How to run the program?

1. In your IDE, activate “onebyzero_env” virtual environment to avoid “module not found” errors. (If you don’t follow this, you might need to install missing packages in your local environment.)
2. Open the jupyter notebook file named “solution.ipynb”.
3. This notebook has three sections:
    a. INPUTS: Here you have to provide:
        i. the correct path for order data and driver data csv files
        ii. Set “one_driver_one_order_bool”. **When this is set to True, we are solving the variation 1 of the problem**. **When this is set to False, we are solving the variation 2 of the problem.**
        iii. The path of the output file containing the final routes and driver-order assignments are also set according to the above boolean variable.

    b. DATA PREPARATION: This part prepares input data in the required format to be consumed by ortools routing engine.

    c. SOLVE: This part calls the main function to model the problem and run the routing engine. Also, this will publish the output in a .txt file.

How are the results produced?

This problem has been solved using google ortools routing engine.

First data preparation is done.

1. Distance Matrix: We created the distance matrix for the problem using geodesics.
2. Index locations by dividing them into pickup locations and drop locations.
3. Set resources like demands, order counts for all nodes.
4. Set time windows for all nodes.

Steps followed are:

a. Create nodes for locations.
b. Set vehicle start and end nodes.
c. Set travel costs for the routes.
d. Apply constraints:
    i. Pickup and Delivery constraints
    ii. Capacity constraints
    iii. One order at a time constraint (in variation 1)
    iv. Soft Time Window constraints with penalty
e. Set solution search parameters.
f. Search for the solution.
g. Parse the solution to get all routes, driver - order assignments and route metrics.

h. Dump the output file in .txt format.

About program files:

solution.ipynb : Main solution notebook which needs to be run by the end user.

input.py : This reads data from given paths and does basic data sanity checks.

distance_matrix.py : This calculates distance matrix using geodesics and caches it for reusability.

data.py : Main data processing file which prepares everything to become suitable for consumption by ortools.

manager.py : This initiates the ortools routing engine and sets the index manager.

solver.py : This sets the search parameters for the routing engine.

solution_reader.py : This reads and parses the solution in required format.

model.py : This is the main file which calls and combines above files to dump the final solution as a text file.