Univerzitet u Sarajevu
Elektrotehnički fakultet
Sarajevo

UNIVERSITÉ MARIE & LOUIS PASTEUR
UFR STGI

Eco-friendly Carpool Optimization Using Clustering and Travel Data

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

Examiners: Philippe Canalda, Hakim Mabed, Anna Joliot

Montbéliard, 2025

Content

1. Introduction

This project focuses on making travel more efficient and eco-friendly by creating smart carpooling groups. The goal is to reduce both the total cost of trips and the amount of CO_2 emissions by grouping people who travel in similar directions, with a special emphasis on reducing CO2 emissions. Consequently, these were the main metrics monitored for each algorithm. To accomplish this, we used simulated data about trips, cars and drivers. First, we used clustering methods (KMeans and AGNES) to group people based on features like where they start and end their trip, the time they travel, and their car availability. Then, we improved those groups using optimization techniques, and finally, we attempted to use reinforcement learning (Q-learning) to learn better group assignments over time. However, due to limitations in resources, we had to abandon this approach. This combination helped us compare different methods and find the best way to organize carpooling that saves money and protects the environment.

2. Related Work

The study by Shaheen, Cohen, and Bayen [1] outlines the environmental and economic advantages of carpooling, emphasizing reductions in energy consumption, emissions, and congestion. It also discusses how technological innovations, such as real-time ride-matching apps and route optimization algorithms and demographic shifts, like increasing urbanization and the rise of digital-native commuters, are influencing shared mobility trends. The paper provided foundational insights into the broader benefits of carpooling, reinforcing the importance of our project's objectives: namely, minimizing CO_2 emissions, improving carpool efficiency, and reducing overall travel costs. These insights helped us frame our research within the context of sustainable transportation and informed our implementation of optimized clustering strategies for carpool group formation. Building on this, the work by Das, Kalbar, and Velaga [2] introduced a dynamic stock model that evaluates the full lifecycle impact of carpooling, including avoided trips and material emissions. The study's balanced view, recognizing the benefits of shared travel while acknowledging its trade-offs, prompted us to incorporate both direct trip data and vehicle usage characteristics when calculating emissions in our clustering and optimization pipeline. Additionally, [3] explores the integration of machine learning techniques, specifically supervised and unsupervised learning methods for optimizing carpooling solutions. The author emphasizes challenges such as traffic congestion, fuel overuse, and inefficient routing, and evaluates ML-based strategies to improve vehicle utilization and system responsiveness. A key difference between this study and our project lies in the scope and implementation focus. While Pramanik's work presents a conceptual framework and general ML techniques applicable to carpooling, our

project builds a concrete pipeline using clustering (KMeans and AGNES) and optimization steps to form real-world carpool groups based on synthetic mission data. Notably, Pramanik discusses the applicability of algorithms such as decision trees, neural networks, and support vector machines (SVMs), whereas our study employs unsupervised clustering with Gower distance to match travelers based on trip features.The conclusion of Pramanik's work highlights that ML tools, when appropriately selected and trained, can significantly enhance the efficiency of shared mobility services. However, it also notes practical limitations like dataset availability, real-time adaptability, and scalability, which are areas we also acknowledged—particularly the challenge of forming groups larger than two people. Our project validates these findings through empirical results, showing the potential of clustering but also revealing its limitations in maximizing car occupancy. Chang et al. [4] proposed a model that combines machine learning-based predictions with vehicle dispatching in ride-hailing services to maximize emission savings. It introduces machine learning models such as XGBoost-based arrival time prediction and reinforcement learning for vehicle assignment, aiming to enhance operational efficiency and ride-matching accuracy. While our project does not implement time-series forecasting or reinforcement learning models directly, we adopted a comparable approach to trip optimization by leveraging clustering algorithms (KMeans and AGNES) based on temporal, geographic, and vehicle-related features. In a similar sense, we used rule-based predictive scheduling by grouping users whose trips overlapped in location and time (±2 days) and allocating them to carpool groups based on capacity constraints. Both approaches aim to maximize vehicle occupancy and reduce emissions through data-driven trip coordination. Lastly, the work by Campana, Delmastro, and Bruno [5] explores the personalization of carpooling services using machine-learned ranking systems. Their emphasis on user preferences for route, timing, and compatibility informed our consideration of integrating user-centric features into future iterations of our matching logic. Although our current system is primarily based on trip features, this paper highlighted a valuable direction for refining our model using dynamic and personalized inputs.

3. Global architecture

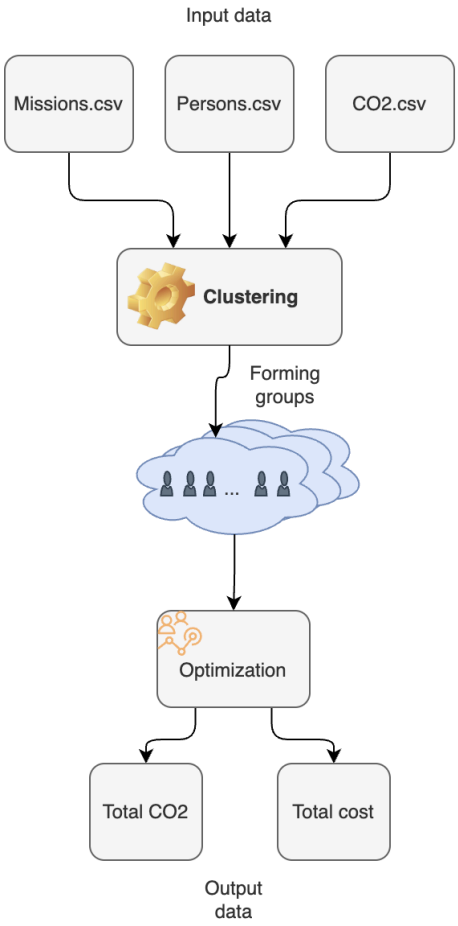

Diagram 1 - global architecture of the project

The global architecture of our carpool optimization system begins with three primary input datasets:

- Missions.csv, which contains detailed information about individual travel missions;
- Persons.csv, providing data about different passengers;
- CO2.csv, which holds emission rates for various vehicle models.

These heterogeneous datasets are integrated and fed into the Clustering module, where machine learning algorithms such as KMeans and AGNES are applied to group travelers with similar trip characteristics. The resulting carpool groups are then passed to the Optimization module, which filters formed groups by taking into account limitations such as vehicle capacity and dates of travel. Additionally, this module considers train travel where applicable. The final output consists of optimized total CO_2 emissions and travel costs, which serve as key metrics for evaluating the sustainability and efficiency of the proposed carpooling solutions, with the goal to minimize these metrics. This global architecture showcases flexible and scalable workflow.

4. Workflow

The project was done in several iterations, with an aim to improve results each time, and various experiments utilized. As can be seen from the previous section 3 (Global architecture), our project consisted of several stages. First of all, we applied a greedy approach, merely putting people together in groups with no optimization whatsoever. These results were used as a baseline, from which we tried to achieve even better results. Afterwards, we applied clustering, followed by the optimization stage. We tried out several different clustering methods, including KMeans, Agnes, DBSCAN, continuing onto the next stage with the most promising ones. Similar was done with the parameters for the chosen clustering algorithms, where fine tuning was used to see which hyperparameters gave the best results in the case of this dataset. Additionally, experiments were done with the number of features as well. All of these experiments will be analyzed in greater detail in the following sections of this work.

Initially, in the optimization stage, we were focusing only on carpooling as a means of travel, without taking into account train or plane. Therefore, we were grouping people together so that every group has at least two passengers, in order to reduce CO2 as much as we can. However, due to some dates of travel not overlapping and some solo passengers not having a car, we had to change these restrictions so that all the people could have been assigned to travel. The additional hotel costs were also taken into account in the cases of passengers who were arriving earlier or leaving later compared to the dates in the dataset, so that they could carpool with someone.

5. Dataset

Due to privacy and GDPR restrictions, the dataset used in this project was simulated and modelled according to the existing dataset of the University Marie and Louis Pasteur (former University of Franche Comte). The original dataset included data on the trips made by university staff to various events, such as conferences, including details such as means of travel, costs made during the trip, the start date and end date of each trip, etc.

The synthetic data was carefully designed to resemble real-world travel missions, including realistic values for departure and arrival locations, travel distances, costs, vehicle types, CO_2 emissions, and time of travel. By simulating this data, we ensured that the testing of clustering, optimization and reinforcement learning techniques could still be done effectively and fairly. Our dataset consists of three core files: personsnew.csv, co22.csv, and allmissions.csv. Several Python libraries were utilized in Google Colab in order to generate and manipulate realistic data. The pandas library was used for creating, transforming, and exporting structured tabular data. numpy

was used to deal with numerical operations and statistical distributions necessary for generating values representing distances, durations, or costs. Python's built-in random module was used to introduce randomness into the dataset, playing the role of a generator for categorical, numerical, and probabilistic choices from predefined values such as a list of addresses. The faker library played a key role in producing realistic fake personal data, including names, addresses, and dates, giving the dataset a human-like quality. In order to make the dataset as similar to the original one, the destinations of the events were all cities in France, and the starting points were the addresses where the campuses of University of Marie and Louis Pasteur were located. Similarly, the names of the people travelling were the ones popular in France. Regarding the generating of travel , the datetime module (specifically datetime and timedelta - difference between two dates or times) was employed. As the result, the three core csv files were obtained:

- Personsnew.csv - contains information about 300 individuals, including their id, personal details (first and last names, home and administrative addresses and cities), car ownership status (if the person has a car) and car attributes (type, model, capacity, fiscal horsepower). These are stored in the columns *person_id*, *first_name*, *last_name*, *home_address*, *admin_address*, *admin_city*, *has_car*, *car_type*, *car_model*, *car_capacity*, *fiscal_hp*. About 64% of the people own a car, with a typical car capacity of 5 and car types like diesel, electric, and hybrid.
- Co22.csv - provides CO_2 emission data for 10 different vehicle models (including train and plane). Each entry contains the model (*car_model*), fuel type (e.g., gasoline, diesel, electric, contained in the *car_type* column), the CO_2 emission rate (*co2_per_km*), and the energy consumption per kilometer, used for electric vehicles (*energy_per_km*). This file was used to enrich the person-car data with environmental impact metrics. The CO2 per kilometer for each type of transportation was chosen based on manual internet research form reliable sources such as European Environment Agency (https://www.eea.europa.eu) and SNFC Voyageurs web site (https://www.sncf-voyageurs.com).
- Allmissions.csv - includes 300 travel missions, each associated with a person, and its id (*mission_id*).. It details the mission's origin and destination (*start_city*, *end_city*), dates (*start_date*, *end_date*), event type (conference, training, meeting, contained in the column *event*), a flag *real_move*, transport mode saved in the *travel_type* column (car, train, or plane), vehicle type (*vehicle_type*), return trip indicator (*is_return_trip*), travel distance in kilometers (*km*), associated costs (parking, hotel per night, plane) contained in the columns *parking_cost*, *hotel_cost* and *plane_cost* respectively, total reimbursement (*reimbursement*), the calculated total trip cost as a sum of parking, hotel and plane costs (*total_cost*), and the nearest airport to the mission's origin (*nearest_airport*). The vehicle_type indicated a type of car for each person with a car, otherwise it was an empty

cell. Regarding the is_return_trip, value “TRUE” indicated that a trip was made in both directions, both from origin to destination, and from destination to origin. The flag real_move was utilized to filter out simulated or test entries. This column indicates whether a trip was actually carried out (value “TRUE”) or merely planned and never took place (value “FALSE”). The cells of the columns containing costs (parking_cost, hotel_cost, plane_cost and total_cost) were never empty, containing zeros if no money was spent. It is also worth noting that all the prices in the table are in euros, as this is the official currency in France. Most trips originate from three main cities: Belfort, Besançon, and Montbéliard, and target seven major cities like Paris, Strasbourg, Marseille, and Toulouse. For simplicity of testing, all of the trips in the dataset took place in May 2025, with a total of 300 missions spread across 11 different days within the month. The busiest travel days were May 4th and May 6th with 36 trips each, followed by May 11th (31 trips), and May 1st and May 9th with 30 trips each.

This approach allowed us to develop and evaluate carpooling strategies in a controlled environment, while maintaining the structure and complexity of a real mobility dataset.

6. Methodology

6.1 Greedy algorithm

As an initial step in our project, we implemented a greedy algorithm to simulate the process of forming carpooling groups based on a real-world scenario. The goal was to assign travelers into groups that respect practical constraints such as driver availability, car capacity, route alignment, and departure timing. Route alignment in this context refers to ensuring that both the start city and end city of a potential passenger exactly match those of the driver's route. In this case, a driver was a person selected from those who own a car, and every other person in that carpooling group was considered to be a passenger. This constraint guarantees that all members of a carpool are traveling along the same path, avoiding detours and ensuring practical feasibility. This algorithm served as a simple but effective baseline for comparison with more complex machine learning approaches that would be explored later.

The algorithm begins by loading and merging three datasets: one containing mission details, another with personal information, and a third with CO_2 emissions per vehicle model, as explained in detail in the previous section. It performs standard data cleaning procedures, such as handling missing values (putting FALSE or 0), converting date

columns to datetime format, and ensuring valid entries for key travel attributes like start and end cities. From the cleaned dataset, the algorithm works by iterating over each traveler. If a person owns a car (indicated by a Boolean flag) and has available capacity, they are selected as a potential driver. The algorithm then searches for passengers whose travel routes exactly match the driver's start and end cities, and whose departure dates fall within a ±2 day window of the driver's scheduled departure, until the capacity of the person's car is filled. This temporal flexibility reflects a realistic assumption that travelers might adjust their plans slightly to enable carpooling, while still respecting individual scheduling constraints. Once a driver is selected, the algorithm fills their car by assigning up to car_capacity - 1 passengers to their group. A group_id is assigned to each formed carpool. This process continues greedily: once a traveler is assigned to a group, they are marked as "used" and are no longer considered for other groups, in order to prevent duplication. Importantly, this ensures that each group has exactly one driver and avoids inefficient situations where multiple drivers travel together or where passengers are left without a designated driver.

Once all possible driver-based groups are formed, the algorithm addresses the remaining unassigned people. These individuals are typically non-drivers who couldn't find a matching carpool group. Instead of leaving them ungrouped, the algorithm creates train-based fallback groups. It looks for others with the same travel route and similar start dates (again within ±2 days) and groups them together under the assumption they would take a train together (or solo if no match is found). These train groups simulate realistic shared travel scenarios for those who couldn't carpool.

After forming the groups, we calculated important metrics: total CO_2 emissions, hotel and travel costs (total cost), and the number of people per group. CO_2 was computed by multiplying distance (km) with vehicle-specific emission values (co2_per_km) from the CO_2 reference dataset. Additionally, we incorporated logic to account for multi-day trips by computing hotel costs based on the number of nights. This version of the algorithm served as a baseline for later evaluations and is visualized in the figure below.

```
Total drivers available: 192
                 co2_total  total_trip_cost  number_of_people transport_mode
carpool_group_id
group_1             62.310              376                 1            car
group_10            15.210              569                 1            car
group_100           14.280             3041                 2            car
group_101           69.200              297                 1            car
group_102          116.640             2287                 4            car
group_103           18.720             1716                 1            car
group_104           42.315              361                 1            car
group_105           69.190              984                 1            car
group_106          122.760              832                 1            car
group_107           18.090              379                 1            car

Total groups formed: 197
Total people assigned: 300
Total CO₂: 11466.47 kg
Total cost: 277350.00 €

Group types:
transport_mode
car      192
train      5
Name: count, dtype: int64
```

Fig. 1 - Greedy algorithm overall results

```
Group group_1 | Mode: car
Route: Besançon → Paris
Date range: 2025-05-01 – 2025-05-01
  - 123 | Victoire Vincent

Group group_10 | Mode: car
Route: Montbéliard → Paris
Date range: 2025-05-01 – 2025-05-01
  - 3 | Christophe Marty

Group group_100 | Mode: car
Route: Belfort → Paris
Date range: 2025-05-06 – 2025-05-07
  - 291 | Richard Guillet
  - 136 | Maggie Perrin
```

Fig. 2 - Greedy algorithm results - carpool groups

This initial greedy algorithm helped us understand the structure and distribution of trips and allowed us to test our early assumptions about carpool group formation. It generated 197 groups and successfully assigned all 300 people, either to cars (192 groups) or train (5 groups). The total cost (€277350.00) and CO_2 emissions (11466.47 kg) from this method are not meant to represent an optimal solution but rather serve as a reference point for evaluating more advanced approaches. Even though these results were solid for an initial step, the limitations of this approach motivated the application of machine learning-based clustering techniques, which aim to create more balanced and efficient groupings by taking a broader view of the data and constraints. The following

pseudocode summarizes the logic of the greedy algorithm in a simplified and abstracted way:

Algorithm 1 Greedy Carpool Matching

```
Sort drivers by start date
for each driver not yet assigned do
    Find matching passengers (same route, within ±2 days)
    Select up to car capacity
    Assign driver + passengers to group
    Mark all participants as assigned
end for
for each unassigned person do
    if has_car = False then
        Assign to train
    else
        Assign to solo car
    end if
end for
Compute total cost and emissions for each group =0
```

Pseudocode 1 - Greedy algorithm

6.2 Clustering algorithms

All clustering algorithms were applied with a different number of features from the dataset. Experiment one was done with only four features, including:

- km: travel distance,
- total_cost: cost of the trip,
- co2_per_km: CO_2 emission per kilometer,
- is_return_trip: categorical variable indicating round trips.

As for experiment two, eleven features in total were used and the results compared afterwards. These eleven features included:

- start_city
- end_city
- travel_type
- vehicle_type
- has_car
- car_capacity
- start_hour (derived from start_date, in ordinal numeric format)
- km

- total_cost
- co2_per_km
- is_return_trip.

Finally, upon analysing results from the first two experiments and upon further reflection, experiment three was conducted as well, including the following features:

- start_city
- end_city
- start_date_ord (derived from start_date, in ordinal numeric format)
- end_date_ord (derived from end_date, in ordinal numeric format)
- is_return_trip.

To prepare these features for clustering, we applied in every experiment:

- **OneHotEncoding** on is_return_trip,
- **StandardScaler** to standardize numerical features,
- **Gower distance matrix** on the transformed dataset.

OneHotEncoding creates a binary column for each category, and it is recommended to be used with nominal data, where each category is equally important, as is the case with this data.

In this clustering phase, we presented and analyzed all experiments and their results, both before and after the optimization stage. Two algorithms: *evaluate_clustered_carpooling()* and *evaluate_optimized_carpooling()* were used for this purpose and will be described in detail in Section 6.3. The core implementation is done using Python and executed in Google Colab. The complete notebook contains approximately 800 lines of code, spanning data preprocessing, clustering, optimization logic and result analysis. The runtime for the full pipeline is under 2 minutes on a standard Colab environment. The code runs without crashing and handles edge cases like missing data or unassignable individuals. Below, we present simplified pseudocode for the key algorithmic components used in the project.

Input: Travel missions, participant data, vehicle CO_2 data (CSV)
Load and merge datasets by `person_id` and `car_model`
Preprocess: convert dates, fill missing values, drop invalid rows
Compute derived features (e.g., `start_hour`)
Apply Gower distance matrix for mixed-type features
Perform clustering (e.g., KMeans or AGNES)
Evaluate clustering with silhouette scores, visualize with PCA
Select best clustering configuration (based on score)
Apply **Greedy Algorithm** for initial carpool formation
Compute emissions, cost, and number of groups
Save baseline results
Apply **Optimization Phase 1: Clustered Carpooling**
Assign people into cluster-based carpools and fallback solo trips
Compute extra hotel costs and CO_2 tax
Apply **Optimization Phase 2: Travel Mode Switching**
Reassign people to train if travel time saving 2 hours
Recalculate CO_2 emissions and final travel cost
Output: Final emissions, cost, tax, number of groups, assignments =0

Pseudocode 2 - global logic

6.3 Optimization algorithms

Following the clustering phase, the next step in our methodology was to refine each cluster by assigning travelers into carpool groups in a way that respects temporal, spatial, and vehicle-related constraints, such as travel dates, car capacity, number of drivers, etc. This optimization layer was essential for transforming the clustering results into realistic group travel plans. The main objective of this phase was to reduce CO_2 emissions, followed by travel costs. To achieve this, we implemented two optimization functions that we used in previous experiments: the carpool group assignment and the one with additional train substitution logic.

Before going into greater detail with these two functions, it is important to mention that, as a result of conducting additional research on CO2 and fuel cost of different means of transportation, in order to create realistic scenarios, travel type “plane” was taken out of consideration and “train” was considered only if the trip was longer than two hours.

We discovered multiple reports [9] and articles [10] stating that the plane emits the highest amount of CO2 in all scenarios, in addition to usually high prices of plane tickets (Fig. 2). Therefore, “plane” was never used in the following algorithms, as it would never be chosen as a means to reduce CO2, which was the primary goal. This is also reflected

in the co2.csv, where it is visible that the least amount of CO2 is emitted by train, followed by electric cars, with the plane emitting the most CO2.

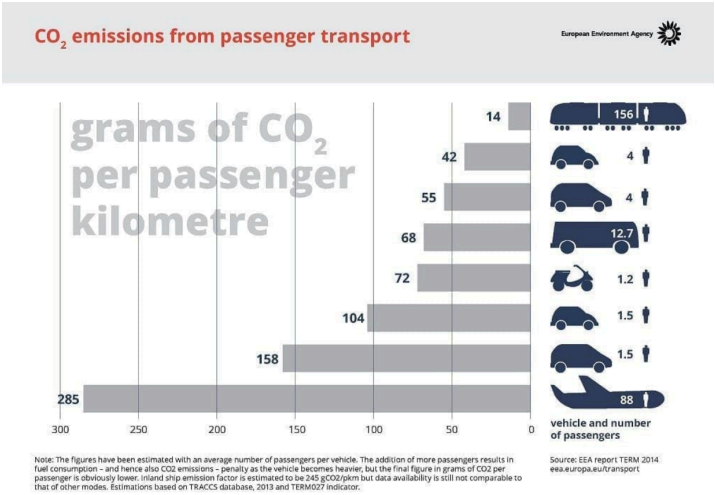

Fig. 21 - Grams of CO2 emissions per passenger transport [9]

Algorithm 2 Initial carpooling fuction

```
Load and preprocess input data from CSV files
Merge missions with persons and CO2 data
Assign participants to clusters using KMeans
for each cluster do
    for each (start_city, end_city) route do
        Identify eligible drivers (has_car = True, capacity > 0)
        for each driver do
            Select up to capacity passengers (same route, ±2 days)
            Assign group with travel_mode = "carpool"
            Mark IDs as assigned
        end for
    end for
end for
for each unassigned person do
    if has_car = False then
        Assign to "train (solo)"
        Compute CO2 using train value
    else
        Assign to "car (solo)"
    end if
end for
Drop duplicates by mission ID
Compute CO2 emissions and extra hotel costs
Calculate total cost, emissions, and CO2 tax
Print group details and summary =0
```

Pseudocode 3 - evaluate_clustered_carpooling() function

The first optimization function, implemented as evaluate_clustered_carpooling(), takes two input files: a CSV containing clustered participant travel data (csv_path) and a CSV with vehicle CO_2 emissions data (co2_path). The function reads both files into DataFrames: df for travel/participant info and co2_df for emission rates, and then converts the start_date and end_date columns to datetime format to allow time-based operations, and fills missing values in key columns: has_car is set to False if not specified, car_capacity to 0, and car_model to "average_carpool" (one of the cells in co2.csv file) for default emission lookups, similar as in the greedy algorithm function. The two datasets are merged by car_model using a left join so that each participant entry has a corresponding co2_per_km emission value from the emissions dataset.

The actual algorithm iterates over each unique cluster label in the dataset (provided by the cluster_col, e.g. from KMeans clustering) and, within each cluster, over each unique (start_city, end_city) travel route. Within each route group, it identifies eligible drivers—those who have a car (has_car = True), have capacity (car_capacity > 0), and have not already been assigned. For each such driver, it selects a limited number of compatible passengers (not already assigned, no car, same route, and whose travel start_date falls within a ±2 day window of the driver's departure). The driver and selected passengers form a carpool group, (travel mode marked as "carpool").
After attempting to group all eligible drivers and passengers, the function handles unassigned participants via a fallback logic. It identifies unassigned individuals from df by checking used_person_ids, and for each one, assigns a solo travel mode: either "car (solo)" if they have a car, or "train (solo)" otherwise, since, as described before, this travel mode emits the least amount of CO2. The CO_2 emissions for solo travelers are computed using their individual co2_per_km (for car) or the TER train rate (for train), and added to carpool_groups.

Finally, all formed groups are stored into a single DataFrame (carpool_df) and total CO_2 emissions across all trips computed, and hotel costs are calculated for any additional nights and the price added to total cost, and all the relevant information is printed out for further analysis. The total cost was calculated by summing the hotel and parking costs from the dataset. The CO2 tax is calculated as a product of total CO2 and CO2 tax rate, in euros. The tax rate is a constant defined in the function as 0.0446, based on the carbon taxes in Europe [20], specifically, France.

Algorithm 3 Evaluate Optimized Carpooling (Mode Switching)

```
Load clustered data and CO2 reference
Preprocess columns and fill missing values
for each cluster do
    for each route (start_city, end_city) do
        Identify eligible drivers
        for each driver do
            Select matching passengers (same route, ±2 days)
            Form group
            if train faster by ≥ 2 hours then
                Assign travel_mode = "train"
            else
                Assign travel_mode = "car"
            end if
            Save group and mark IDs
        end for
    end for
end for
for each unassigned person do
    if has no car or train is faster then
        Assign to "train"
    else
        Assign to "car"
    end if
    Compute CO2 for solo route
end for
Recalculate emissions and costs using travel mode
Compute extra hotel nights and parking costs
Calculate total cost, total CO2, and CO2 tax
Print group summaries and metrics =0
```

Pseudocode 4 - evaluate_optimized_carpooling() function

The evaluate_optimized_carpooling() function further improves the carpooling logic of the clustered version by introducing more intelligent fallback decisions for both groups as well as solo travellers. While both functions group participants into carpools based on clusters and route similarity (matching start/end cities within ±2 days), this optimized version also considers relative travel durations between car and train. For each route, if the train is at least two hours faster than the car, the entire group's travel_mode_optimized is switched from car to train.

Additionally, unassigned individuals (those who weren't placed in any carpool group) are reassessed with an optimized fallback: if they don't have a car or if the train would be significantly faster, they are assigned to solo train travel; otherwise, they travel solo by car. This logic is more sustainable and time-efficient than the simple fallback in the first

function, which always defaulted to car if available. The rest of the evaluation, including hotel cost calculation and the computation of total emissions, travel cost, and carbon tax, remains logically the same but are applied to a more intelligently optimized set of assignments.

Regarding the train, to determine when a car trip should be substituted with a train journey for environmental optimization, we performed a manual estimation of travel durations using trusted online routing platforms such as Google Maps for car travel and SNCF Voyageurs for train travel. These estimates were collected for several common origin-destination routes across France, in accordance with the earlier described dataset. These durations were stored in Python dictionaries (train_durations, car_durations) and later integrated with the optimization algorithms to determine more sustainable transport options where appropriate.

To evaluate the cost-effectiveness of car travel, we calculated the average fuel cost per 100 kilometers using real-world data. According to the ODYSSEE-MURE project [11], the average fuel consumption for new thermal (internal combustion engine) cars in France is approximately 5.2 liters per 100 km. As of May 2025, the average gasoline price in France is €1.67 per liter, based on data from GlobalPetrolPrices.com [12]. By multiplying these figures, the estimated fuel cost per 100 km is: $5.2 \times 1.67 = 8.685$.

As the next step, we considered train travel, particularly via high-speed and regional services such as OUIGO and TER, was also evaluated in terms of cost. OUIGO, the low-cost branch of the French high-speed rail network (TGV), offers ticket prices ranging from €10 to €49, depending on route and booking conditions [13]. TER (Transport Express Régional) fares vary by region and distance, with some round-trip examples (e.g., Bellegarde to Lyon) costing around €52 [14]. Taking into account variability across regions and booking windows, the average train fare for trips exceeding 2 hours was estimated to fall between €40 and €60.

Depending on the type of car, model, age, speed of driving, etc, from these calculations it is obvious that the train ride, despite emitting less CO2, would probably exceed the car ride in terms of cost, which was defined as a secondary metric to be observed and optimized. Therefore, we concluded that a train ride would be more efficient in terms of both cost and CO2 emissions if it is used exclusively for longer rides, which we defined to be the ones exceeding 2 hours. This was taken into account after clustering in order to optimize and refine the groups created from raw data. Since all of the listed origin addresses in the dataset are located in the cities with a train station, usually within a walking distance from the admin addresses, the trip to the train station was neglected at

this stage. Even if a transportation such as bus, car, or tram was to be used to reach the train station from admin addresses, this would be a rather small contribution to the overall CO2 emissions and cost, compared to how much CO2 (and with it, money) is saved by taking a train, as previous calculations show.

6.4 Reinforcement algorithms

Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards through trial and error. Unlike supervised learning, RL does not require labeled data but relies on feedback in the form of rewards or penalties to guide learning. Key elements include the agent, environment, states, actions, rewards, policies (strategies for choosing actions), and value functions (expected future rewards). The goal is to find an optimal policy that maximizes long-term reward. RL algorithms such as Q-learning and deep reinforcement learning algorithms have been successfully applied in diverse areas including game playing, robotics, and autonomous control [19], and were therefore attempted on this dataset as well.

The highest rewards were given to the learning agent for CO2 emission reduction, followed by total cost reduction. However, due to limitations in resources, the work with reinforcement learning had to be abandoned since large quantities of RAM and computing units would have to be spent in order for this amount of data to be processed in reasonable time.

7. Results

7.1 KMeans algorithm (experiment one)

To improve carpool group formation while considering a mix of numerical and categorical variables, we applied the KMeans clustering algorithm using Gower's distance as a similarity metric. Gower's distance is well-suited for heterogeneous data and enabled us to incorporate features of different types effectively.

KMeans is a widely used unsupervised machine learning algorithm that partitions a dataset into k distinct clusters by minimizing the within-cluster variance. The algorithm begins by randomly selecting k initial centroids. It then iteratively performs two steps: (1) assigning each data point to the nearest centroid based on a distance metric (typically Euclidean distance), and (2) recalculating the centroids as the mean position of all points within each cluster. This process repeats until the centroids stabilize or a maximum number of iterations is reached. KMeans assumes that clusters are convex and isotropic (spherical in shape), and that they contain similar numbers of points. It is sensitive to the initial placement of centroids, which may lead to different clustering outcomes on different runs unless a consistent initialization strategy is applied.

Despite these limitations, KMeans is highly efficient and scalable, making it suitable for large datasets. It has found wide applications in various domains such as customer segmentation, image compression, anomaly detection, and document clustering where it helps uncover natural groupings in data for targeted analysis and decision-making [15]–[18]. Therefore, we decided to apply KMeans in our project to cluster travel-related data featuring both categorical and numerical attributes. To address the challenge of mixed data types, we used Gower's distance metric, which allows effective integration of heterogeneous features.

This preprocessing ensured that distance calculations reflected both the numerical magnitude and the categorical structure of the data. We trained the KMeans model on the Gower matrix using three different values for the number of clusters: 10, 30 and 50. These values were selected based on both research and our own results. In particular, Phaam, Dimov and Nguyen [21] argue in their paper, describing methods for selecting *k* in K-means, that many studies evaluate **k** at values such as 10, 20, 30, 40, and 50, particularly on datasets of a few hundred samples, such as this one. Furthermore, another survey [22] highlights discrete values such as 10, 30 and 50 a common choice when it comes to applying KMeans algorithm, together with metrics such as Silhouette score to assess the results. The Silhouette Score [23] is a metric used to evaluate the quality of a clustering solution by measuring how similar a data point is to its own cluster compared to other clusters. It combines two key aspects: cohesion (how close a point is to others in the same cluster) and separation (how far a point is from points in other clusters). The score ranges from –1 to +1, where a high value indicates that the point is well matched to its own cluster and poorly matched to neighboring clusters. A score close to +1 reflects dense and well-separated clusters, a score around 0 indicates overlapping clusters, and a negative score suggests that points may have been assigned to the wrong cluster.
To evaluate the clustering quality, we used the Silhouette Score as well:

- k=10: 0.269
- k=30: 0.258
- k=50: 0.281.

In our case, all the scores are similar and all are showing clustering of moderate quality, with k=50 showing a slight advantage over other values. PCA was applied for dimensionality reduction and cluster visualization, another common practice when it comes to clustering. The PCA plots (shown on Figures 3, 4 and 5) show moderate separation, with more compact and balanced cluster distribution at higher k.

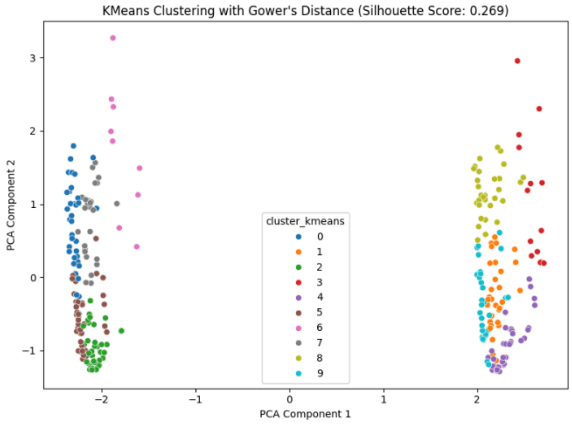

Fig. 3 - Kmeans k=10

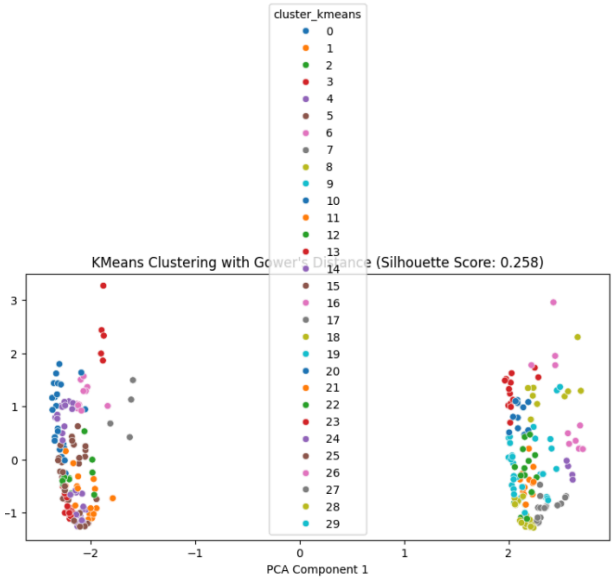

Fig. 4 - Kmeans k=30

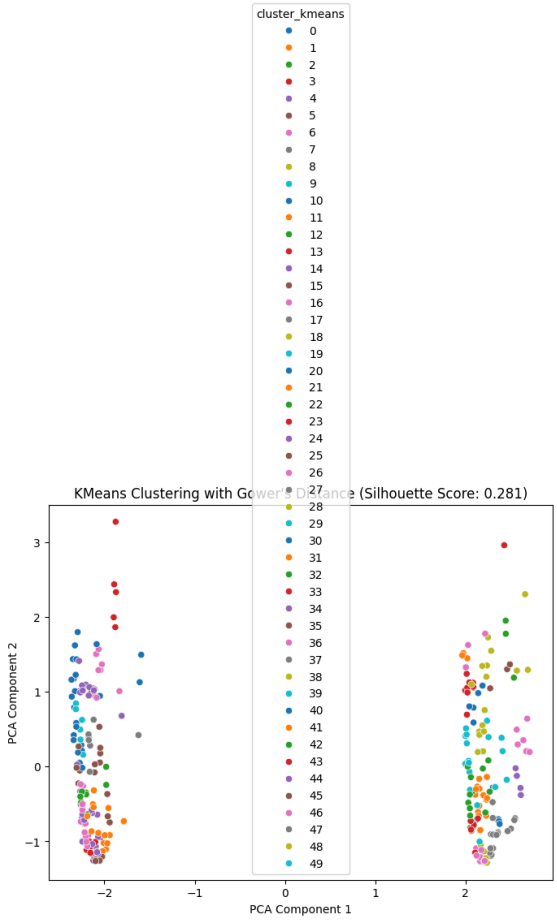

Fig. 5 - Kmeans k=50

After clustering, we formed carpool groups from each cluster by applying the function *evaluate_clustered_carpooling()*, as described in the Section *6.3,*

We ran this evaluation script separately for k=10, k=30, and k=50 (results are shown below on Tab. 1), calculating:

- Total cost of all groups,
- Total CO_2 emissions,
- Number of people traveling.

k	Total cost (€) before opt.	Total groups before opt.	Total cost (€) after opt.	Total groups after opt.	CO_2 (kg) before opt.	CO_2 (kg) after opt.
10	**62020**	**272**	**62020**	**249**	**11909.28**	**8951.28**
30	**62020**	**288**	**62020**	**283**	**11983.22**	**8653.8**
50	**62020**	**294**	**62020**	**291**	**12021.06**	**8563.72**

Tab. 1 - Results before and after optimization - Kmeans (Experiment 1)

The table (Tab.1) shows results before optimization (upon applying a more simple evaluate_clustered_carpooling function on clusters, used purely for evaluation) and results after optimization (after applying evaluate_optimized_carpooling function, with some optimization mechanisms).

So, increasing the number of clusters leads to smaller and more homogeneous groups, However, a trend of increasing the number of groups needed to be formed and CO2 emissions as k increases is noticeable, with total cost remaining constant. To visualize and compare the results, a log-scale barplot was used to highlight absolute differences even when metrics differ by an order of magnitude. This visualization (Fig. 6) confirms that k=10 is the optimal setting, minimizing both cost and emissions (accounted for in CO2 per kg and CO2 tax).

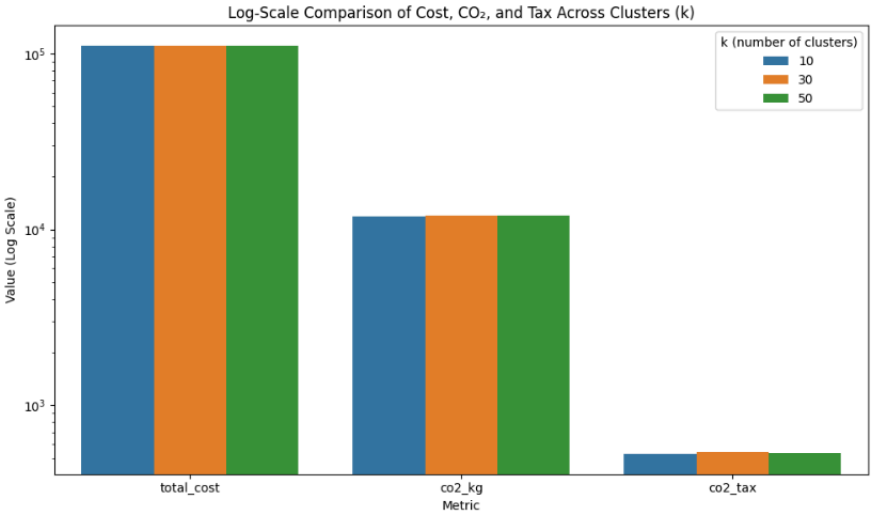

Fig. 6 - Visualization of suboptimal results

Although the cost is reduced compared to the greedy algorithm, CO2 emissions are increased after the first phase (columns in the Tab.1 noting the results before the optimization), which is the opposite of the goal. Therefore, we introduced a second optimization phase, earlier mentioned as *evaluate_optimized_carpooling()* function.

In the Tab.1, "before opt." results represent the application of the *evaluate_clustered_carpooling()* function, while "after opt." values are the results obtained from the *evaluate_optimized_carpooling()* function.

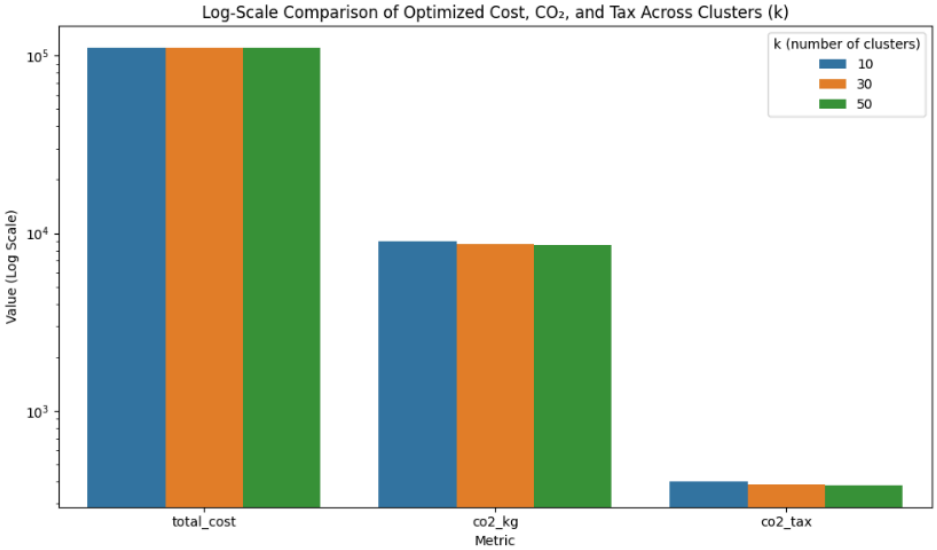

Fig. 7 - Visualization of optimized results

This final optimization decreased total emissions significantly (as well as CO2 taxes), while keeping the price constant. The fact that the price remained constant across different clusters points to a conclusion that, most likely, the clustering consistently grouped people travelling on the same dates, therefore no additional hotel costs occured, or at least similar dates, balancing out the additional hotel costs at the end. Therefore, consistent clustering was achieved with various k values, as expected given that their Silhouette values were similar. Also expected from the Silhouette score, the clustering with k=50 gave the best results, with the amount of emitted CO2 minimized.

```
Group group_1 | Cluster: 37
Route: Besançon → Dijon
Start Dates: 2025-05-09 – 2025-05-09
Driver: Gabrielle Peltier (ID: 270)
(No passengers – solo driver)

Group group_10 | Cluster: 26
Route: Besançon → Paris
Start Dates: 2025-05-03 – 2025-05-03
Driver: Xavier Carlier (ID: 116)
(No passengers – solo driver)

Group group_100 | Cluster: 17
Route: Belfort → Nice
Start Dates: 2025-05-05 – 2025-05-05
Driver: Gabriel Cohen (ID: 35)
(No passengers – solo driver)

Group group_101 | Cluster: 17
Route: Besançon → Strasbourg
Start Dates: 2025-05-05 – 2025-05-05
Driver: Nicolas Léger (ID: 98)
(No passengers – solo driver)

Group group_102 | Cluster: 17
Route: Montbéliard → Dijon
Start Dates: 2025-05-09 – 2025-05-09
Driver: Emmanuel Pelletier (ID: 97)
(No passengers – solo driver)
```

Fig. 8 - Detailed carpool groups - KMeans k = 50 (Experiment 1)

7.2 AGNES algorithm (experiment one)

After evaluating KMeans clustering, we implemented AGNES (Agglomerative Nesting) as a hierarchical alternative. Unlike KMeans, which partitions data into flat clusters based on centroid minimization, AGNES constructs a hierarchy of nested clusters through a bottom-up approach. The algorithm begins by treating each data point as its own singleton cluster. At each iteration, it identifies the two clusters that are closest to each other according to a chosen distance metric and merges them into a new cluster. This process of successive merging continues until all data points are grouped into a single

cluster. AGNES offers several advantages in our context. Firstly, it is deterministic and less sensitive to initialization randomness, which improves stability and reproducibility. Secondly, it can capture multi level groupings in the data, revealing both small, tight subgroups and larger overarching clusters. This hierarchical structure is particularly useful when the true number of clusters is not known in advance, or when interpretability is important.

As emphasized by Murtagh and Contreras [6], hierarchical clustering, especially with average linkage, is well-suited for datasets that combine numerical and categorical features and is widely applied in behavioral and transport data modeling. To accommodate the mixed data types in our dataset, we computed Gower's distance, which handles heterogeneous variables by combining normalized differences for numerical features and similarity matching for categorical ones. The same preprocessing steps applied in the KMeans experiment were used here.

After preprocessing, we applied AGNES using the cosine distance metric [8], which measures angular similarity between data points and is especially effective for high-dimensional spaces where direction matters more than magnitude. We paired this with average linkage [8], which computes the average pairwise distance between all elements of two clusters when determining which to merge. This combination, cosine distance and average linkage, balances sensitivity to the shape and spread of clusters. It has shown effectiveness in transportation and behavioral modeling, as demonstrated by Tan et al. [7], where cosine distance captures directional similarity and average linkage ensures consistent cluster cohesion.

We tested three different values for the number of clusters, denoted as $n \in \{10, 30, 50\}$, based on research and experimentation. Values are the same as those for k in KMeans, with the purpose of a fair comparison. The silhouette scores obtained were 0.423 for n = 10 and 0.416 for n = 30 (both shown in Fig. 9), while the highest score of 0.477 was achieved at n = 50 (Fig. 10). These results indicate that increasing the number of clusters generally improved intra-group cohesion and inter-group separation, as reflected in the silhouette metric.

Visually (based on PCA projection), the clusters were fairly well separated in all three cases, with n=50 showing the best compactness and separation.

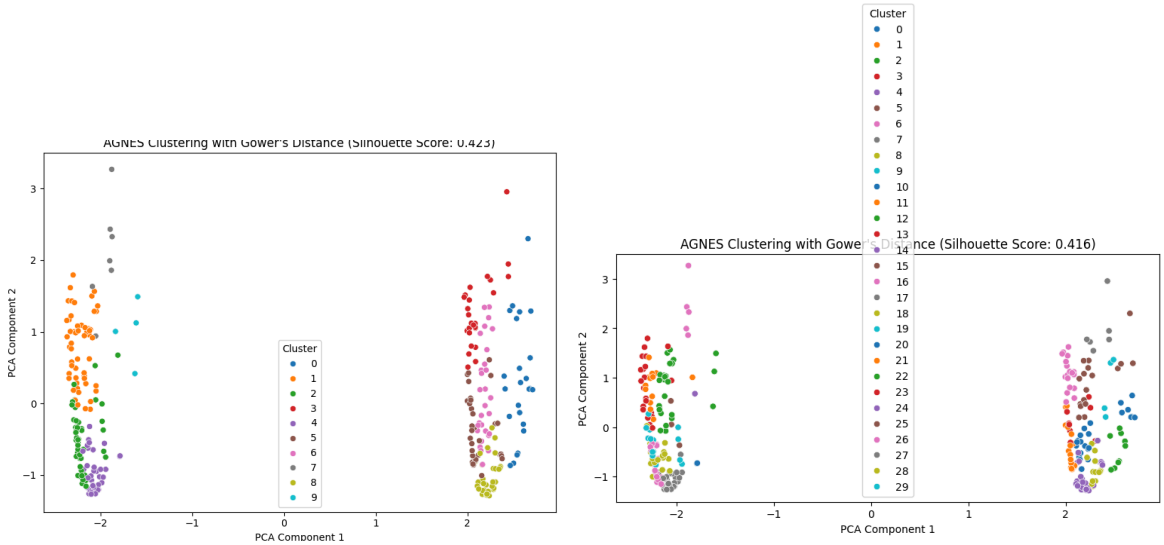

Fig. 9 - AGNES n=10 (left) and n=30 (right)

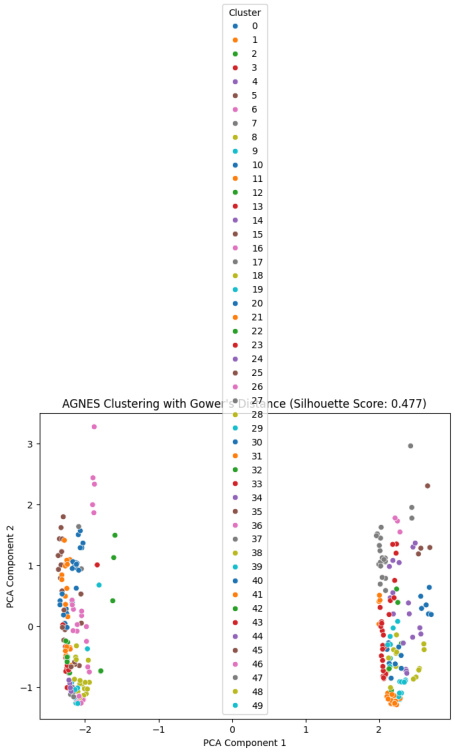

Fig. 10 - AGNES n=50

We formed carpool groups within each AGNES cluster by the same way as for the KMeans. Then we calculated total CO_2 emissions and trip cost for each n (Tab. 2). We observed that increasing *n* reduced both cost and emissions, similar to the pattern in

KMeans. A higher number of clusters (e.g., $n = 50$) results in finer granularity, meaning that travelers are grouped into smaller, more homogeneous clusters based on their travel characteristics. This allows for more precise matching of individuals with similar routes and schedules, thereby improving the efficiency of carpool group formation. To visualize comparison between these versions we made a barplot, which confirmed that n=50 is the optimal setting, minimizing both cost and emissions.

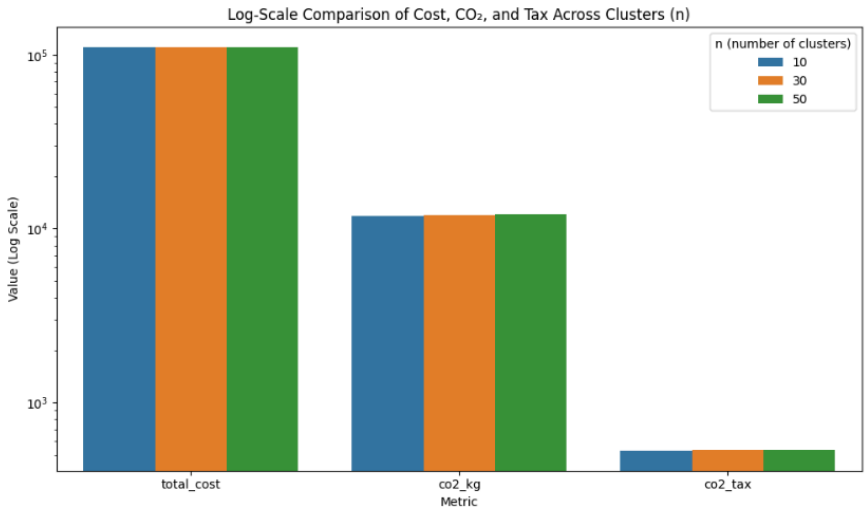

Fig. 11 - Visualization of suboptimal results

Next, we optimized the carpooling groups by switching transport mode from car to train if train travel was at least 2 hours faster (with *evaluate_optimized_carpooling()* function).

n	Total cost (€) before opt.	Total groups before opt.	Total cost (€) after opt.	Total groups after opt.	CO_2 (kg) before opt.	CO_2 (kg) after opt.
10	**62020**	**270**	**62020**	**245**	**11889.39**	**8939.09**
30	**62020**	**290**	**62020**	**282**	**12002.32**	**8611.17**
50	**62020**	**298**	**62020**	**293**	**12041.48**	**8447.94**

Tab. 2 - Results before and after optimization - AGNES (Experiment 1)

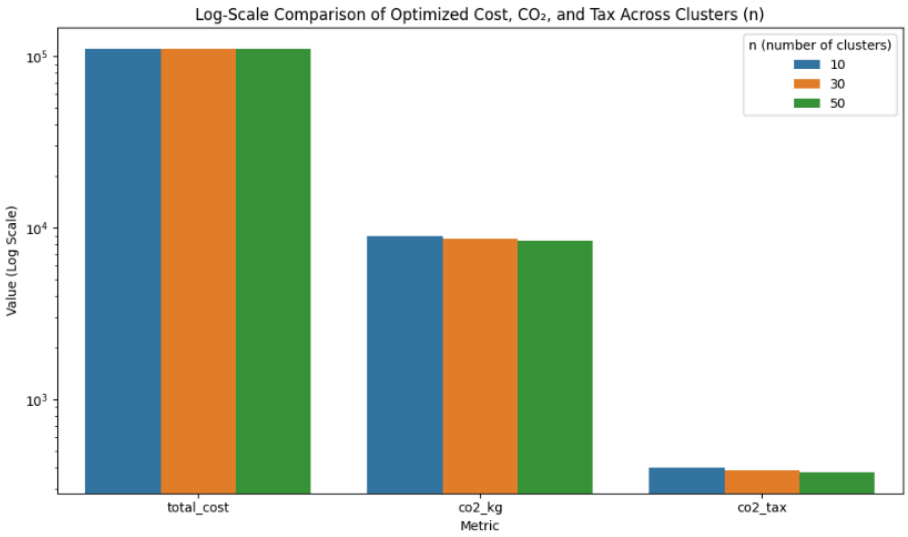

Fig. 12 - Visualization of optimized results

The AGNES clustering approach with Gower's distance yielded meaningful group structures for carpool optimization. However, as seen in Table 2, the best trade-off between cost and CO_2 emissions was achieved with n=50 for the optimized version, resulting in the lowest CO_2 emissions and tax. The best results before the optimization were achieved with n=10, similar to KMeans results. This confirms that finer clustering (higher n), combined with more intelligent optimization, leads to more precise grouping and significant environmental and economic benefits. Regarding the cost, a similar conclusion can be made as in the case of the KMeans algorithm, which shows that the total cost is more dependent on the restrictions set in the optimization function, rather than clustering algorithms. Additionally, we can conclude that Agnes algorithm showed improvement in the optimization results, compared to KMeans.

7.3 KMeans algorithm (experiment two)

An identical approach was used, but with a different logic when it came to selecting features, as explained at the beginning of this section. Using KMeans with Gower's distance, we tested k=10, 30, and 50 clusters and evaluated both clustering compactness and optimization performance. Although the silhouette scores remained relatively low and similar to each other, with k=50 again yielding the best results (0.176 for k=10, 0.181 for k=30, and 0.187 for k=50).

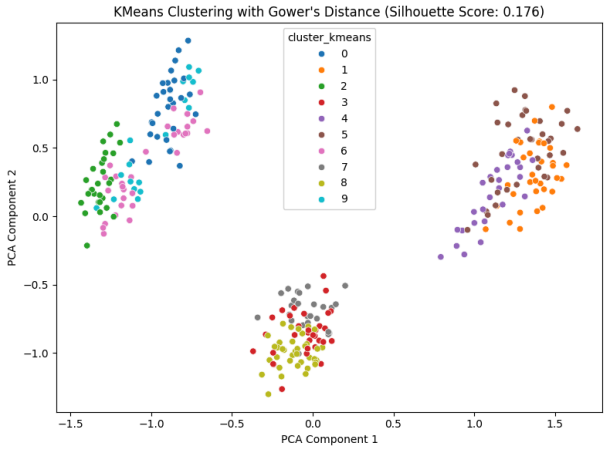

Fig. 13 - KMeans k=10 (Experiment 2)

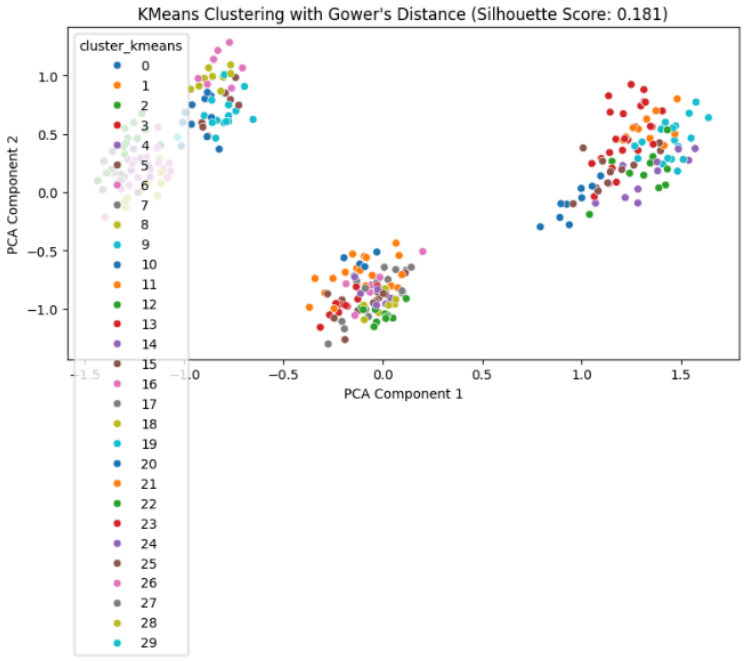

Fig. 14 - KMeans k=30 (Experiment 2)

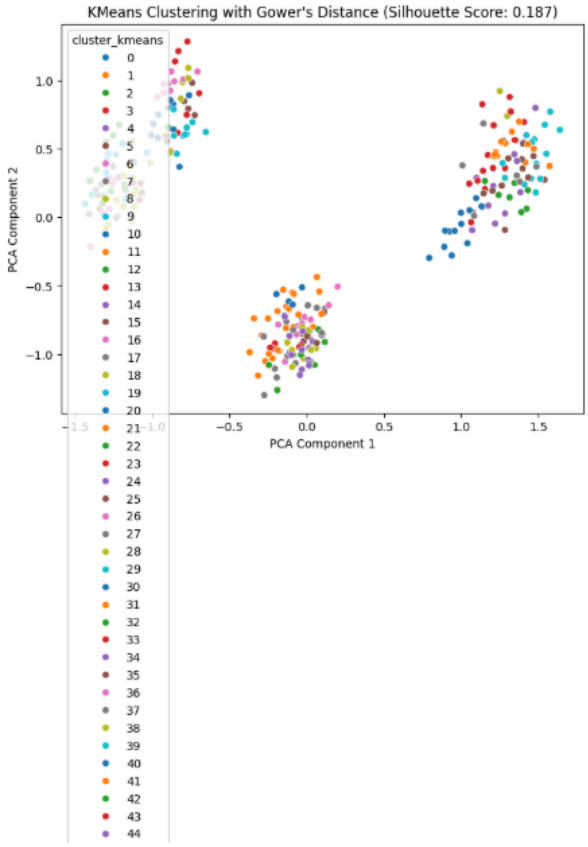

Fig. 15 - KMeans k=50 (Experiment 2)

k	Total cost (€) before opt.	Total groups before opt.	Total cost (€) after opt.	Total groups after opt.	CO_2 (kg) before opt.	CO_2 (kg) after opt.
10	**62020**	**300**	**62020**	**231**	**12051.48**	**8398.82**
30	**62020**	**300**	**62020**	**252**	**12051.48**	**8398.82**
50	**62020**	**300**	**62020**	**252**	**12051.48**	**8398.81**

Tab. 3 - Results before and after optimization - Kmeans (Experiment 2)

Based on results shown in Table 3 we had several conclusions. As before, the cost is constant across all the variations in k and the evaluation functions. Another pattern that appears is the consistency of total groups formed, and evaluated CO2 before the final optimization, as well as the optimized CO2. The same is with CO2 tax (537.5 for before *evaluation_optimized_carpooling()* and 374.59 after). Combined with low and similar Silhouette scores for all three k values, it can be concluded that the clustering structure may be weak. This suggests that even from a data perspective, the clusters aren't strongly distinct. Thus, changing k slightly does not significantly impact the environment, because the clusters do not reflect strong underlying travel patterns (even for a different number of groups for k=50, the CO2 emission remains the same). However, even if the number of clusters does not greatly affect the optimization, we can still see that an improvement has been made regarding CO2 emissions (8398.82 for the optimized CO2, compared to previous 8563.72 as the k=50 score). This is not the case for suboptimal clustering evaluation (12051.48 as opposed to 11909.28). This is to be expected as well, since the results before final optimization made no significant groups (300 groups for 300 people). Again, this indicates weak clustering structures.

7.4 AGNES algorithm (experiment two)

To improve upon the earlier AGNES clustering with only 4 features, we extended the methodology by incorporating a richer set of the same 11 features. We tested the clustering performance for n=10, n=15, and n=30 clusters, analyzing both Silhouette scores and carpool formation and optimization results. After clustering, carpool groups were formed within each cluster based on the same algorithms as earlier. Images presented below are showing results of AGNES when n=10, n=30 and n=50, respectively.

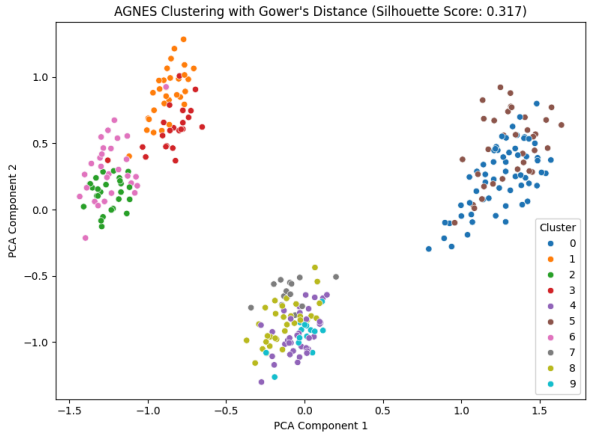

Fig. 16 - AGNES n=10 (Experiment 2)

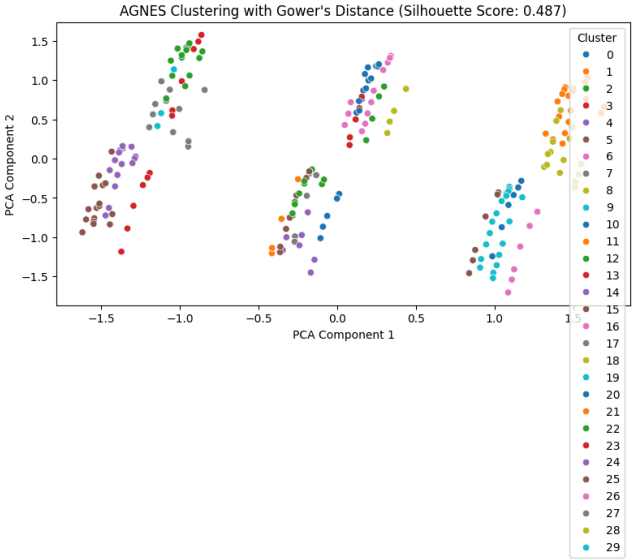

Fig. 17 - AGNES n=30 (Experiment 2)

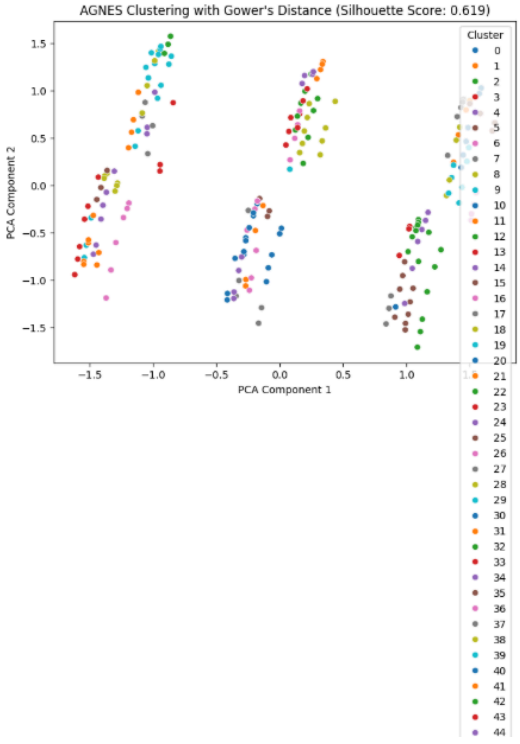

Fig. 18 - AGNES n=50 (Experiment 2)

As visible in Fig. 16-18, Silhouette scores were as follows: 0.317 for n=10, 0.487 for n=30 and 0.619 for n=50. This indicates better clustering than in the case of KMeans for 11 features, with sharper distances between Silhouette scores, meaning that n value has an impact on clustering formation. This shows promise of a strong and distinct clustering, again with n=50 having the best score.

n	Total cost (€) before opt.	Total groups before opt.	Total cost (€) after opt.	Total groups after opt.	CO_2 (kg) before opt.	CO_2 (kg) after opt.
10	**62020**	**300**	**62020**	**237**	**12051.48**	**8398.82**
30	**62020**	**300**	**62020**	**248**	**12051.48**	**8398.82**
50	**62020**	**300**	**62020**	**249**	**12051.48**	**8398.82**

Tab. 4 - Results before and after optimization - Agnes (Experiment 2)

Even though the Silhouette scores showed better clustering tendency, all the other metrics remained the same as for KMeans, with the exception of total groups formed after optimization showing a slight difference (Table 4). This leads to the conclusion that clustering with 11 features seems to be ineffective and weak, having low environmental impact, regardless of hyperparameter values.

7.5 KMeans algorithm (experiment three)

In the case of experiment 3, a slightly different approach was applied. The aim of this experiment was to make an attempt at even more improved results upon detailed analysis of Experiment one and two. First of all, as mentioned at the beginning of the Clustering chapter, features which were supposed to be optimized were taken out of the account, with only start and end cities and dates (and is_return_trip) used for clustering. Furthermore, a bigger range of k and n was used for KMeans and Agnes, with Grid Search algorithm instead of manual fine tuning.

Grid Search is a hyperparameter optimization technique used to find the best combination of parameters that yields the highest model performance. It systematically explores a

predefined set of hyperparameter values by training and evaluating a model for every possible combination. For each combination, a performance metric (e.g. silhouette score for clustering) is computed, and the configuration that produces the best score is selected [24].

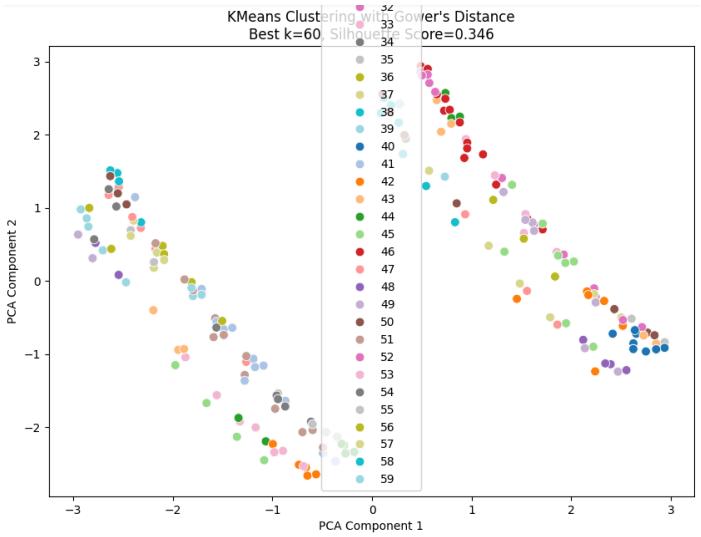

Fig. 19 - KMeans k=60 (Experiment 3)

As visible in Fig. 19, the best results were achieved with k=60, even when compared to the silhouette scores from the two previous experiments. This shows promise of a strong and distinct clustering.

k	Total cost (€) before opt.	Total groups before opt.	Total cost (€) after opt.	Total groups after opt.	CO_2 (kg) before opt.	CO_2 (kg) after opt.
60	**62020**	**233**	**62020**	**161**	**11670.28**	**9758.29**

Tab. 5 - Results before and after optimization - Kmeans (Experiment 3)

The CO2 tax was reduced compared to the first two experiments' final optimization phase as well (520.49 as compared to 530.27, the best CO2 tax score for Experiment one, and 537.5 for Experiment two). As seen in Table 5, the number of total groups dropped significantly as well, indicating more efficient grouping in this experiment. However,

total CO2, even though being less for the results before final optimization compared to previous experiments, was higher after the final optimization.

This contrast suggests that although the clustering in Experiment 3 led to better initial grouping (fewer groups with lower pre-optimization emissions and a better Silhouette score for KMeans), the optimization phase did not yield as effective results in terms of CO_2 reduction. One possible explanation is that by excluding impactful features like vehicle type or CO_2 per km from the clustering process, the resulting groups might not have been that suitable for optimal carpooling. Therefore, while the clustering was sharper (as confirmed by a higher Silhouette score), it lacked the environmental context needed to guide group formation toward low-emission configurations.

7.6 AGNES algorithm (experiment three)

The similar procedure as for KMeans Experiment 3 was followed, but this time for AGNES clustering algorithm. Additionally, linkage was also fine tuned. In hierarchical clustering, such as AGNES, the linkage strategy determines how the distance between clusters is calculated during the merging process. The linkage strategies most suitable for our task were determined to be average and complete, since they tend to form more balanced, compact clusters, or tight and small clusters, respectively. Average calculates the distance between the closest pair of points (one in each cluster), while complete represents distance between the furthest pair of points (one in each cluster) [25]. For this case, the average linkage type achieved the optimal results.

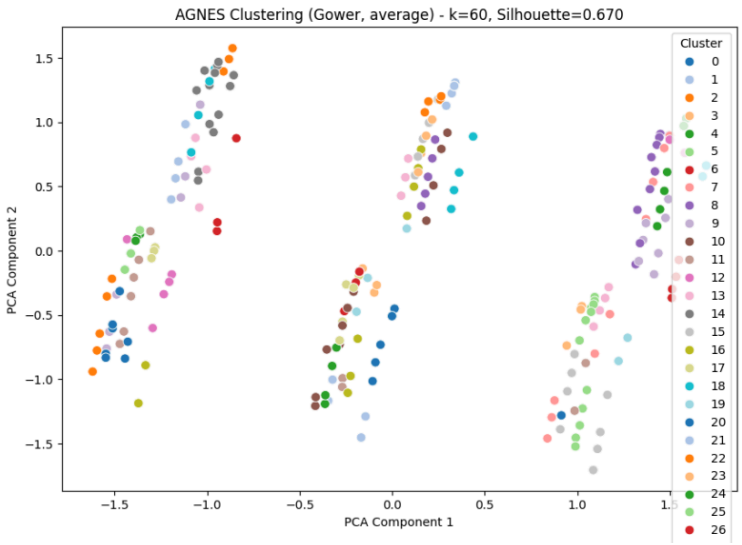

Fig. 20 - AGNES n=60 (Experiment 3)

n	Total cost (€) before opt.	Total groups before opt.	Total cost (€) after opt.	Total groups after opt.	CO_2 (kg) before opt.	CO_2 (kg) after opt.
60	**62020**	**226**	**62020**	**148**	**11628.58**	**9806.33**

Tab. 6 - Results before and after optimization - Agnes (Experiment 3)

In terms of clustering quality, AGNES with Gower distance and average linkage achieved a noticeably higher Silhouette Score (0.670) than any of the KMeans or AGNES configurations from Experiments 1 and 2. This indicates significantly better internal cohesion and separation of clusters using AGNES. Results are shown in Table 6.

From a sustainability perspective, the optimized CO_2 emissions for AGNES in Experiment 3 were 9806.33 kg, which is more than the optimized values from KMeans Experiment 1 and Experiment 2 (the lowest, therefore the best value, being 8398.82 kg). However, AGNES produced fewer total carpool groups after optimization (148) and a lower CO2 value for before the optimization stage, suggesting more compact and possibly more efficient groupings. This also contributed to a reduced CO_2 tax of 518.63 euros, which is not the case for the optimized version (437.36 euros).

7.7 Comparing all experiments

Algorithm	Total cost (€) before opt.	Total cost (€) after opt.	Total groups before opt.	Total groups after opt.	CO_2 (kg) before opt.	CO_2 (kg) after opt.
No algorithm	**110350**	/	/	/	**24624.42**	/
Greedy	**277350**	/	**197**	/	**11466.47**	/
KMeans (Exp.1)	**62020**	**62020**	**272 (k=10)**	**291 (k-=50)**	**11909.28 (k=10)**	**8536.72 (k-=50)**
KMeans (Exp. 2)	**62020**	**62020**	**300**	**231 (k=10)**	**12051.48**	**8398.82**
KMeans (Exp. 3)	**62020**	**62020**	**233 (k=60)**	**161 (k=60)**	**11670.28 (k=60)**	**9758.29 (k=60)**
AGNES (Exp. 1)	**62020**	**62020**	**270 (n=10)**	**293 (n=50)**	**11889.39 (n=10)**	**8447.94 (n=50)**
AGNES (Exp. 2)	**62020**	**62020**	**300**	**237 (n=10)**	**12051.48**	**8398.82**
AGNES (Exp. 3)	**62020**	**62020**	**226 (n=60)**	**148 (n=60)**	**11628.58 (n=60)**	**9806.33 (n=60)**

Tab. 7 - Comparison of relevant metrics

When analyzing the results of the clustering and optimization approaches, each key metric reveals important trade-offs and insights, as shown in Table 7. For each algorithm, the best result is displayed across all experiments (noting which k and n values achieved those results). If there are no k and n values noted, it means that the same result was obtained for all the tried out k and n values. Furthermore, the greedy algorithm only outputs suboptimal results, and no algorithm refers to the mere summation of CO2 emissions and total costs from the csv files. In the case of all other experiments, the suboptimal results were achieved by applying *evaluate_clustered_carpooling()*, and the optimal ones upon applying *evaluate_optimized_carpooling()* functions. When comparing the no algorithm results to the greedy algorithm, we can notice that the CO2 emissions significantly dropped, more than double (from 24624.42 to 11466.47), meaning that the algorithm served its purpose, with room for improvement.. However, the total cost increased. This happens because of the additional logistics introduced by group formation and fallback strategies. In the greedy algorithm, not all participants can be efficiently grouped into carpools due to mismatched availability, route constraints, or capacity limits. As a result, many individuals are assigned to solo travel, either by car or train, and experience additional hotel charges if leaving earlier or later. Therefore, the greedy method realistically reflects logistical inefficiencies and extra expenses involved in scheduling around participant constraints, leading to a higher overall travel cost compared to simple summation.

The Silhouette Score, which measures clustering quality, was highest in AGNES Experiment 3 (0.670), indicating strong cohesion and separation among clusters. However, this did not directly translate to the best environmental outcome, showing that well-formed clusters are not necessarily aligned with sustainability goals if the right features (e.g., CO_2 impact) are excluded. In contrast, the number of groups formed varied significantly across experiments. The Greedy method formed 197 groups, while AGNES Experiment 3 achieved only 148 after optimization, showing a high degree of compactness and efficient passenger grouping. Fewer optimized groups often indicate better carpool efficiency, though excessively small clusters may ignore travel constraints or CO_2 efficiency. The total cost, while constant across all clustering experiments

(€62020), was dramatically reduced from the Greedy baseline (€277 350), as well as the no algorithm sum (€110350), thanks to careful optimization of shared travel and hotel costs, as well as the exclusion of plain tickets (they are included in the dataset sum). This consistency suggests that clustering method choice had less influence on cost than did the optimization function's logic, since the same routes and trip durations were considered regardless of the combination of people, therefore justifying the constant toal cost. Even upon applying a train fallback, the price remained the same, meaning that no significant costs were incurred to the usage of more expensive transportation (train), while still reducing CO2, which was the goal. Lastly, the total CO_2 emissions highlight the most critical distinction: the best reductions were achieved in KMeans and AGNES Experiments 1 and 2 (as low as 8398.82 kg), particularly when a richer feature set was used to cluster travelers with similar environmental impact profiles. In contrast, experiments with higher Silhouette scores but simplified features (Experiment 3) resulted in higher emissions, proving that meaningful clustering for sustainability must include variables like car type and CO_2 per km. Overall, AGNES gave slightly better results than KMeans algorithm. This can be explained by several reasons: AGNES is more flexible with cluster shapes (which are common in real-life data), works better with Gower distance and mixed data types, In conclusion, the most effective results came not from the highest-quality clustering, but from a balance between feature grouping and practical optimization.

7.8 Testing the pipeline

After successfully creating the pipeline which produced optimal carpool groups with a minimal amount of CO2, it was tested on larger and more diverse datasets. The new datasets were generated with 600 and 1200 instances for missions and persons csv files, while keeping the co2 the same (as it only contains hardcoded values of CO2 emissions). The date range was larger in the case of the newly generated datasets, going back for the last 5 years, which is more in line with the university dataset that this simulation was modelled after.The main aim of this testing was to inspect how the pipeline would act with a considerably larger data load. The execution time remained almost the same, with only a few added seconds for cell execution on the larger datasets, even when tested on over a 1000 data. This made total execution time longer, but not considerably so.

Regarding the qualitative measures of the code quality, the algorithm did not crash or show unexpected behaviour in any of the datasets, and was successful in finding the optimal solution based on the earlier described algorithms in all of the experiments.

Upon inspecting the obtained results, it can be confirmed that the algorithms behaved consistently even on larger and more diverse datasets. The clustering methods proved to be quite consistent, with similar Silhouette scores as compared to the first dataset, and with larger values of k and n always showing better clustering tendencies. Furthermore, with the optimized results, the larger k and n values scored better, once again generating the less amount of CO2 emissions (the total cost remained consistent, as for the first dataset), although other values came close as well. A trend which was noticed on both larger datasets results was that KMeans and AGNES gave similar, sometimes identical results when compared to each other across all experiments, proving the efficiency of both methods on larger datasets and thus justifying us choosing them. This was to be expected, since, with more data points, both algorithms are guided by stronger statistical structure, more averaged distances, all contributing to making their clustering decisions naturally converge.
As for the original dataset, suboptimal results (*evaluate_clustered_carpooling()*) did not improve compared to the greedy algorithm, but the optimized results (*evaluate_optimized_carpooling()*) showed a sharper improvement in comparison, when speaking about CO2 emissions. For example, on the 600 instances dataset, the greedy algorithm scored 26054.49 kg of CO2 emissions, while Experiment 1 outputted 19096.86 kg (k=50 and n=50), and, for the 1200 instances dataset 37958.36 kg (k=50) and 37958.37 kg (n=50), as compared to 50527.27 achieved with greedy algorithm. Other experiments achieved similar or identical results to these, as mentioned above. This further proves the efficiency of the optimized algorithm combined with clustering, compared to a greedy approach.

Therefore, a conclusion can be derived that, overall, the pipeline is a simple, efficient and easily scalable method, providing consistent results for different datasets.

8. Conclusion

This project presented a comprehensive approach to eco-friendly carpool optimization by combining clustering algorithms, optimization techniques, and an initial exploration of

reinforcement learning on a simulated dataset reflecting university staff travel to various events. We evaluated four clustering experiments, three versions of KMeans and AGNES, differing in feature complexity, number of clusters, and hyperparameters, alongside a baseline greedy algorithm.

The results demonstrated that simpler clustering approaches (KMeans and AGNES in Experiment 1), using fewer features, achieved the lowest CO_2 emissions (the total cost remained the same across all three experiments), especially in the case of the AGNES algorithm. Additionally, which features matter. The best results were achieved in Experiment 1 and not Experiment 3, despite both experiments working with only a few features (four and five respectively), since co2_per_km was among the features in the first experiment, therefore taking account of the environmental factors when performing the clustering. A trend which was noticed across all experiments was that the higher values of k and n gave the best Silhouette scores, as well as optimized results. Furthermore, when the algorithms were tested on larger and more varied datasets, they demonstrated consistent results and reliable performance, pointing to an effective yet scalable solution.

While reinforcement learning was considered a promising next step for dynamic and adaptive group formation, computational resource limitations prevented its full application.

These results highlight an important trade-off between forming groups and optimizing sustainability metrics. The hierarchical AGNES algorithm's stability and interpretability proved advantageous for forming appropriate travel groups, while the use of Gower's distance enabled effective handling of mixed data types. The integration of travel mode optimization, substituting car rides with train journeys when effective, further improved environmental outcomes without increasing costs.

Overall, the proposed solution demonstrated effectiveness in reducing emissions and travel costs compared to initial naive grouping, and the scalable architecture supports future application onto real datasets and with more complex constraints. This work

demonstrated the practical potential of data-driven carpool optimization for reducing carbon footprints and travel expenses.

9. Future work

As a first step, future work regarding this project should focus on validating and replicating these results using real-life travel data and authentic databases rather than simulated datasets, such as this one. Applying the models to data covering not only recent years but ongoing trips as well could enable dynamic carpool group formation that adapts to last-minute changes and cancellations.

Additionally, machine learning is a vast field, developing more with each day. There are multiple algorithms and experiments which could be tried on this particular dataset, over a longer time period, with more advanced fine tuning of parameters, which would also require more computing resources. This particular research is a good building block for an even more sophisticated machine learning and optimization approaches, which are currently out of the scope of this particular project, at this time. One possible improvement for the optimization function would be to add a logic for driver optimization, making sure that a car with least amount of CO2 emission was chosen for each carpool group. In the case of this dataset, the difference between the majority of the car models was not that great, and, based on the obtained carpool group structures, would not play a significant role in reducing CO2 emissions. However, this may be a useful next-level optimization for other datasets.

Furthermore, exploring other advanced optimization methods such as linear programming (GbFCA, MILP, etc.) could provide more precise trip scheduling and cost minimization, potentially even outperforming machine learning approaches. Incorporating predictive models based on historical travel behavior and external factors (e.g., weather, public transport schedules, delays on the road) may further enhance group matching and user satisfaction.

Building upon this, a mobile or web application could be developed which would allow users to find their carpool group for their next environmentally and cost friendly trip.The functionalities of this app could include allowing individuals to input conference or travel times, home addresses, budget and other preferences, with automated generation of personalized carpool groups, all while taking user preferences into account. This app and research in general could be extended beyond university campuses and France to include broader geographical areas, individual home locations, and carpooling to the airport. Additionally, if a journey is over 2 hours long or if a carpool group of individuals do not have a car, a train fallback was used. However, depending on the train schedules, real-time occurrences such as possible delays, cost, user preference etc., other means of travel could be taken into account as well (e.g., bus), while still having in mind the goal to minimize CO2 emissions.

Finally, investing into computational resources required for reinforcement learning techniques and larger quantities of data could enable more adaptive, learning-based carpooling systems that optimize for multiple objectives in real time.

10. Literature

[1] S. Shaheen, A. Cohen, and A. Bayen, *The Benefits of Carpooling*. Berkeley, CA: University of California, 2018.

[2] D. Das, P. P. Kalbar, and N. R. Velaga, "Dynamic stock model based assessment of carpooling in passenger transportation carbon emissions: Will avoided trips and material credits help?," *Sustainable Production and Consumption*, vol. 30, pp. 486–498, 2022.

[3] S. Pramanik, *Carpooling Solutions Using Machine Learning Tools*, Advances in IT Standards and Standardization Research Series. IGI Global, 2022.

[4] X. Chang, J. Wu, Z. Kang, et al., "Estimating Emissions Reductions with Carpooling and Vehicle Dispatching in Ridesourcing Mobility," *Sustainable Mobility and Transport*, Nature, 2024.

[5] M. G. Campana, F. Delmastro, and R. Bruno, "A Machine-Learned Ranking Algorithm for Dynamic and Personalized Car Pooling Services," *arXiv preprint*, 2023.

[6] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[7] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, 2nd ed., Pearson, 2018.

[8] L. Rokach and O. Maimon, "Clustering methods," in *Data Mining and Knowledge Discovery Handbook*, Springer, 2005, pp. 321–352. doi: 10.1007/0-387-25465-X_15.

[9] J. Smits, *Climate Change and Tourism: Plane, Car, Train or Boat?*, Travel Begins at 40, 29-Sep-2019. [Online]. Available: https://www.travelbeginsat40.com/2019/09/climate-change-and-tourism-plane-car-train-or-boat/. [Accessed: 26-May-2025].

[10] BBC News, *Climate change: Should you fly, drive or take the train?*, 24-Aug-2019. [Online]. Available: https://www.bbc.com/news/science-environment-49349566. [Accessed: 26-May-2025].

[11] ODYSSEE-MURE, "Specific consumption of new cars," [Online]. Available: https://www.odyssee-mure.eu/publications/efficiency-by-sector/transport/specific-consumption-new-cars-country.html. [Accessed: 26-May-2025].

[12] GlobalPetrolPrices.com, "France gasoline prices, May 2025," [Online]. Available: https://www.globalpetrolprices.com/France/gasoline_prices/. [Accessed: 26-May-2025].

[13] SNCF Connect, "OUIGO fares and booking conditions," [Online]. Available: https://www.sncf-connect.com/en-en/ouigo/fares. [Accessed: 26-May-2025].

[14] SNCF Connect, "TER pricing and regional fare structures," [Online]. Available: https://www.sncf-connect.com/en-en/ter/fares. [Accessed: 26-May-2025].

[15] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proc. 5th Berkeley Symp. on Math. Statist. and Prob.*, 1967, pp. 281–297.

[16] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, Jan. 1980.

[17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[18] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," in *KDD Workshop on Text Mining*, 2000, pp. 525–530.

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

[20] "Carbon Taxes in Europe," *Green Fiscal Policy Network*, Jun. 8, 2021. [Online]. Available: greenfiscalpolicy.org/carbon-taxes-in-europe/. [Accessed: May. 20, 2025].

[21] D. T. Pham, S. S. Dimov, C. D. Nguyen, "Selection of K in K-means clustering," IMechE Journal of Mechanical Engineering Science, vol. 219, part C, 2005.

[22] M. Shukla and N. Naganna, "A Review on K-means Data Clustering Approach," International Journal of Innovative Research in Computer and Communication Technology, Special Issue, 2013.

[23] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, 1987. [Online]. Available: https://doi.org/10.1016/0377-0427(87)90125-7. [Accessed: Jun. 13, 2025].

[24] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, Feb. 2012. [Online]. Available: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf. [Accessed: 8-Jun-2025].

[25] Scikit-learn developers, "sklearn.cluster.AgglomerativeClustering," *Scikit-learn Machine Learning Library*, [Online]. Available:

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html. [Accessed: 9-Jun-2025].