Aristotle University of Thessaloniki

Mechanical Engineering Department

Industrial Management Division

Laboratory of Statistics and Quantitative Analysis Methods

ARISTOTLE UNIVERSITY OF THESSALONIKI

Instructions on using the VRP variant solver application

Supervising Professor: Vlachos Dimitrios

Developed by:

Alkiviadis Aleiferis (https://www.linkedin.com/in/alkiviadis-aleiferis-495686190/)

Thanasis Sidiropoulos

Description

The current developed application aims at solving the famous Vehicle Routing Problem in many variant forms, with the Fuel consumption, Heterogenous Capacitated fleet, Vehicle Routing Problem, with Time Windows and Depot loading constraint, the most complex form available. The possible variants are as follows:

VRP: VRP without demands or time windows, obligated to start from depot, contact all nodes and return to depot. Minimizes distance travelled.

C-VRP: VRP with demands and capacitated vehicles (non-heterogenous fleet, m^3 and kg capacities). Node demands must be satisfied with available vehicle capacities.

H-C-VRP: Heterogenous fleet. Many available vehicles with different capacities in m^3 and kg.

H-C-VRP-TW: HCVRP with time windows constraints.

F-H-C-VRP-TW: HCVRPTW with fuel consumption minimization.

F-H-C-VRP-TW-D: FHCVRPTW with constraint loading of the vehicles at the depot.

All the above models can utilize only a single vehicle (if only one is registered in the excel file), making the above models a **TSP** variant (excluding the heterogenous fleet characteristic).

The application can form all the possible subset models. For example, “F-C-VRP”, “VRP-TW-D”, “F-C-TSP-TW”. The only dependencies are that only a capacitated problem can have heterogenous fleet, and only a time windows problem can have a depot loading constraint.

The model is formed by the GUI settings checkboxes (“F”, “C”, “TW”, “D”, characteristics) and the structure of the input excel data (“H”, one or many vehicles). Input data and checked settings must conform for proper functionality (Time windows can’t be enabled without provided windows from the file, FCR must be provided if fuel consumption minimization is enabled).

A template file is available for downloading.

Libraries Used:

Google OR-Tools: Provides the main routing solver engine.

Open Route Service: Provides the distance and duration matrices with API call.

Pandas: Enables the excel data input.

Input Excel File Structure

The programs data input takes place through a properly structured excel file, whose columns formation and data type must be specific. Using the pandas library, the excel file data is stored into a “dataframe” object and then to a Python dictionary whose keys are the Excel file’s columns titles and corresponding values are the columns values stored in a Python list. **The Choose File button opens a file dialog window for selecting the file**.

Every row of the file must be thoroughly complete giving all the necessary data of every location and vehicle. From this rule columns *Depot Docking Capacity (m)*, *Total demand kg,* FCR, Total Demand (m3), **Total avail cap (m3), Total Weight are excluded.**

SHEET: Orders and Locations Data

Locations: In this column appear the location numbers described with a string which can be personally arranged. For example, “Location 10” is a proper way to describe the 10^{th} location, although it can have any other possible string definition. Note: the first location (location zero) is the depot. Data type: String

Description: It is a description of the location formed into a string. Data type: String

Longitude/Latitude: They are each corresponding location’s Longitude and Latitude. They are necessary for the distance and time matrix formulation through the API request. Data type: Decimal

Lower/Upper Time Window: They’re each locations time windows in which the visit must take place, constraining the model. An acceptable time format is 24 hours, for example 18 hours, 0 minutes, 0 seconds (18:00:00 = 6 PM).

Total Demand Capacity kg: It is the total demand of the location in kg. The amount can be manually entered, or a user specific formula can take place inside the file calculating the demand for each location. The depot demand must be zero. Data type = Integer

Total Demand Capacity m3: Same as above but in cubic meters. The depot demand must be zero. Data type = Integer

Service Times: It is the time necessary for servicing the point of delivery. It can be internally calculated same as demand. The depot service time must be zero. Data type = Integer

SHEET: Vehicle Data

Vehicle No: Each vehicles number beginning from 0 and ascending. Data type = integer

Model: It is the vehicles model. Data type = string

Driver id: A string format representation of the driver's identity. Data type= string

Vehicle Cargo Capacity(kg): Corresponding vehicle's capacity in kg. Data type = integer

Vehicle Cargo Capacity(m3): Same as above but in cubic meters. Data type = integer

Vehicle Depot Docking Space Req (m): It's the corresponding vehicle's space in meters required for loading it at the depot. Data type = integer

Vehicle Weight (kg): The corresponding empty vehicle's weight in kg. Data type = integer

Loading Time: It's the vehicle's time necessary for loading it at the depot. Data type = integer

Depot Docking Capacity (m): It is the depot's total capacity in meters for loading vehicles. For example, a docking space of 9 meters can load simultaneously vehicles whose total depot docking space required is 8. Surplus vehicles must wait. The sum of the space required must be less than the depot's capacity. **Note: This is a standalone data. It doesn't correspond to vehicles. Only one value needed**. Data type = integer

Total demand kg/ Total available cap (kg)/ Total Demand (m3)/ Total avail cap (m3)/ Total Weight: These standalone cells are automated and must not be tempered. NOTE: One can choose not to include them, but in case of active Fuel Consumption model checkbox, and FCR value must be provided for the vehicle (liter/km fuel consumption). Otherwise using the template excel file the FCR cell is automatically calculated.

FCR: This cell is the Fuel Consumption Ratio and is also automated and should not be tempered. If the user needs to provide a personally approximated value, should do so if the Fuel Consumption Problem is activated.

Note: cells in the excel template file with this color are automated and must not be changed.

SHEET: FCR func

This sheet stands only for supporting automated calculations and must not be changed.

Tabs Description

Points/Vehicles

These tabs get filed with the excel input data when the file insertion takes place. One can inspect the input data before solving.

Settings

The settings tab enables the ability to choose between solver options, as well as the solving model, ranging between simple distance VRP/TSP and full HCVRPTW with depot resource constraints and fuel consumption minimization. Below each entry and checkbox is thoroughly explained.

Max distance per Vehicle (Meters): It is the maximum distance a vehicle can travel. Works as a restraint in the model. Entering extremely large numbers can alleviate the constraint. Defaults at 250km. Data type = integer

Depot Docking Capacity (Meters): It is the depots docking capacity. It gets filled automatically at file insertion. Changeable if needed in this entry by the user. Data type = integer

Max Solution Time (seconds): It is the max time given to the solver for returning a solution to the model. Data type = float

Select Solver First Solution Strategy: In this menu we can choose the algorithm for the first solution of the model.

Select Local Search (Metaheuristic) Strategy: In this menu we can choose the metaheuristic used for optimizing the first solution. Metaheuristics “kick-in” only after the first solution. Note: They will continue searching the solution space as long as we allow the solver from the max solution time entry. Metaheuristics don’t have an internal verification of optimality.

Is Fuel Consumption Problem: This checkbox enables the fuel consumption model, minimizing the total fuel consumption of all the vehicles. The effect on the objective value will be explained later.

Is Time Problem: This checkbox enables time window constraints. The effect on the objective value will be explained later.

Is Resource Problem: This checkbox enables the loading constraint at the depot, calculating and adding the excessive time for loading in the vehicle's route time cumulative variable.

Note: In case of surplus vehicles necessary for satisfying the demand, the excessive vehicles will still be loaded as if they would deliver. Only way of bypassing this mechanic is checking the surplus vehicle capacity in advance and engaging less vehicles.

Is Distance Problem: It enables the distance dimension of the problem. The effect on the objective value will be explained later.

Is Demand Problem: It enables the demand satisfying constraint of the problem.

Vehicle Max Time (minutes): It's the vehicles max operational time in minutes, constraining the routes maximum available time. Data type = integer

Vehicle Slack Time (minutes): It's the time dimensions cumulative variable slack. This enables the vehicle to wait between stops to achieve the time window, for maximum up to the entered amount of time. Data type = integer

Can solver Drop Locations: Enabling this checkbox gives the model the ability to do two things. First it can alleviate some constraints of the model for no penalty and no node dropping. For example, time windows and vehicle max distances and times, but not demand related constraints, without dropping the whole node for a penalty. Second it can drop a node if the vehicle's capacity cannot satisfy the whole route's demand. This adds a penalty to the objective value, which is user related, and can be seen in the **Penalty Value entry**. *Note: if the penalty is less than the actual cost of the transportation the model will drop all the locations. Enter a very large amount for node dropping only for capacity shortage.* Data type = integer

Objective value configuration

The objective value function is formulated according to the problem checkboxes enabled in the settings tab. Each checkbox enables a different dimension of the model and sets the objective value according to a hierarchy of the dimensions. *However*, the demand dimension is only constraining the model, not affecting its objective function directly. Also, the resource problem checkbox adds a constraint to the already enabled time dimension.

“Is Time Problem” checkbox enables the time dimension, “Is Distance Problem” enables the distance dimension and “Is Fuel Consumption Problem” enables the fuel consumption dimension. Each dimension creates variables attached to the vehicles. *According to the total dimensions enabled a different dimension enforces itself on the objective value.* The hierarchy is as follow: First the fuel consumption dimension if enabled bypasses all the other dimensions enforcing its value on the objective function (liters consumed). Second the distance dimension, if fuel dimension is disabled, sets the objective value (in meters travelled). Last if all other dimensions (fuel and distance, demand doesn’t affect this) are disabled, time dimension sets the objective value in minutes travelled, minimizing operating time. As expected, the model minimizes the current objective value function each time.

Reading the console output

The console output begins with the following lines:

Current First Solution Algorithm: The current choice

Current Metaheuristic: The current choice

Is distance problem: True or False (Boolean)

Is time window problem: True or False (Boolean)

Resource Constraint enabled: True or False (Boolean)

Is demand problem: True or False (Boolean)

Drop visits enabled: True or False (Boolean)

ORS_key: The current Open Route Service Key for the API call.

Total Execution Time: The total time of execution, including model formation times.

Total Solving Time: The solving time alone

Status of solution:

(For example “ROUTING_SUCCESS: Problem solved successfully.”)

Objective value: The objective value of the solution.

After those lines all route’s details are printed concatenated in the following format:

##

ROUTE FOR VEHICLE i:

[0] --> [ ] --> … --> [0] (Sequence of route's nodes formed with index numbers)

Cumulative Load/Distance: [0]: [Load(0 m3 , 0 kg) , Dist(0 m)] --> [node index]: [Load(1 m3 , 100 kg) , Dist(23765 m)] --> ……. --> [0] Load(2 m3 , 200 kg)

(The load is cumulating in contrast to the real scenario, each time by the nodes demand.)

Load of the route in m3: (The total load of the route in m^3)

Load of the route in kg: (The total load of the route in kg)

Distance of the route: (Total route's distance in meters)

Cumulative Time: [0]:Time(0,0) --> [node index]:Time(earliest possible, latest possible) --> …… -->[0]: Time(returning time, returning time)

The parenthesis at each node indicates the time window the vehicle should be in the spot. For example, if the time window of the next node hasn't opened yet, a waiting time (slack time) can occur. Same values in the parenthesis means the vehicle departed instantly after servicing.

Note: Service times are implemented in the model, by adding them in the transport between the nodes.

Time of the route: (Total route's total time in minutes)

Depending on the model activated the following can be read at the end of the output:

##

Total time of all routes: (Sum of all routes' durations)

##

Total distance of all routes: (Sum of all routes' distances)

##

Total m3 load of all routes: (Sum of all routes' load in m^3)

Total kg load of all routes: (Sum of all routes' load in kg)

##