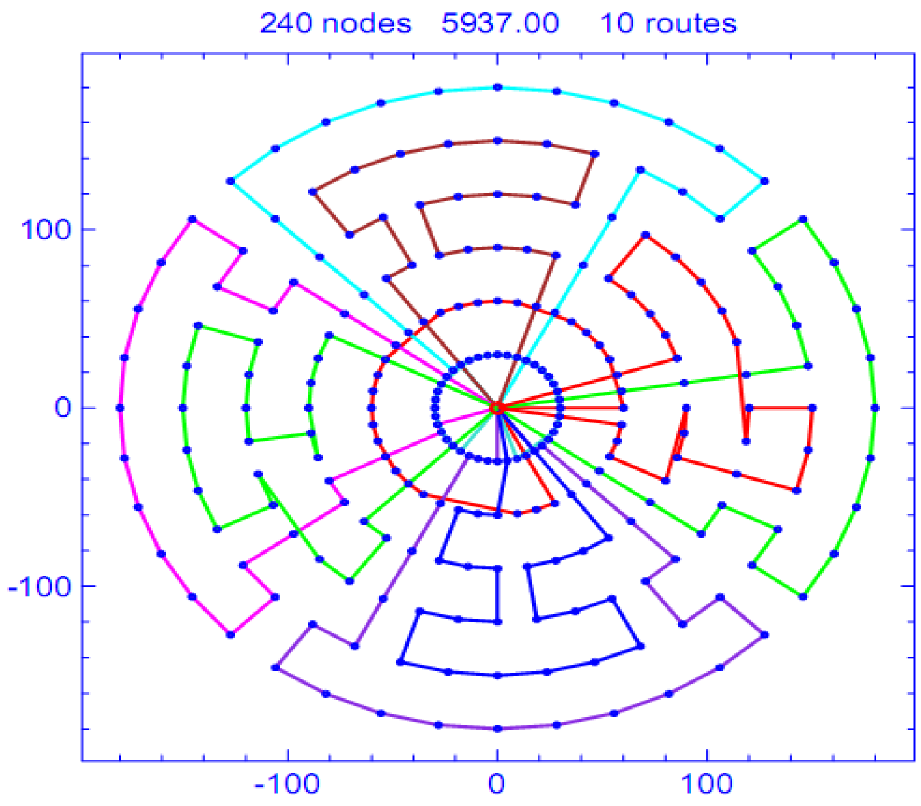
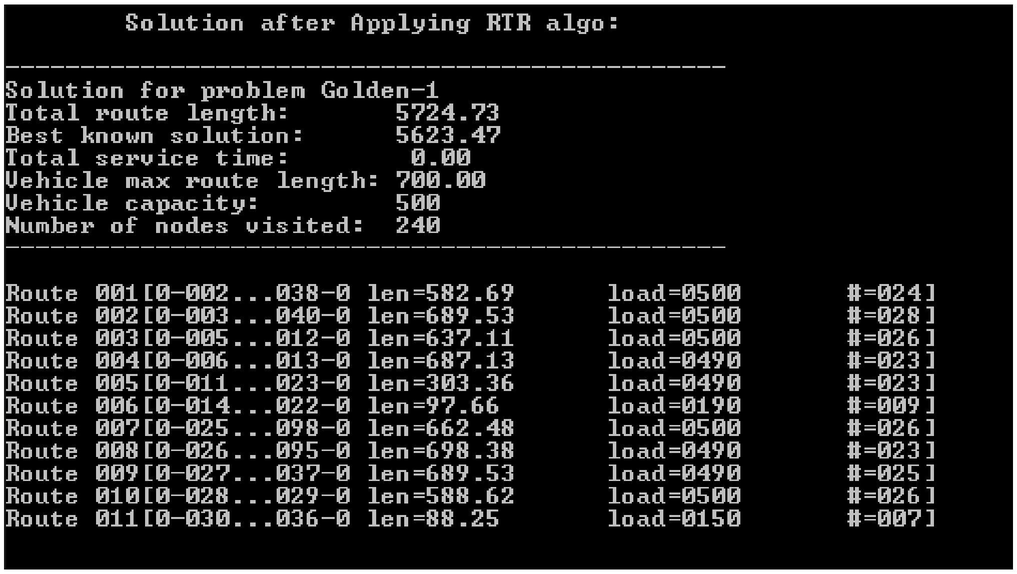


Fig. 4.11(b): Plot of routes for Golden_01.

4.1.12. Golden_01.vrp – After applying RTR algorithm.



```
        Solution after Applying RTR algo:

------------------------------------------------
Solution for problem Golden-1
Total route length:          5724.73
Best known solution:         5623.47
Total service time:            0.00
Vehicle max route length: 700.00
Vehicle capacity:            500
Number of nodes visited:   240
------------------------------------------------

Route 001[0-002...038-0 len=582.69      load=0500       #=024]
Route 002[0-003...040-0 len=689.53      load=0500       #=028]
Route 003[0-005...012-0 len=637.11      load=0500       #=026]
Route 004[0-006...013-0 len=687.13      load=0490       #=023]
Route 005[0-011...023-0 len=303.36      load=0490       #=023]
Route 006[0-014...022-0 len=97.66       load=0190       #=009]
Route 007[0-025...098-0 len=662.48      load=0500       #=026]
Route 008[0-026...095-0 len=698.38      load=0490       #=023]
Route 009[0-027...037-0 len=689.53      load=0490       #=025]
Route 010[0-028...029-0 len=588.62      load=0500       #=026]
Route 011[0-030...036-0 len=88.25       load=0150       #=007]
```

Fig. 4.12(a) Optimized route details for Golden_01 after applying RTR algorithm.

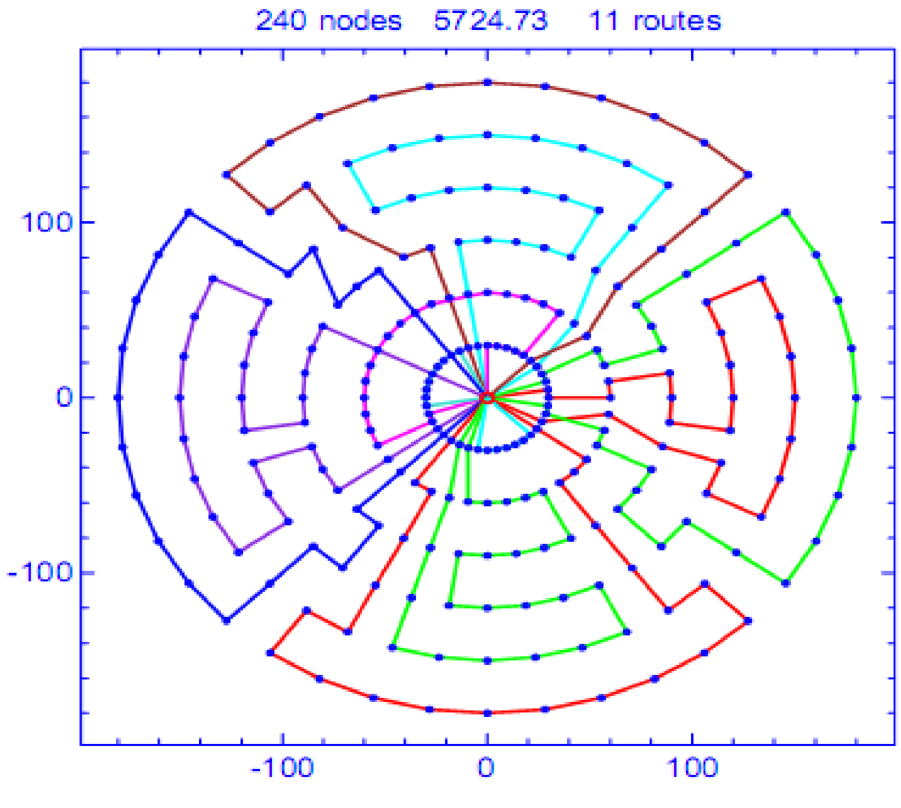
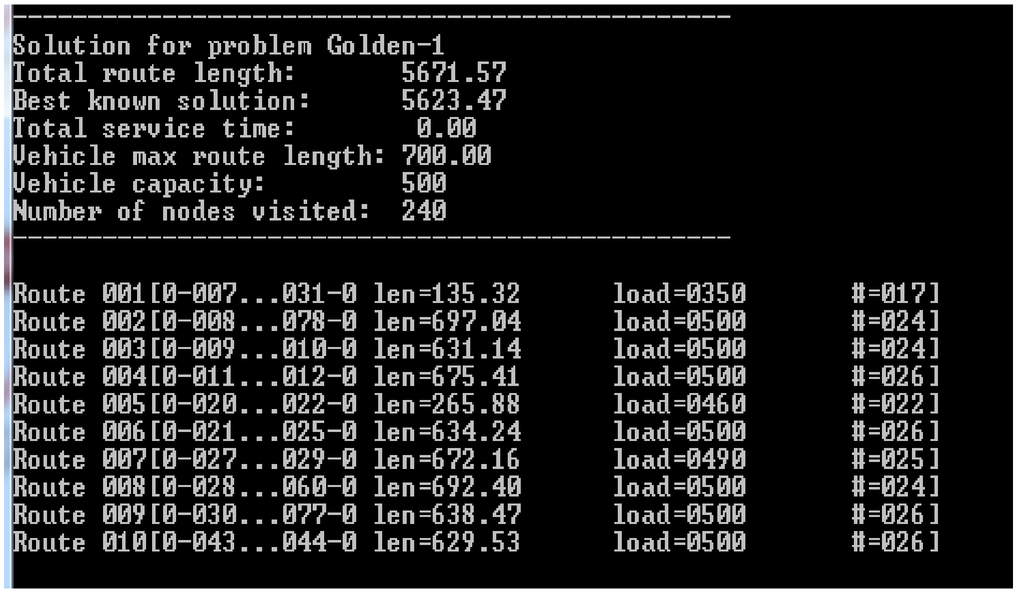


Fig. 4.12(b): Plot of routes for Golden_01 after optimization by RTR algo.

4.1.13. Golden_01.vrp – After applying New_RTR algorithm.



```
-----------------------------------------------
Solution for problem Golden-1
Total route length:        5671.57
Best known solution:       5623.47
Total service time:          0.00
Vehicle max route length: 700.00
Vehicle capacity:          500
Number of nodes visited:   240
-----------------------------------------------

Route 001[0-007...031-0 len=135.32      load=0350       #=017]
Route 002[0-008...078-0 len=697.04      load=0500       #=024]
Route 003[0-009...010-0 len=631.14      load=0500       #=024]
Route 004[0-011...012-0 len=675.41      load=0500       #=026]
Route 005[0-020...022-0 len=265.88      load=0460       #=022]
Route 006[0-021...025-0 len=634.24      load=0500       #=026]
Route 007[0-027...029-0 len=672.16      load=0490       #=025]
Route 008[0-028...060-0 len=692.40      load=0500       #=024]
Route 009[0-030...077-0 len=638.47      load=0500       #=026]
Route 010[0-043...044-0 len=629.53      load=0500       #=026]
```

Fig. 4.13(a) Optimized route details for Golden_01 after applying New_RTR algorithm.

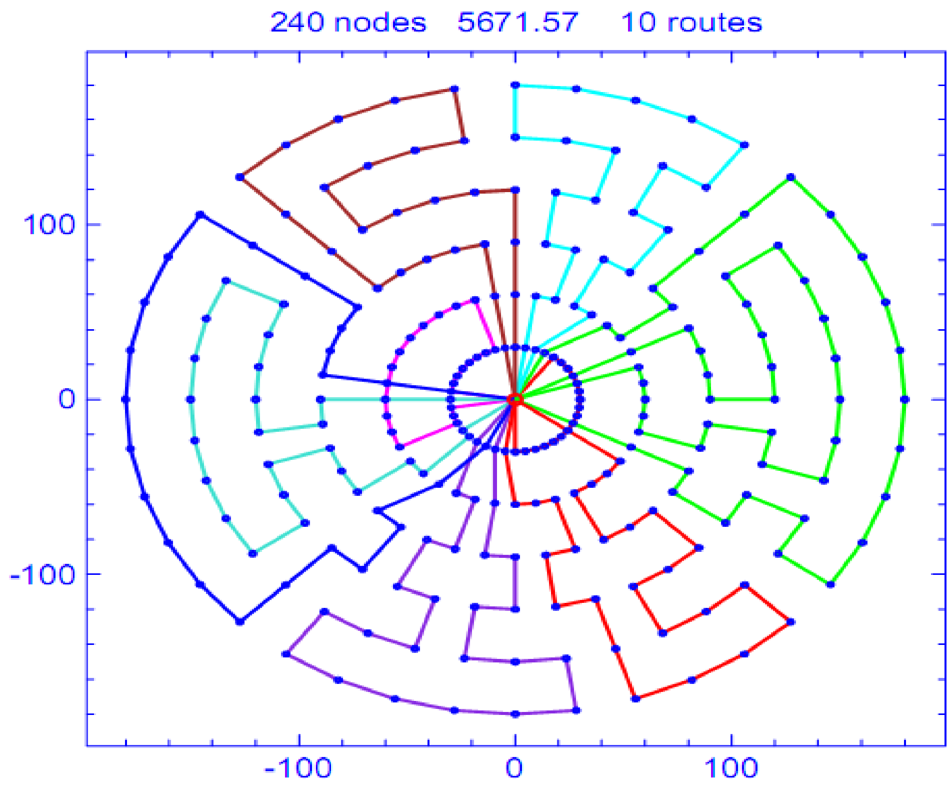
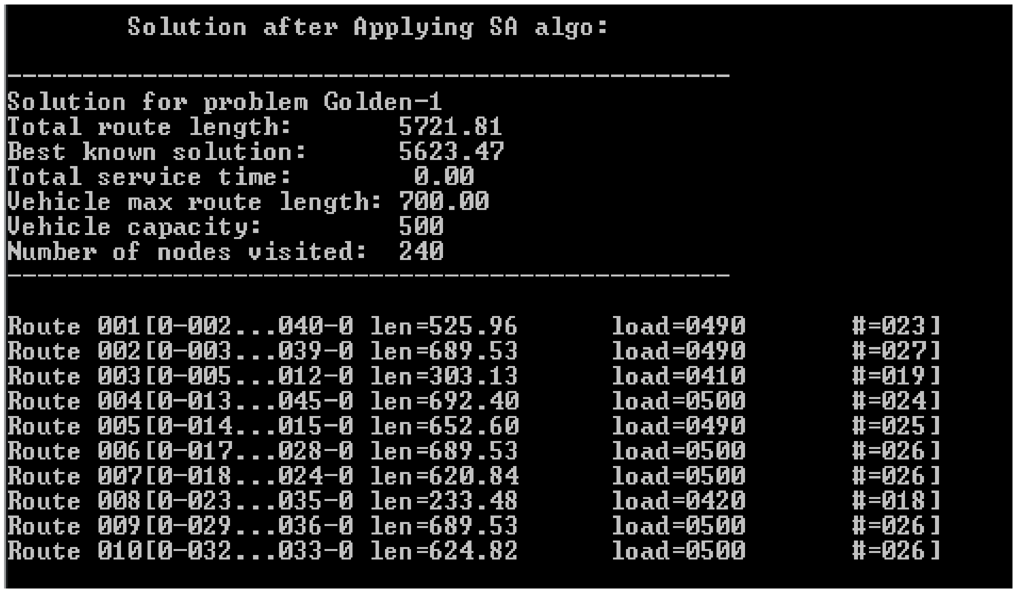


Fig. 4.13(b): Plot of routes for Golden_01 after optimization by New_RTR algo.

4.1.14. Golden_01.vrp – After applying SA algorithm.



```
        Solution after Applying SA algo:

------------------------------------------------
Solution for problem Golden-1
Total route length:        5721.81
Best known solution:       5623.47
Total service time:         0.00
Vehicle max route length: 700.00
Vehicle capacity:          500
Number of nodes visited:   240
------------------------------------------------

Route 001[0-002...040-0 len=525.96      load=0490       #=023]
Route 002[0-003...039-0 len=689.53      load=0490       #=027]
Route 003[0-005...012-0 len=303.13      load=0410       #=019]
Route 004[0-013...045-0 len=692.40      load=0500       #=024]
Route 005[0-014...015-0 len=652.60      load=0490       #=025]
Route 006[0-017...028-0 len=689.53      load=0500       #=026]
Route 007[0-018...024-0 len=620.84      load=0500       #=026]
Route 008[0-023...035-0 len=233.48      load=0420       #=018]
Route 009[0-029...036-0 len=689.53      load=0500       #=026]
Route 010[0-032...033-0 len=624.82      load=0500       #=026]
```

Fig. 4.14(a) Optimized route details for Golden_01 after applying SA algorithm.

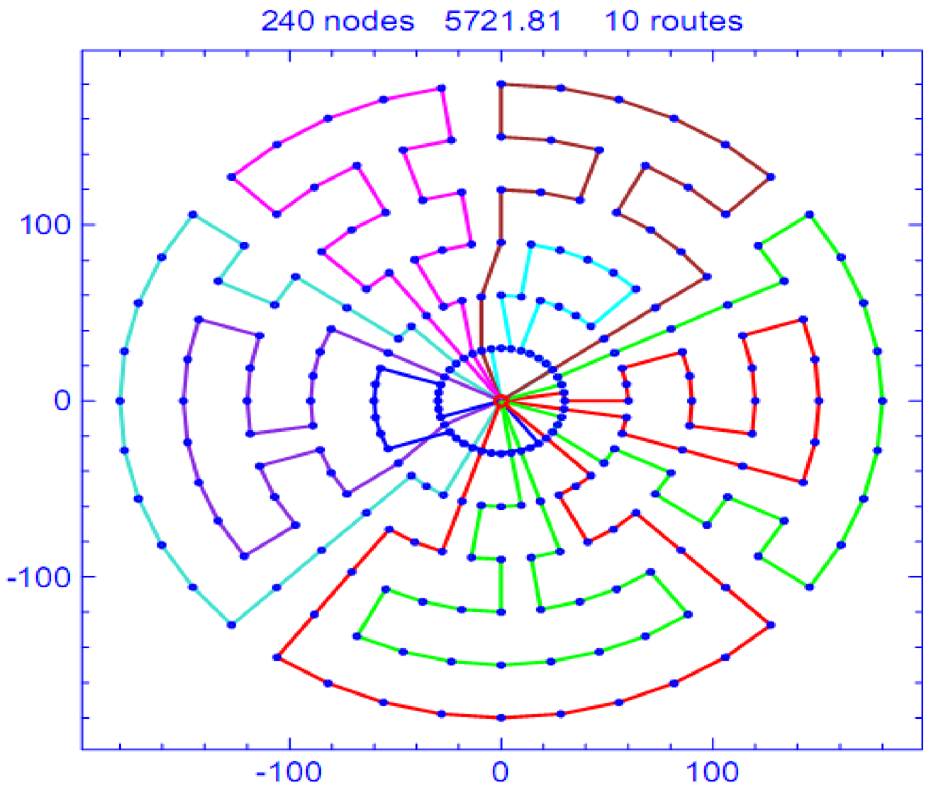
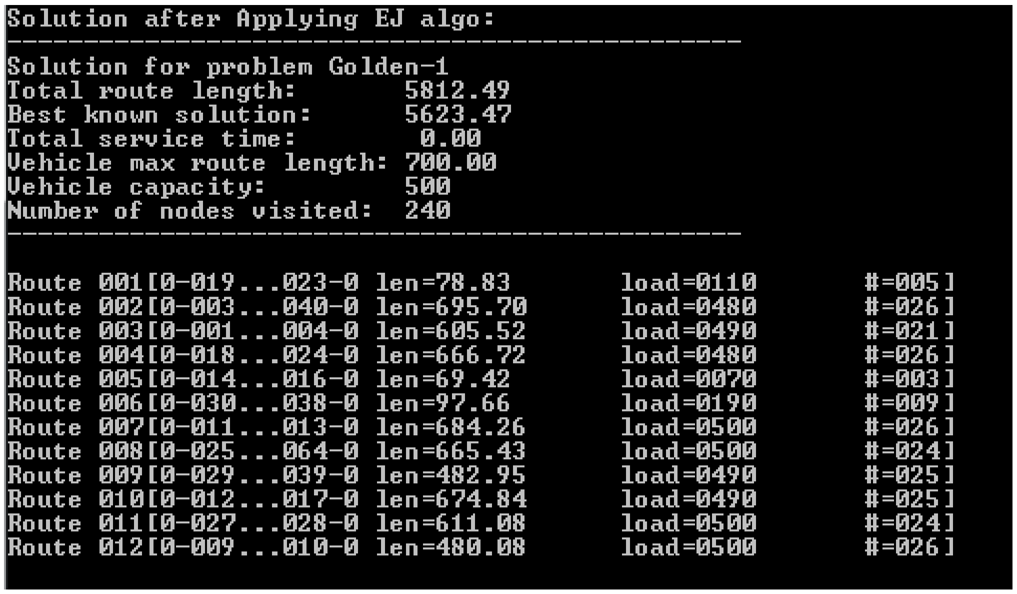


Fig. 4.14(b): Plot of routes for Golden_01 after optimization by SA algo.

4.1.15. Golden_01.vrp – After applying EJ algorithm.



```
Solution after Applying EJ algo:
------------------------------------------------

Solution for problem Golden-1
Total route length:        5812.49
Best known solution:       5623.47
Total service time:         0.00
Vehicle max route length: 700.00
Vehicle capacity:          500
Number of nodes visited:   240
------------------------------------------------

Route 001[0-019...023-0 len=78.83       load=0110        #=005]
Route 002[0-003...040-0 len=695.70      load=0480        #=026]
Route 003[0-001...004-0 len=605.52      load=0490        #=021]
Route 004[0-018...024-0 len=666.72      load=0480        #=026]
Route 005[0-014...016-0 len=69.42       load=0070        #=003]
Route 006[0-030...038-0 len=97.66       load=0190        #=009]
Route 007[0-011...013-0 len=684.26      load=0500        #=026]
Route 008[0-025...064-0 len=665.43      load=0500        #=024]
Route 009[0-029...039-0 len=482.95      load=0490        #=025]
Route 010[0-012...017-0 len=674.84      load=0490        #=025]
Route 011[0-027...028-0 len=611.08      load=0500        #=024]
Route 012[0-009...010-0 len=480.08      load=0500        #=026]
```

Fig. 4.15(a) Optimized route details for Golden_01 after applying EJ algorithm.

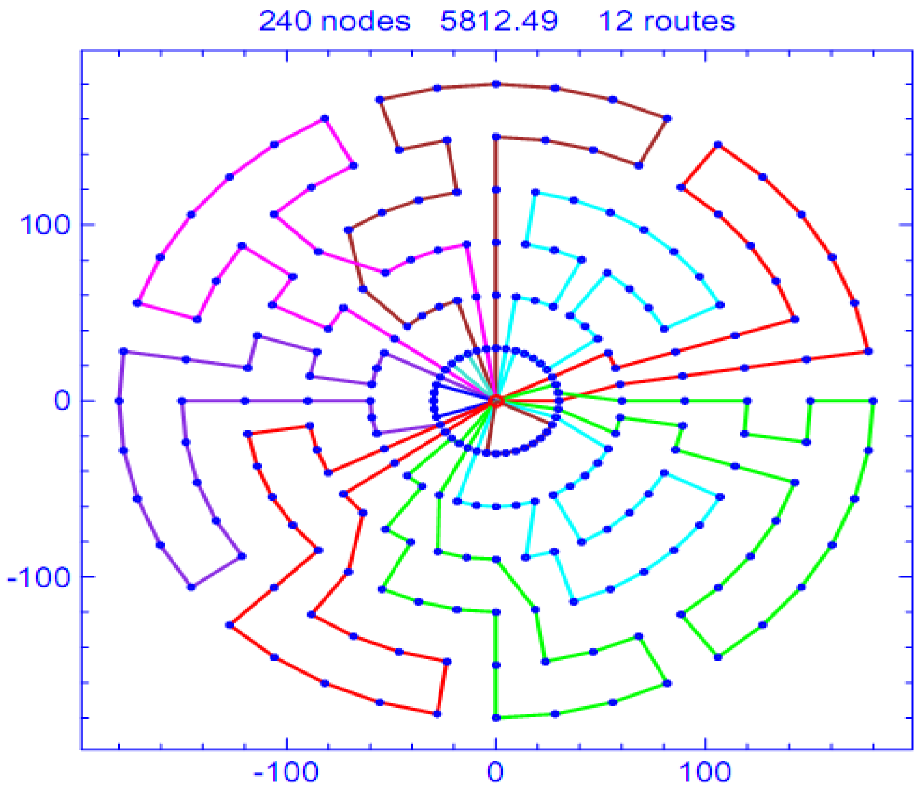
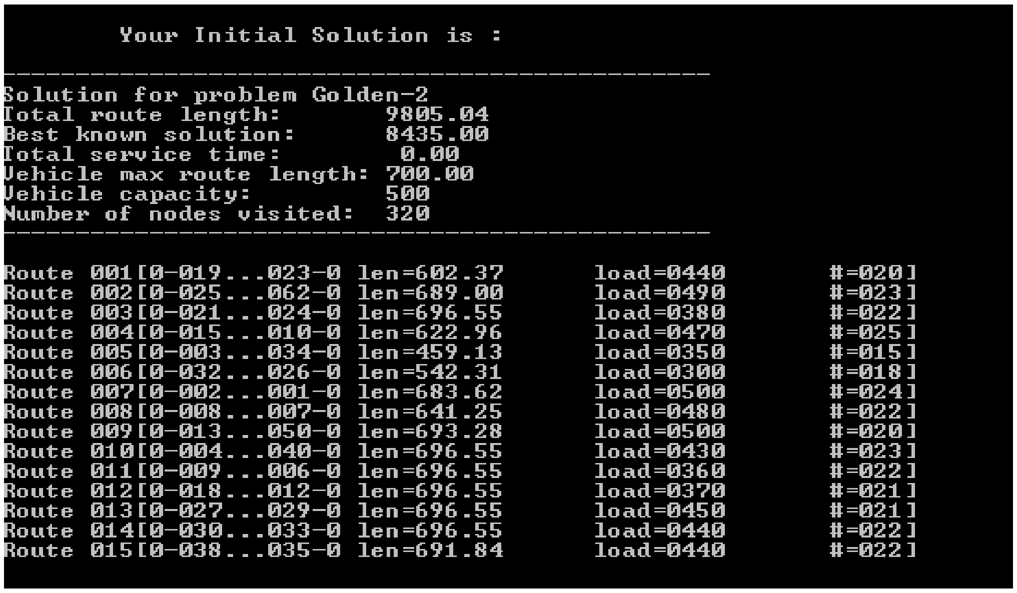


Fig. 4.15(b): Plot of routes for Golden_01 after optimization by EJ algo.

4.1.16. Golden_02.vrp – Initial Solution.



```
        Your Initial Solution is :

------------------------------------------------
Solution for problem Golden-2
Total route length:        9805.04
Best known solution:       8435.00
Total service time:          0.00
Vehicle max route length: 700.00
Vehicle capacity:          500
Number of nodes visited:   320
------------------------------------------------

Route 001[0-019...023-0 len=602.37       load=0440        #=020]
Route 002[0-025...062-0 len=689.00       load=0490        #=023]
Route 003[0-021...024-0 len=696.55       load=0380        #=022]
Route 004[0-015...010-0 len=622.96       load=0470        #=025]
Route 005[0-003...034-0 len=459.13       load=0350        #=015]
Route 006[0-032...026-0 len=542.31       load=0300        #=018]
Route 007[0-002...001-0 len=683.62       load=0500        #=024]
Route 008[0-008...007-0 len=641.25       load=0480        #=022]
Route 009[0-013...050-0 len=693.28       load=0500        #=020]
Route 010[0-004...040-0 len=696.55       load=0430        #=023]
Route 011[0-009...006-0 len=696.55       load=0360        #=022]
Route 012[0-018...012-0 len=696.55       load=0370        #=021]
Route 013[0-027...029-0 len=696.55       load=0450        #=021]
Route 014[0-030...033-0 len=696.55       load=0440        #=022]
Route 015[0-038...035-0 len=691.84       load=0440        #=022]
```

Fig. 4.16(a) Route details for Golden_02 after applying Clarke-Wright algorithm.

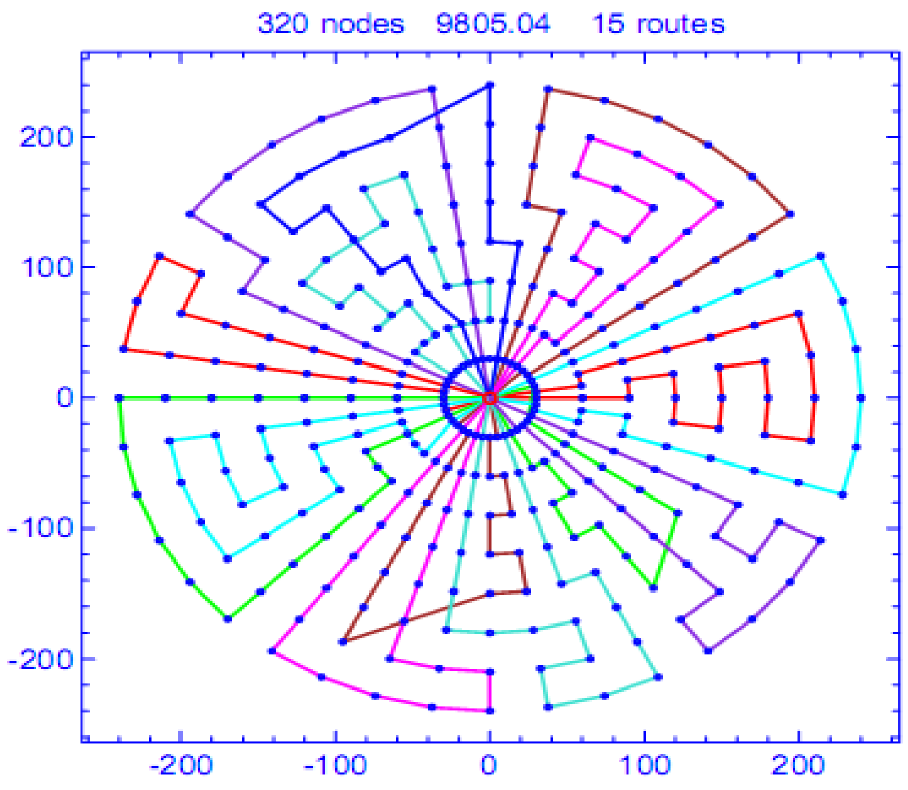
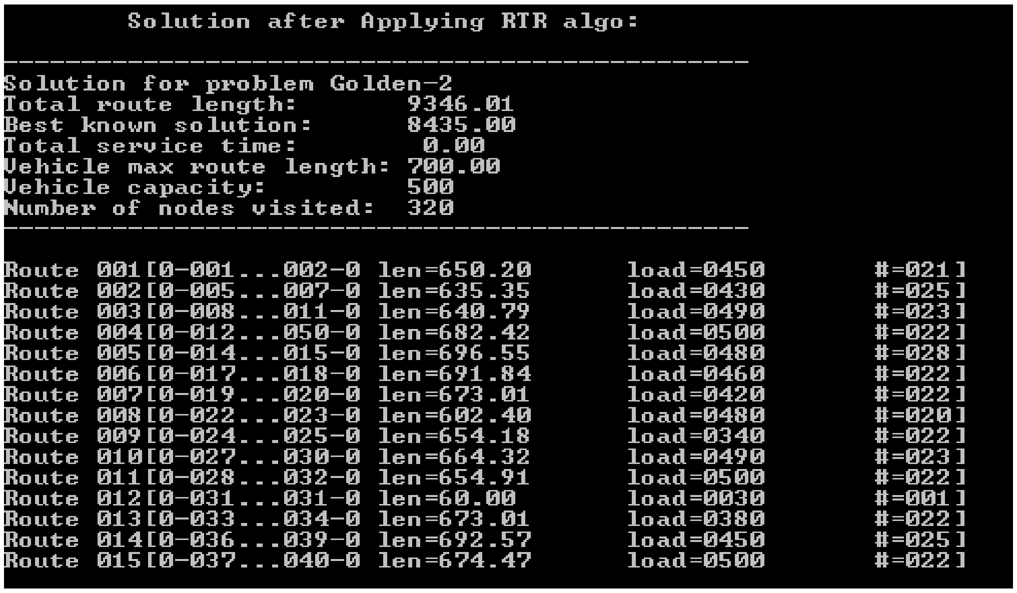


Fig. 4.16(b): Plot of routes for Golden_02.

4.1.17. Golden_02.vrp – After applying RTR algorithm.



```
        Solution after Applying RTR algo:

------------------------------------------------
Solution for problem Golden-2
Total route length:        9346.01
Best known solution:       8435.00
Total service time:         0.00
Vehicle max route length: 700.00
Vehicle capacity:          500
Number of nodes visited:   320
------------------------------------------------

Route 001[0-001...002-0 len=650.20      load=0450       #=021]
Route 002[0-005...007-0 len=635.35      load=0430       #=025]
Route 003[0-008...011-0 len=640.79      load=0490       #=023]
Route 004[0-012...050-0 len=682.42      load=0500       #=022]
Route 005[0-014...015-0 len=696.55      load=0480       #=028]
Route 006[0-017...018-0 len=691.84      load=0460       #=022]
Route 007[0-019...020-0 len=673.01      load=0420       #=022]
Route 008[0-022...023-0 len=602.40      load=0480       #=020]
Route 009[0-024...025-0 len=654.18      load=0340       #=022]
Route 010[0-027...030-0 len=664.32      load=0490       #=023]
Route 011[0-028...032-0 len=654.91      load=0500       #=022]
Route 012[0-031...031-0 len=60.00       load=0030       #=001]
Route 013[0-033...034-0 len=673.01      load=0380       #=022]
Route 014[0-036...039-0 len=692.57      load=0450       #=025]
Route 015[0-037...040-0 len=674.47      load=0500       #=022]
```

Fig. 4.17(a) Optimized route details for Golden_02 after applying RTR algorithm.

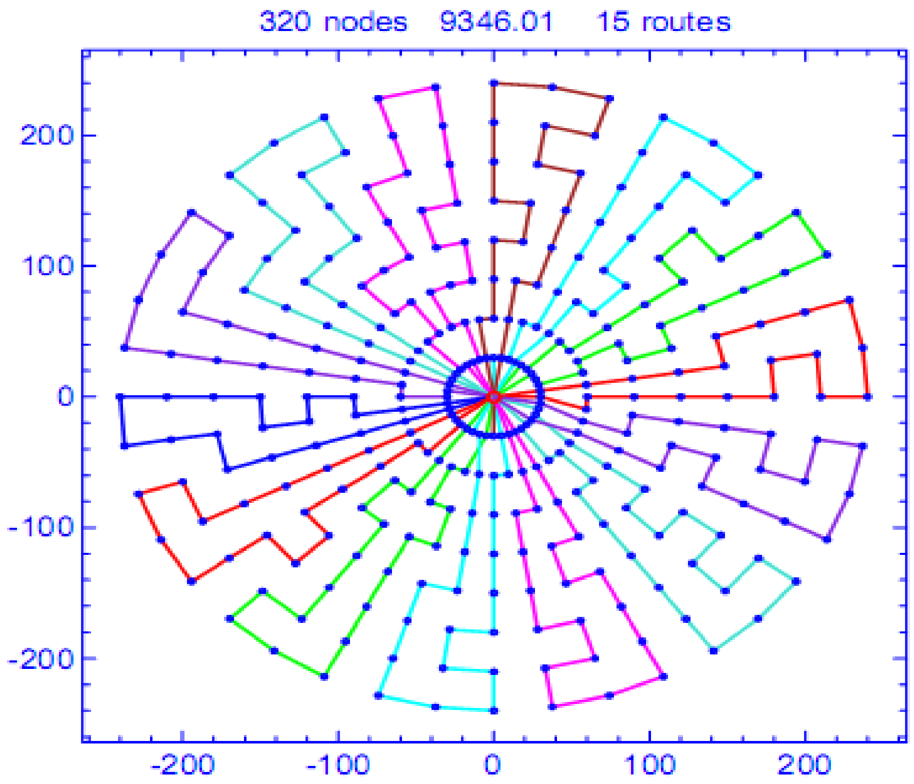
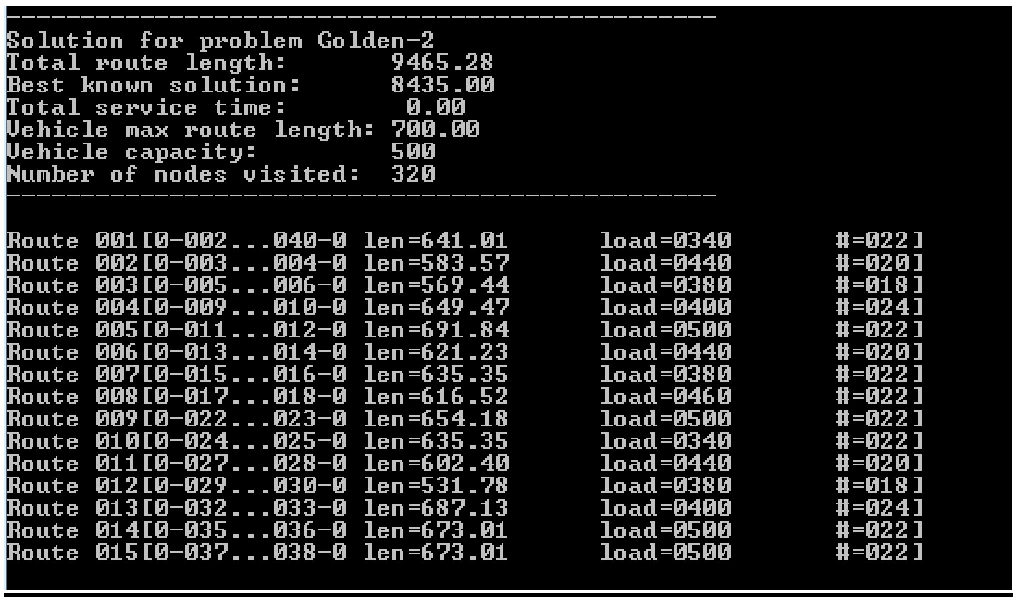


Fig. 4.17(b): Plot of routes for Golden_02 after optimization by RTR algo.

4.1.18. Golden_02.vrp – After applying New_RTR algorithm.



```
-----------------------------------------------
Solution for problem Golden-2
Total route length:        9465.28
Best known solution:       8435.00
Total service time:         0.00
Vehicle max route length: 700.00
Vehicle capacity:          500
Number of nodes visited:   320
-----------------------------------------------

Route 001[0-002...040-0 len=641.01      load=0340       #=022]
Route 002[0-003...004-0 len=583.57      load=0440       #=020]
Route 003[0-005...006-0 len=569.44      load=0380       #=018]
Route 004[0-009...010-0 len=649.47      load=0400       #=024]
Route 005[0-011...012-0 len=691.84      load=0500       #=022]
Route 006[0-013...014-0 len=621.23      load=0440       #=020]
Route 007[0-015...016-0 len=635.35      load=0380       #=022]
Route 008[0-017...018-0 len=616.52      load=0460       #=022]
Route 009[0-022...023-0 len=654.18      load=0500       #=022]
Route 010[0-024...025-0 len=635.35      load=0340       #=022]
Route 011[0-027...028-0 len=602.40      load=0440       #=020]
Route 012[0-029...030-0 len=531.78      load=0380       #=018]
Route 013[0-032...033-0 len=687.13      load=0400       #=024]
Route 014[0-035...036-0 len=673.01      load=0500       #=022]
Route 015[0-037...038-0 len=673.01      load=0500       #=022]
```

Fig. 4.18(a) Optimized route details for Golden_02 after applying New_RTR algorithm.

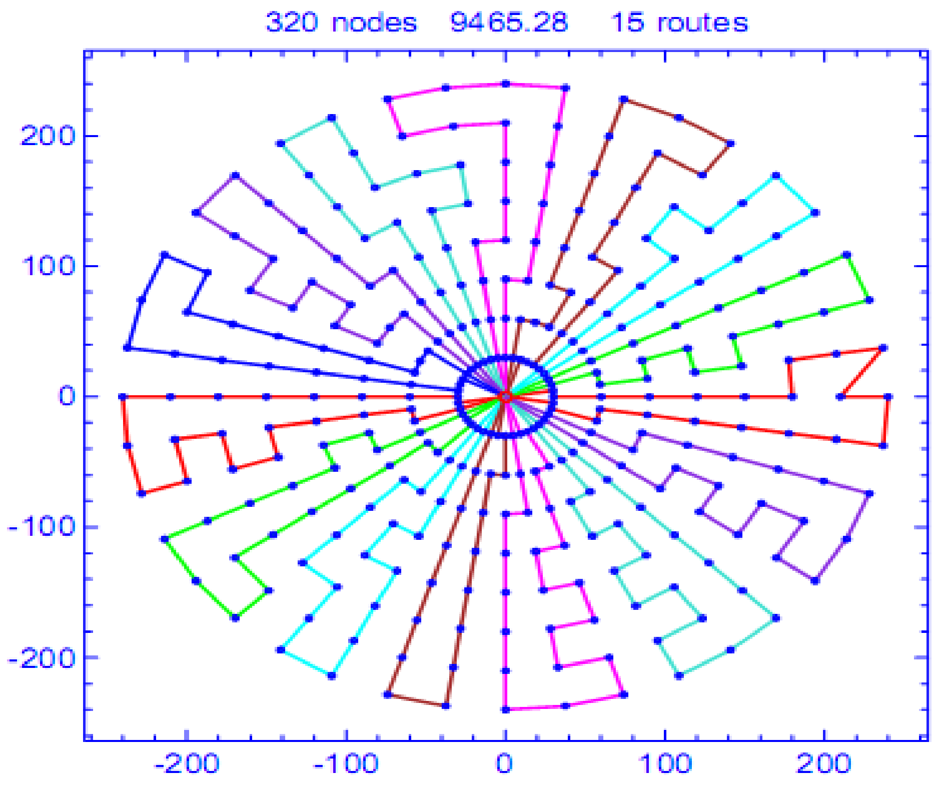
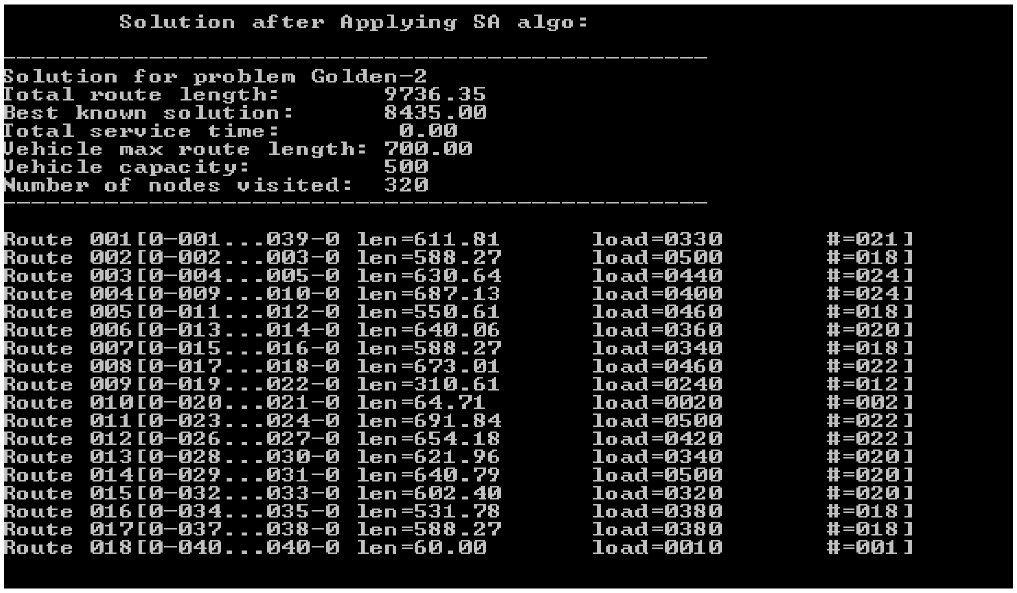


Fig. 4.18(b): Plot of routes for Golden_02 after optimization by RTR algo.

4.1.19. Golden_02.vrp – After applying SA algorithm.



```
        Solution after Applying SA algo:

-------------------------------------------------
Solution for problem Golden-2
Total route length:        9736.35
Best known solution:       8435.00
Total service time:          0.00
Vehicle max route length: 700.00
Vehicle capacity:          500
Number of nodes visited:   320
-------------------------------------------------

Route 001[0-001...039-0 len=611.81      load=0330       #=021]
Route 002[0-002...003-0 len=588.27      load=0500       #=018]
Route 003[0-004...005-0 len=630.64      load=0440       #=024]
Route 004[0-009...010-0 len=687.13      load=0400       #=024]
Route 005[0-011...012-0 len=550.61      load=0460       #=018]
Route 006[0-013...014-0 len=640.06      load=0360       #=020]
Route 007[0-015...016-0 len=588.27      load=0340       #=018]
Route 008[0-017...018-0 len=673.01      load=0460       #=022]
Route 009[0-019...022-0 len=310.61      load=0240       #=012]
Route 010[0-020...021-0 len=64.71       load=0020       #=002]
Route 011[0-023...024-0 len=691.84      load=0500       #=022]
Route 012[0-026...027-0 len=654.18      load=0420       #=022]
Route 013[0-028...030-0 len=621.96      load=0340       #=020]
Route 014[0-029...031-0 len=640.79      load=0500       #=020]
Route 015[0-032...033-0 len=602.40      load=0320       #=020]
Route 016[0-034...035-0 len=531.78      load=0380       #=018]
Route 017[0-037...038-0 len=588.27      load=0380       #=018]
Route 018[0-040...040-0 len=60.00       load=0010       #=001]
```

Fig. 4.19(a) Optimized route details for Golden_02 after applying SA algorithm.

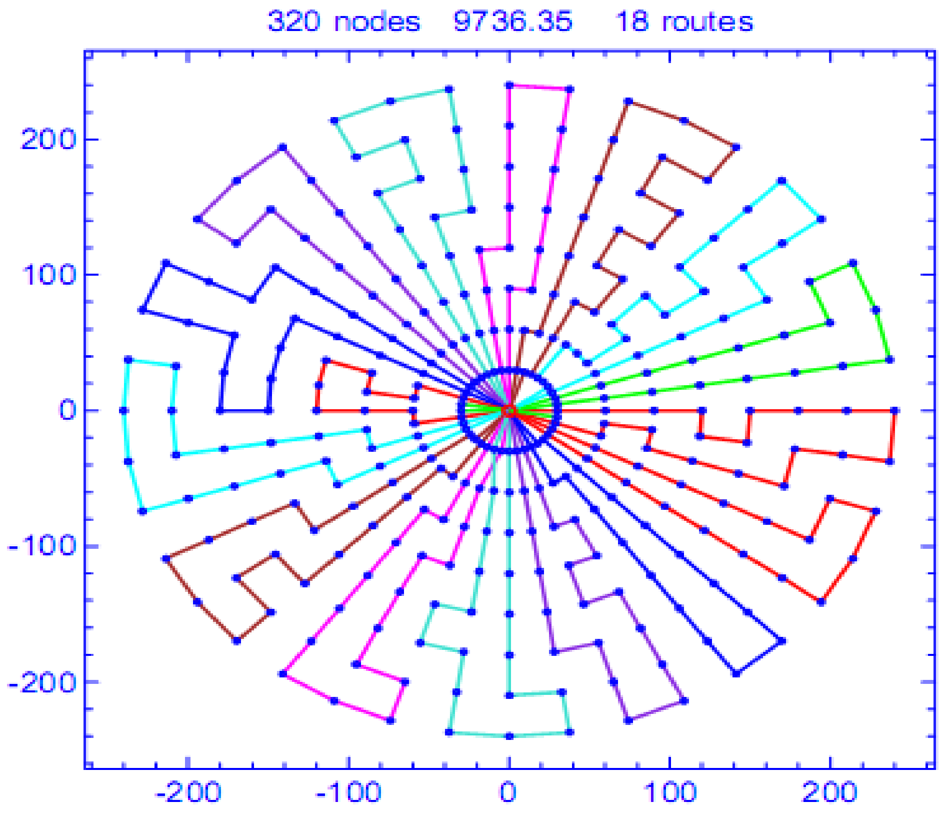
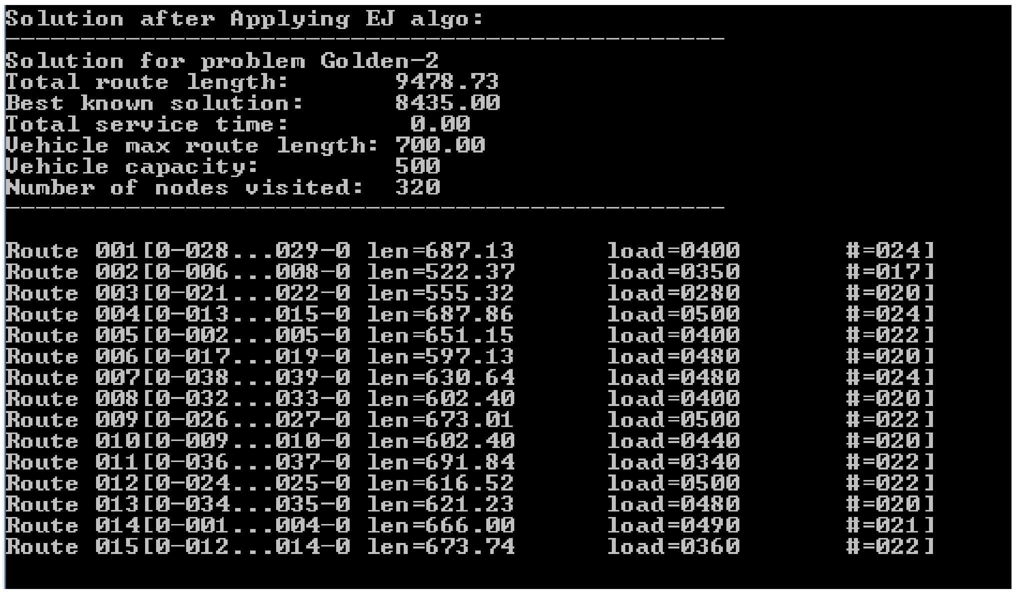


Fig. 4.19(b): Plot of routes for Golden_02 after optimization by SA algo.

4.1.20. Golden_02.vrp – After applying EJ algorithm.



```
Solution after Applying EJ algo:
------------------------------------------------

Solution for problem Golden-2
Total route length:        9478.73
Best known solution:       8435.00
Total service time:          0.00
Vehicle max route length: 700.00
Vehicle capacity:          500
Number of nodes visited:   320
------------------------------------------------

Route 001[0-028...029-0 len=687.13      load=0400        #=024]
Route 002[0-006...008-0 len=522.37      load=0350        #=017]
Route 003[0-021...022-0 len=555.32      load=0280        #=020]
Route 004[0-013...015-0 len=687.86      load=0500        #=024]
Route 005[0-002...005-0 len=651.15      load=0400        #=022]
Route 006[0-017...019-0 len=597.13      load=0480        #=020]
Route 007[0-038...039-0 len=630.64      load=0480        #=024]
Route 008[0-032...033-0 len=602.40      load=0400        #=020]
Route 009[0-026...027-0 len=673.01      load=0500        #=022]
Route 010[0-009...010-0 len=602.40      load=0440        #=020]
Route 011[0-036...037-0 len=691.84      load=0340        #=022]
Route 012[0-024...025-0 len=616.52      load=0500        #=022]
Route 013[0-034...035-0 len=621.23      load=0480        #=020]
Route 014[0-001...004-0 len=666.00      load=0490        #=021]
Route 015[0-012...014-0 len=673.74      load=0360        #=022]
```

Fig. 4.20(a) Optimized route details for Golden_02 after applying EJ algorithm.

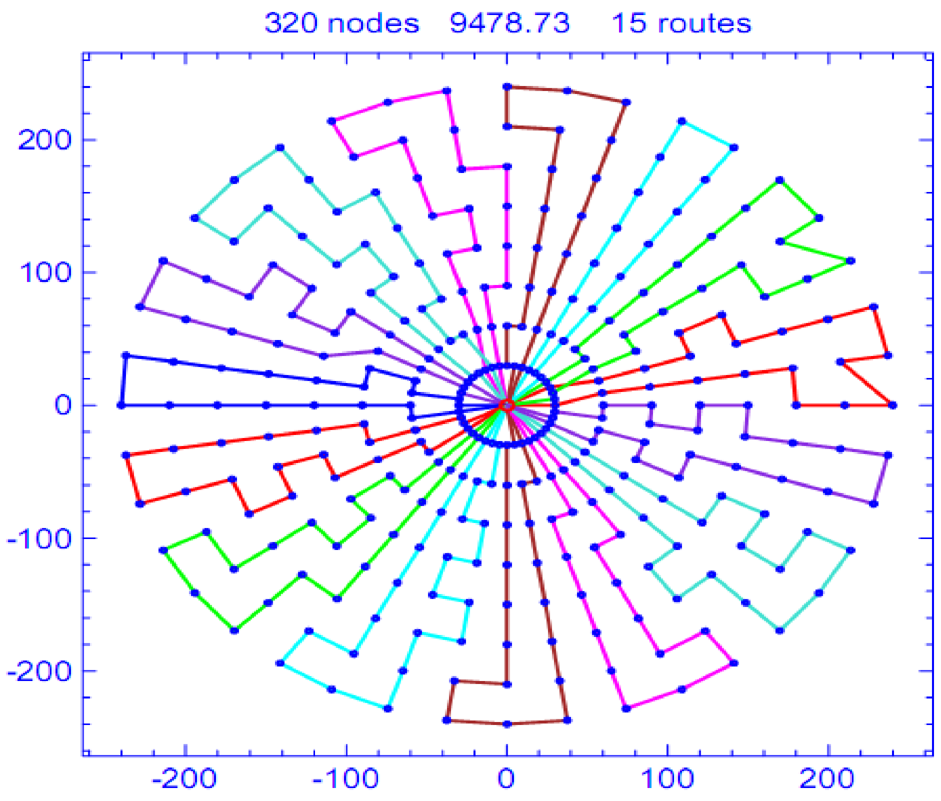
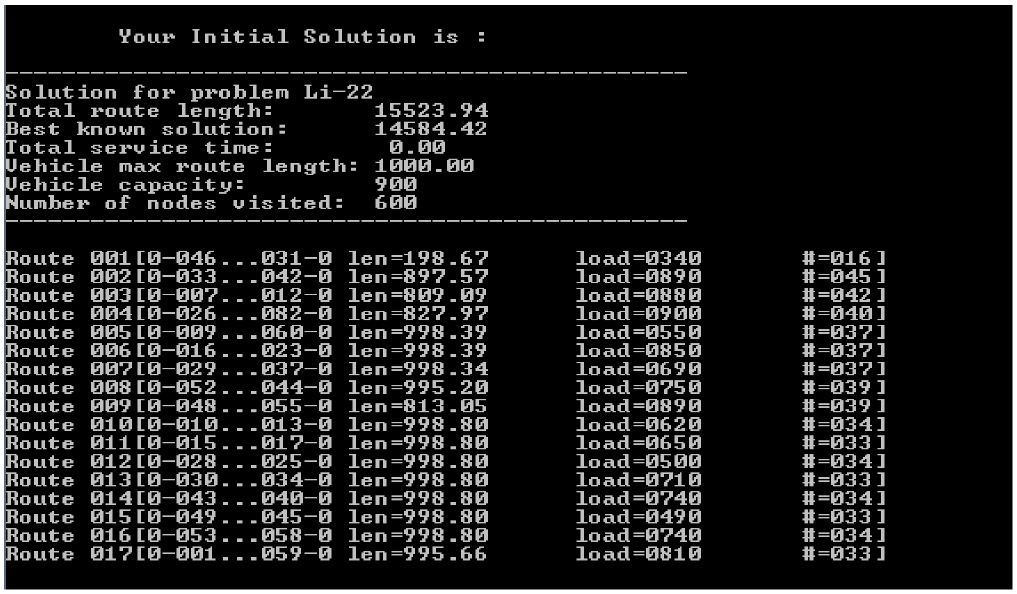


Fig. 4.20(b): Plot of routes for Golden_02 after optimization by EJ algo.

4.1.21. Li_22.vrp – Initial Solution.



```
        Your Initial Solution is :

------------------------------------------------
Solution for problem Li-22
Total route length:        15523.94
Best known solution:       14584.42
Total service time:         0.00
Vehicle max route length: 1000.00
Vehicle capacity:          900
Number of nodes visited:   600
------------------------------------------------

Route 001[0-046...031-0 len=198.67      load=0340        #=016]
Route 002[0-033...042-0 len=897.57      load=0890        #=045]
Route 003[0-007...012-0 len=809.09      load=0880        #=042]
Route 004[0-026...082-0 len=827.97      load=0900        #=040]
Route 005[0-009...060-0 len=998.39      load=0550        #=037]
Route 006[0-016...023-0 len=998.39      load=0850        #=037]
Route 007[0-029...037-0 len=998.34      load=0690        #=037]
Route 008[0-052...044-0 len=995.20      load=0750        #=039]
Route 009[0-048...055-0 len=813.05      load=0890        #=039]
Route 010[0-010...013-0 len=998.80      load=0620        #=034]
Route 011[0-015...017-0 len=998.80      load=0650        #=033]
Route 012[0-028...025-0 len=998.80      load=0500        #=034]
Route 013[0-030...034-0 len=998.80      load=0710        #=033]
Route 014[0-043...040-0 len=998.80      load=0740        #=034]
Route 015[0-049...045-0 len=998.80      load=0490        #=033]
Route 016[0-053...058-0 len=998.80      load=0740        #=034]
Route 017[0-001...059-0 len=995.66      load=0810        #=033]
```

Fig. 4.21(a) Route details for Li_22 after applying Clarke-Wright algorithm.

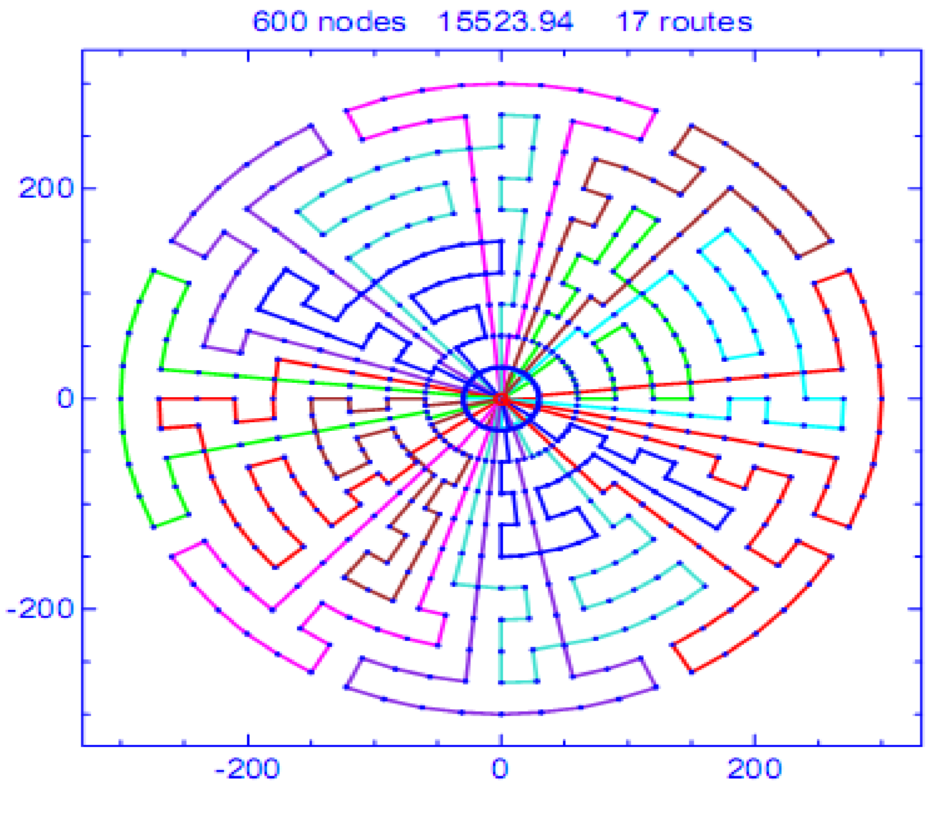
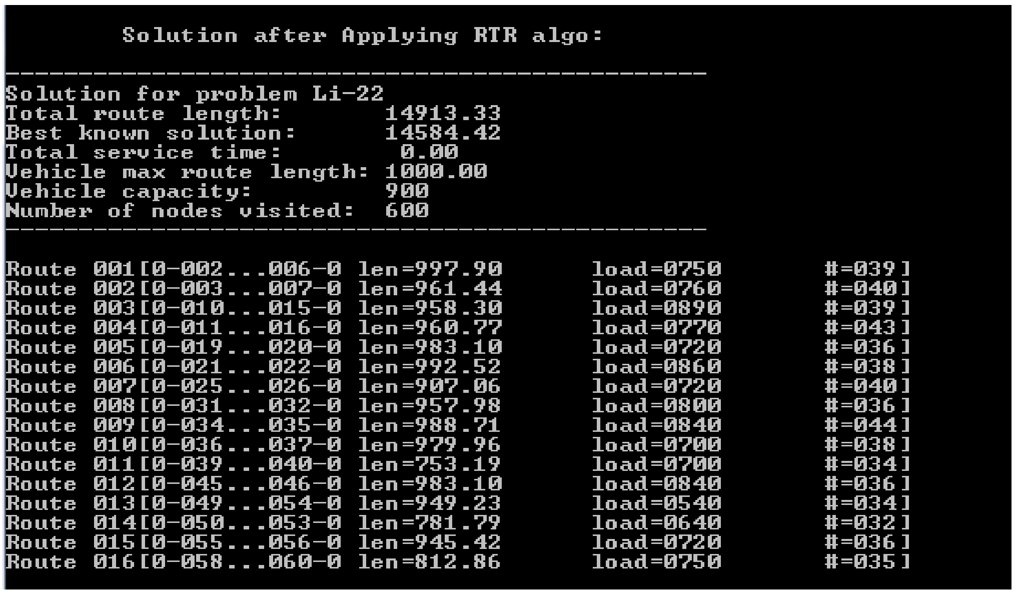


Fig. 4.21(b): Plot of routes for Li_22.

4.1.22. Li_22.vrp – After applying RTR algorithm.



```
        Solution after Applying RTR algo:

------------------------------------------------
Solution for problem Li-22
Total route length:       14913.33
Best known solution:      14584.42
Total service time:         0.00
Vehicle max route length: 1000.00
Vehicle capacity:         900
Number of nodes visited:  600
------------------------------------------------

Route 001[0-002...006-0 len=997.90      load=0750       #=039]
Route 002[0-003...007-0 len=961.44      load=0760       #=040]
Route 003[0-010...015-0 len=958.30      load=0890       #=039]
Route 004[0-011...016-0 len=960.77      load=0770       #=043]
Route 005[0-019...020-0 len=983.10      load=0720       #=036]
Route 006[0-021...022-0 len=992.52      load=0860       #=038]
Route 007[0-025...026-0 len=907.06      load=0720       #=040]
Route 008[0-031...032-0 len=957.98      load=0800       #=036]
Route 009[0-034...035-0 len=988.71      load=0840       #=044]
Route 010[0-036...037-0 len=979.96      load=0700       #=038]
Route 011[0-039...040-0 len=753.19      load=0700       #=034]
Route 012[0-045...046-0 len=983.10      load=0840       #=036]
Route 013[0-049...054-0 len=949.23      load=0540       #=034]
Route 014[0-050...053-0 len=781.79      load=0640       #=032]
Route 015[0-055...056-0 len=945.42      load=0720       #=036]
Route 016[0-058...060-0 len=812.86      load=0750       #=035]
```

Fig. 4.22(a) Optimized route details for Li_22 after applying RTR algorithm.

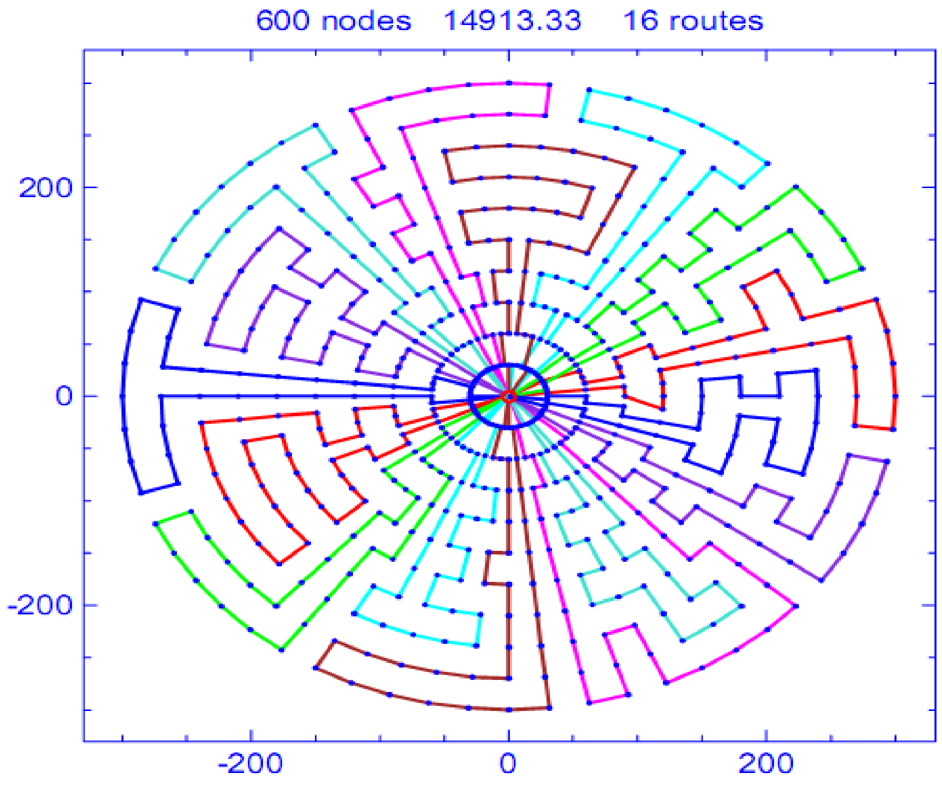
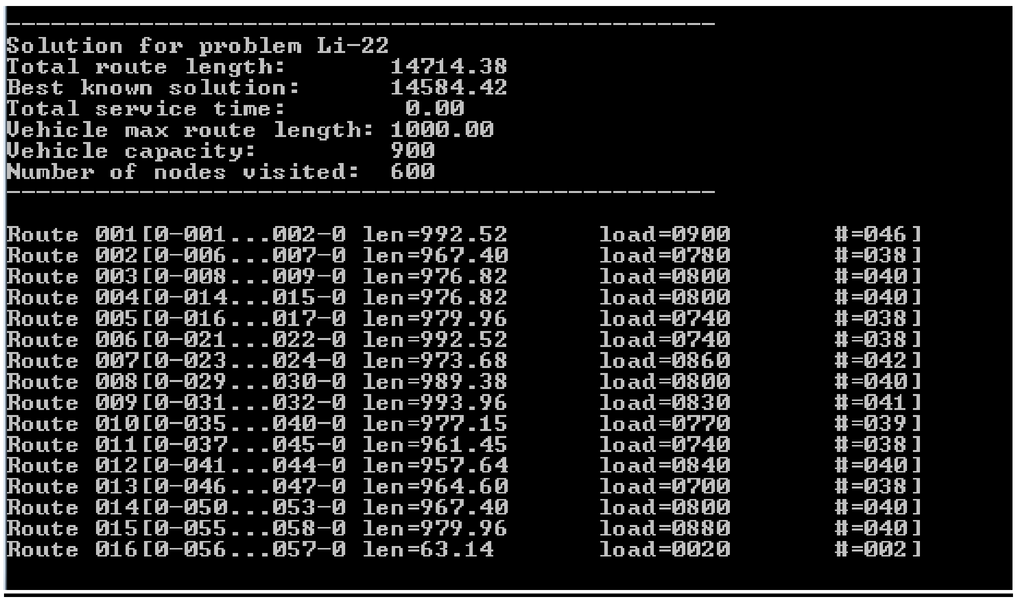


Fig. 4.22(b): Plot of routes for Li_22 after optimization by RTR algo.

4.1.23. Li_22.vrp – After applying New_RTR algorithm.



```
------------------------------------------------
Solution for problem Li-22
Total route length:          14714.38
Best known solution:         14584.42
Total service time:              0.00
Vehicle max route length:  1000.00
Vehicle capacity:            900
Number of nodes visited:     600
------------------------------------------------

Route 001[0-001...002-0 len=992.52       load=0900        #=046]
Route 002[0-006...007-0 len=967.40       load=0780        #=038]
Route 003[0-008...009-0 len=976.82       load=0800        #=040]
Route 004[0-014...015-0 len=976.82       load=0800        #=040]
Route 005[0-016...017-0 len=979.96       load=0740        #=038]
Route 006[0-021...022-0 len=992.52       load=0740        #=038]
Route 007[0-023...024-0 len=973.68       load=0860        #=042]
Route 008[0-029...030-0 len=989.38       load=0800        #=040]
Route 009[0-031...032-0 len=993.96       load=0830        #=041]
Route 010[0-035...040-0 len=977.15       load=0770        #=039]
Route 011[0-037...045-0 len=961.45       load=0740        #=038]
Route 012[0-041...044-0 len=957.64       load=0840        #=040]
Route 013[0-046...047-0 len=964.60       load=0700        #=038]
Route 014[0-050...053-0 len=967.40       load=0800        #=040]
Route 015[0-055...058-0 len=979.96       load=0880        #=040]
Route 016[0-056...057-0 len=63.14        load=0020        #=002]
```

Fig. 4.23(a) Optimized route details for Li_22 after applying New_RTR algorithm.

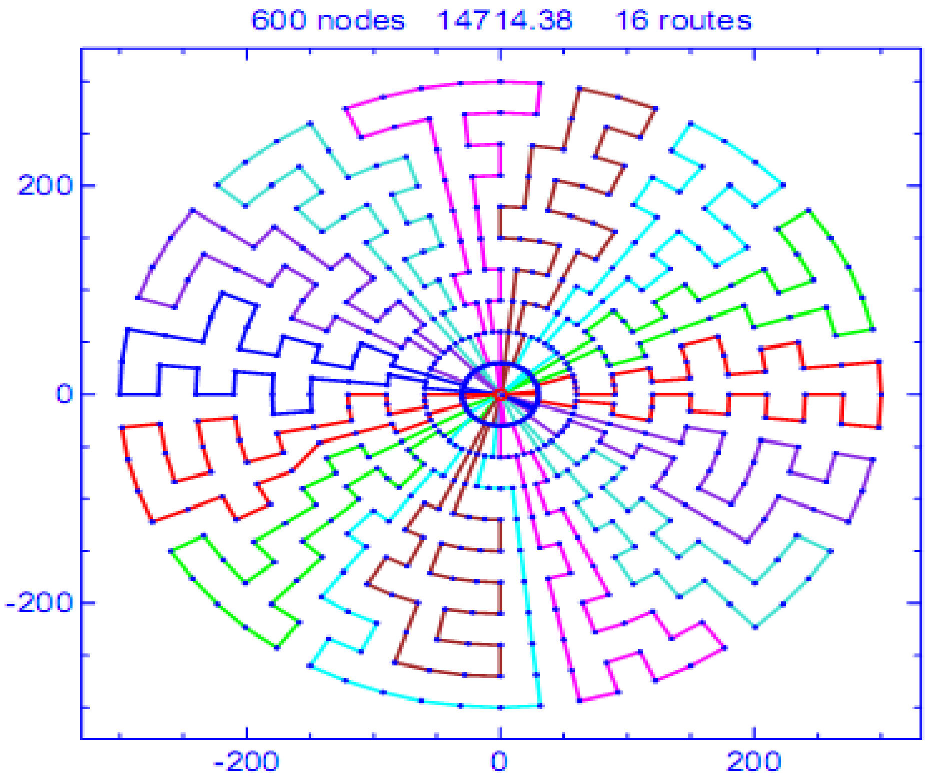
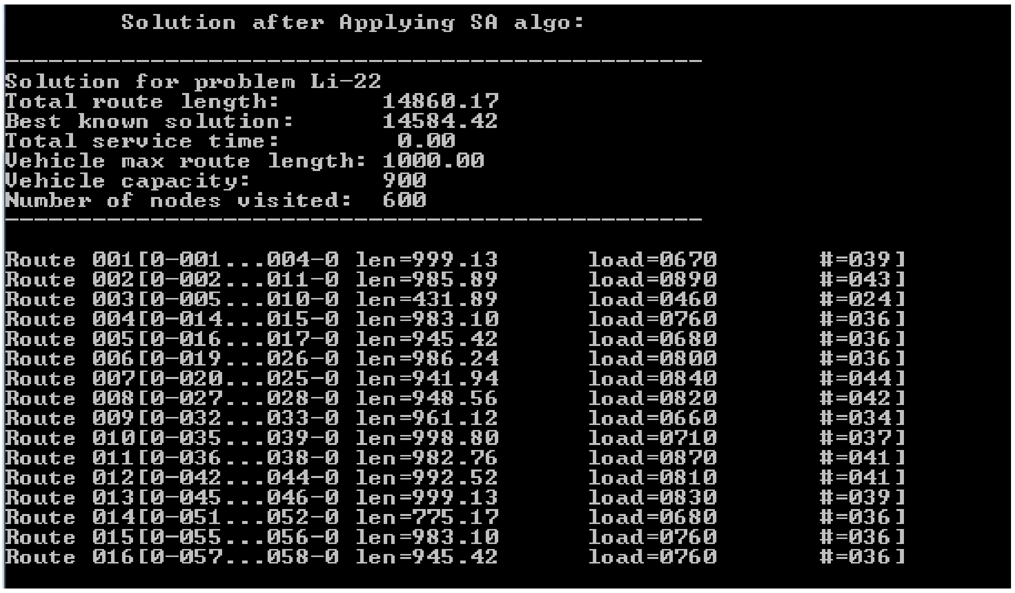


Fig. 4.23(b): Plot of routes for Li_22 after optimization by New_RTR algo.

4.1.24. Li_22.vrp – After applying SA algorithm.



```
        Solution after Applying SA algo:

------------------------------------------------
Solution for problem Li-22
Total route length:        14860.17
Best known solution:       14584.42
Total service time:         0.00
Vehicle max route length: 1000.00
Vehicle capacity:          900
Number of nodes visited:   600
------------------------------------------------

Route 001[0-001...004-0 len=999.13      load=0670       #=039]
Route 002[0-002...011-0 len=985.89      load=0890       #=043]
Route 003[0-005...010-0 len=431.89      load=0460       #=024]
Route 004[0-014...015-0 len=983.10      load=0760       #=036]
Route 005[0-016...017-0 len=945.42      load=0680       #=036]
Route 006[0-019...026-0 len=986.24      load=0800       #=036]
Route 007[0-020...025-0 len=941.94      load=0840       #=044]
Route 008[0-027...028-0 len=948.56      load=0820       #=042]
Route 009[0-032...033-0 len=961.12      load=0660       #=034]
Route 010[0-035...039-0 len=998.80      load=0710       #=037]
Route 011[0-036...038-0 len=982.76      load=0870       #=041]
Route 012[0-042...044-0 len=992.52      load=0810       #=041]
Route 013[0-045...046-0 len=999.13      load=0830       #=039]
Route 014[0-051...052-0 len=775.17      load=0680       #=036]
Route 015[0-055...056-0 len=983.10      load=0760       #=036]
Route 016[0-057...058-0 len=945.42      load=0760       #=036]
```

Fig. 4.24(a) Optimized route details for Li_22 after applying SA algorithm.

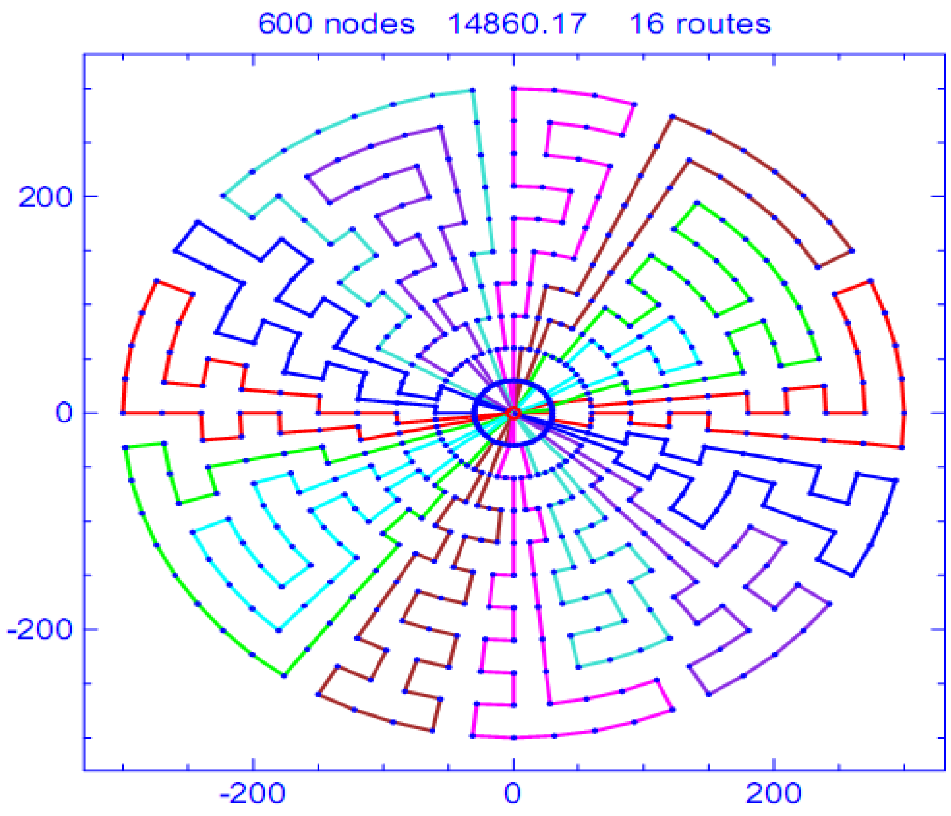
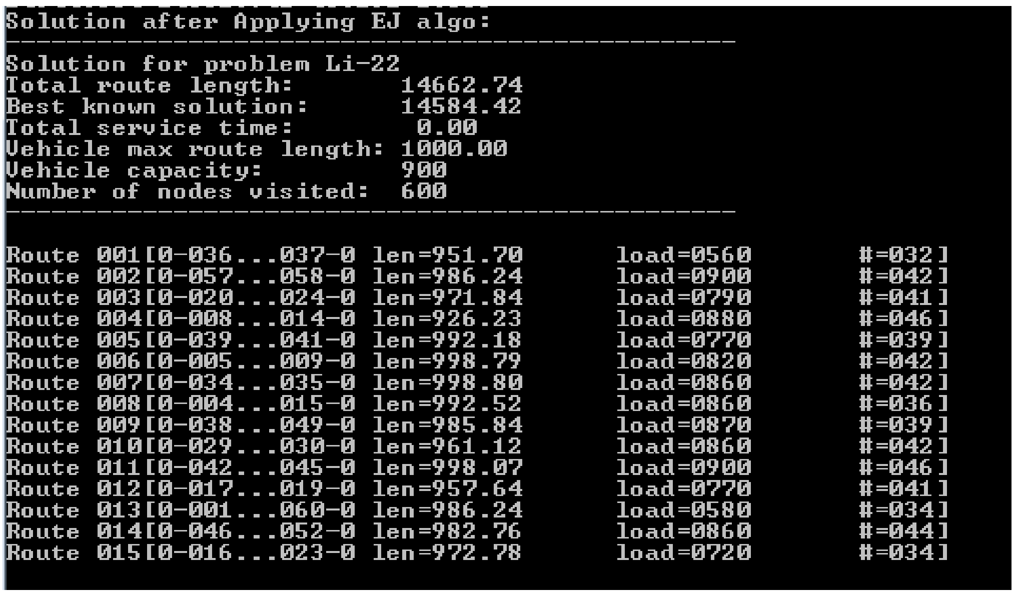


Fig. 4.24(b): Plot of routes for Li_22 after optimization by SA algo.

4.1.25. Li_22.vrp – After applying EJ algorithm.



```
Solution after Applying EJ algo:
------------------------------------------------
Solution for problem Li-22
Total route length:        14662.74
Best known solution:       14584.42
Total service time:         0.00
Vehicle max route length: 1000.00
Vehicle capacity:          900
Number of nodes visited:   600
------------------------------------------------

Route 001[0-036...037-0 len=951.70       load=0560        #=032]
Route 002[0-057...058-0 len=986.24       load=0900        #=042]
Route 003[0-020...024-0 len=971.84       load=0790        #=041]
Route 004[0-008...014-0 len=926.23       load=0880        #=046]
Route 005[0-039...041-0 len=992.18       load=0770        #=039]
Route 006[0-005...009-0 len=998.79       load=0820        #=042]
Route 007[0-034...035-0 len=998.80       load=0860        #=042]
Route 008[0-004...015-0 len=992.52       load=0860        #=036]
Route 009[0-038...049-0 len=985.84       load=0870        #=039]
Route 010[0-029...030-0 len=961.12       load=0860        #=042]
Route 011[0-042...045-0 len=998.07       load=0900        #=046]
Route 012[0-017...019-0 len=957.64       load=0770        #=041]
Route 013[0-001...060-0 len=986.24       load=0580        #=034]
Route 014[0-046...052-0 len=982.76       load=0860        #=044]
Route 015[0-016...023-0 len=972.78       load=0720        #=034]
```

Fig. 4.25(a) Optimized route details for Li_22 after applying EJ algorithm.

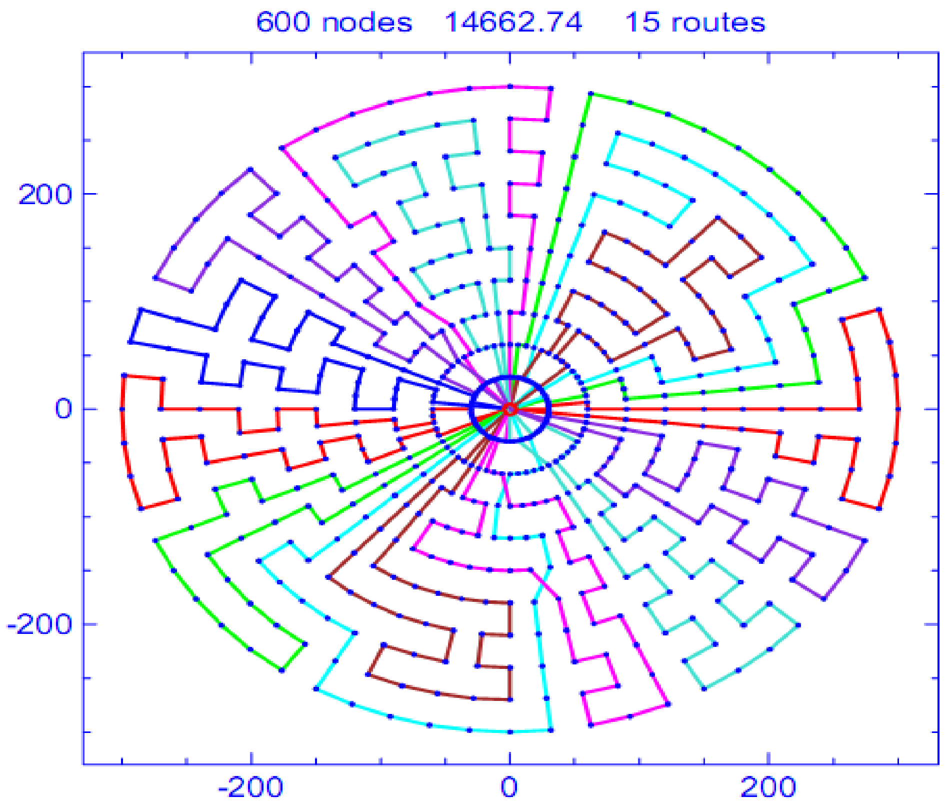


Fig. 4.25(b): Plot of routes for Li_22 after optimization by EJ algo.

4.2 Comparison of RTR & New_RTR algorithm

The table 4.1 shows the comparison of maximum_route_length computed by traditional record to record travel algorithm and our proposed New_RTR algorithm. It also summarizes the percentage improvement gained.

Figure 4.26 shows the relation between the improvement gained and the number of non-depot nodes in the problem. It can be easily assessed from the graph that New_RTR performance increases with the increase in number of non-depot nodes.

Figure 4.27 summarizes the relation between the improvement gained and the deviation used during New_RTR algorithm. It can be observed from the plot that most of the improvements are at deviation 0.01 which is the same as used by traditional RTR algorithm. Hence it can be concluded that, the values of lambda play an important role in finding optimal solution.

Table 4.1: Performance of New_RTR and RTR.

Benchmark	Nodes	Dev	Total_Max_Route_Length		Best_λ	%age Imp
			New_RTR	RTR		
Christofides_01	50	0.02	526.61	526.97	1.3	0.068
Christofides_02	75	0.05	841.09	849.54	1.1	0.995
Christofides_03	100	0.03	828.66	830.10	1.2	0.173
Christofides_04	150	0.03	1041.13	1043.01	1.6	0.181
Christofides_06	160	0.01	556.68	558.99	1.3	0.413
Golden_02	320	0.01	8468.12	8547.09	0.7	0.924
Golden_03	400	0.02	11137.48	11152.30	1.8	0.133
Golden_04	480	0.01	13652.13	13685.98	1.8	0.247
Li_21	560	0.03	16345.49	16463.17	1.4	0.715
Li_22	600	0.01	14714.38	14913.33	1.0	1.334
Li_23	640	0.01	18856.98	19131.67	1.4	1.436
Li_24	720	0.01	21466.38	21935.82	1.2	2.140
Li_26	800	0.03	24223.09	24673.69	1.8	1.826
Taillard_75A	75	0.03	1618.36	1624.00	1.0	0.348
Taillard_75B	75	0.04	1344.63	1348.23	1.0	0.267
Taillard_75C	75	0.04	1292.61	1294.56	1.0	0.151
Taillard_100B	100	0.01	1945.55	1957.85	1.1	0.628

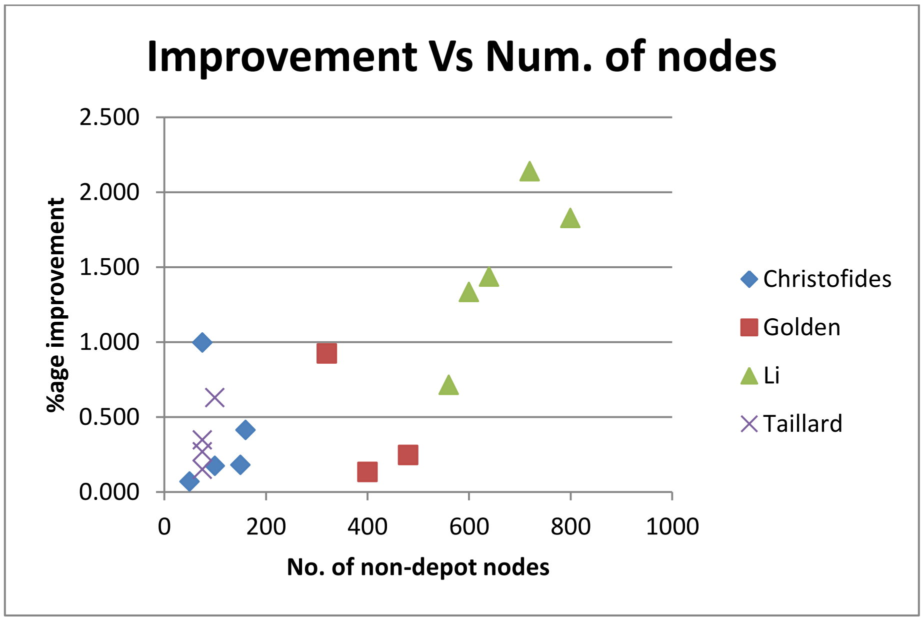


Figure 4.26: Improved performance of New_RTR w.r.t. Number of nodes

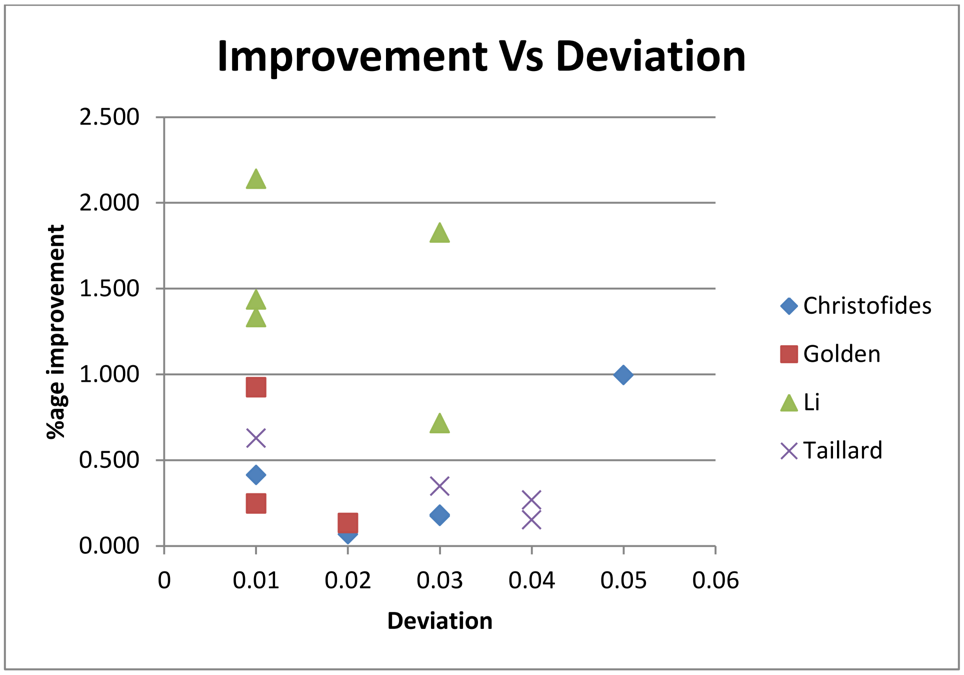


Figure 4.27: Improvement of New_RTR w.r.t. Deviation

5.1 CONCLUSION

EazyRoute has proved to be a better contender for a comprehensive solution of Vehicle Routing Problem with Capacity Constraints. It's heart lies in the underlying heuristic and meta-heuristic algorithms used to compute and optimize the solutions. The variety of algorithms provide user with a wide choice to optimize the solution to the best. Also it provides users with the flexibility in terms of input parameters i.e. Number of vehicles, Maximum route length and Capacity of each vehicle.

Our proposed algorithm, New_RTR, which has been proved to perform better to its counterpart, when embedded with EazyRoute will provide the user to generate more optimized solutions.

EazyRoute's functionality can further be extended as discussed in section 5.2, to solve the problem that closely resembles the real world applications.

5.2 FUTURE WORK

As a part of this research process, we have developed EazyRoute, which works well for Capacitated Vehicle Routing Problem. But we have also seen that there are other variants of VRP that maps to real world problems. EazyRoute can further be extended to handle all other variations like Multiple-depots, Multi-Vehicle, Split-Delivery, etc.

REFERENCES

[1] G.B. Dantzig, and J.H. Ramser, “The truck dispatching problem”, Management Science 6 (1959): 80–91.

[2] Smith, J. Cole, and Z. C. Taskin. "A Tutorial Guide to Mixed Integer Programming Models and Solution Techniques." *Optimization in Medicine and Biology* (2008): 521-548.

[3] Tasan, A. Serdar, and Mitsuo Gen. "A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries." *Computers & Industrial Engineering* 62.3 (2012): 755-761.

[4] Zhen, Tong, Yuhua Zhu, and Qiuwen Zhang. "A Particle Swarm Optimization Algorithm for the Open Vehicle Routing Problem." *Environmental Science and Information Application Technology, 2009. ESIAT 2009. International Conference on*. Vol. 2. IEEE, 2009.

[5] Narasimha, Koushik S. Venkata, Elad Kivelevitch, and Manish Kumar. "Ant Colony optimization technique to Solve the min-max multi depot vehicle routing problem." *American Control Conference (ACC), 2012*. IEEE, 2012.

[6] Bertsimas, Dimitris J. "A vehicle routing problem with stochastic demand." *Operations Research* 40.3 (1992): 574-585.

[7] C. Groer. The VRPH software. http://sites.google.com/site/vrphlibrary/, 2010.

[8] PLPlot. The PLPlot software package. http://plplot.sourceforge.net/, 2010.

[9] Groer Chris, Bruce Golden, and Edward Wasil. "A library of local search heuristics for the vehicle routing problem." *Mathematical Programming Computation* 2.2 (2010): 79-101.

[10] Laporte, Gilbert. "What you should know about the vehicle routing problem."*Naval Research Logistics (NRL)* 54.8 (2007): 811-819.

[11] G. Laporte, and F.V. Louveaux. “Solving Stochastic Routing Problems with the Integer L-shaped Method”. In Fleet Management and Logistics, T.G. Crainic and G. Laporte (eds.)(1998), 159-167, Kluwer Academic Publishers, Boston.

[12] W. Burrows. “The Vehicle Routing Problem with Loadsplitting: A Heuristic Approach”. In 24th Annual Conference of the Operational Research Society of New Zealand, (1988): 33-38.

[13] M. M. Solomon. “Algorithms for the Vehicle Routing Problem with Time Windows”. Transportation Science, 29(2), (1995): 156-166.

[14] Laporte, Gilbert, and Yves Nobert. "Exact algorithms for the vehicle routing problem." *Surveys in Combinatorial Optimization* 31 (1987): 147-184.

[15] P. Toth, and D. Vigo, “Exact solution of the vehicle routing problem,” Fleet Management and Logistics, T.G. Crainic, and G. Laporte (Editors), Kluwer, Boston,(1998): 1–31.

[16] Laporte, Gilbert, Yves Nobert, and Martin Desrochers. "Optimal routing under capacity and distance restrictions." *Operations research* 33.5 (1985): 1050-1073.

[17] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson. “Solution of a large-scale traveling-salesman problem”, Operations Research 2 (1954): 393–410.

[18] D. Naddef, and G. Rinaldi, “Branch-and-cut algorithms for the capacitated VRP,” The Vehicle routing Problem, P. Toth, and D. Vigo (Editors), SIAM Monographs on Discrete Mathematics and Applications, Philadelphia,(2002): 53–81.

[19] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, “An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation”, Operations Research 52 (2004): 723–738.

[20] G.A. Finke, A. Claus, and E. Gunn, “A two-commodity network flow approach to the traveling salesman problem”, Congressus Numerantium 41 (1984): 167–178.

[21] M. Balinski, and R. Quandt, “On an integer program for a delivery problem”, Operations Research 12 (1964): 300–304.

[22] A.N. Letchford, R.W. Eglese, and J. Lysgaard, Multistars, “Partial multi-stars and the capacitated vehicle routing problem”, Mathematical Progamming Series A 94 (2002): 21–40.

[23] E. Balas, and M.W. Padberg, “Set partitioning: A survey”, SIAM Review 18 (1976): 710–760.

[24] K. Hoffman, and M.W. Padberg, “Solving airline crew scheduling problems by branch and cut”, Management Science 39 (1993): 657–682.

[25] R. Baldacci, N. Christofides, and A. Mingozzi, “An exact algorithm for the vehicle routing Problem based on the set partitioning formulation with additional cuts”, Mathematical Programming Series A (2007).

[26] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragao, M. Reis, E. Uchoa, and R.F. Werneck, “Robust branch-and-cut-and-price for the capacitated vehicle routing problem”, Mathematical Programming Series A 106 (2006): 491–511.

[27] N. Christofides, A. Mingozzi, and P. Toth, “The vehicle routing problem,” Combinatorial Optimization, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi (Editors), Wiley, Chichester,(1979): 315–338.

[28] B.L. Golden, E.A. Wasil, J.P. Kelly, and I-M. Chao, “Metaheuristics in vehicle routing,” Fleet Management and Logistics, T.G. Crainic, and G. Laporte (Editors), Kluwer, Boston, (1998): 33–56.

[29] G. Clarke, and J.V. Wright, “Scheduling of vehicles from a central depot to a number of delivery points”, Operations Research 12 (1964): 568–581.

[30] G. Laporte, and F. Semet, “Classical heuristics for the capacitated VRP,” The Vehicle Routing Problem, P. Toth, and D. Vigo (Editors), SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, (2002): 109–128.

[31] M.D. Nelson, K.E. Nygard, J.H. Griffin, and W.E. Shreve, “Implementation techniques for the vehicle routing problem”, Computers & Operations Research, 12 (1985): 273–283.

[32] K. Altinkemer, and B. Gavish, “Parallel savings based heuristic for the delivery problem”, Operations Research 39 (1991): 456–469.

[33] P. Wark, and J. Holt, “A repeated matching heuristic for the vehicle routing problem”, Journal of the Operational Research Society 45 (1994): 1156–1167.

[34] B.L. Golden, T.L. Magnanti, and H.Q. Nguyen, “Implementing vehicle routing algorithms”, Networks 7 (1977): 113–148.

[35] B.E. Gillett, and L.R. Miller, “A heuristic algorithm for the vehicle dispatch problem”, Operations Research 22 (1974): 340–349.

[36] B.A. Foster, and D.M. Ryan, “An integer programming approach to the vehicle scheduling problem”, Operations Research 27 (1976): 367–384.

[37] D.M. Ryan, C. Hjorring, and F. Glover, “Extentions of the petal method for vehicle routing”, Journal of the Operational Research Society 44 (1993), 289–296.

[38] J. Renaud, F.F. Boctor, and G. Laporte, “An improved petal heuristic for the vehicle routing problem”, Journal of the Operational Research Society 47 (1996): 329–336.

[39] M.L. Fisher, and R. Jaikumar, “A generalized assignment heuristic for the vehicle routing problem”, Networks 11 (1981): 109–124.

[40] B.M. Baker, and J. Sheasby, “Extensions to the generalized assignment heuristic for vehicle routing”, European Journal of Operational Research 119 (1999): 147–157.

[41] P.M. Thompson, and H.M. Psaraftis, “Cyclic transfer algorithms for multi-vehicle routing and scheduling problems”, Operations Research 41 (1993): 935–946.

[42] A. Van Breedam, “An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selection of Problems with Vehicle-Related, Customer-Related, and Time Related Constraints”, Ph.D. Dissertation, University of Antwerp, 1994.

[43] G.A.P. Kindervater, and M.W.P. Savelsbergh, “Vehicle routing: Handling edge exchanges,” Local Search in Combinatorial Optimization, E.H.L. Aarts, and J.K. Lenstra (Editors), Wiley, Chichester, (1997): 337–360.

[44] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany, “New heuristics for the vehicle routing problem,” Logistics Systems: Design and Optimization, A. Langevin, and D. Riopel (Editors) Springer, New York, (2005): 279–297.

[45] M. Gendreau, J.-Y. Potvin, O. Braysy, G. Hasle, and A. Løkketangen, “Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography,” The Vehicle Routing Problem: Latest Advances and Challenges, B.L. Golden, S. Raghavan, and E.A. Wasil (Editors), Springer, Boston, 2007.

[46] G. Dueck, New optimization heuristics: “The great deluge algorithm and the record-to-record travel”, Journal of Computational Physics, 104 (1993): 86–92.

[47] F. Glover, “Future paths for integer programming and links to artificial intelligence”, Computers & Operations Research 13 (1986): 533–549.

[48] J.-F. Cordeau, G. Laporte, and A. Mercier, “A unified tabu search heuristic for vehicle routing problems with time windows”, Journal of the Operational Research Society 52 (2001): 928–936.

[49] M. Gendreau, A. Hertz, and G. Laporte, “A tabu search heuristic for the vehicle routing problem”, Management Science 40 (1994): 1276–1290.

[50] N. Mladenovic, and P. Hansen, Variable neighborhood search, Computers & Operations Research 24 (1997): 1097–1100.

[51] O. Ergun, “New neighborhood search algorithms based on exponentially large scale neighborhoods”, Ph.D. Thesis, Massachusetts Institute of Technology, 2001.

[52] D. Pisinger, and S. Ropke, “A general heuristic for vehicle routing problems”, Computers & Operations Research, 34 (2007): 2403–2435.

[53] J.H. Holland, “Adaptation in natural and artificial systems”, The university of michigan press, Ann Arbor, MI, 1975.

[54] C. Prins, “A simple and effective evolutionary algorithm for the vehicle routing problem”, Computers & Operations Research 31 (2004): 1985–2002.

[55] Y. Rochat, and E.D. Taillard, “Probabilistic diversification and intensification in local search for vehicle routing”, Journal of Heuristics 1 (1995): 147–167.

[56] C.-D. Tarantilis, and C.T. Kiranoudis, BoneRoute, “An adaptive memory-based method for effective fleet management”, Annals of Operations Research 115 (2002): 227–241.

[57] H. Ghaziri, “Solving routing problems by a self-organizing map,” Artificial Neural Networks, T. Kohonen, K. Makisara, O. Simula, and J. Kangas (Editors), North-Holland, Amsterdam,(1991):829–834.

[58] M. Schumann, and R. Retzko, “Self-organizing maps for vehicle routing problems – minimizing an explicit cost function,” Proceedings of the International Conference on Artificial Neural Networks, F. Fogelman-Soulie (Editor), Paris,(1995): 401–406.

[59] J. Kytojoki, T. Nuortio, O. Braysy, and M. Gendreau. “An efficient variable neighborhood search heuristic for very large scale vehicle routing problems”, Computers & Operations Research, 47(2), (2005):329-336.

[60] G. Reinelt. “TSPLIB - A Traveling Salesman Problem Library”. ORSA Journal on Computing-3, (1991):376-384.

[61] P.C. Yellow. “A computational modification to the savings method of vehicle scheduling”, Operational Research Quarterly, 21, (1970):281-293.

[62] Lin, S-W., et al. "Applying simulated annealing approach for capacitated vehicle routing problems." *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*. Vol. 1. IEEE, 2006.

[63] R. Hassin and A. Keinan. “Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP”. Operations Research Letters, 36(2),(2008):243-246.

[64] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem Operations Research, 21, (1973):2245-2269.