Univerzitet u Sarajevu
Elektrotehnički fakultet
Sarajevo

UNIVERSITÉ MARIE & LOUIS PASTEUR
STGI

Eco-friendly Carpool Optimization Using Clustering and Travel Data - Developer's Guide

Students: Amina Hromić, Nerma Kadrić, Nadina Miralem, Azra Žunić

The carpooling algorithm was implemented in Google Colab, Eco-friendly Carpool Optimization Using Clustering and Travel Data.ipynb. In order to run the code, an access to Google Colab service is required, or a locally available environment with at least Python 3.8+ installed.

Generating datasets

In order for the algorithm to compile, three inputs in the form of a csv are required: a csv containing information about missions, a csv containing data about passengers, and one with CO2 emissions for various vehicles. The missions csv contains following columns:

- mission_id (int64) - unique ID for each mission
- person_id (int64) - unique identifier for each person
- start_city (string) - city of departure
- end_city (string) - destination city
- start_date (datetime) - date and time when travel begins
- end_date (datetime) - date and time when travel ends
- event (string) - type of event attended (e.g. conference)
- real_move (bool) - TRUE if the person actually travelled, FALSE if it was a planned but cancelled mission
- travel_type (string) - Original travel method used (e.g. plane, car, train)
- vehicle_type (string) - type of car used, if applicable (e.g. diesel, electric)
- is_return_trip (bool) - TRUE if the mission includes a return trip from origin to destination and the other way round, otherwise FALSE
- km (int64) - distance traveled in kilometers
- parking_cost (int64) - parking fees in euros
- hotel_cost (int64) - cost of hotel per night in euros
- plane_cost (int64) - plane ticket cost if applicable in euros
- reimbursement (int64) - amount reimbursed to the traveler
- total_cost (int64) - initial total cost of the mission (excluding optimization effects)
- nearest_airport (string) - name of the airport closest to the departure city

The person csv contains:

- person_id (int64) - unique identifier for each person
- first_name (string) - first name of the person
- last_name (string) - last name of the person
- home_address (string) - full residential address of the person
- admin_address (string) - address of the administrative office the person is affiliated with
- admin_city (string) - city where the administrative office is located
- has_car (bool) - TRUE if the person owns a car, otherwise FALSE
- car_type (string) - type of the person's car (e.g. diesel, electric)
- car_model (string) - specific model of the car
- car_capacity (float64) - number of passengers the car can accommodate (excluding driver)
- fiscal_hp (float64) - fiscal horsepower value of the car used for emissions calculations

For the CO2 csv, the following columns are required:

- car_model (string) - name or label of the vehicle model (e.g. Peugeot 208, average_carpool, TER)
- car_type (string) - fuel or energy type of the vehicle (e.g. petrol, diesel, electric)
- co2_per_km (float64) - average CO_2 emissions in kilograms per kilometer
- energy_per_km (float64) - energy consumption in kWh per kilometer

These csv files can be generated using the Simulating data.ipynb file, where parameters such as the amount of data, content of different columns, output file paths etc., can be adjusted as needed, according to the detailed instructions in the file itself.

Running the code

Once all the data is in the correct format, the user can run the project directly from the provided GitHub repository. To get started, the user can either:

- Clone the repository to their local machine using:

git clone
https://github.com/AHromic1/Eco-friendly-Carpool-Optimization-Using-Clustering-and-Travel-Data.git

- Or open the main notebook file directly in Google Colab by uploading the .ipynb notebook and using the pre-structured Dataset folder included in the repo.

```
missions = pd.read_csv("Dataset/allmissions.csv")
persons = pd.read_csv("Dataset/personsnew.csv")
co2 = pd.read_csv("Dataset/co22.csv")
```

Once this step is completed, all the cells need to be run in order they appear in, either by clicking on the play icon in each cell, or choosing Runtime -> Run all from the toolbar at the top. After a while, the algorithms will produce the results which can be plotted by running the following cells and the results interpreted. It is worth noting that the algorithms were trained on datasets with 300 instances. In case of a slow execution for larger datasets, runtime may need to be upgraded to GPU or TPU, or additional computing units bought through Google Colab services.

Environment setup

In order for the algorithms to run properly, the required Python libraries must be included, via the following imports:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import gower
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from datetime import timedelta
from sklearn.cluster import AgglomerativeClustering
```

In Google Colab, most libraries are preinstalled. Locally, you can install dependencies via "pip install <name of the library>" command, like in the example below:

```
pip install gower
```

To simplify the installation of all required Python libraries, a requirements.txt file is included in the GitHub repository. This file lists all dependencies needed to run the project. Users can install the necessary packages in a single command.

- If running locally, execute the following command in your terminal from the project directory:

  pip install -r requirements.txt

- If using Google Colab, you can either upload the requirements.txt file or clone the repository, then run:

  !pip install -r requirements.txt

Preprocessing of data

The code first preprocesses data by encoding relevant categories using OneHotEncoding and scaling numerical features using StandardScaler.

Clustering algorithms

Once this step is completed, clustering algorithms, KMeans and Agnes are applied, using the Gower distance matrix to handle mixed data effectively. Clustering parameters, as well as the number of features, were adjusted accordingly so that the best results are achieved. This was decided after evaluation and visualization, where Silhouette scores were calculated to assess cluster quality, Principal Component Analysis (PCA) used for dimensionality reduction and to visualize clusters, and intermediate results such as cluster assignments, group sizes, and

cost/emission metrics output for inspection.

Optimization phase

Finally, the code enters the optimization phase, where carpool groups are formed by matching users with similar start/end cities, close travel dates, and available car capacity.
Additionally, car trips are replaced with train trips when train travel is at least 2 hours faster, in order to further reduce CO2 emissions. Finally, updated cost and CO2 emissions are recalculated accordingly. For more details on what each cell does, please consult the detailed explanations in Google Colab notebook.