I4 Challenge: Ideation at Isolation

Challenge IV: Grocery Routing Problem

Solution Report

Name: Sanket Gandhi

Entry no: 2017TT10896

Department: Department of Textile & Fibre Engineering

Email: tt1170896@iitd.ac.in

Sangandhi29@gmail.com

Cell phone no: 9421058059

Challenge No: 4

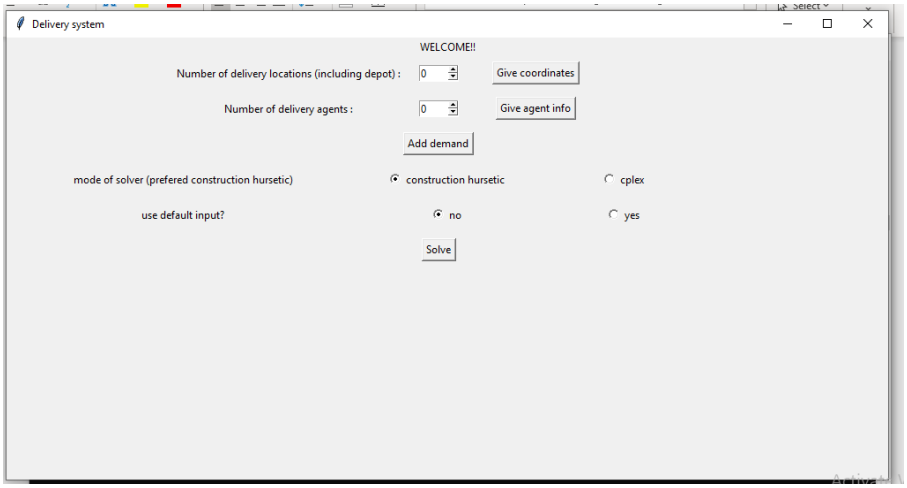
Challenge title: Grocery Routing Problem

Solution

So the problem was divided into two parts as Graphical User Interface and backend Problem Solver. Let's look at it one by one

1. **GUI :**

   As the problem statement mentions we were supposed to make simple GUI which takes delivery location coordinates, demand and delivery agent's capacities. So, we used python programming language to make GUI. We used Tkinter, Numpy and Matplotlib libraries. Here is screen shot of GUI:



   Note: We are not using google map coordinates as input because for use of google map we require API key. This API key requires billing information. Instead we used Euclidean 2D coordinates of location. We can easily switch to google map system if we get api key. Also the conversion of graph information from Distance Matrix to Euclidean 2D coordinate is done by Principle Component

Analysis which only shows around 5% error for 40 nodes when generated randomly.

2. **Problem Solver**:
   The problem statement suggested to use Cplex for solving this VRP and we used it. There are certain limitations for this which are discussed later.
   So we have to satisfy the demand of every customer in given time interval. Now if we convert our objective to minimize total time of operation the solution will not change given it exists. And time minimization is same as distance minimization. So we formulated the problem as total distance minimizing problem given all demands are satisfied
   We used two method to solve the problem.
   a. Constraint Solving
   b. Construction Heuristic

a. Constraint Solving:

This is done by Cplex python module. We defined constraints and variables as following:

N : set of delivery locations
Q: set of delivery agents
C:set of capacity of delivery agent
D: set of demands of customers
V: set of vertices in graph
V = N U (depot)

d_{ij} for all i,jϵ V :distance between i^{th} and j^{th} vertice

x_{ijk} for all i,jϵ V , kϵQ: is binary variable so if it is true it denotes that k^{th} delivery agent will go from j^{th} vertex to i^{th}

u_k for all kϵQ : is continuous integer variable with upper bound as capacity of k^{th} delivery agent and lower bound of zero. Initially it is zero. It denotes the

Objective function:

$\min \sum_{i,j,k} x_{ijk} d_{ij}$ over i,jϵ V , kϵQ , i != j

Constraints:

a. Only one delivery agent should visit customer

$\sum_{j,k} x_{ijk} = 1$ for all i ϵ N and i != j

b. Number of delivery agent coming in and going out of location must be same. That is outgoing number of outgoing agents should be 1

$\sum_{j,k} x_{jik} = 1$ for all i ϵ N and i != j

c. Now third constraint is on the u_k. If delivery agent visits the customer then it's load should increase

($u_k = u_k + D_i$ if x_{ijk} is true for some j ϵ V) for all i ϵ N and k ϵ Q

d. The constraint on capacity of the delivery agent $0 <= u_k <= c_k$ for all k ϵ Q

Limitation of Cplex:

Cplex gives optimum solution but it consumes lots of time. Given n as number of vertices it has time complexity as $O(2^n n^2)$. Now Cplex does not allow free use over certain size of problem. Given I have used free version of Cplex for our problem it does not give answer able 17 vertices.

So, to overcome time complexity for bigger problem we can compensate on solution quality.

b. Construction Heuristic:

In this method we build solution by adding node to given graph such no constraint is violated.

Now there are two important decisions

i) Which node to insert
ii) And where to insert

Node selection:

In our implementation we used maximum regret method to choose the node

Let d_{ki} denotes increase in objective function if i^{th} node is inserted in k^{th} best possible insertion

So, we choose i such that

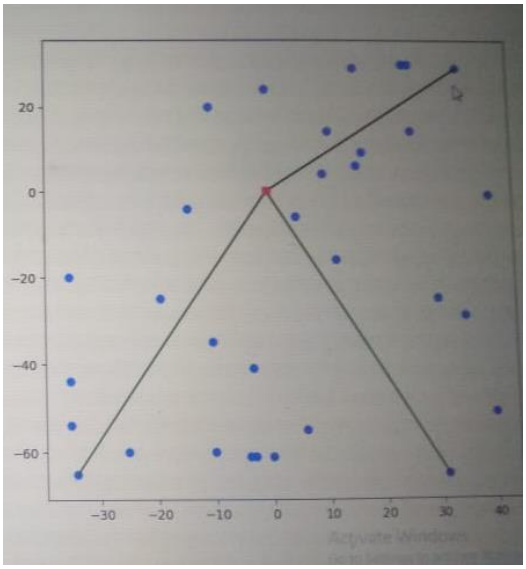
$$\mathbf{(d_{ki} - d_{1i}) \text{ is maximum}}$$

We choose k that is regret depth depending on problem size.

Insertion Location:

We insert chosen node to insertion corresponding to minimum increase in objective function that is location corresponding to d_{1i}

Also, we start with some initial routes. This are selected based on q farthest point.

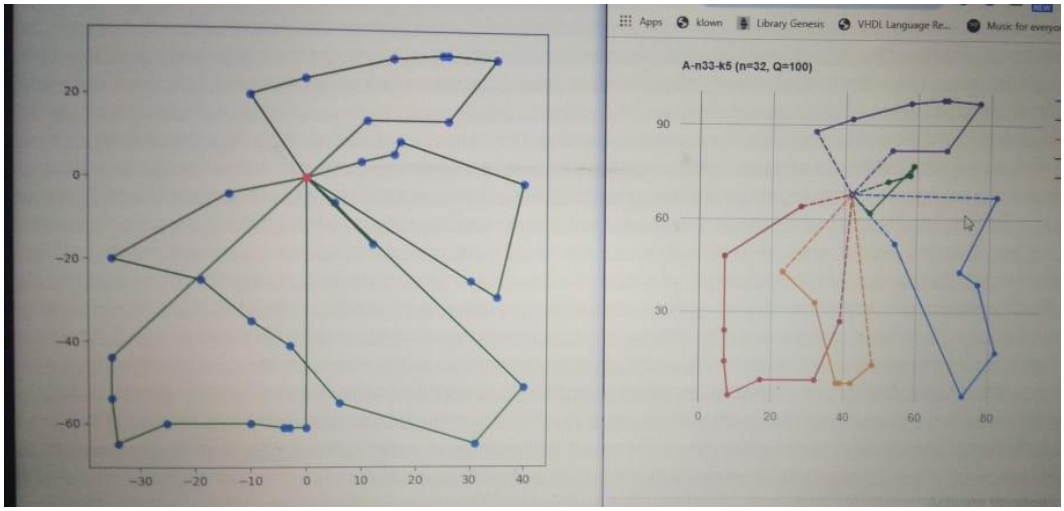


Here q = 3 so we initialize solution with 3 routes

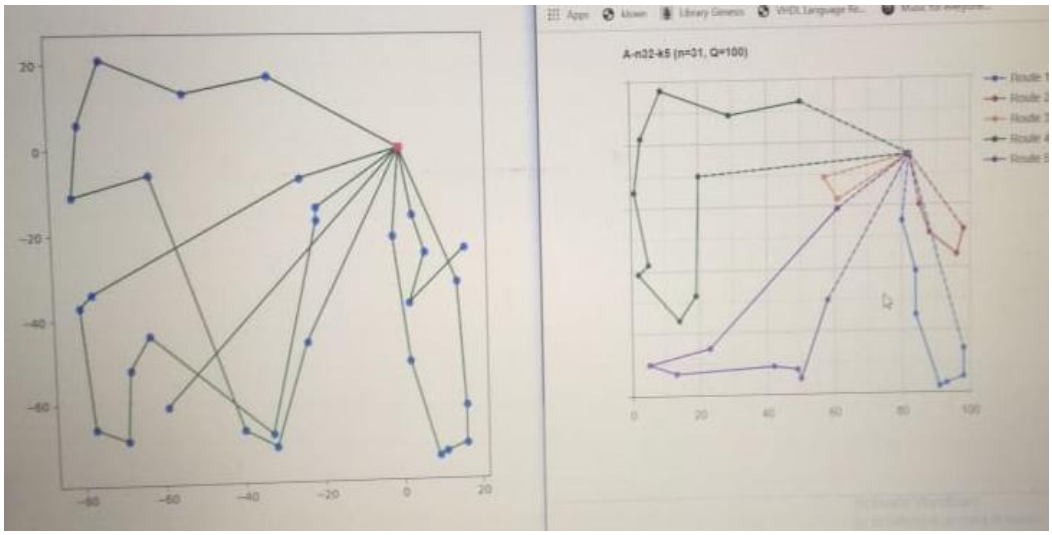
Limitation:

The answer we get from this method may not be optimal.

But by using meta-heuristic we can increase solution quality. The meta-heuristics explore the neighbourhood of solution and take best one.



Hurestic cost: 667 , optimal cost: 630



Hurestic cost: 855 , optimal cost: 784

REFINMENT :

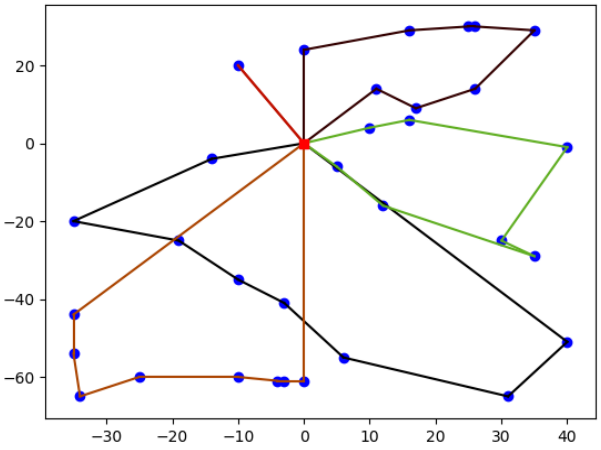
Large Neighbourhood set: (LNS)

This method destroys some part of solution and recreate it. It also accepts large cost solution in order to escape local minimum. Known as *simulated annealing*.

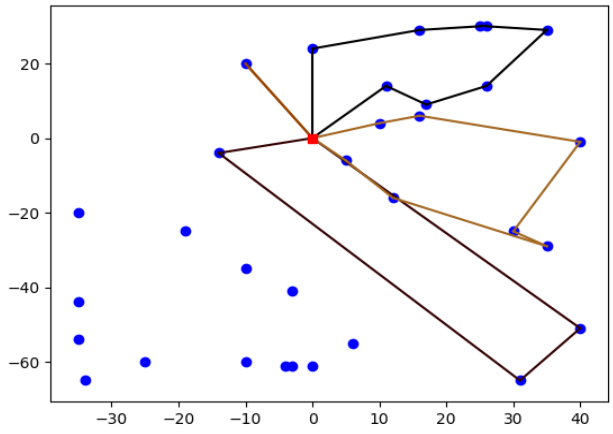
Right one are optimal and left one are solved by construction heuristic without LNS

Example of LNS:

The following one is initial solution without LNS. The cost is 658

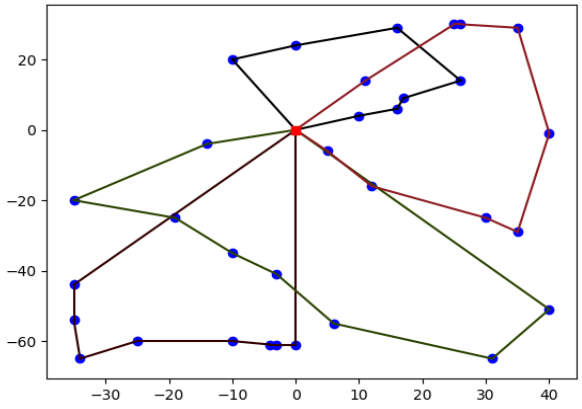


When we apply LNS one time it destroys the solution as



The destroyer take random point and some random radius circle and unassigned all nodes in that circle and then recreate solution.

After few 100 LNS operations



The cost is 635 which is 0.7% larger than optimal with just 100 LNS operations.

Submission

The submission contains following files:

1. Main.py: main file which convert data from GUI to graph.data file and calls appropriate solver
2. Construction_hurestic.cpp: Reads input from graph.data file and solve problem using construction heuristic without using meta heuristic
3. Gui.py: Generates graphical user interface
4. Cp.py: Solves problem using cplex
5. Read_output.py: Generate output graph

Requirement of system:

1. C++14 compiler (optional)
2. Python interpreter
3. Python libraries as numpy, tkinter, matplotlib, cplex

4. To get these libraries go to folder which has this downloaded solution in CMD windows. Run following command
   python requirement.py

How to run platform?

After requirements are completed go for command

python main.py

in same folder. Now fill GUI form. If you just need to see working keep input mode “default”

The **vrp.sol** contain solution as

Agent_0 0 1 4 7 0

Agent_1 0 8 5 0

This means agent 0 should go from 0 (depot) to 4th customer then to 7th and then return to depot.

Novelty and Innovation:

We have implemented the construction heuristics using GUI along with constraint program solving on same platform. The user can decide whether he wants fast answer or he wants optimal answer.

Now when delivery of small items is to be done in particular area getting solution quick is more important than it's optimality and when things have to deliver in different cities optimality is important.

This platform gives both option to users.

Impact:

The cost and time of delivering assignments will be reduced by significant amount. This will cause less pollution. Also, during this COVID19 pandemic the functionality of system should be fast and optimal. The people who wants grocery will not need to step out of there house as this application will be in hand of shopkeepers which will make them optimal to deliver grocery to people. This will reduce interaction between peoples and will help to stop spread of COVID19

Feasibility of implementation:

The platform does not make any assumption on customer numbers and delivery agent numbers so solution can be used for any problem and will be feasible to use it.

Some future ideas:

1.Meta-Heurestic : By adding LSN+ meta heuristic over construction heuristic will increase quality of heuristic solutions

2.Google map: It will be very easy to replace Euclidian coordinates with google coordinates by attaching API Key based patch to platform. So real time tracking and traffic can be incorporated

3.Mobile Application: By adding more java files we will easily be able develop mobile application which will used by both customer and delivery to track and navigate respectively.

THANK YOU!!