Review 3

TRANSPORTATION SERVICE SYSTEM FOR GROCERY KART

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

By

DEEPANK KARTIKEY (14BIT0051)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLGY AND ENGINEERING
March 2018

Guide Signature

(Prof. Vanmathi C)

ABSTRACT

Transportation is a major problem domain in logistics, and so it represents a substantial task in the activities of many companies. In some market sectors, transportation means a high percentage of value added to goods. Therefore, the utilization of computerized methods for transportation often results in significant savings ranging from 5% to 20% in the total costs. As per this day there is no deterministic approach invented which will give best result and solves the problem of Vehicle routing with time windows. In the Vehicle Routing Problem (VRP), the aim is to design a set of *m* minimum cost vehicle routes through *n* customer locations, so that each route starts and ends at a common location and some side constraints are satisfied. Many heuristic approaches have been invented to approximate the solution and improve the solution previously achieved by different researchers. I have accepted this challenge which tends to address the delivery of grocery to various stores from warehouses and determine the best paths based on route evaluation, best available routes, determine optimum number of trucks required, profit, cost and efficiency analysis. Various comparative reports will be generated for various user and city data. More specifically, we consider the distance, fixed cost, time, together with risk simultaneously. To get a quicker and more accurate solution, several improvements are proposed in applying GA for optimization search. I am using multi-objective genetic algorithm (GA) to address the problem to obtain optimum solutions. I will be comparing an existing crossover operator and my new crossover operator. Genetic Algorithms are powerful and broadly applicable stochastic search and optimization techniques that work for many problems that are difficult to solve by conventional techniques.

Since multiple objective problems rarely have points that maximize all the objectives simultaneously, I am typically in a situation to maximize each objective to greatest extent possible. I have an interest in multiple objective transportation problems.

Problem Statement

GroceryKart.com is responsible for supplying fresh vegetables and produce to all retail stores in Mumbai and adjoining suburbs. There are about 600 stores in their network. They have a fleet of old Matador vans that are being used to haul the goods from their central warehouse in Andheri. Given the perishable nature of the goods, all vans leave at 5:00 AM and all deliveries should happen before 8:00 AM. Each van can make multiple stop(s) but the union contract prevents any truck from making more than 20 stops or travelling more than 65 KM in any trip.

In effort to reduce costs and minimize greenhouse emissions, GroceryKart.com is planning to retire all the old Matador vans and purchase a new fleet of Mahindra vans. They believe the cost of purchasing the Mahindra vans will more/less be covered by the sale of Matador vans and the rest of the money will come from the existing savings fund. However, maintaining each truck costs Rs. 20,000 a month (including driver, etc.). Also, the variable cost/km of travel is around Rs. 30 for the first three years of the truck usage. Each Mahindra truck can carry up to 1000 kg at full capacity and at the end of the trip, all trucks return to the warehouse. Each stop at the store takes 5 minutes to download (irrespective of the weight) and every KM travelled takes 90 seconds on an average.

I should determine the number of Mahindra vans required to be purchased and the total KM that will be travelled to service the daily demand of all the stores so as to minimize the total cost over the first three years?
Also, determine

- The number of trucks needed
- Routing for each truck and
- The total miles travelled each day

Transportation Service System for Grocery kart is a multi-objective optimization problem, which needs a suitable approach to obtain a global optimum solution. So, for this high complexity problem without any known sophisticated solution technique, a genetic approach is well suited. In the optimization process, due to the problem specific encoding, the genetic algorithm enables me to compute and optimize the routing quiet easily as compared to the other deterministic approaches. Genetic algorithm is a search heuristic that mimics the process of natural evolution. Genetic algorithms generate solutions to optimization problems using techniques inspired by natural evolution, such as selection, crossover and mutation.

1. Introduction

1.1 Purpose

The purpose of this project is to reduce the Time complexity and optimize the resources involved in routing techniques consisting of large data sets using Genetic Algorithm.

1.2 Scope

Genetic Algorithms are powerful and broadly applicable stochastic search and optimization techniques that work for many problems that are difficult to solve by conventional techniques. A genetic algorithm (GA) is a method for solving both constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution. The algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm randomly selects individuals from the current population and uses them as parents to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution.

Some basic terms related to Genetic algorithms:

- **Population** − It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- **Chromosomes** − A chromosome is one such solution to the given problem.
- **Gene** − A gene is one element position of a chromosome.
- **Allele** − It is the value a gene takes for a chromosome.
- **Genotype** − Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
- **Phenotype** − Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.
- **Decoding and Encoding** − For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

- **Fitness Function** − A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.
- **Genetic Operators** − These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

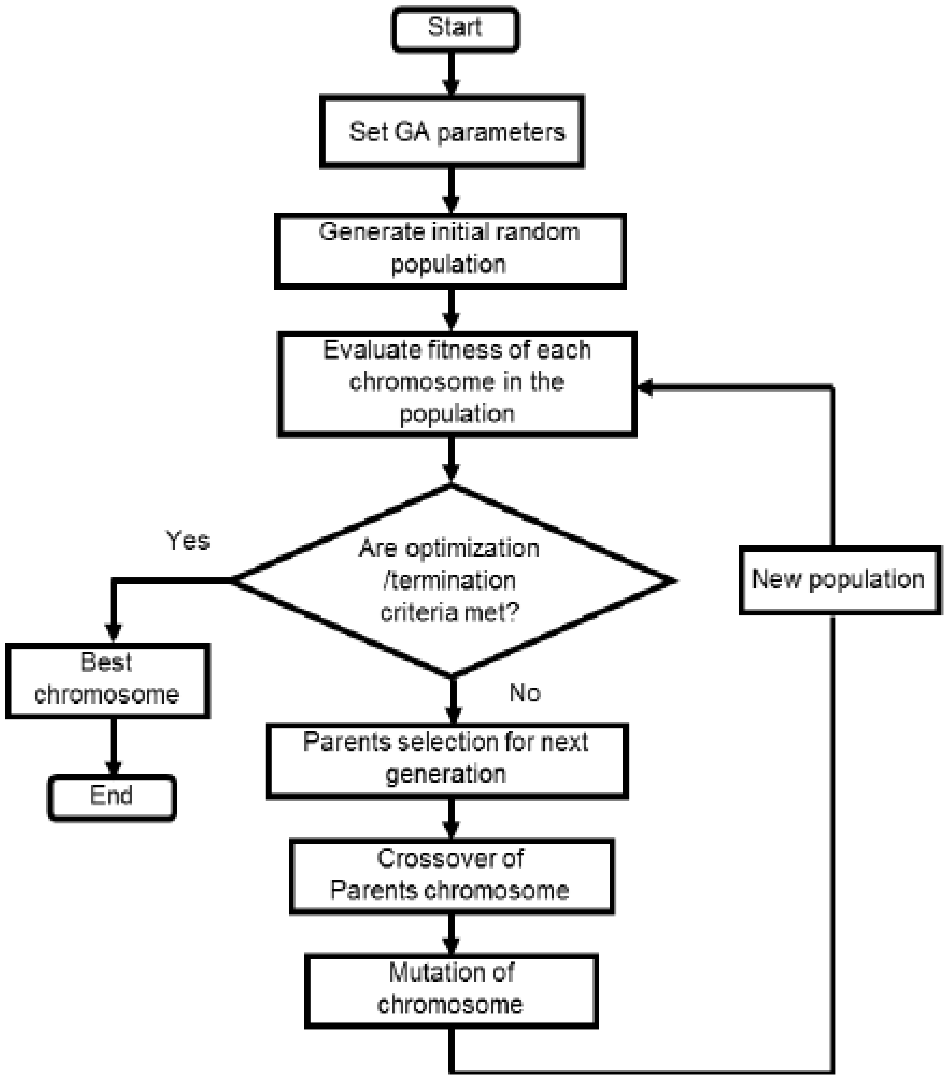

Start
Set GA parameters
Generate initial random population
Evaluate fitness of each chromosome in the population
Yes
Are optimization /termination criteria met?
New population
Best chromosome
No
End
Parents selection for next generation
Crossover of Parents chromosome
Mutation of chromosome

Fig. 1. Genetic Algorithm working

1.2 Applet

An **applet** is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal. An applet is a Java class that extends the java.Applet class. A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment. The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.

Life Cycle of an Applet

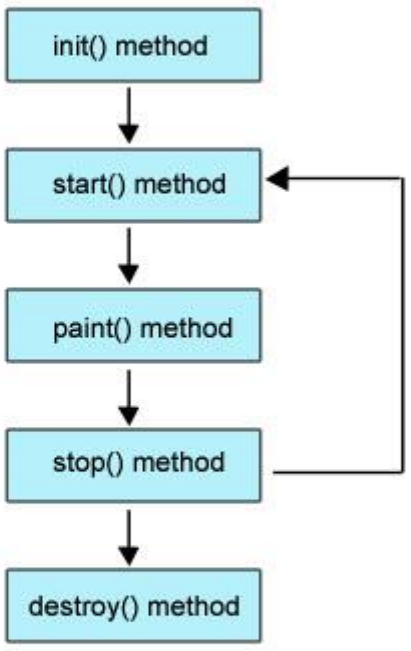


Figure: Life cycle of Applet

The methods in the Applet class gives you the framework on which you build any serious applet −

- **init** − This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
- **start** − This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
- **stop** − This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.
- **destroy** − This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.
- **paint** − Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt.

These methods are known as Applet Life Cycle methods. These methods are defined in java.applet.Applet class except paint() method. The paint() method is defined in java.awt.Component class, an indirect super class of Applet.

2. Literature Survey

The project focuses on few existing applications which partially accomplish what I am trying to do through the project “Transportation Service System for Grocery Kart”. The purpose of this project is to reduce the Time complexity and optimize the resources involved in routing techniques consisting of large data sets using Genetic Algorithm.

The problem of transport of goods, commodities and people is as relevant as when it was raised in 1959 by Dantzig and Ramser (1959), where it was considered as a generalization of the traveling salesman problem. The first article in which the phrase “Vehicle routing” appeared is attributed to Golden [1]. Other versions of Vehicle Routing Problem (VRP) appeared in the early 70s. Liebman and Marks [3] presented mathematical models that represent routing problem associated with the collection of wastes; Levin [4] posed the problem of fleet vehicles; Wilson [2] presented the problem of telephone request for transportation service for persons with disabilities and Marks and Stricker presented a model for routing public service vehicles. Some probabilistic concepts were associated with the problem by Golden and Stewart. In Solomon [5] constraints of time windows were included; Sariklis and Powell [6] proposed a problem where the vehicle does not return to the point where the journey begins. Recently, some specific conditions have been added to approach VRP to real-life problems, resulting in several variants which modify constraints or objective function of the basic VRP optimization model.

In its general form the objective of the VRP (Vehicle Routing Problem) is to design a set of minimum cost routes that serves many places, geographically dispersed, and fulfilling specific constraints of the problem. Since its first formulation in 1959, there have been many publications and has expanded its scope. Initial studies were conducted for analysing the distribution management. In the last decade, there have been significant advances in terms of the technical solution to resolve large instances. Another aspect that has gained interest is the inclusion of technological innovations in the VRP. These include global positioning systems, radio frequency identification and use of high-capacity computer information processing.

The Vehicle routing problem can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several customer demands, finds the set of routes with overall minimum route cost which service all the demands [7]. All the itineraries start and end at the depot and they must be designed in such a way that each customer is served only once and just by one vehicle.

Genetic algorithms have been inspired by the natural selection mechanism introduced by Darwin [8]. They apply certain operators to a population os solutions of the problem at hand, in such a way that the new population is improved compared with the previous one according to a prespecified criterion function. This procedure is applied for a preselected number of iterations and the output of the algorithm is the best solution found in the last population or, in some cases, the best solution found during the evolution of the algorithm. In general, the solutions of the problem at hand are coded and the operators are applied to the coded versions of the solutions. The way the solutions are coded plays an important role in the performance of a genetic algorithm. Inappropriate coding may lead to poor performance.

The operators used by genetic algorithms simulate the way natural selection is carried out. The most well-known operators used are the reproduction, crossover, and mutation operators applied in that order to the current population. The reproduction operator ensures that, in probability, the better a solution in the current population is, the more (less) replicates it has in the next population. The crossover operator, which is applied to the temporary population produced after the application of the reproduction operator, selects pairs of solutions randomly, splits them at a random position, and exchanges their second parts. Finally, the mutation operator, which is applied after the application of the reproduction and crossover operators, selects randomly an element of a solution and alters it with some probability. Hence genetic algorithms provide a search technique used in computing to find true or approximate solutions to optimization and search problems.

The solutions techniques for vehicle routing problems can be depicted as given in figure below:

2.1 Exact techniques

The VRP in its different forms, was initially addressed using exact techniques such as Branch and Bound algorithm proposed by Laporte and Nobert [9]; Fischetti, Toth and Vigo[10] and Baldacci and Branch and Cut and Price proposed by Baldacci et al.,[11].

2.2 Heuristics

Heuristics for the VRP are divided into three classes: two phases,
construction and iterative improvement.

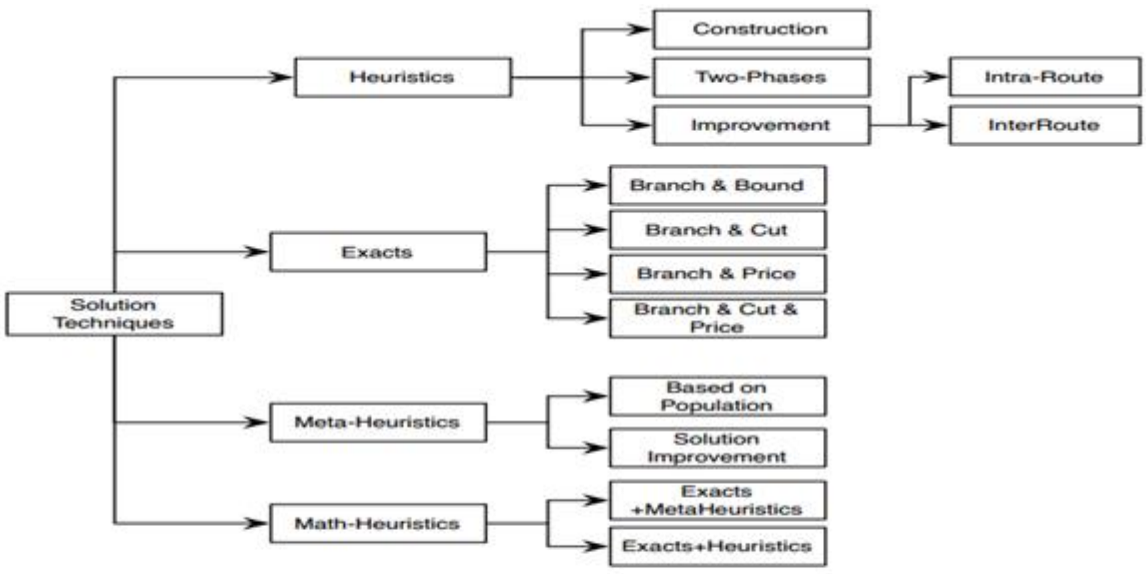


Figure 1. VRP Solution Techniques.

The **two-phase heuristics** divide the problem into two stages:
Customer allocation to route and determination the order of the visit. The iterative improvement receives an initial solution and then another heuristic by inter and intra route movements between customers to explore the solution space. Inside this classification Beasley [12] proposed the strategy Cluster First-Route second and Routed First-Cluster Second.

2.2.1 Construction heuristics

Construction heuristic starts from an empty solution which is filled considering feasible alternatives. This kind of heuristics includes savings algorithm in two versions parallel and sequential posed by Clar and Wright [13]. Insertion heuristics, the solution is created by inserting customers who have not been assigned to a route proposed by Mole and Jameson [14]; and Laporte [9].

2.2.3 Iterative improvement heuristics. It is for the betterment of the individual routes, and use concepts such as Lambda exchange, Or - OPT, Van-Breedam Operators, Breedam, GENI and GENIUS, widespread insertions proposed by Gendreau, Hertz and Laporte [9], cyclic transfers, Cross - exchange neighbourhood, which is the generalization three neighbourhoods: insert, swap and Two - OPT. A detailed explanation of these techniques is found

2.3 Metaheuristics

Metaheuristics have been adapted and intensively used for solve different variants of VRP. The main types of metaheuristics which have been applied to solve to the VRP are: Simulated Annealing, Determinist Annealing, Genetic Algorithms, Ant colony optimization, Iterative Location Search and Variable neighbourhood Search To apply such strategies is necessary to define a sufficiently clear and robust coding that allows representing an alternative solution where the value of the objective function and its feasibility is efficiently calculated for all the configurations.

It is also necessary to identify heuristics that can improve the performance of the technique chosen. It is important to define strategies that deliver quality settings and local search mechanisms that allow for efficient and effective exploration of the solution space. Only the metaheuristics applied to VRP deserve space on multiple items, because many authors not mentioned in this document have made quite creative proposals to improve this methodology type where the performance is evident in the responses and times computer. So the study of the variation of VRP interest requires special attention when a metaheuristic strategy is defined as an alternative solution.

2.4 Math-heuristics

This kind of methods consists in hybrid strategies that combine elements of heuristic techniques, metaheuristic techniques and methodologies exact solution as BB, BC and BCP. Here the sub-problems are identified and modelled in exact way then are solved by commercial integer linear programming solver. Dynamic programming or Lagrangean relaxation sometimes is used for this step.

To obtain best results and overcome disadvantages of other classical search methods, one needs to combine Genetic Algorithms with other optimization techniques.

3 Detailed Design

3.1 System Architecture

I have chosen Object Oriented Designing for the Transportation Service System for Grocery Kart. **Object-oriented design** is the process of planning a system of interacting objects for solving a software problem. It is one approach to software design. An object contains encapsulated data and procedures grouped together to represent an entity.

The 'object interface', how the object can be interacted with, is also defined. An object-oriented program is described by the interaction of these objects. Object-oriented design is the discipline of defining the objects and their interactions to solve a problem that was identified and documented during object-oriented analysis.

I have chosen this architecture because of the following reasons:

- Easier maintenance.
- Objects may be understood as stand-alone entities.
- Objects are potentially reusable components.
- For some systems, there may be an obvious mapping from real world entities to system objects

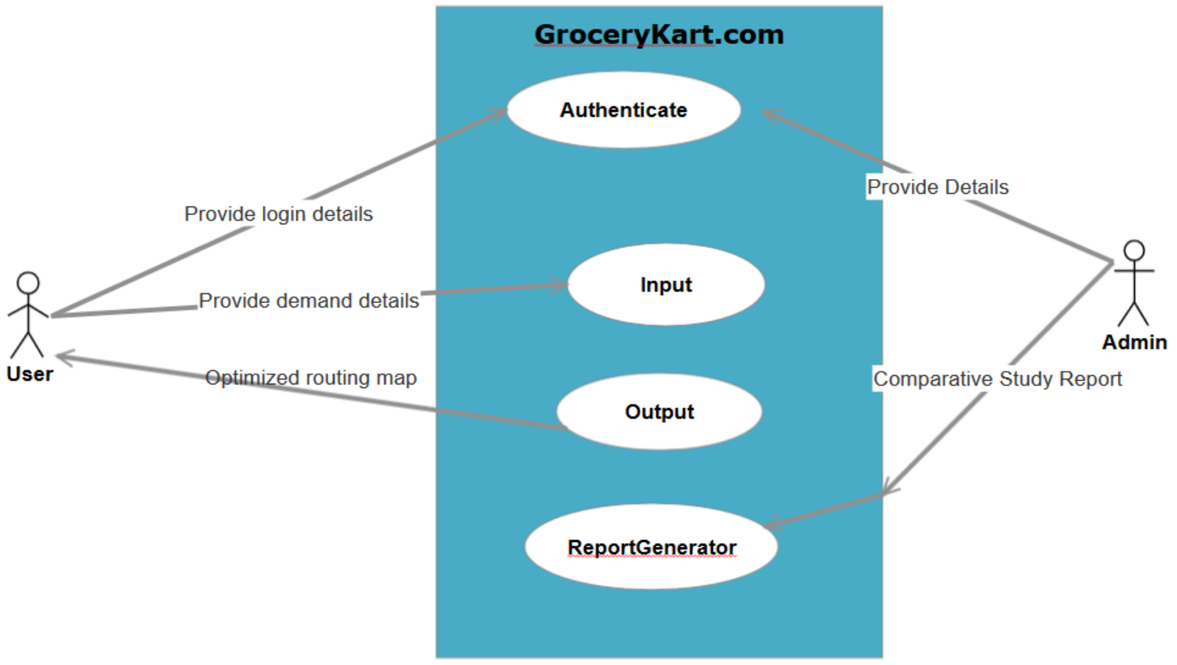
Java applet is a small application that is written in the Java programming language, or another programming language that compiles to Java Bytecode, and delivered to users in the form of Java bytecode. The user launches the Java applet from a web page, and the applet is then executed within a Java Virtual Machine (JVM) in a process separate from the web browser itself. A Java applet can appear in a frame of the web page, a new application window, Sun's Applet Viewer, or a stand-alone tool for testing applets. Java applets were introduced in the first version of the Java language, which was released in 1995.

Java applets are usually written in Java, but other languages such as JPython , JRuby, Pascal, Scala, or Eiffel(via SmartEiffel) may be used as well.

Java applets run at very fast speeds and, until 2011, they were many times faster than JavaScript. Unlike JavaScript, Java applets had access to 3Dhardware acceleration, making them well-suited for non-trivial, computation-intensive visualizations. As browsers, have gained support for hardware-accelerated graphics thanks to the canvas technology (or specifically WebGL in the case of 3D graphics), as well as Just-in Compiled Time JavaScript, the speed difference has become less noticeable.

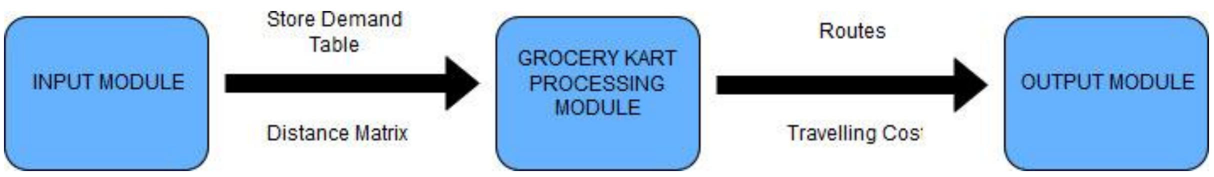
Since Java bytecode is cross-platform (or platform independent), Java applets can be executed by browsers (or other clients) for many platforms, including Microsoft Windows, FreeBSD, Unix, macOS and LINUX.

3.2 UML diagram



Data Flow Diagram:

The overall information flow in the system can be shown by the following diagram:



3.3 Module Description:

The system will be constructed from multiple distinct components:

3.3.1 Input Module

Initially, **user authentication** takes place in authentication window where he needs to provide the user_id and password to start working of the Grocery Kart service system . Its function includes:

authenticate()

The system will check with the login details and if correct it will allow user to proceed else it will show error message “Enter correct login details”.

If user can login with correct user_id and password then he will be redirected to main welcome window in which user chooses some option out of the various multi-objectives mentioned like Route Evaluation, Best Available Routes, Trucks Analysis, Stores Analysis etc. After selecting the desired option user must enter inputs for the various fields to calculate optimum results. The inputs can be in various forms like Data Centre store matrix, Store

Demand Table or even discrete tables. System will take data sets of the distance and requirement of goods from each store.

These inputs will then be passed on to the second module where the entire processing will take place with the help of classes: grocerykartform, candidate etc.

3.3.2 Processing Module

The processing modules contains the mechanism of the genetic algorithm. It defines the flow of randomly generated possible solutions and optimising them to the best solution.

There are 3 proposed classes:

- GroceryKart
- Store
- Candidate

GroceryKart class

This class will be the main class of the project, the main functions involved in calculating the optimum solution such as Selection, Mutation, Crossover will be invoked by this class. Some of the functions could be:

- generate Population ()
- selection ()
- mutation ()
- crossover ()

Store Class

This class will have all the details about stores in the city where the grocery is to be delivered. It will serve as an entity and will describe all the demand of a store.

Candidate Class

This class will model the solutions to be generated. It will include the best routes available and their respective fitness values.

3.3.3 Output Module

The Output module will provide the required results for the user.

Proposed classes for this module are:

- Store analysis: It will include all the demands of a store and will provide which truck satisfies it.
- Truck analysis: It will provide best path for a truck and total distance travelled by it.

Report Generation

It is a segment of Output module which will conclude the results. It will have a function which will generate 'x' best solutions with

- Maintenance Cost
- Yearly Cost
- No. Of Trucks to be bought
- Total distance travelled

4 Implementation

The Genetic Algorithm has five basic stages:

a. Population Initialization
b. Selection
c. Fitness Evaluation
d. Crossover
e. Mutation

Population initialization, two methods are being used:

1. generatePopulation(): it generates a diverse population

2. greedyGeneratePopulation(): it generates optimum solutions

Selection is being performed by using Tournament Selection concept, which involves running several "tournaments" among a few individuals (or chromosomes) chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover.

Fitness Evaluation is done by using following two methods:
Weighted Sum approach
Pareto Ranking : Pareto is a state of allocation of resources from which it is impossible to reallocate so as to make any one individual or preference criterion better off without making at least one individual or preference criterion worse off.

For **Crossover** , user will be given two choices:
BCRC: Best Cost Route Crossover

BCRC not only maintains the candidates' good nature but improves it too, to drive the force of obtaining better solutions by manipulating existing good candidates.

PMX: Partially Mapped Crossover

PMX aims at keeping as many positions from the parents
To achieve this , a sub-string is swapped like in two-point crossover and the values are kept in all other non-conflicting positions.
Then conflicting positions are replaced by values which were swapped to other offspring.

Example:

p1= (1 2 3 4 5 6 7 8 9)

p2= (4 5 2 1 8 7 6 9 3)

Assume the position 4-7 are selected for swapping.

Then the two offspring are given as follows if we omit the conflicting positions.

o1= (* 2 3|1 8 7 6|* 9)

o2= (* * 2|4 5 6 7|9 3)

Now we take the conflicting positions & in what was swapped to the other offspring. For eg 1 and 4 were swapped. Therefore we have to replace the 1 in first position o1 by 4 & so on.

o1= (4 2 3 1 8 7 6 5 9)

o2= (1 8 2 4 5 6 7 9 3)

Mutation will be done by using Inversion Operator. The length of inversion will be fixed to 2 or 3 which gets selected randomly and assigned to get flipped.

4.1 AUTHENTICATE

Authentication involves the process of identifying an individual, which is usually based on a username and password.

Authenticate(): Default Constructor of Authenticate class which initializes a new User taking User Id and Password as input .The **message** is returned as output by this function.

4.2 SYSTEM DATA

System Data Contains Global Variables, with their Prototypes and default values already set. The Members Of this class are also inherited by Other Subclasses.

Globally declared variables are:

No of trucks	Routing maps
distanceTravelled	CostIncurred
Profit	Efficiency
deliveryTime	userId
storeDemandTable	DCStore Matrix
Generations	populationsize
crossoverProbability	mutationProbability
SelectionPercentage	

Grocerykartform (): Default Constructor of **Super Class** Grocerykartform is used to initialize all the global variables mentioned above.

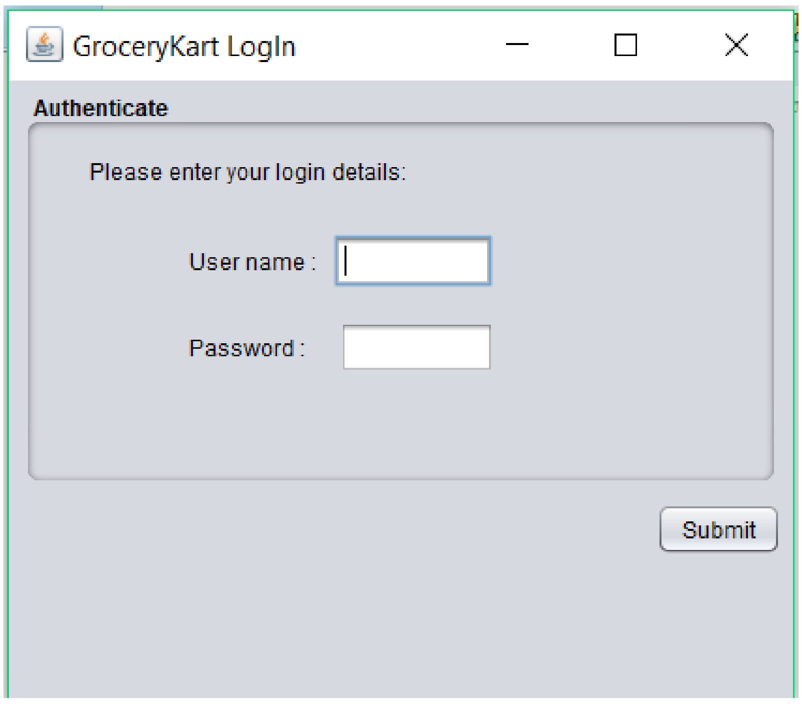
4.3 INPUT MODULE

This class is responsible for retrieving input from the user which mainly consists of Store Demand Table and DC Store Matrix, while all other factors are already containing default values, which can be altered using AdjustFactors class described in section . This class is a Subclass of System data.

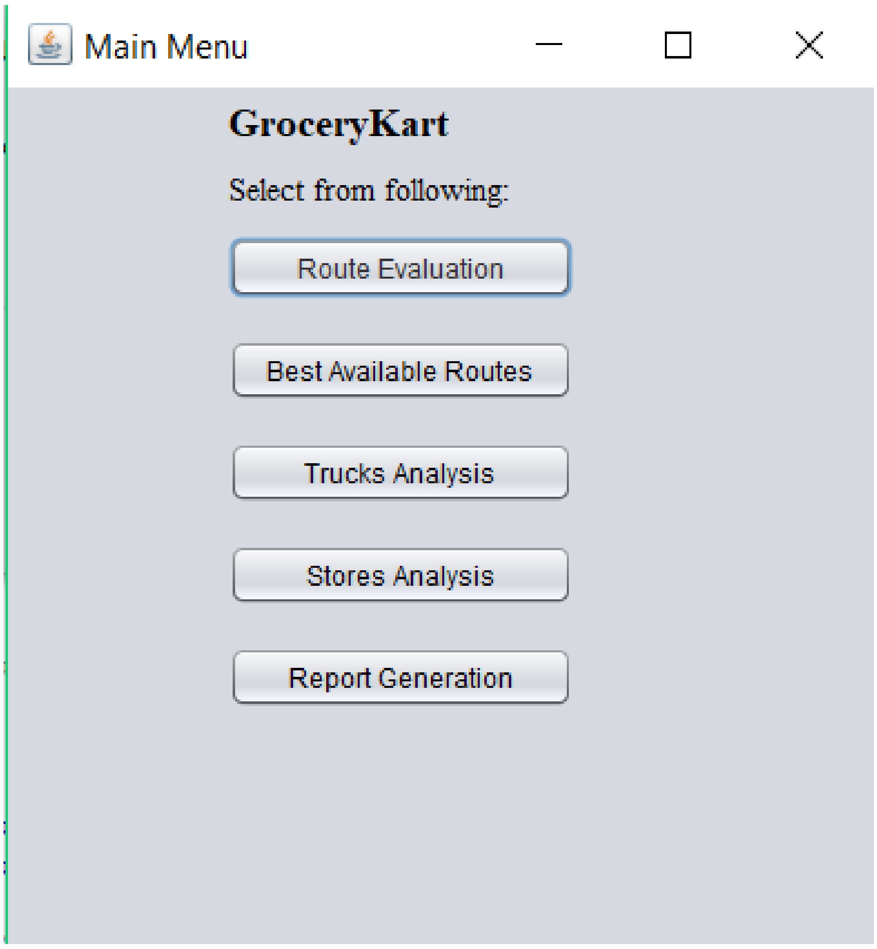
Variables used in this module :

No of stores : Total number of stores.
Maxstops : maximum number of stops at which truck an stop
MaxDemand : total demand of all stores
MaxDistance : Total distance covered by all trucks in one day
MaxDeliveryTime : maximum Delivery time within which demand must be fulfilled.
Read data(): default constructor of class Read data. It invokes mainly two functions:
ReadDemandTable(): it provides the demand of each store.
ReadDistanceMatrix(): it provides the distance of each store from the warehouse and the distance among the stores.

It is the authentication window which requires the username and password as the input from the user to start the working of the grocery kart.

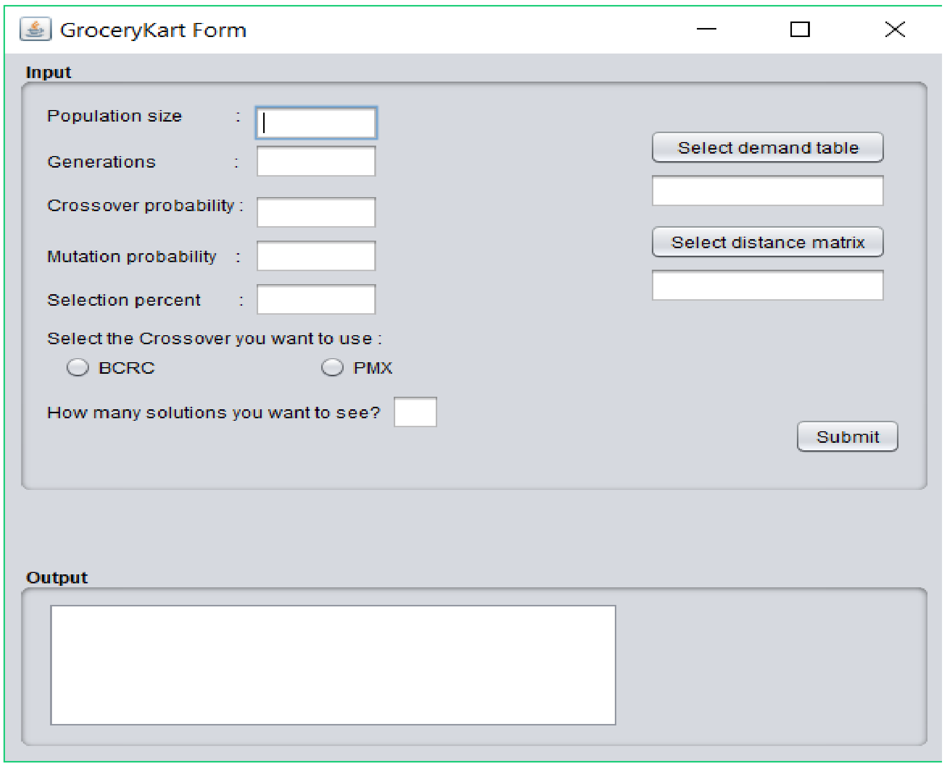


This window is the main welcome window in which the user chooses its options.

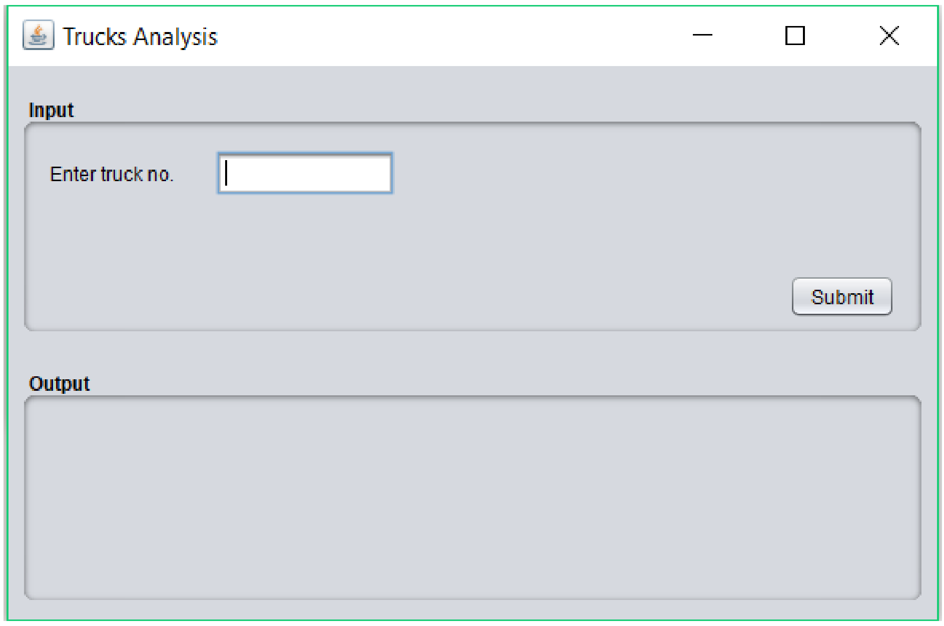


There are 5 choices for the user. The functionality of Best Available Routes, Trucks Analysis, Stores Analysis, Report Generation are dependent on Route Evaluation which takes the required inputs from the user.

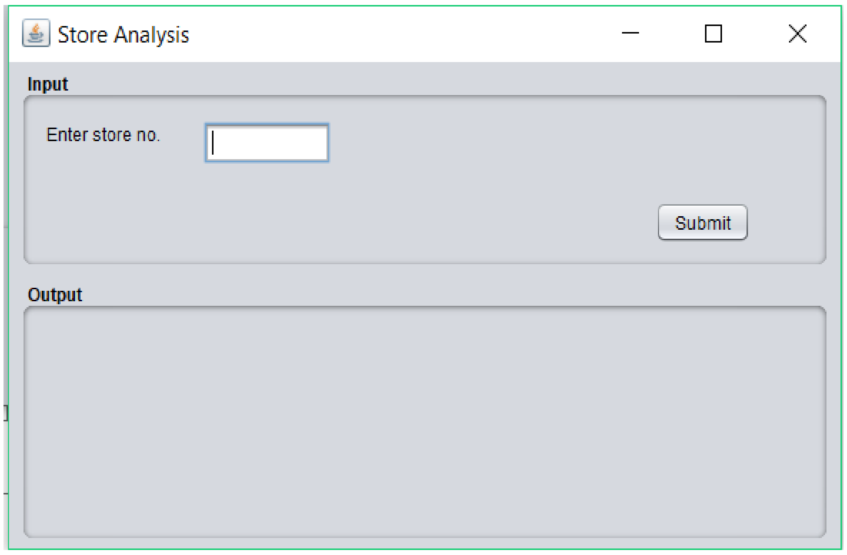
This is the route evaluation window. The user inputs the various fields in order to calculate the various optimum results produced providing the trucks required and the routes and distance travelled by these trucks.



This is the Truck Analysis window. It takes the truck number as the input value from the user to provide various analysis or result on that particular truck bought by the Grocery Kart.



This is Store Analysis window. It takes the Store number as the input value from the user to provide various analysis or other information on that particular store.



AdjustFactors Class

It is responsible for adjusting some factors according to the requirement of the user. Factors such as **mutation probability, crossover probability, and maximum number of stops** can be adjusted easily.

AdjustFactor(): default constructor reinitialises the values of factors to be adjusted.

4.4 Processing Module

The Processing module contains the working mechanism of the algorithm. It defines the flow of randomly generated possible solutions and optimising them to the best solution. It contains the following classes :

GroceryKart
Store
Candidate

4.4.1 GroceryKart Class

This class is the main class in the project, the main functions involved in calculating the optimum solution such as Selection, Mutation, Crossover are invoked by this class. Functions invoked by GroceryKart class are :

Generatepopulation() :
ParetoRankinh() :
TournamentSelection() :
Crossover() :
Mutation() :

4.4.2 Store class

This class contains all the detailing about all the stores in the city where the grocery is to be delivered. It serve as an entity and describes all the demand of a store.

The methods which are invoked in this class are **:**

toString() : to show the representation of store.

4.4.3 Candidate Class

This class models the solution to be generated. It includes the best routes available and the fitness.

4.5 Output Module

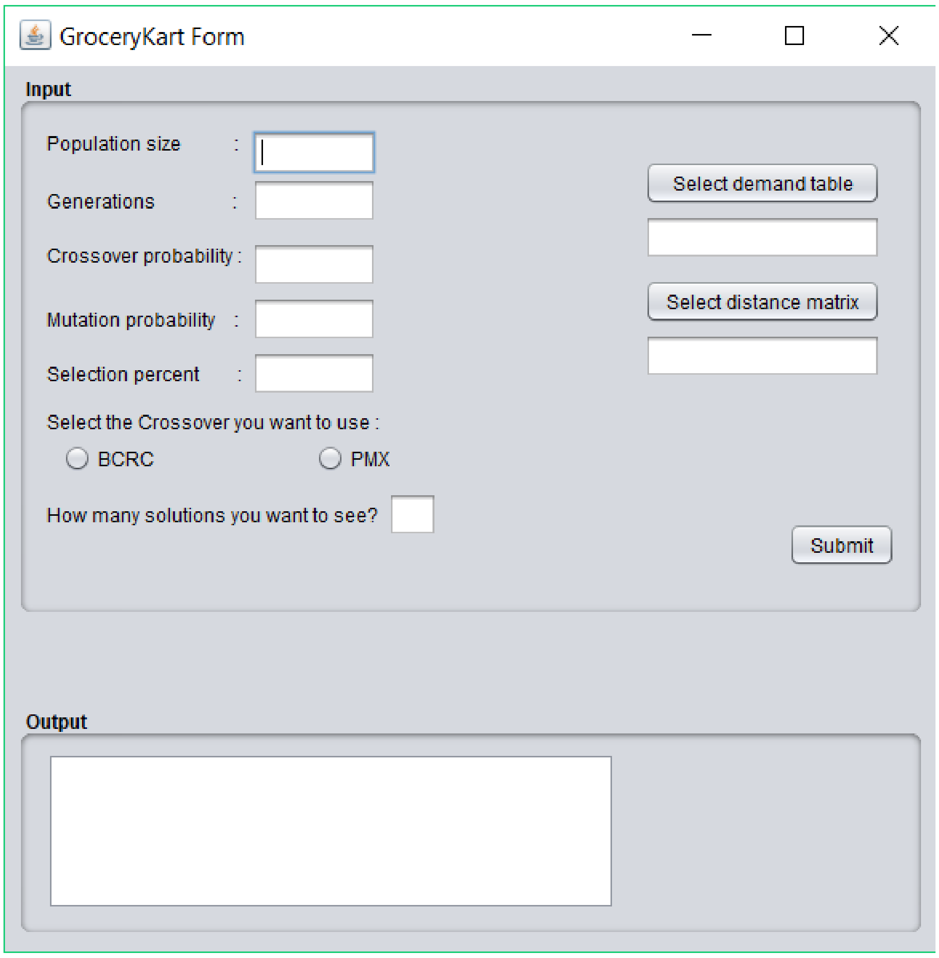
Output Module provides the required results for the user.

Classes used in this module are:

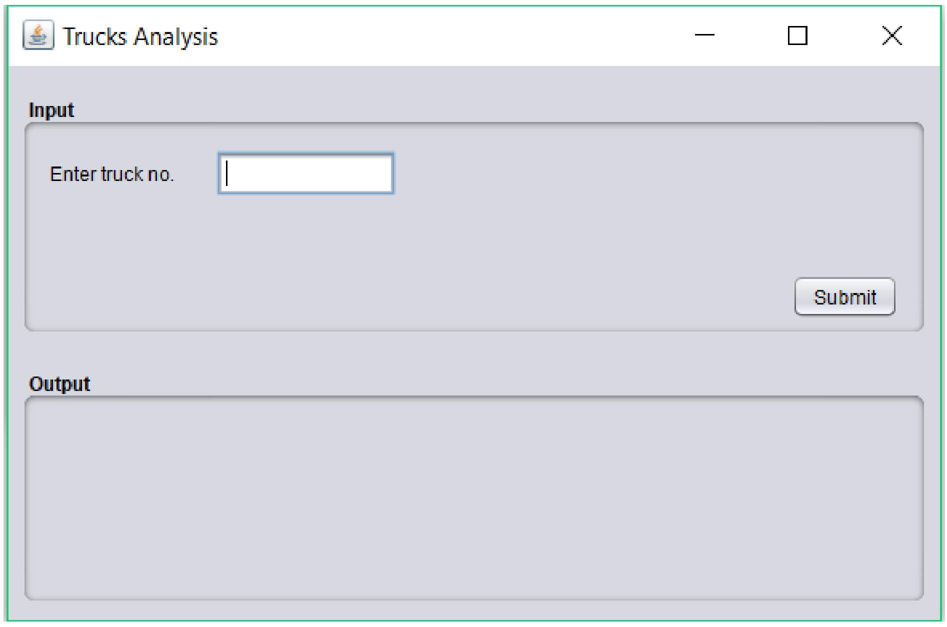
Store analysis : It includes all the demands of a store and provides which truck satisfies it.

Truck Analysis : It provides the best path for a truck and total distance travelled by it.

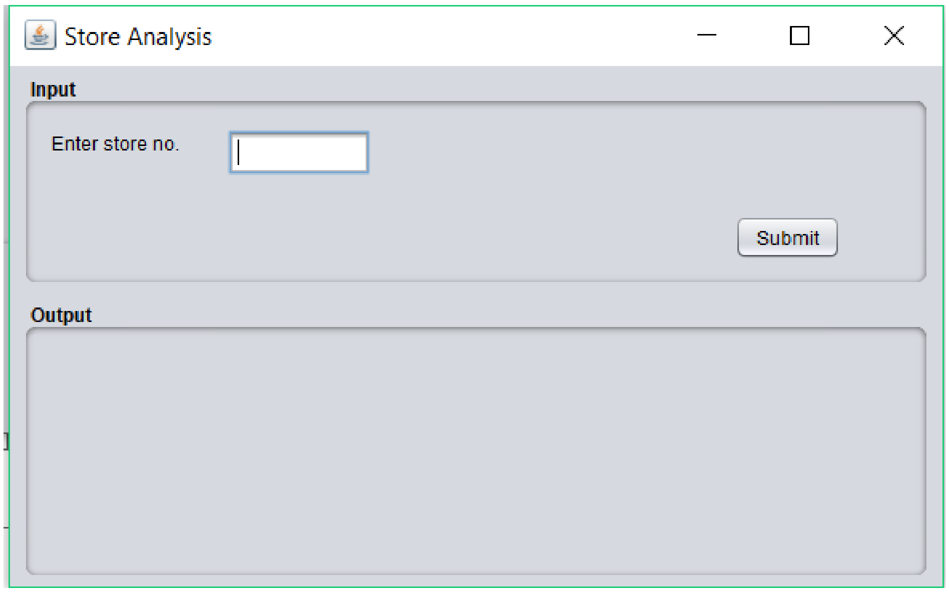
This is window provides various candidate generation with the total distance travelled and maximum number of trucks required the sorts the result according to the ranks.



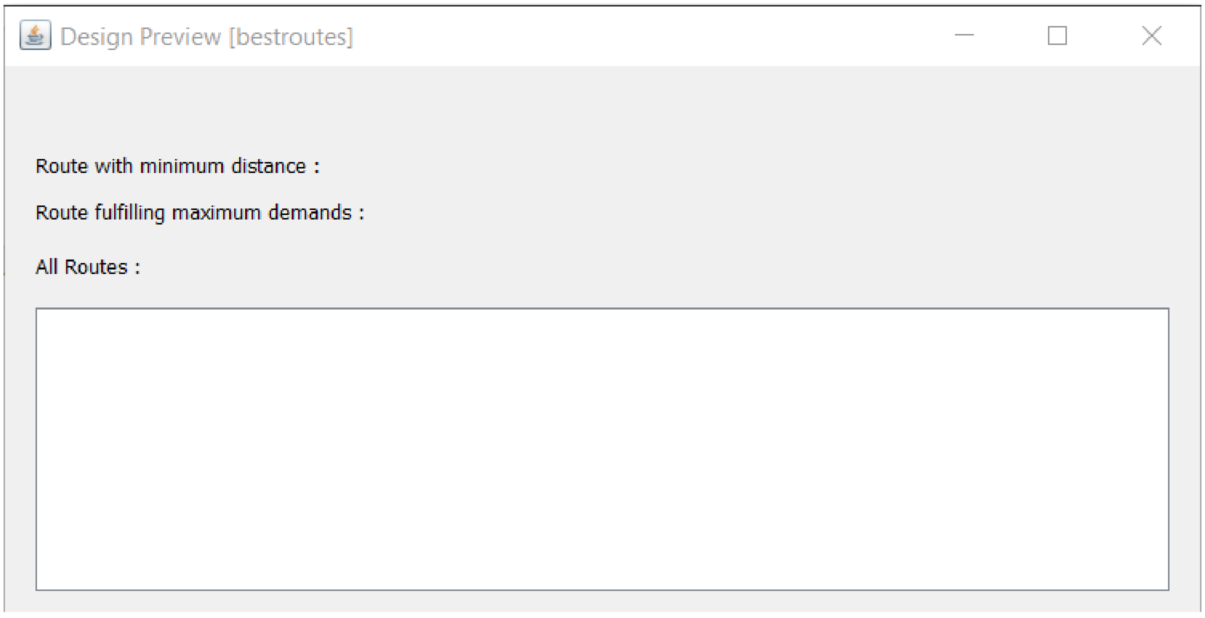
This window shows the result for the truck number specified by the user. It provides the path of that truck (i.e. which all stores it visits) and the distance travelled by that truck and the cost incurred.



This window shows the result for the store number specified by the user. It provides the demand of that store and which truck fulfils it



This window provides us the best available routes for the distribution of Grocery from the warehouse to various stores.


Design Preview [bestroutes]
Route with minimum distance :
Route fulfilling maximum demands :
All Routes :

REFRENCES

[1] Golden, B., Magnanti, L. and Nguyan, H. (1972). Implementing Vehicle Routing Algorithms. Networks, 7(2), 113-148

[2] Wilson, N., Sussman, J., Wang, H. and Higonnet, B. (1971). Scheduling algorithms for dial-a-ride Systems. Massachusetts Institute of Technology, Cambridge.Technical report USL TR-70-13.

[3] Liebman, J. and Marks, D. (1970). Mathematical analysis of solid waste collections. Public Health Service. Publ No 2065. Ms Depot of Health. Education and Welfare

[4] Levin, A. (1971). Scheduling and Fleet Routing Models For Transportation Systems. Transportation Science 5(3), 232-256.

[5] Solomon, M. (1987). Algorithms for Vehicle Scheduling Routing Problem with Time Windows. Operations Research, 35(2), 354-265

[6] Sariklis, D. and Powell, S. (2000). A Heuristic Method, The Open Vehicle Routing Problem. JORS, 51, 564-573.

[7] K.C.Tan, L.H.Lee, Q.L.Zhu, K.OU,2000. Heuristic methods for vehicle routing problem with time windows, Artificial Intelligence in Engineering, 1-15

[8] Mitsuo Gen and Runwei Cheng,2000. Genetic Algorithms and Engineering Optimization, Wiley Series, 340-400

[9] Laporte, G. and Nobert, Y. (1987). Exact algorithms for the Vehicle Routing Problem. Discrete Mathematics, 31, 147-184.

[10] Fischetti, M., Toth P. and Vigo, D. (1994). A branch and bound algorithm for the capacitated vehicle routing problem on direct graphs. Operations Research, 42, 846-859.Fukasawa, R., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E. and Werneck, R. F. (2004). Robust branch-and-cut-and-price for the capacitated vehicle routing problem.

[11] Baldacci, R., Toth, P. and Vigo, D. (2010). Exact Algorithms For Routing Problems Under Vehicle Capacity Constraints. Annals of Operations Research, 175(1), 213-245

[12] Beasley, J. (1983). Route first-cluster second methods for VRP. Omega, 1, 403-408

[13] Clarke, G. and Wright, J.(1964). Scheduling of Vehicles from a Central Depot a Number of Delivery Points. Operations Research, 12, 568-581.

[14] Mole, R. and S. Jameson, S. (1976). A sequential route-building algorithm employing a generalized saving criterion. Operation Research Quarterly, 11, 503-511

[15] Randy L.Haupt ,2004. Practical Genetic Algorithms -2nd Edition , Wiley Series, 200-250

[16] Beatrice Ombuki,B.J.Ross and Franklin Hanshar,2006. Multi Objective Vehicle Routing Problem with Time Windows, Applied Intelligence, 1-14

[17] S.M.SIVANANDAM, S.N.DEEPA,2008. Introduction to Genetic Algorithms, Springer, 170-200

[18] Mitchell Melanie,1999. An introduction to Genetic Algorithm, MIT Press, 160-165

[19] http://www.mu-sigma.com/muphoria/