The Comparative Analysis of Car-Pooling Algorithms for Ride-Sharing Systems

Julien Baudru
Université Libre de Bruxelles
8 PUBLICATIONS **1** CITATION

The Comparative Analysis of Car-Pooling Algorithms for Ride-Sharing Systems

Julien Baudru[1,2][0000−0002−8771−2494] and Hugues Bersini[1,2][1111−2222−3333−4444]

[1] IRIDIA - Université Libre de Bruxelles, Brussels, Belgium
[2] FARI - AI Institut for the Common Good, Brussels, Belgium
julien.baudru@ulb.be

Abstract. In this paper, we present a comparative analysis of five methods for constructing ride-sharing pools of users, focusing on their efficiency in terms of execution time, the percentage of user requests fulfilled, the distance of the detour made by the driver and the waiting time of the passenger. Furthermore, we introduce a model able to simulate user demand, based on car usage data across different time intervals in Belgium. Then, we use the proposed model as a basis for evaluating the performance of the five methods and their variants: OD Similarity, OD Clustering, OD Time Alignment, Trip Similarity, and Trip Buffering.

Keywords: Car-Pooling · Trip-Matching · Ride-Sharing · Sustainable Mobility · Smart Transportation · Road Networks.

1 Introduction

For several decades, the use of private vehicles has exploded, leading to an increase in traffic congestion, pollution and accidents. Various solutions already exist to reduce the reliance on private cars, we can first think of public transport, for instance, as a promising alternative. However, since not all cities around the world have a well-developed public transport network, ride-sharing seems to be the most viable alternative for users in terms of economy, ecology and comfort [4]. In addition, according to [20], Peer-to-Peer (P2P) ride-sharing is the most flexible and lowest-cost mobility alternative.

Reducing the number of vehicles on the road appears to be essential from an economical and ecological point of view. Indeed, in Europe Union (EU), inefficient urban mobility, and road congestion in particular, costs the EU around 119 billion dollars a year (2020) [15]. In the United States of America (USA), the road congestion is estimated to be responsible for a cost of around 121 billion dollars a year (2019) [23]. From an ecological point of view, in EU, between 2014 and 2017, CO_2 emissions from road transport rose by 45 million tonnes, or 5% [19]. In addition, the use of personal vehicles is a major source of urban air pollution in Brussels, Stuttgart and Milan [28]. In USA, in 2021, CO_2 emissions from transportation in the United States totaled 1700 million tonnes, the most from any sector of the economy. And in 2019, 40 million tonnes of greenhouse gases, about 2% of all transportation-related emissions, were emitted because of traffic congestion [24]. As outlined in the aforementioned information, it is clear that solving the challenges posed by ride-sharing, or car-pooling, has become a major mobility issue in most regions.

Ride-pooling can be described as a shared transport system in which multiple users take a common route, and therefore vehicle, to reach their different, or common, destination [12]. This transport system is based on the shared use of private vehicles or the shared use of vehicles provided by Mobility as a Service (MaaS) companies. This system differs from the services of Transportation Network Companies (TNCs), such as Uber and Lyft, which offer a low-cost alternative to taxis [20]. P2P rides-haring seek to gain the advantages of TNCs while reducing their negative impact on the environment. The most suitable definition for the study presented in this paper is ride-sharing through the use of private vehicles. An extended definition provided by [22] is given in Text 1.

Private cars are utilized by various households or organizations in either a centralized (one large, open group) or decentralized system (several small, closed groups). The vehicle is owned by one member of the carpool group or can be jointly owned by several group members.

Text 1: Definition of ride-sharing.

As accurately stated in [9], passengers and drivers might have multiple varying objectives and preferences that must be optimised and satisfied but often these preferences are conflicting. For example, drivers aim to optimize their financial gains, minimize daily service hours, and express preferences for specific service areas, whereas passengers seek to minimize travel time, waiting time and cost and may prioritize a comfortable journey with pleasant personal space. However, according to [9], offering a car sharing service that optimises all the above objectives while simultaneously satisfying all preferences may not be feasible. This is why in the following of this paper we will deal with only two preferences and one objective. These preferences concern the maximum detours travelled by the driver and the maximum waiting time of the passenger and the objective is to maximise the number of satisfied passenger requests.

In this paper, we compare different methods for creating groups of users interested in carpooling. We evaluate these methods based on their execution time, the percentage of user requests satisfied, the distance of the detour made by the driver and the waiting time of the passenger. In addition, we propose a model that simulates user demand based on car usage data across various time periods for a day in Belgium. This approach provide insights into user behavior patterns, allowing a robust comparison of the OD Similarity, OD Clustering, OD Time Alignment, Trip Similarity, and Trip Buffering methods. We start by defining the different mathematical elements used, secondly we present two similarity functions, and then we detail different methods to generate pools of users. Next, we introduce a model to simulate user requests in car-pooling and we compare the results obtained by the different methods tested on this model and on randomized experiments with the two different similarity functions when possible, and finally propose a series of future improvements for further research.

2 Similar works

To the knowledge of the authors, only a few reviews exist in the literature on the ride-sharing problem. One of the most notable is provided by [25], in this review the authors propose a classification of the various existing systems, distinguishing between dynamic and static systems. They then subdivide the category of dynamic systems into three parts: centralised, decentralised and hybrid systems. For each of the static and dynamic systems, they propose a distinction between systems that use heuristics and those that do not. Also, they compare the advantages and disadvantages of each system. Another notable review is that provided by [20], in their article they compare a series of flexible, dynamic, ride-sharing systems on the basis of five criteria, the use of flexible paths, the use of multi-hop, the possibility of having multiple rides and whether the solution is optimal. They note that few systems meet all the criteria at the same time, and that the execution times of the systems presented are too long to be used in large-scale real-life applications. The authors then propose two novel approaches to increase the performance of a ride-sharing system, one of which uses the Ellipsoid Spatiotemporal Accessibility Method (ESTAM), an idea similar was proposed in [2], to find the optimal meeting point between a passenger and a driver.

There exists a diverse array of strategies for user matching methods in carpooling, all aimed at optimizing passenger-driver pairing for efficient transportation solutions. Traditional approaches often rely on heuristic algorithms, such as nearest neighbor, greedy algorithms or evolutionary algorithms, which prioritize proximity and availability. For instance in [8], they present two heuristic algorithms based on greedy method and the time-space network for the case of one origin to many destinations and many origins to one destination in the context of dynamic taxi-pooling problem. In [11], the authors demonstrate the effectiveness of evolutionary algorithms in minimizing total trip costs for distributing passengers traveling from a common origin to different destinations in multiple taxis. Another popular method to solve this problem is to use an operational search approach. In [3], the authors have developed an approximation algorithm for assigning cars to requests while aiming to minimize costs. Their algorithm guarantees solutions with at most 2.5 times the optimal cost, and experiments show that it often achieves a better ratio, around 1.2, on synthetic data.

Additionally, collaborative filtering methods, inspired by recommender systems, consider user preferences and historical data to improve matching accuracy. Among others, the model proposed by [9], MaMoP, uses social reasoning and evolutionary algorithms to simultaneously optimise ride-sharing solutions and take account of user preferences. In [26], they integrate user personality preferences into a matching model for passengers in ride-sharing systems. They modify the stable roommates problem algorithm for one-on-one passenger matching and consider factors such as personality and steadiness. In [1], the authors propose an algorithm to improve the matching optimization, taking into account the gender, age, professional status and the social tendencies of the participants. In addition, the proposed algorithm subdivides the unmatched segments of the path of the drivers, generating new trip requests to create additional matches using these unmatched segments. In [29], they present a model in which riders are matched based on a specific set of human characteristics using machine learning techniques. After trip completion, they record the user feedback and compute two main characteristics that are most important to riders. The registered and the computed characteristics are fed to a classification module, which later predicts the two main characteristics for new riders. And finally in [7], the authors propose a recommender system for carpooling services that leverages on learning-to-rank techniques to automatically derive the personalised ranking model of each user from the history of her choices (i.e., the type of accepted or rejected shared rides). Then, the system builds the list of recommended rides in order to maximise the success rate of the offered matches.

Nevertheless, there appears to be a lack of practical evaluations of existing algorithms for generating user pools in the context of ride-sharing. Consequently, this article aims to address this gap in the current literature.

3 Problem formulation

In this section, we describe the theoretical background from which the pooling methods presented further on have been developed.

First, it is worth noting that the ride-sharing issue is classified under the classic Dial-a-Ride Problem (DARP) [21], known for its NP-hard complexity [27]. Within DARP, passengers request rides from designated origins o_p to specific destinations d_p. Therefore, we define the directed weighted graph $G = \{V, E\}$ with $V = \{v_1, v_2, ..., v_n / v_i \in \cap G\}$, i.e. the set of road intersections. The edges of $G = \{V, E\}$ are define as $E = \{arc(i, j)/i \in V, j \in V\}$, i.e. the set of roads between these intersections. For each $arc(i, j)$, a non-negative travel cost $\delta(i, j)$ is associated, which corresponds to the distance of the road between intersections i and j, each $arc(i, j)$ also have a travel speed $\sigma_u(i, j)$ depending on the user u. We denote by $p_u(i, j)$ the subset of V containing the sequence of nodes $\{v_1, v_2, ..., v_n\}$ from the arcs included in path of user u to travel from the source i to the destination j. The travel time for user u to complete path $p_u(i, j)$, $\tau_u(i, j)$, is given by the Equation 1.

$$\tau_u(i, j) = \sum_{v,v' \in p_u(i,j)} \left(\frac{\delta(v, v')}{\sigma_u(v, v')} \right) \tag{1}$$

In this review, we distinguish between two types of user u, drivers u_d and passengers u_p. Let U be the set of users, P the set of pools, P_i the pool containing one user u_d and multiple users u_p, and n_p the number of passengers in the pool P_i. In each of the methods presented, we aim to create pools consisting of a single driver u_d accompanied by at least one passenger u_p, the number of passengers n_{u_p} not exceeding the maximum number of seats available in the vehicle $v_{capacity}$.

4 Similarity functions

In this section, we describe the different similarity functions that will be used in the rest of this article to create user pools thanks to the different methods presented in section 5. These two functions are deployed to determine how similar two users are according to distance or time criteria. One uses only the origin and destination, while the other uses the path taken by the users. In the following, all computations of distance or time between two points are based on the shortest path found by the Dijsktra [10] algorithm.

4.1 Distance similarity

A common metric used to quantify the similarity between two points is the Euclidean distance. Thus, the Euclidean distances between the origins and the destinations are given by $\delta(o_i, o_j)$ and $\delta(d_i, d_j)$ respectively. By combining these two distances, we obtain the similarity function between two users i and j given by the Equation 2.

$$sim(i,j) = \exp\left(-\alpha \cdot \left(\frac{\delta(o_i, o_j)}{\gamma_o} + \frac{\delta(d_i, d_j)}{\gamma_d}\right)\right) \tag{2}$$

Where α is a scaling factor to adjust the importance of the distance and γ_o and γ_d are scaling parameters that control the spread of the similarity function. The larger the value of α, the more emphasis is placed on the proximity of both origins and destinations.

This similarity function can be modified to take into account the average origin and destination of a group of users, a pool, enabling it to be used iteratively in one of the methods presented later. This function is defined in the same way as before, but with the addition of the current pool's average origin o_{pool} and average destination d_{pool}. Where γ_{pool_o} and γ_{pool_d} are scaling parameters that control the spread of the similarity function. This modified similarity function is given by Equation 3.

$$sim(i,j) = \exp\left(-\alpha \cdot \left(\frac{\delta(o_i, o_j)}{\gamma_o} + \frac{\delta(d_i, d_j)}{\gamma_d} + \frac{\delta(o_{\text{pool}}, o_j)}{\gamma_{\text{pool}_o}} + \frac{\delta(d_{\text{pool}}, d_j)}{\gamma_{\text{pool}_d}}\right)\right) \tag{3}$$

4.2 Trip similarity

Another way of creating user pools is to define the similarity between users on the basis of their paths rather than just their origin and destination. For this we use the similarity function defined by Equation 4, where the length of a path is given by n and the node v_u^i is the i^{th} node on the path of the user u.

$$sim(p_{u_d}, p_{u_p}) = \frac{\sum_{i=0}^{n} psim(v_{u_d}^i, v_{u_d}^i)}{n} \tag{4}$$

In [16] the authors use the spatio-temporal similarity measure between two nodes in the graph define by the Equation 5.

$$psim(v_{u_d}, v_{u_d}) = \exp\left(\frac{w_1 \times \ln\left(\frac{1}{1+\delta(v_{u_d}, v_{u_d})}\right) + w_2 \times \ln\left(\frac{1}{1+\Delta\tau(v_{u_d}, v_{u_d})}\right)}{w_1 + w_2}\right) \tag{5}$$

Where $\delta(v_{u_d}, v_{u_d})$ is the spatial Euclidean distance of two points, $\Delta\tau(v_{u_d}, v_{u_d})$ is the absolute difference of the points in time and $w1$ and $w2$ are the weight given to the distance and time factors. Thus, when $w_1 = 1$ and $w_2 = 0$ this similarity function only takes into account the distance separating the pairs of points on the two paths. Conversely, when $w_1 = 0$ and $w_2 = 1$, the function only takes into account the travel time separating the pairs of points on the two paths.

5 Pooling methods

In this section, we present and describe five methods to create user groups, pools, for ride-sharing. Although these methods are different, they have a number of common characteristics. The Trip Similarity and OD Time Alignment methods both use the time criterion, the Trip Similarity and Trip Buffering methods both use the paths of the users $p_u(o, d)$, the OD Similarity and OD Clustering methods are based on the origin and destination points of the users (o, d), and finally all the methods except OD Time Alignment use the notion of distance δ.

5.1 OD Similarity

This method consists of matching users who have similar origin and destination locations. The objective is to find pairs of users i and j whose origins o_i and o_j are close to each other and whose destinations d_i and d_j are also close to each other. To create pools of users with a maximum size of $v_{capacity}$ and ensuring that each pool contains at least one driver u_d, we use the following steps:

1. Select a driver u_d from the set of available users and assign it to a pool.
2. For each remaining user u_p:
   (a) Calculate the similarity between the origin and destination of u_p and the average origin and average destination of the current pool using the Equation 3.
   (b) If the similarity exceeds a certain *threshold*, add u_p to the current pool.
   (c) Repeat steps 2 and 3 until the pool reaches its maximum size $v_{capacity}$.
3. Remove all users in the current pool from the set of available users.
4. Repeat steps 2-4 until all users are assigned to a pool or stop the algorithm if no more allocations are possible.

5.2 OD Clustering

This method consists of creating user groups by creating clusters based on the distance similarity of the origin o and destination d points of users using the Equation 2. First, we create a graph where the nodes are the users who have sent their requests. We define this complete directed weighted graph as $G_u = \{V, E\}$ with $V = \{u_1, u_2, ..., u_n / u_i \in \cap U\}$, i.e. the set of user in U. The edges of $G_u = \{V, E\}$ are define as $E = \{sim(i, j)/i \in V, j \in V\}$, i.e. the similarity values between each users. Then we use the Louvain algorithm [5] to create the clusters, pools, based on the weights of the edges. The final pool construction is subject to two constraints applied to the clusters found.

$$n_{u_p} \leq v_{capacity} \tag{6}$$

$$n_{u_d} = 1 \tag{7}$$

The constraint 6 ensures that the size of each cluster should be at most $v_{capacity}$ and the constraint 7 ensures that each cluster contains at least one driver user.

5.3 OD Time Alignment

This method consists of optimizing the formation of the pools by matching users with similar departure and arrival times. This approach minimizes waiting times for users and ensures efficient utilization of vehicles. Let $\Delta\tau(o_{u_p}, o_{u_d})$ denote the difference between the departure times of a passenger u_p and a drive u_d. Similarly, let $\Delta\tau(d_{u_p}, d_{u_d})$ denote the difference between the arrival times of a passenger u_p and a drive u_d. A time threshold $\tau_{\text{threshold}}$ is defined to determine whether the departure and arrival times of a passenger and a driver are sufficiently aligned for carpooling. Mathematically, the conditions for time alignment is expressed by the Equations 8 and 9.

$$\Delta\tau(o_{u_p}, o_{u_d}) \leq \tau_{\text{threshold}_o} \tag{8}$$

$$\Delta\tau(d_{u_p}, d_{u_d}) \leq \tau_{\text{threshold}_d} \tag{9}$$

Then, to create the pools, we assign a number of $v_{capacity}$ passengers u_p to each driver u_d, based on a ranking of the most similar departure and arrival times.

5.4 Trip Similarity

This method consists of comparing different points on the path of two users from a spatial and temporal point of view. Pools are created based on the groups of users with the largest combined trip similarities. To define each individual trip i based on the origin o and destination d we use Dijsktra [10] to find the shortest path, $p_{u_i}(o, d)$. The similarity between two paths is given by the Equation 4. Then we create pools of users based on this similarity measures. Let P_{ij} be the pool containing users u_i and u_j, and n_{ij} the total number of users in the pool P_{ij}. For each pair of users u_i and u_j with $u_i, u_j \in U$ and $i \neq j$:

1. Compute the similarity between the trips of u_i and u_j using Equation 4.
2. If $sim(p_{u_d}, p_{u_p})$ is greater than a certain *threshold*, then add u_i and u_j to the same pool if there is exactly one driver user u_d in the pool P_{ij} and if $n_{ij} \leq v_{capacity}$.

Finally, instead of comparing each of the trip points of two users, we compare their origins and destinations as well as N randomly selected trip points. This choice of implementation greatly reduces the computation time while preserving the characteristics of the method.

5.5 Trip Buffering

This method consists of creating pools based on the users that the driver meets during his trip. Each driver has a buffer of a given distance δ_{buff} or time τ_{buff}. If the origin o_p of an user u_p is in the buffer, the driver u_d will make a detour to pick him up and add it to his pool, otherwise he ignores it and continues on his way to his destination. Let v_d be the current position of u_d in G. Mathematically, the fact of taking a passenger can be expressed by the binary variable p_{u_p} given in the Equation 10.

$$p_{u_d} = \begin{cases} 1, & \text{if } \exists \text{ user } u_p \text{ such that } \delta(v_d, o_p) \leq \delta_{buff} \text{ or } \tau(v_d, o_p) \leq \tau_{buff} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

To ensure correctness of the method, two constraints must be satisfied. The first Constraint 11 ensures that the length of the detour is less than the maximum detour length $\delta_{detourMAX}$ tolerated by u_d and the second Constraint 13 ensures that there are still seats available in the vehicle. Let n_{u_p} be the current number of user u_p in the vehicle of user u_d. If the variant of this method using detour time rather than distance is selected, then the Constraint 11 is replaced by the Constraint 12.

$$\delta_{detour} \leq \delta_{detourMAX} \cdot p_{u_d} \tag{11}$$

$$\tau_{detour} \leq \tau_{detourMAX} \cdot p_{u_d} \tag{12}$$

$$n_{u_p} \leq v_{capacity} \cdot p_{u_d} \tag{13}$$

For this method, there is no need to specify a constraint on the number of drivers in the pool u_d, as the pools are constructed from these drivers. It is also important to specify that the detour constraint, 11 or 12 depending on the preference, is also defined for the destination point of the passenger according to the destination point of the driver. This avoids taking on passengers who are on the same route but whose destination is not at all similar to the destination of the driver.

As shown, this method can be modified to use travel time instead of distance as presented above, so the distance buffer becomes a time buffer, the detour distance becomes the time taken by that detour and the maximum detour distance becomes the maximum detour time tolerated by the driver.

6 Experimental setting

In this section, the various results obtained are presented. All experiments were carried out on a Windows 11 OS equipped with an 8-core AMD Rizen 7 5800X processor with a frequency of 3.80 GHz and 32 GB of RAM. For the sake of quick prototyping the algorithms have been written in Python 3.10.11. The road network, the graph, is stored in the form of a dictionary of dictionaries thanks to the NetworkX library [14].

6.1 Road network

Table 1 shows the properties of the real road network G used in the two experiments, the randomized and the simulation. These data come from information available on Open Street Map thanks to the Python library OSMnx [6], each node is a road intersection and each intersection is linked by roads, the edges. These edges carry information such as the distance of the road, the speed limit, the transit time, the type of road and its name.

Table 1: Brussels road network specifications

Network	**Nodes**	**Edges**	**Max deg**	**Avg. deg**
Brussels	18547	40890	12	4.41

6.2 Modeling user requests

To evaluate the different methods presented, we simulated the user requests by varying their occurrence according to the hour of the day. Fig. 1 shows the number of requests as a function of time; this model is based on real data collected by [18] in Belgium, shown in Fig. 2. It should be noted that the proposed model only takes into account the number of requests at each time of day, without taking into account the real origin and destination points of users, which are chosen randomly from the road network G.

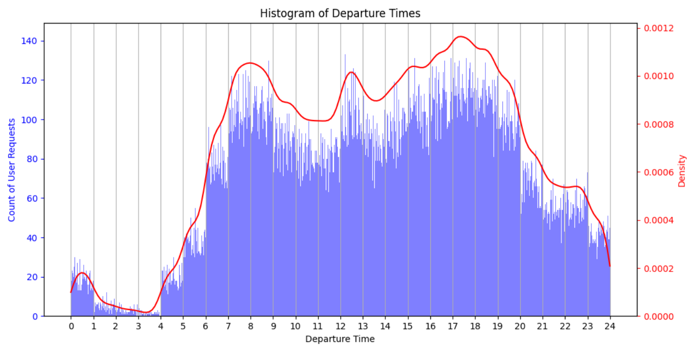

Fig. 1: Simulation data generated by our model.

To create this model we used a combination of uniform random function. The construction of the proposed model can be described as follow: Let N represent the total number of user in the simulation and d_h be the proportion of users at time h with h in $0, 1, ..., 23$. The d_h values have been experimentally defined to best match the real data from [18]. Let $U(h, h+1)$ represent a uniform distribution over the interval $[h, h+1)$, and let n_h represent the number of users in hour h, which is calculated as $n_h = N \times d_h$. The model generates n_h random numbers from $U(h, h+1)$ for each hour h from 0 to 23. The distribution of users for a hour h, D_h, is given by the Equation 14.

$$D_h = x_1, x_2, ..., x_{n_h} \quad \text{where} \quad x_i \sim U(h, h+1) \tag{14}$$

The final distribution D is the union of all D_h and is given by the Equation 15. Thus, the distribution D contains all the sub-distributions $[D_0, D_1, ..., D_{23}]$.

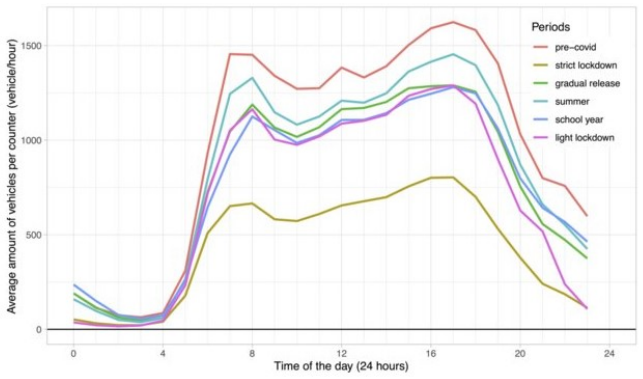

Fig. 2: Real world data from [18].

$$D = D_0 \cup D_1 \cup ... \cup D_{23} \tag{15}$$

Then, for each user u joining the simulation at time τ_a, a request r_u is established. A query contains information on the point of origin o and destination d in the graph G, the shortest path sp to join the od points, the departure time τ_a in the simulation and the type of user m being the mode, either driver or passenger. We denote a request as the set $r_u = (o, d, sp, \tau_a, m)$

7 Results

In this section, we present the results obtained by the five methods presented in Section 5 as well as variants of these methods using one or other of the similarity functions presented in Section 4. In total, we compare eight algorithms designed to build pools of users for car-pooling.

The parameters of the tested methods are detailed in Table 2. For the OD Time Alignment method, the $\tau_{\text{threshold}_o}$ and $\tau_{\text{threshold}_d}$ values are equal and are counted in seconds in the *Threshold* column. Similarly, the maximal detour in term of distance and time for the two variants of the Trip Buffering methods are noted in the *Threshold* column. For the OD Similarity method, the value of γ_{pool_o} is equal to the value of γ_o and the value of γ_{pool_d} is equal to the value of γ_d. Note that each of these methods is scalable to any vehicle capacity and is therefore adaptable to different scenarios. In the following, for each of the violin-shaped plots, we carried out 50 experiments, for each experiment we selected a random number, from 10 to 200, of simultaneous requests. The other results were performed on a proportion of simultaneous user requests based on the model presented in Section 6.2 with data points every 15 minutes throughout the day.

Table 2: Methods parameters

Method	***Criteria***	$v_{capacity}$	***Threshold***	α	γ_o	γ_d	N	$buff$	w_1	w_2
OD Similarity	δ	4	0.0005	0.0001	5000	5000	-	-	-	-
OD Clustering	δ	4	0.0005	0.0001	5000	5000	-	-	-	-
OD Time Alignment	τ	4	40	-	-	-	-	-	-	-
Trip Similarity	δ, τ	4	0.00055	-	-	-	2	-	0.5	0.5
Trip Similarity Dist	δ, τ	4	0.00055	-	-	-	2	-	0.9	0.1
Trip Similarity Time	δ, τ	4	0.00055	-	-	-	2	-	0.1	0.9
Trip Buffering Dist	δ	4	5000	-	-	-	-	8000	-	-
Trip Buffering Time	τ	4	10	-	-	-	-	30	-	-

7.1 Runtime

In this section we compare the running times of the different methods presented in the Section 5.

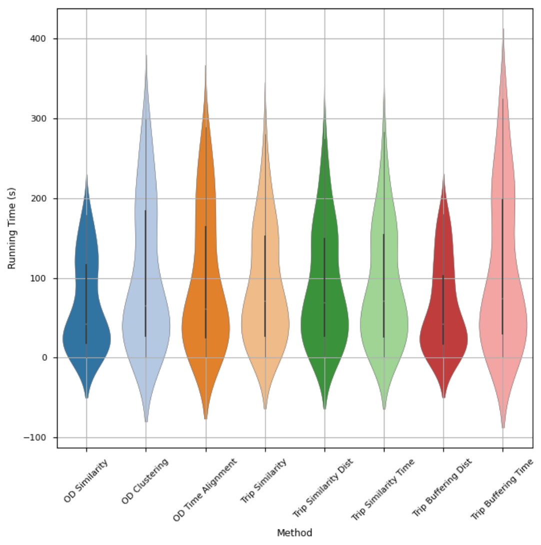

Fig. 3: Average running time of the pooling methods.

On average, the methods OD Similarity and Trip Buffering using the distance similarity function perform best as shown in the Fig. 3. On the other hand, the OD Clustering method and the Trip Buffering method using the time similarity are the two slowest methods on average. The reason why the Trip Buffering method using time similarity is the slowest is due to the extra computation needed to retrieve the driver's travel time. And the reason why the OD Clustering method is slow may be explained by the extra time needed to build the user network and to cluster this network using the Louvain method.

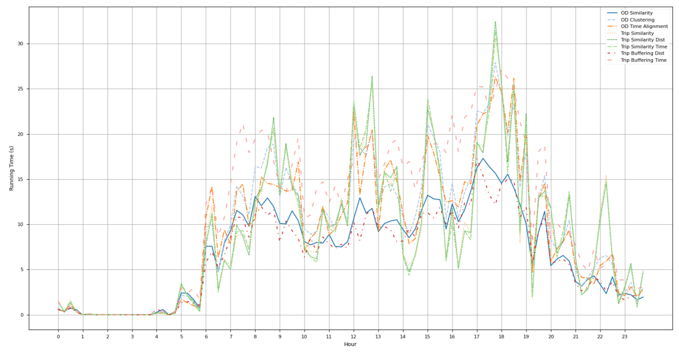

Fig. 4: Running time of the pooling methods according to the number of user requests.

As can be seen in Fig. 4, the execution time for each of the different methods tested depends directly on the number of simultaneous requests received. We also note that the three variants of the Trip Similarity method are much more sensitive to the number of queries than the other methods. Also, the two methods that seem most robust to variations in the number of queries are OD Similarity and Trip Buffering using the distance similarity function. During our simulation, the best method, the Trip Buffering using the detour distance similarity, certifies an average running time of between 3 and 17 seconds at peak times. It is also worth to note that the execution time of the methods also depends on the size of the graph being processed since all the methods require shortest path calculations to obtain similarity values.

7.2 Requests satisfaction

In this section, we compare the average size of pools created as well as the average satisfaction of the user requests of the different methods presented in Section 5.

Fig. 5 shows the average number of users, drivers and pedestrians, in each of the pools created. Based on our user request simulation model, we can see that the average vehicle occupancy across all methods exceeds 3 users, indicating that the cars are nearly at maximum capacity. It can be seen that the OD Similarity method is the ones for which the number of users is the most constant. The smaller average pool size in the OD Time Alignment method can be attributed to the random selection of requests. This randomness can result in significant time differences and consequently lower similarities.

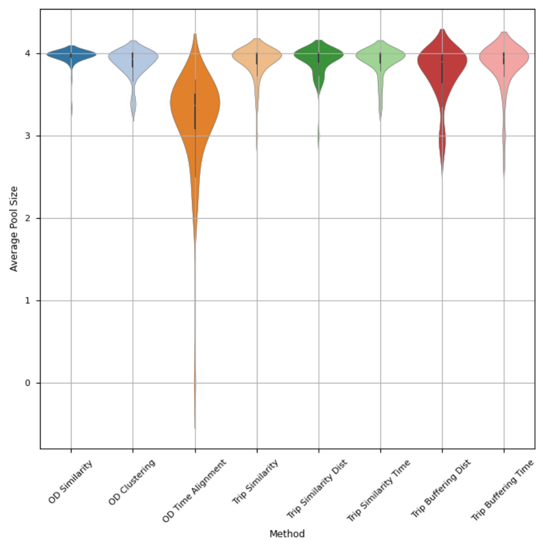

Fig. 5: Average pool size of the pooling methods.

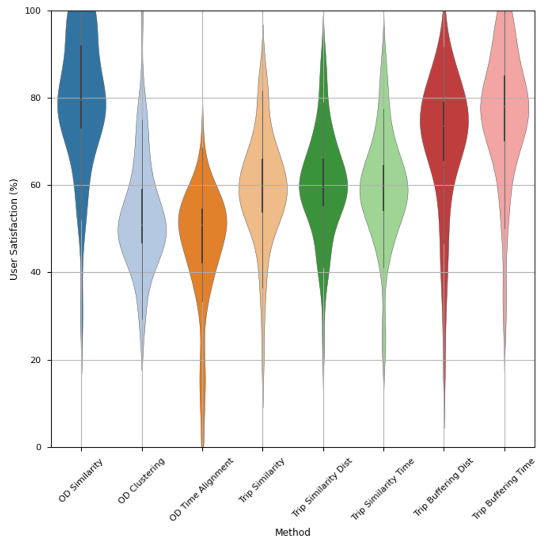

Fig. 6: Average satisfaction percentage of the pooling methods.

The satisfaction percentage is defined as the ratio between the number of passengers in a pool and the total number of passengers multiplied by 100, overall average results are given by the Fig. 6 and the average results for each hour of the day are given by the Fig. 7. Thanks to these two figures, we can see that, whether in randomized experiments or in the simulation, the OD Similarity method performs most successfully, followed by the two variations of Trip Buffering and Trip Similarity. The very high user demand satisfaction results of the OD Similarity method can be attributed to the fact that the average pool size is highly constant for this method and almost always equals the maximum vehicle capacity $v_{capacity} = 4$. One of the reasons for the poor performance of the OD Clustering method is that it depends on the Louvain algorithm to form the pools, which is not deterministic. Also, on the basis of our user request simulation model, we can see that on average all the methods manage to satisfy at least 30% of requests during rush hours.

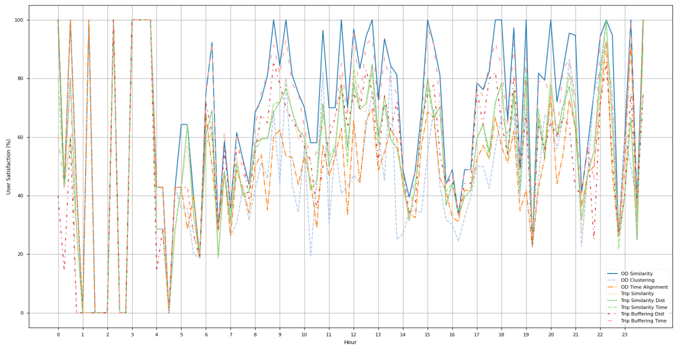

Fig. 7: Average satisfaction percentage of the pooling methods according to the number of user requests.

Furthermore, it is important to note that the percentage of satisfied users also depends directly on the number of randomly selected drivers and the capacity of their vehicles. Indeed, if there are not enough drivers and/or seats available at time t, then unavoidably some passengers cannot be satisfied. So when there are many requests, the probability of having more drivers increases, as does the percentage of requests satisfied, as shown in Fig. 7. Similarly, when there are fewer requests, only a few drivers are needed to satisfy the majority of requests. As with the execution time criterion, we can see that all methods follow a similar behavior, but this time it's not correlated with the number of requests. In our simulation, the best method, OD Similarity, certifies an average user satisfaction between 35% and 100% during rush hours.

7.3 Driver detour distance

In this section, we compare the different methods based on the criterion of the average detour made by the driver to pick up passengers. The detour is calculated by comparing the length of the direct path with the length of the path where the driver picks up the passenger. The result is the average of these detour distances for all pools of a method, in kilometers.

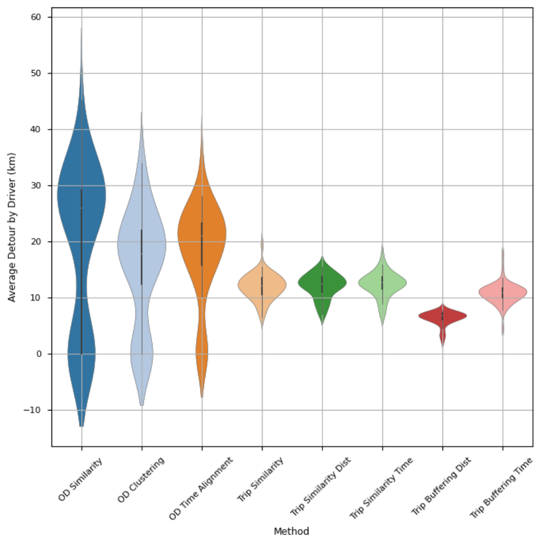

Fig. 8: Average detour distance for the driver by pooling methods.

In Fig. 8, we see that the method with the smallest average driver detour is the variant of the Trip Buffering method using the detour distance as criteria. This result is expected since it is the only method with a constraint on the maximum detour accepted by the driver. Also, We observe that the version of Trip Buffering based on the detour distance perform better in term of detour than the one based on the detour time. We also note that the three variants of Trip Similarity method perform equally well. This can be explained by the fact that these methods take into account multiple nodes on the path, unlike the OD Similarity and OD Clustering methods, which have the greatest deviation and only take into account the origin and destination.

In Fig. 9, we note that for certain instances, the detour distance of several methods is 0. This is due to the fact that the methods use the shortest path calculation to determine similarity, which in some cases does not exist, only the Trip Buffering method is not sensitive to this condition. For instance, for the hours in $[2, 4]$, all the detour distances are 0 because no user request can be satisfied. The best method, the Trip Buffering using the detour distance similarity, certifies an average driver detour between 4 and 10 kilometers during rush hours.

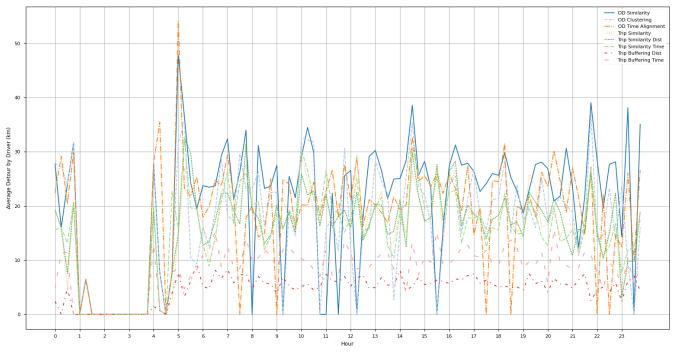

Fig. 9: Average detour distance for the driver by pooling methods according to the number of user requests.

7.4 Passenger waiting time

In this section, we compare the different methods on the basis of the average waiting time of passengers before the driver comes to pick them up.

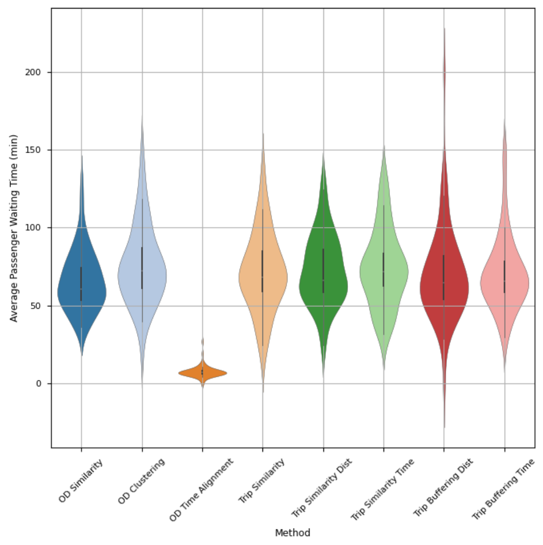

Fig. 10: Average waiting time for the passenger by pooling methods.

Fig. 10 shows that all the methods performed similarly in our randomized experiment except for the OD Time Alignment method. Indeed, this method achieves significantly better results because the user pools are composed of passengers and a driver with similar departure times, consequently reducing the average waiting time.

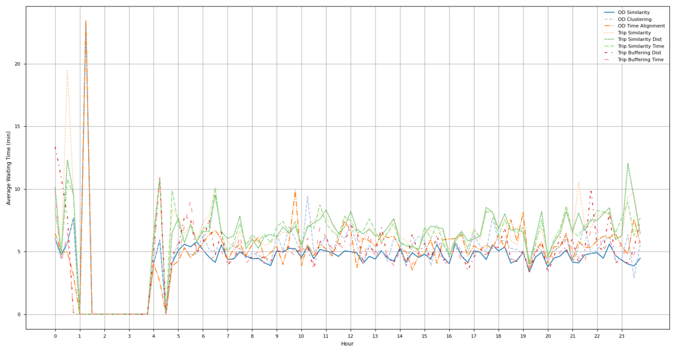

Fig. 11: Average waiting time for the passenger by pooling methods according to the number of user requests.

Fig. 11 illustrates that passenger waiting time depends on the number of user requests only when this number is below a certain threshold. Indeed, when the number of requests is low, not enough drivers are among the users, and passengers are therefore forced to wait longer. This suggests that increasing the number of participants will lead to better performance in terms of waiting time. This observation has also been made by [20] concerning the matching rate, which also seems to be our case for the percentage of satisfied requests. Once this threshold is reached, the waiting time for each method is stable. Although the OD Time Alignment method is the best on average over 50 random experiments, we can see from Fig. 11 that during our simulation the most constant method is the OD Similarity, certifying an average passenger waiting time between 4 and 6 minutes during rush hours.

7.5 Overall comparison

To summarize the results discussed in this section, based on the different parameters presented earlier, we compared the eight methods by establishing a score that is weighted sum of the criteria for the randomized experiment and for the simulation. The weights assigned to the respective criteria are given in Table 3.

We are looking for the method that minimizes the execution time, the driver's detour and the passenger's waiting time. This method must also maximize the percentage of satisfied users and the number of users in each car. Thus, the method with the highest score should be the one that best corresponds to the

Table 3: Criteria weight

Criterion	Weight
Running Time (s)	-1.5
User Satisfaction (%)	1
Average Pool Size	1
Number of Pools	0
Average Detour by Driver (km)	-1
Average Passenger Waiting Time (min)	-1

objectives and preferences pursued in this research. The scores obtained are given in Table 4.

Table 4: Method scores

Method	Randomized experiment score	Simulation score
Trip Buffering Dist	-99.706951	36.447783
OD Similarity	-100.205970	35.640391
OD Time Alignment	-122.408349	16.648278
Trip Similarity Dist	-158.499280	22.717071
Trip Similarity	-159.199678	22.693013
Trip Similarity Time	-161.330231	22.379183
Trip Buffering Time	-169.952633	33.921981
OD Clustering	-186.394580	18.173433

According to the weights chosen, the two methods that best satisfy the criteria are, for both experiments, the Trip Buffering method using distance similarity and the OD Similarity method.

8 Future works

For future improvements, we would like to modify the current proposed model. First, it would be interesting to take a more extensive period of time to build a more realistic model. Taking, for example, a full year, this would enable us to study in detail the adaptability of the different methods to changes in demand within the road network. Then, the model accuracy could be improved by taking intervals in minutes rather than hours, and by defining uniformly random functions for 5 minute intervals for example. In addition, it would be interesting to consider other types of distribution than Uniform, such as a model using a Gaussian mixture, as was done in [3]. Finally, concerning the model, we made the hypothesis that the number of requests in the city of Brussels followed the same distribution as for the entire country of Belgium, therefore it would be more advisable to define a model for a specific city rather than for an entire country, using to do so the population of the city comparatively to the population of whole country, for example.

Still on the subject of the data used for the experiments, in the future we plan to test the different methods and even combinations of methods on a real sample of students from one university. Indeed, in the particular context of universities, solving the ride-sharing challenge becomes important not only for the reasons listed above but also because according to [17] around 78% of students travel alone by car. In addition, according to [13], the propensity to practice peer-to-peer car-pooling is higher among younger people. Thus, we will be able to compare the theoretical results presented here with the results obtained under real world conditions.

Concerning the methods themselves, as a first step, it would be interesting to analyze the evolution of execution times for each method in relation to different road network sizes, as well as to explore scalability issues and execution times across larger datasets or varied urban environments other than the Brussels road network, as some may scale very well and others not. In a second step, we would like to test combinations of methods to try and get the best out of each of them. For example, we could imagine using one method during rush hour and another during calm periods. It would be valuable to analyse if this type of hybrid strategy could have a positive impact on the percentage of satisfied requests in the system.

Finally, regarding the preferences and objectives studied here, in the future we would like to take into account the criterion of destination arrival time guarantees for passengers and drivers. Indeed, even if the waiting time is important, such a system can only be viable in practice if it offers guarantees on the arrival time. This new parameter should therefore be taken into account in the scoring of the different methods.

9 Conclusion

We have proposed a comprehensive comparison of five methods for forming user pools for ride-sharing. In addition, we presented two different similarity functions, one based on time and the other on distance. We have proposed variants of the five initial methods using these two similarity functions, and finally compared the eight methods obtained. These methods were evaluated according to five main criteria: running time, percentage of request satisfaction, pool size, passenger waiting time and driver detour distance. In our experiments, whether randomized or during simulation, the Trip Buffering method using the distance similarity function always outperformed the other methods. Our results indicate that the Trip Buffering method using the distance similarity function is the most effective for optimizing the key metrics in ride-sharing. The presented results were based on a user request simulation model that we have proposed for Belgium, further supporting the robustness and applicability of our findings.

Acknowledgments. This project was supported by the FARI - AI for the Common Good Institute (ULB-VUB), financed by the European Union, with the support of the Brussels Capital Region (Innoviris and Paradigm).

References

1. Aydin, O.F., Gokasar, I., Kalan, O.: Matching algorithm for improving ride-sharing by incorporating route splits and social factors. PLOS ONE **15**(3), 1–23 (03 2020). https://doi.org/10.1371/journal.pone.0229674, https://doi.org/10.1371/journal.pone.0229674
2. Baudru, J., Bersini, H.: Heuristic optimal meeting point algorithm for car-sharing in large multimodal road networks. In: Proceedings of the 10th International Conference on Vehicle Technology and Intelligent Transport Systems. VEHITS (May 2024). https://doi.org/10.5220/0000186800003702
3. Bei, X., Zhang, S.: Algorithms for trip-vehicle assignment in ride-sharing. Proceedings of the AAAI Conference on Artificial Intelligence **32**(1) (Apr 2018). https://doi.org/10.1609/aaai.v32i1.11298, https://ojs.aaai.org/index.php/AAAI/article/view/11298
4. Biying, Y., Ye, M., Meimei, X., Baojun, T., Bin, W., Jinyue, Y., Yi-Ming, W.: Environmental benefits from ridesharing: A case of beijing. Applied Energy **191**, 141–152 (2017). https://doi.org/https://doi.org/10.1016/j.apenergy.2017.01.052, https://www.sciencedirect.com/science/article/pii/S0306261917300600
5. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment (10), P10008 (2008)
6. Boeing, G.: OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. Computers, Environment and Urban Systems **65**, 126–139 (2017). https://doi.org/10.1016/j.compenvurbsys.2017.05.004, https://doi.org/10.1016/j.compenvurbsys.2017.05.004
7. Campana Mattia, G., Delmastro, F., Bruno, R.: A machine-learned ranking algorithm for dynamic and personalised car pooling services. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). pp. 1856–1862 (2016). https://doi.org/10.1109/ITSC.2016.7795857
8. Chi-Chung, T., Chun-Ying, C.: Heuristic algorithms for the dynamic taxipooling problem based on intelligent transportation system technologies. Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007) **3**, 590–595 (2007), https://api.semanticscholar.org/CorpusID:589680
9. De Carvalho, V.R., Golpayegani, F.: Satisfying user preferences in optimised ridesharing services:. Applied Intelligence **52**, 11257 – 11272 (2022), https://api.semanticscholar.org/CorpusID:246221061
10. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik **1**, 269–271 (1959), http://eudml.org/doc/131436
11. Fagundez, G., Massobrio, R., Nesmachnown, S.: Online taxi sharing optimization using evolutionary algorithms. 2014 XL Latin American Computing Conference (CLEI) pp. 1–12 (2014), https://api.semanticscholar.org/CorpusID:35643693
12. Furuhata, M., Dessouky, M.M., Ordóñez, F., Brunet, M.E., Wang, X., Koenig, S.: Ridesharing : The state-of-the-art and future directions (2013), https://api.semanticscholar.org/CorpusID:17974805

13. Gärling, T., Gärling, A., Johansson, A.: Household choices of car-use reduction measures. Transportation Research Part A: Policy and Practice **34**(5), 309–320 (2000). https://doi.org/https://doi.org/10.1016/S0965-8564(99)00039-7, https://www.sciencedirect.com/science/article/pii/S0965856499000397
14. Hagberg, A., Swart, P., Schult, D.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008)
15. Ivanova, I., Stefanov, M., Verity, J., Brokopp, N.E., Grassi, E., Munoz Mula, P., Pesce, P., Zych, A., Byalkova, M.: Sustainable urban mobility in the eu: No substantial improvement is possible without member states' commitment. European Court of Auditor (2020)
16. Ketabi, R., Alipour, B., Helmy, A.: Playing with matches: Vehicular mobility through analysis of trip similarity and matching (09 2018)
17. Luè, A., Colorni, A.: A software tool for commute carpooling: a case study on university students in milan. International Journal of Services Sciences - Int J Serv Sci **2** (01 2009). https://doi.org/10.1504/IJSSCI.2009.026540
18. Macharis, C., Tori, S., Séjournet, A., Keseru, I., Vanhaverbeke, L.: Can the covid-19 crisis be a catalyst for transition to sustainable urban mobility? assessment of the medium- and longer-term impact of the covid-19 crisis on mobility in brussels. Frontiers in Sustainability **2** (08 2021). https://doi.org/10.3389/frsus.2021.725689
19. Mandl, N., Pinterits, M., Anderson, G., Carmona, G., Gager, M., Gaisbauer, S., Goll, M., Grassi, G., Gschrey, B., Gueguen, C., Gugele, B., Jeannot, C., Jespers, K., Jevsejenko, S., Juvyns, O., Kastori, M., Krtkova, E., Lanza, F., Leip, A., Matthews, B., Mellios, G., Moosmann, L., Moorkens, I., Nicco, L., Ondrusova, B., Osterheld, S., Maria, P., Rigler, E., Schmidt, G., Sosa, C.R., Szemesova, J., Abad Viñas, R., Vincent, J.: Annual european union greenhouse gas inventory 1990-2017 and inventory report 2019. European Environment Agency (2019)
20. Neda, M., Jayakrishnan, R.: A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. Transportation Research Part B: Methodological **106**, 218–236 (2017). https://doi.org/https://doi.org/10.1016/j.trb.2017.10.006, https://www.sciencedirect.com/science/article/pii/S0191261517301169
21. Psaraftis, H.N.: A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science **14**(2), 130–154 (1980), http://www.jstor.org/stable/25767975
22. Rodenbach, J., Matthijs, J., Seeuws, B., Ryvers, S., Decombel, S.: Impact report, car-sharing in belgium in 2022 (2022), https://www.autodelen.net/wp-content/uploads/2023/03/Impact-report-Car-sharing-in-Belgium-in-2022.pdf
23. Schrank, D., Eisele, B., Lomax, T.: 2012 urban mobility report. Texas Transportation Institute;Southwest Region University Transportation Center (U.S.) (2012), https://rosap.ntl.bts.gov/view/dot/61387
24. Shirley, C., Gecan, R., Kile, J., Adler, D., Austin, D., Chase, N., Hopkins, B., Krupkin, A., Prendergast, T., Reese, R., Rosenberg, J., Tawil, N., Willie, S., Dwyer, M., Lattanzio, R., Kling, J., Sunshine, R.: Emissions of carbon dioxide in the transportation sector. Congressional Budget Office, Nonpartisan Analysis for U.S. Congress (2022)
25. Silwal, S., Gani, M.O., Raychoudhury, V.: A survey of taxi ride sharing system architectures. pp. 144–149 (06 2019). https://doi.org/10.1109/SMARTCOMP.2019.00044

26. Thaithatkul, P., Seo, T., Kusakabe, T., ASAKURA, Y.: A passengers matching problem in ridesharing systems by considering user preference. Journal of the Eastern Asia Society for Transportation Studies **11**, 1416–1432 (12 2015). https://doi.org/10.11175/easts.11.1416
27. Toth, P., Vigo, D., Toth, P., Vigo, D.: Vehicle routing: Problems, methods, and applications, second edition (2014)
28. Wojciechowski, J., Wisniewska-Danek, K., Radecka-Moroz, K., Friel, C., Coelho, J., Soblet, F., Niemenmaa, V., Happach, B., Kubat, J., Otto, J., Pirelli, L., Simeonova, R., Zalegan, A., O'Doherty, R.: Air pollution: Our health still insufficiently protected. European Court of Auditor (2018)
29. Yatnalkar, P., Narman, H.: A matching model for vehicle sharing based on user characteristics and tolerated-time (10 2019). https://doi.org/10.1109/HONET.2019.8908058