Can dynamic ride-sharing reduce traffic congestion?

Negin Alisoltani, Ludovic Leclercq, Mahdi Zargayouna

Transportation Research Part B 145 (2021) 212–246

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

Can dynamic ride-sharing reduce traffic congestion?

Negin Alisoltani[a,b], Ludovic Leclercq[a,*], Mahdi Zargayouna[b]

[a] Univ. Gustave Eiffel, Université de Lyon, ENTPE, LICIT, Lyon F-69518, France
[b] Univ. Gustave Eiffel, IFSTTAR, COSYS-GRETTIA, Marne-la-Vallée F-77454, France

ARTICLE INFO

Article history:
Received 31 July 2019
Revised 12 January 2021
Accepted 13 January 2021

Keywords:
Dynamic ride-sharing
Traffic congestion
Trip-based MFD
Optimal fleet management

ABSTRACT

Can dynamic ride-sharing reduce traffic congestion? In this paper we show that the answer is yes if the trip density is high, which is usually the case in large-scale networks but not in medium-scale networks where opportunities for sharing in time and space become rather limited. When the demand density is high, the dynamic ride-sharing system can significantly improve traffic conditions, especially during peak hours. Sharing can compensate extra travel distances related to operating a mobility service. The situation is entirely different in small and medium-scale cities when trip shareability is small, even if the ride-sharing system is fully optimized based on the perfect demand prediction in the near future. The reason is simple, mobility services significantly increase the total travel distance, and sharing is simply a means of combating this trend without eliminating it when the trip density is not high enough. This paper proposes a complete framework to represent the functioning of the ride-sharing system and multiple steps to tackle the curse of dimensionality when solving the problem. We address the problem for two city scales in order to compare different trip densities. A city scale of 25 km^2 with a total market of 11,235 shareable trips for the medium-scale network and a city scale of 80 km^2 with 205,308 demand for service vehicles for the large-scale network over a 4-hour period with a rolling horizon of 20 minutes. The solutions are assessed using a dynamic trip-based macroscopic simulation to account for the congestion effect and dynamic travel times that may influence the optimal solution obtained with predicted travel times. This outperforms most previous studies on optimal fleet management that usually consider constant and fully deterministic travel time functions.

1. Introduction

In recent years, intelligent transportation systems have reshaped traditional transportation supply with the rapid introduction of new mobility services. Among these services, ride-sharing is becoming popular (Tahmasseby et al., 2014).

Ride-sharing consists in sharing a car using an e-hailing application to save costs and resources. Dynamic ride-sharing refers to a system that supports an automatic ride-matching process between participants at very short notice or even en-route (Agatz et al., 2012). This service is often advertised as a significant way of alleviating congestion and more generally as being eco-friendly, but few results exist to support this claim, and some claims to the contrary have also been expressed

* Corresponding author.
E-mail address: ludovic.leclercq@univ-eiffel.fr (L. Leclercq).

https://doi.org/10.1016/j.trb.2021.01.004

(Caulfield, 2009). Indeed, ride-sharing definitely reduces the number of cars travelling, but it can also increase travel distance. Both phenomena must be considered to evaluate the actual impact of ride-sharing. This study tackles this question for a medium-size network by simulating and solving an optimal fleet management problem.

The dynamic ride-sharing problem involves two sub-problems:

1. Satisfying demand and managing a fleet of vehicles
2. Accurately predicting travel times to determine vehicle availability and pickup/drop off times.

The first sub-problem corresponds to a fleet management optimization problem with multiple objectives and has recently attracted much attention.

1.1. Problem formulation

In a great deal of research, the optimal assignment is formulated as an integer linear programming problem and then different approaches are taken to optimize the problem (Zargayouna and Zeddini, 2012). In this paper, we try to rigorously formulate the problem considering the essential objectives and constraints for the passengers and providers.

Much research work on ride-sharing has tried to minimize the total distance for cars to accommodate the trips requested (Ota et al., 2017; d'Orey and Ferreira, 2014; Qian et al., 2017). This leads to minimizing the provider's costs. In ride-sharing, the participant's willingness to share their ride is critical. Therefore, it is important to consider the riders objectives too. In the dynamic ride-sharing method proposed by Agatz et al. (2011), the objective was to minimize the total vehicle-miles driven by all participants. They showed that this objective is aligned with societal objectives for reducing emissions and traffic congestion.

Travel time is an important feature for the participants (Naoum-Sawaya et al., 2015). Another important objective for the passengers is the time that they have to wait for the ride. As Stiglic et al. (2016) showed, if no match is found before a specified time, the passenger is likely to leave the system. Previous research usually focused on one of these objectives at a time, but it is important to take them into account simultaneously. This paper uses an objective function that combines all these criteria.

One of the most important considerations in dynamic ride-sharing is the time constraint. Many of the systems proposed in the literature let users choose their earliest and latest pickup times (Linares et al., 2016; Agatz et al., 2011). Besides the constraints on travel time, this paper introduces the desired maximum number of ride-sharers for all passengers. Thus, in a given vehicle, the number of on-board passengers cannot exceed the lowest number of passengers willing to share for all the on-board passengers. This parameter can also affect traffic.

1.2. Problem solving

The ride-sharing assignment is a pickup and delivery problem with time windows (PDPTW) (Mahmoudi and Zhou, 2016). There is a vast literature on solution methods and algorithms for these problems. However, there is still room for improvement in these methods. Recently Mourad et al. (2019) has presented a survey of models and algorithms for optimizing shared mobility, and they have shown that one of the most important problems in the solution for these systems is computation time and the quality of the results. The presented algorithm in this study can make big progress in solving the dynamic ride-sharing problem by providing high-quality solutions in a short time. Some studies in the literature use branch and bound methods to solve PDPTW (Ghilas et al., 2018). The algorithm proposed here is based on the branch and bound concept but with specific configurations and in particular cluster decomposition to find the exact solution for the matching problem over small instances, e.g., a low number of requests.

Most approaches attempt to find the near-optimal solution to the matching problem in ride-sharing systems by considering specific constraints like vehicle capacity and the time window, to minimize the total additional distance (Ota et al., 2017; Qian et al., 2017; d'Orey and Ferreira, 2014) and maximize the match between vehicles and passengers (Stiglic et al., 2016; Ma et al., 2013; Goel et al., 2017). They usually rank the possible, feasible matches for passengers and cars close to each other, based on the objective function and then choose the best match for the requests.

Hyland and Mahmassani (2018) assigned the passenger to a vehicle only if the latter is 20% closer to the passenger than any idle shared car. The assignment problem is solved with different heuristic methods in the literature. Herbawi and Weber (2012b) used a genetic algorithm to find a sub-optimal solution for the ride-matching problem, and then an insertion heuristic took care of the newly received requests by modifying the solution of a genetic algorithm when possible.

In this paper, we aim to approach the global optimal solution. The search for the global solution may be computationally expensive, but it permits answering the question of the maximum gain we can expect from ride-sharing in the transportation system. For the same reason, we introduce no uncertainty in the trip demand and consider that all requests are known over a rolling horizon of 20 minutes. To approximate the global solution, we resort to an algorithm to solve an integer linear program.

However, as we are targeting problems with large instances, we are still faced with the curse of dimensionality. Our solution approach is designed to be exact for small samples. It is then extended with several heuristics that keep the general design for the solution method but significantly reduce its computation time. In the large-scale problems, the number of

received requests at every time is huge. It has been indicated in the literature that the patterns of demands and the patterns of supplies are spatially-temporally dependent (Wang et al., 2019).

A lot of researches on this domain uses different clustering methods to consider these dependencies. They use methods like dividing the time into several time slots or dividing the space into several clusters, road segments, or cells (Gonzalez et al., 2008; Davis et al., 2018; Qi and Liu, 2018; Yuan et al., 2012).

In Chen et al. (2020) all pickup points are partitioned into several clusters and the taxi dispatching problem is solved in each cluster. The authors in Bard and Jarrah (2009) show that for large-scale problems, an appropriate solution is clustering the demand nodes and downsizing the network. Some researches try to limit the feasible region with clustering methods to speed up the computation. They usually divide the demand nodes in the network into geographically dense clusters (Özdamar and Demir, 2012; Sáez et al., 2008). Although the clustering approach is widely used in vehicle routing problems, there are relatively limited number of studies employing clustering methods in the ride-sharing problem (Li and Chung, 2020).

The main strategy here is to cluster the requests depending on a shareability index to create smaller samples that are faster to solve. This method narrows the exploration of the space to feasible and promising states only. As the number of assigned passengers increases for a car, the intersection of feasible areas becomes smaller, and the algorithm can compute the assignment of running cars.

1.3. Considering traffic dynamics

The second sub-problem has been given less attention in the literature, but is very important if an operational deployment is envisioned. Network congestion can have significant impacts on the ride-sharing service.

The optimization system of the ride-sharing service uses estimates for the predicted travel time obtained from a so called "prediction model". When the rides are executed, a gap usually exists between the estimation and the real traffic condition. The so called "plant model" represents the real traffic condition and it may require dynamic adjustment of the initial assignment to fit with the conditions observed. When simulating a dynamic ride-sharing service, it is essential to accurately distinguish the prediction and the plant models to provide a realistic service.

In most research, the plant model and the prediction model are the same (Zou and Dessouky, 2018; Goel et al., 2017; Ma et al., 2015). There is no benchmark considering traffic conditions, but a few studies have considered the impact of traffic conditions on ride-sharing (Ordóñez and Dessouky, 2017; Wang et al., 2016). For instance, Goel et al. (2017) proposed an approach where the pick-up and drop off locations for passengers are selected from a fixed set. They considered a randomly chosen overhead of 10–20 percent to reflect different traffic conditions when computing the end time for a driver. Even with this consideration, the authors used only the prediction model and assumed that the travel times during the assignment process stayed the same during the execution of the vehicle schedules. Other works used only static travel times in the optimization process (Herbawi and Weber, 2012a).

In some research, only the plant model is considered. For instance, in (Linares et al., 2016), (Ma et al., 2015) and (Jia et al., 2017), the authors used a simulator to assess the dynamic ride-sharing but they did not optimize vehicle allocation. In Ban et al. (2019), the authors develop a general economic equilibrium model at the macroscopic level to describe the equilibrium state of a transportation system composed of solo drivers and the e-hailing service providers. Experimental results show that when there is little in the symmetry in the network demand, the travelled distances increase significantly with the service usage due to the increase in deadhead miles. However, as the symmetry increases the impact on deadhead miles significantly reduces with increased service usage. These results are coherent with our findings, where the mobility services significantly increase the total travel distance, and sharing is a means of combating this trend without eliminating it.

In our method, we define the plant model in addition to the prediction model to assess the impact of traffic conditions on the performance of the dynamic ride-sharing system for large-scale problems and see how the dynamic ride-sharing system can impact traffic congestion. We use real data from the Lyon network in our simulations. In fact, our plant model shows the real equilibrium in the system. The prediction model considered is based on the last observed travel times, while the plant model considered is a trip-based Macroscopic Fundamental Diagram (MFD) model able to reproduce the evolution over time of mean traffic conditions for a full road network using the MFD as a global behavioral curve (Lamotte and Geroliminis, 2016; Mariotte et al., 2017; Mariotte and Leclercq, 2019). The macroscopic fundamental diagram (MFD) shows the rapid evolution and gives a synthetic overview of network states (Ameli et al., 2020).

Given the review above, we have the following assumptions about the impact of dynamic ride-sharing on traffic congestion :

1. Increasing the market-share (the percentage of service vehicles) when the number of sharing is 0 (no sharing) increases the vehicles' total travel time and distance and, consequently, the traffic congestion in the network. Sharing the trips by increasing the number of sharing leads to reduce the total travel time and distance in the network.
2. In the medium-scale and small-scale networks, sharing can combat the increase in total travel time and distance, but it can not eliminate it. Thus, it can not make a considerable improvement to the traffic situation
3. In the large-scale networks, the accumulation of demand for the service is very high compared to the medium and small-scale networks. Therefore, dynamic ride-sharing is different in large-scale, and it can significantly reduce traffic congestion.

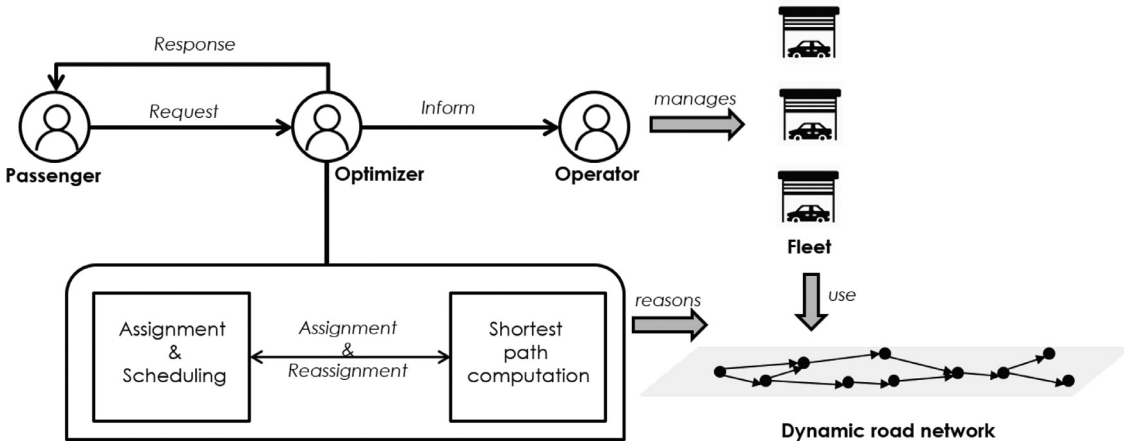


Fig. 1. System components.

In this paper, we perform an extensive simulation study (based on real-world traffic patterns) to assess the influence of dynamic ride-sharing systems on traffic congestion in medium-scale and large scale networks. Different situations (five different market-shares and three numbers of sharing) are investigated in terms of traffic conditions compared with a base traffic situation where all the requests are served with personal cars.

1.4. Contributions

The main contributions of this paper can be summarized as follows:

- We present a new and efficient exact solving algorithm for small instances of ride-sharing problem.
- We define a new shareability index and a new smart decomposition of the optimization problem based on the index for the ride-sharing problem. This allows us to find near-optimal solutions for much larger instances.
- We build a new modeling framework with the plant and the prediction model for the dynamic ride-sharing problem using a trip-based dynamic model to assess the evolution of travel time and the congestion dynamic in the system while reproducing the behavior of all vehicles in the system and not only the service cars.

The paper is organized as follows. In Section 2, we present the characteristics of the system and introduce a mathematical model of the ride-share optimization problem. In Section 3, we discuss the solution algorithm we have developed for this optimization problem. In Section 4, we design the experimental platform and study the quality of the solution method. In Section 5, we present and analyze the results to assess the influence of the system on congestion and, finally, in Section 6, we summarize the key findings and suggest directions for future research.

2. Dynamic ride-sharing design and optimization framework

The main characteristics of the ride-sharing problem we investigate are:

- Door-to-door dynamic ride-sharing (the passenger obtains service at the exact defined origin and destination).
- Passengers define the earliest pickup time and the latest arrival time. The passenger must be picked up, transported and dropped off at the destination inside this time window.
- All requests over the next prediction horizon (usually 20 minutes) are considered known at the beginning of the horizon.
- Each passenger defines the maximum number of persons they are ready to share a trip with. We call it "number of sharing". The service has to guarantee that the number of sharing constraint is always satisfied for all the cars.
- Service time is added to each trip to reflect the time to stop and get in and out of the car.
- The service is provided by a limited number of vehicles that are initially all in the central depot. Local depots are uniformly distributed over the network to represent locations where cars can wait for further assignments. When an idle waiting car is needed to serve a passenger, it comes from the nearest non-empty depot. Note that the central depot can always generate new cars if necessary. When a car ends a trip without any further short-term assignment, it goes to the nearest depot and waits there.

The global functioning of the system is as follows (Fig. 1). At the beginning of a new rolling horizon, the fleet management components solve the optimal assignment problem based on predicted travel times. Then, the simulation component implements this assignment over the next horizon to determine the actual evaluation of the system and the effective pickup/drop off and travel times for all the vehicles.

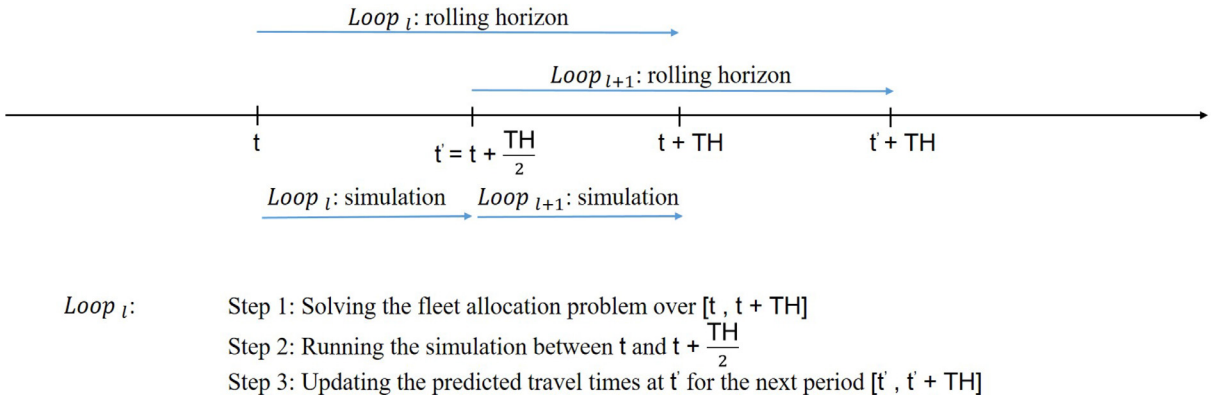


Fig. 2. Rolling horizon and simulation period.

The effective value of the objective function (objective function realized) can be computed at the end of each horizon and be compared to the optimal objective value (estimated objective function) derived from solving the assignment problem with predicted travel times.

In the following, the mathematical description of both components (fleet management component and dynamic simulation component) is provided.

Note that the predicted travel times used in the fleet management correspond to the trip length divided by the mean speed over the full network at the end of the previous simulation period. Note also that we stop the simulations halfway on the rolling horizon ($\frac{TH}{2}$) and solve a new fleet allocation problem over a new full rolling horizon (TH). This prevents the system from being myopic to the new demand that may arrive just after the end of a simulation period, see Fig. 2.

2.1. Fleet management component

At each period (rolling horizon width), the system receives n passenger requests. Every request has a pickup point i and a drop-off point $i+n$. Thus, there will be $2n$ nodes ($\{1,\dots,2n\}$) plus nodes 0 (corresponding to the depot from where the vehicle comes) and $2n+1$ (corresponding to the depot where the vehicle goes after finishing all assignments). If vehicle k is assigned to passenger i, the decision variable y_i^k equals 1 and if not it equals 0. If vehicle k takes the passenger from point i to j, the decision variable $x_{i,j}^k$ equals 1 and if not it equals 0.

Table 1 gives the list of the notations that we use in this paper. The predicted direct travel time between every two points at each time is DTT_{ij}^t. Other variables in the model are computed based on DTT_{ij}^t.

The time window for passenger pickup and drop off can be derived from the departure time (earliest pickup time) and arrival time (latest arrival time) defined by the passenger. The difference between the earliest pickup time and the earliest drop off time is the minimum time needed to go from the passenger's origin to their destination (direct free flow travel time):

- pickup time window: $(EP_i, LD_i - DTT_{i,i+n}^t)$
- drop off time window: $(EP_i + DTT_{i,i+n}^t, LD_i)$

Waiting time is the time that the passenger must wait before being picked up. When the passenger defines the earliest pickup time, the vehicle cannot serve them before this time or after the latest pick up time. If the car arrives at point i before EP_i, it must wait to pick up the passenger at their desired time. So, the waiting time, in this case, is zero for the passenger. But if the car arrives after EP_i, the waiting time is the difference between the pickup time and the lower bound of the pickup time window:

$$WT_i = P_i^k - EP_i \qquad \forall i \in P \tag{1}$$

The exact passenger pickup time is the time that the vehicle arrives at the passengers location. This time can be computed as follows:

$$P_i^k = \sum_{g=0}^{i-1} \sum_{j=n+1}^{i+n-1} \sum_{k=1}^{m} DTT_{i,j}^t . x_{gj}^k . y_i^k + DTT_{i+n-1,i}^t + ST_i \qquad \forall i \in P, t \in TH \tag{2}$$

The total travel time for the passenger is defined as the sum of the service time and the predicted travel time.

$$TT_i = P_{i+n}^k - P_i^k + ST \qquad \forall i \in P \tag{3}$$

Table 1
Notation.

m	Fleet size
n	Number of passengers
M	Set of vehicles, $M=\{1,\dots,m\}$
N	Set of passengers, $N=\{1,\dots,n\}$
P	Set of pick up points, $P=\{1,\dots,h\}$
D	Set of drop off points, $D=\{1,\dots,g\}$
A	Set of all pick up and drop off points, $A=P\cup D$
O	Set of all the stop points, $O=A\cup\{0,2n+1\}$
TH	Time horizon
org_i	Origin point of passenger $i\in N$
des_i	Destination point of passenger $i\in N$
n_i^{share}	Maximum number of sharing for passenger $i\in N$
d_i	Number of seats demanded for passenger i
Cap	Vehicle capacity
c_i^k	Capacity of vehicle $k\in M$ at point i
ST_i	Service time for passenger i
EP_i	Earliest pick up time for passenger i
LP_i	Latest pick up time for passenger i
ED_i	Earliest drop off time for passenger i
LD_i	Latest drop off time for passenger i
P_i^k	The time when the vehicle k arrives at point i
TT_i	Total travel time for passenger i
WT_i	Waiting time for passenger i
$WT_i^{'}$	Waiting time for passenger i when served individually
T_k	Travel time for vehicle k
TD_k	Travel distance for vehicle k
N_{trips}	Number of trips
α	Weight of waiting time
β	Weight of passenger total travel time
γ	Weight of vehicle total travel time
δ	Weight of vehicle total travel distance
$DTT_{i,j}^t$	Direct travel time from point i to j at time t, $(i,j\in A, t\in TH)$
$DTD_{i,j}$	Direct travel distance from point i to j, $(i,j\in A)$
$x_{i,j}^k$	Decision variable equal to 1 if vehicle k takes the passenger from point i to j and 0, otherwise
y_i^k	Decision variable equal to 1 if vehicle k is assigned to passenger on point i and 0, otherwise

The total travel time for each vehicle is the summation of the direct travel time for all the trips that are served over the time horizon by this vehicle.

$$T_k=\sum_{i\in O}\sum_{j\in O}DTT_{i,j}^t.x_{i,j}^k \qquad \forall k\in M, t\in TH \tag{4}$$

Also, the total travel distance for a vehicle is the summation of the direct travel distance for all the trips that are served over the time horizon by this vehicle:

$$TD_k=\sum_{i\in O}\sum_{j\in O}DTD_{i,j}.x_{i,j}^k \qquad \forall k\in M \tag{5}$$

The capacity of vehicle k when it arrives at point i is the summation of the number of passengers that are picked up at point i by this car minus the number of passengers that are dropped off from the vehicle at this point.

$$c_i^k=Cap-\sum_{g=1}^{i-1}\sum_{j=1}^{i+n-1}\sum_{k=1}^{m}d_g.x_{g,j}^k.y_i^k+\sum_{g=1}^{i}\sum_{j=1}^{i+n-1}\sum_{k=1}^{m}d_g.x_{g,j}^k.y_i^k \qquad \forall i\in P \tag{6}$$

Here we introduce the fleet management problem that can be expressed as the following integer linear program:

$$\min\sum_{i\in P}(\alpha.WT_i+\beta.TT_i)+\sum_{k\in M}\gamma.T_k+\delta.TD_k \tag{7}$$

subject to:

$$d_i\le\sum_{k\in M}c_i^k.y_i^k,\ \forall i\in P \tag{8}$$

$$\sum_{j\in O}x_{i+n,j}^k.y_i^k(c_{i+n}^k+d_i-c_j^k)=0,\ \forall i\in P,\forall k\in M \tag{9}$$

$$\sum_{j\in A} x_{i,j}^k . y_i^k (c_i^k - d_i - c_j^k) = 0, \forall i \in P, \forall k \in M \tag{10}$$

$$EP_i - P_i^k - ST_i \leq 0, \forall i \in P, \forall k \in M \tag{11}$$

$$P_i^k + ST_i - LD_i + DTT_{i,i+n}^t \leq 0, \forall i \in P, \forall k \in M \tag{12}$$

$$EP_i + DTT_{i,i+n}^t - D_{i+n}^k - ST_i \leq 0, \forall i \in P, \forall k \in M \tag{13}$$

$$D_{i+n}^k + ST_i - LD_i \leq 0, \forall i \in P, \forall k \in M \tag{14}$$

$$x_{i,j}^k (P_i^k + DTT_{i,j}^t + ST_i - P_j^k) \leq 0, \forall i, j \in A \tag{15}$$

$$\sum_{k=1}^{m} y_i^k = 1, \forall i \in P \tag{16}$$

$$\sum_{j\in A} x_{i,j}^k - \sum_{j\in A} x_{j,i+n}^k = 0, \forall i \in P, \forall k \in M \tag{17}$$

$$\sum_{i\in P\cup\{0\}} x_{i,j}^k - \sum_{i\in D\cup\{2n+1\}} x_{j,i}^k = 0, \forall j \in O, \forall k \in M \tag{18}$$

$$\sum_{j\in P} y_j^k . d_j - \sum_{j\in D} y_j^k . d_j - n_i^{share} \leq M1.(1 - y_i^k), \forall k \in M, \forall i \in N \tag{19}$$

$$c_j^k - M2.(1 - x_{0,j}^k) - Cap \leq 0, \forall j \in P, \forall k \in M \tag{20}$$

$$c_j^k + M2.(1 - x_{0,j}^k) - Cap \geq 0, \forall j \in P, \forall k \in M \tag{21}$$

$$m \leq M3 \tag{22}$$

$$x_{i,j}^k \in \{0, 1\}, \forall i, j \in O \tag{23}$$

$$y_i^k \in \{0, 1\}, \forall i \in N, k \in M \tag{24}$$

The objective function is to minimize the waiting time and the total travel time for passengers and the total travel time and distance for vehicles. α, β, γ and δ are the weights for the passenger waiting time and travel time and the vehicle travel time and travel distance respectively (Eq. (7)). The objective weights are determined according to a prior definition of their relative importance. Constraint (8) to(10) are on capacity. The first ensures that at each pickup point, the demand does not exceed the vehicles capacity at that point. The second is to be assured that all the passengers who are picked up at the origin, will be dropped off at the corresponding destination. The third constraint is to ensure that the passengers who are picked up at point i stay in the vehicle up to their destination. Constraint (11) to(15) are on time. The time windows for the pickup and drop off are ensured with constraint (11) to(14). The drop off point must be visited after the pickup point and sufficient time must be guaranteed for service time and the travel time between the origin and destination. This constraint is imposed by inequality (15). Constraint (16) to (18) are related to the assignment. Constraint (16) ensures that just one vehicle is assigned to passenger i. Constraint (17) guarantees that the same vehicle is handling a passenger pickup and drop off. Constraint (18) is the flow constraint, to be sure that the vehicle that enters a service node will also exit from it. Constraint (19) ensures that the number of passengers in a car is lower than or equal to the number of sharing that the passenger has defined. Constraint (20) and (21) work together to guarantee that when a vehicle exits the depot, it has no passenger on board. Constraint (22) ensures that there is a sufficient number of vehicles in the fleet. The possible values of $x_{i,j}^k$ and y_i^k are given by (23) and (24).

2.2. Dynamic simulation component

We use two models in our simulation framework. The plant model implements the results of the optimal fleet allocation problem to provide a dynamic simulation of the full transportation system. The prediction model provides travel time estimations when solving the fleet allocation problem. It is presented in Section 4

The plant model considers all the vehicles in the transportation network and not only the ride-sharing vehicles. That means that we also have a baseline of personal trips composed of: (i) trips that come from or go outside the area studied and that cannot be shared; (ii) trips that are wholly inside the region but for which rides are not requested and personal cars are chosen.

The market-share defines among the several trip categories how many users will request a ride or choose their personal car. Simulating all the trips whatever their mode of transportation mode (personal car or service car) guarantees that we can properly track congestion during peak hours.

In this research, the trip-based MFD is used to accommodate individual trips while keeping a very simple description of traffic dynamics (Lamotte and Geroliminis, 2016; Mariotte et al., 2017; Mariotte and Leclercq, 2019; Leclercq et al., 2017). The general principle of this approach is to derive the inflow and outflow curves, noting that the travel distance L_i by a car i entering at time $t - T(t)$ when $n(t)$ is the number of en-route vehicles at time t and the mean speed of travelers is $V(n(t))$ at every time t, must satisfy the following equation:

$$L_i = \int_{t-T(t)}^{t} V(n(s))ds \tag{25}$$

The function $V(n(t))$ is the speed macroscopic fundamental diagram and can be derived from common observations for a transportation network (Leclercq et al., 2014). For more details on the functioning of trip-based MFD, readers can refer to (Leclercq et al., 2017; Mariotte and Leclercq, 2019).

The service cars can have two situations. They are waiting in depots for new passengers, or they are servicing the assigned passengers. In addition to the shared cars that are circulating to serve the passengers, other cars are serving the rides that are not shared in the network. So, the accumulation at each time t is the summation of the number of circulating service vehicles and the number of personal vehicles in the system. Therefore, at each time t the mean speed of travelers can be computed.

At each time step, the simulator computes the current speed of the cars considering the current traffic situation (the number of en-route vehicles). Then, the vehicle can cover a distance based on the current speed at every time step. So, the situation of cars is updated every time step, with the speed computed in the time and the remaining travel distance to cover. The time step that we use in our plant model is 1 second. So, the state of en-route cars is updated every second in the simulations.

3. Solution methods

If n is the number of requests and m is the number of vehicles, the number of $x^k_{i,j}$ is $m(2n+1)(2n+1) = 4mn^2 + 4mn + m$ and the number of y^k_i is mn, so the number of variables of the model is $NV = 4mn^2 + 5mn + m$. Thus, the complexity of the optimization problem grows exponentially by small increases in the number of requests and vehicles. Also, softening the constraints, for example, increasing the number of sharing will increase complexity. One of the most important problems in the solution approaches for shared mobility systems is computation time and quality of the results (Mourad et al., 2019).

In this paper, we build a solution method with multiple steps that starts from finding the exact solution for small instances. Furthermore, we introduce extensions that speed up the solution method and can address bigger networks, even large-scale networks, while assessing the difference in quality at each step. We show that the proposed heuristics can keep the quality of solutions at an acceptable level (near-optimal solution) while significantly decreasing the computation time. Thus, we design our solution method based on the classical branch and bound algorithm (Ross and Soland, 1975) but with specific properties to cope with a fleet management problem.

3.1. Exact assignment algorithm over the full-time horizon

The algorithm builds a tree of routes and tries to add the feasible points to the best branch of the tree at each step. It checks the feasibility of the points regarding the model constraints. In the beginning, it starts from the closest non-empty depot to the origins and adds the origin points to the branches of the tree (Fig. 3(a)). The algorithm can add a destination point if and only if its related origin point has been added to the route before (Fig. 3(b)). Also, it can add a new origin point if the capacity constraints and the number of sharing constraints are satisfied (Fig. 3(c)). The time window constraints must be checked when adding new stop points to the routes (Fig. 3(d)). When the algorithm finds a feasible point for a route, it creates a new route by adding this possible point and puts the newly created route in the set of paths.

Finally, the best route is the route that has the minimum objective function.

Algorithm 1 shows the optimization algorithm. Each part of the algorithm corresponds to one or multiple equations of our mathematical model (the number of the corresponding equations is shown at each level of the algorithm).

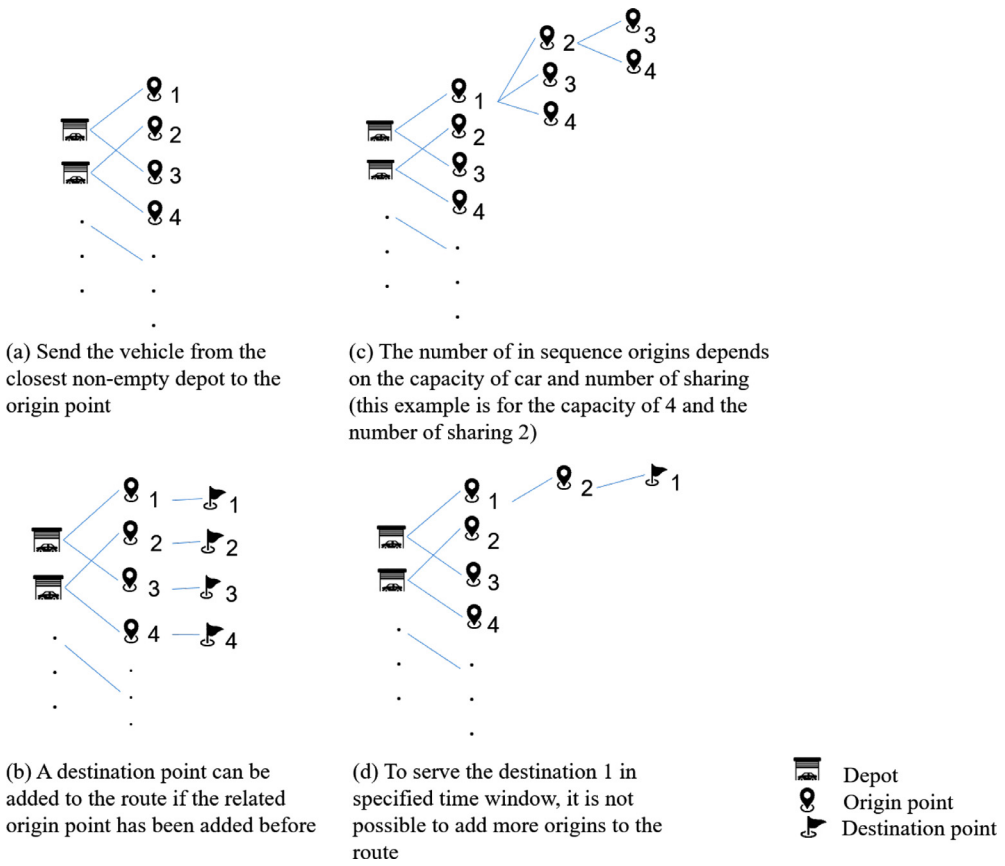


Fig. 3. The assignment algorithm function.

The initial route set *S* contains the origin points of the requests that are not already being assigned to a car. The algorithm builds the primary branches from the depots nearest the origins. Then, it finds the best branch with respect to the objective function (Eq. (7)) among these primary branches. The next branches are created as extensions of this primary branch. To add points to the primary branch, equation 17 of the model must be satisfied. Then the algorithm finds a set of points that can be added to the first branch. The feasibility of adding points to the branch is checked by equations 8–15,19,20 in the model. Then, the algorithm creates new branches by adding the feasible points. At each iteration, the optimal branch in terms of the objective function is selected to be the base branch. At the end, when it is no longer possible to add more points to a branch, if the points on the branch satisfy Eqs. (16) and (18) of the model, the branch can be added to the results set. Finally, the optimal branch is selected from the set of results.

When the schedule is received, the algorithm puts the associated vehicle in the en-route vehicles set, and as the car finishes the assigned schedule, it goes back to the nearest depot. At each iteration of the algorithm, a large number of branches are added to the route set. The critical point that makes our method efficient is that we remove the branches that are not feasible with respect to three kinds of constraint (on time, capacity and number of sharing).

This algorithm is exact and its complexity explodes with the number of branches. This is, for example, the case when we increase the number of sharing or requests. In this case, we need to introduce heuristics that reduce the exploration of the feasible solution.

3.2. Heuristic 1: Rolling horizon with re-scheduling

Considering all the requests over the full-time horizon can provide the global optimum solution. However, this greatly increases the number of variables and is not reasonable in practice. To reduce the number of variables, but also to bring the expression of the problem more in line with common practice, we now implement a rolling horizon, generally about 20 minutes. The requests are assumed known only over the next rolling horizon. The corollary is that we have to introduce a new process to handle travelling cars that have not yet reached their maximal occupancy because of the car or the passenger constraints. We therefore introduce a specific algorithm to assign the new requests in priority to en-route vehicles. The remaining requests are handled by the first algorithm presented in the previous section.

Algorithm 1: Assign requests to the depot vehicles.

input: New requests: direct travel times ($DTT^t_{i,j}$), direct travel distances ($DTD_{i,j}$) set of vehicles (M), set of points (P, D, A), time windows (EP_i, LD_i), number of seats demanded (d_i), number of sharing (n_i^{share}), maximum detours (SQ), vehicle capacity (Cap), weights of objective function (α, β, γ, δ)**output:**Vehicle schedules

```
while Not all the points in A are assigned do
    Create the new car m ∈ M;
    Create initial routes set S from remaining origins in origin set P (equation 21–23);
    while S is not empty do
        Find the optimized route s ∈ S (in terms of objective function (equation 7));
        Find the set of points SP that can be added to s (equation17);
        for sp ∈ SP do
            if sp is feasible for time constraints (equation 11–15) on s then
                Compute new vehicle capacity (equation 9,10);
                if sp is feasible for capacity, number of sharing, detour constraints (Eq. 8,19,20) on s then
                    Create new route ns by adding the point sp to the route s;
                    Add route ns to the routes set S;
        if All sp ∈ SP are non-feasible in route s then
            if number of route origins = number of route destinations (equation 16,18) then
                Put route s in the results set Result;
            else
                Remove route s from routes set S;
    Find the optimized route optimal − route ∈ Result;
    Assign the optimal − route to the car m;
    Remove pickup points on optimal − route from P;
    Remove m from M;
    Add m to en-route vehicles set eM;
```

Algorithm 2: Assign requests to the en-route vehicles.

input: New requests**output:**Vehicles re-schedules

```
for origin p ∈ P do
    for c − schedule, the schedule of car m ∈ eM do
        if Detour is possible from any of the remaining origins on c − schedule then
            Build the re − schedule by adding the p after origin;
            if p is feasible for time window, capacity and number of sharing constraints on c − schedule then
                if d the destination of p is feasible for time window on c − schedule then
                    Create new schedule n − schedule by adding p and d to c − schedule;
                    Put n − schedule to the Result set;
    Find the optimized route optimal − schedule∈ Result;
    Re-assign the optimal − schedule to the car m;
    Remove p from P;
```

The second part of the algorithm to assign requests to the en-route vehicles is shown in Algorithm 2 . First, the algorithm checks the possibility of adding the origin point of the request to the vehicle schedule. It must check the capacity of the car, the number of sharing for all the on-board passengers after adding the new origin as well as the time window for all the stop points. If the vehicle route remains feasible after adding the new origin, the algorithm checks the possibility of adding the related destination point. In this step, it must check the time window for all the points after adding the new origin and destination points. Then, the algorithm puts all the feasible vehicles for the new request in the *Result* set. Finally, it chooses the vehicle that has a minimum increase in the objective function after adding the new request and sends the re-scheduled route to the car.

Fig. 4 shows an example of the dynamic ride-sharing algorithm when the number of sharing is one. In the first part, when the optimizer receives the requests, algorithm 2 searches to put new trips in the en-route vehicle schedule and then in the second part, algorithm 1 starts to work and assign the optimal routes to the vehicles waiting in depots. For example,

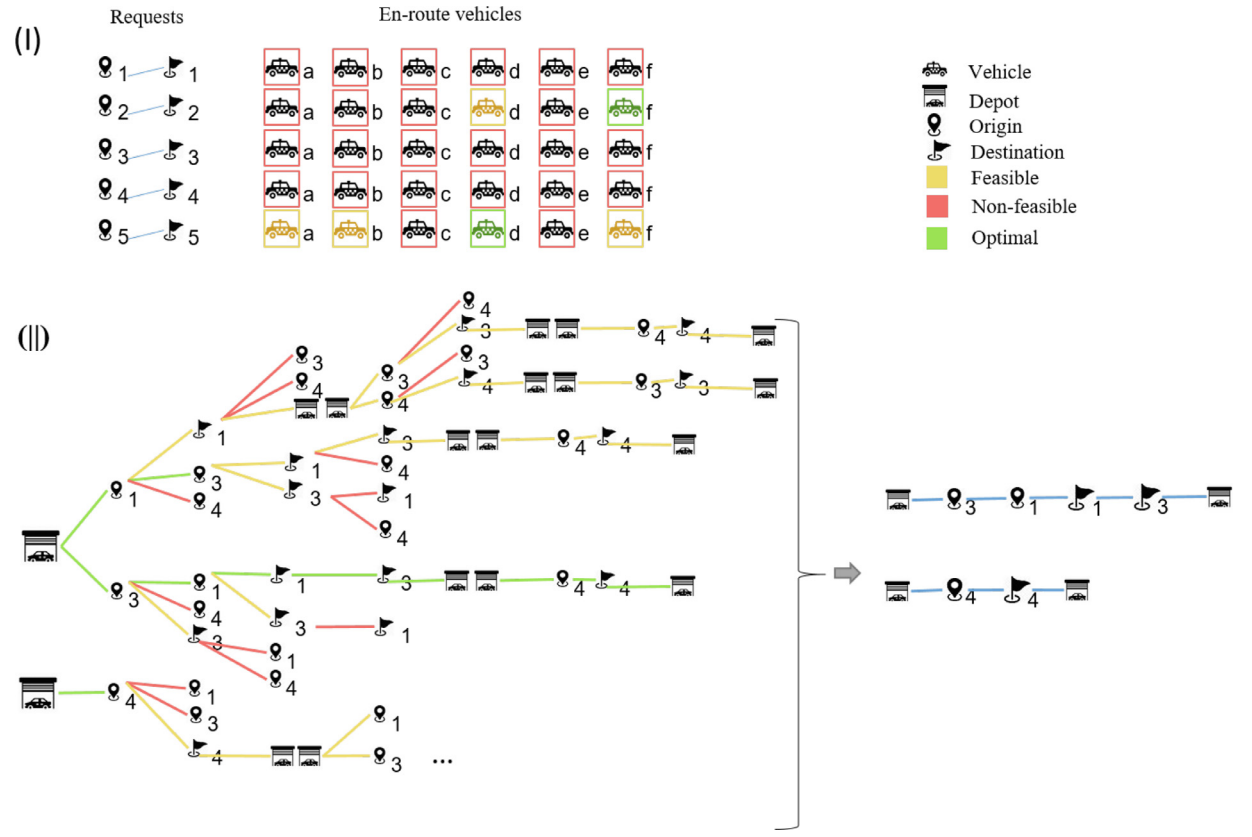
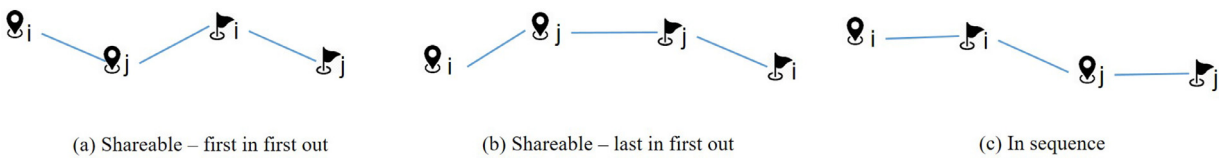


Fig. 4. Dynamic assignment.



(a) Shareable – first in first out (b) Shareable – last in first out (c) In sequence

Fig. 5. Trip situations.

in the Fig. 4, the optimizer has received five new requests. For request number 2, the re-scheduling algorithm finds vehicles d and f which can serve this request, satisfying all the constraints and, finally, vehicle f has a smaller increase in objective function after adding request 2 to its schedule. Likewise, for request 5 and vehicle d. Then for the remaining not-assigned trips, the main algorithm finds the best routes and assigns them to the vehicles in the nearest depots. The optimal solution serves requests 3 and 1 in the same car and request number 4 individually.

3.3. Heuristic 2: Heuristic 1 + clustering method

The algorithm proposed creates branches of origin and destination points as mentioned before. Thus, the computation time increases exponentially as the number of requests increases. Restricting the exploration of the feasible area to the branches that are more likely to create the optimal assignment can narrow the search of feasible solutions. To overcome this limitation, we define a clustering method to make clusters of the requests which are more likely to be shared. Then the algorithm is executed within each cluster independently.

To perform the clustering on the requests received by the system over each rolling horizon, we define the "Shareability Index" ($SF_{i,j}$) between request i and request j ($\forall i, j \in N$). We compute $SF_{i,j}$ for each pair of trips, and the function value is the difference between the travel time when the two trips are shared and the travel time to serve each trip individually.

Three situations exist for every two trips (Fig. 5).

In Fig. 5(a), two trips can be shared, and the first passenger drop off is before that of the second passenger. So, the travel time for the first passenger is the summation of their waiting time, the travel time between the first origin and the second origin and the travel time between the second origin and the first destination. Also, the travel time for the second passenger

is the summation of their waiting time, the travel time between the second origin and the first destination and the travel time between the first destination and the second destination. In situation (b), the vehicle serves the second passenger while the first passenger is on board. Thus, the travel time for the second passenger is the same as when served individually and the travel time for the first passenger is the travel time of all the links from the first stop point to the last one. There is a third situation when the trips are not shared, but the vehicle can serve two passengers sequentially Alisoltani et al. (2020). This situation must be considered in the shareability index in order to put these trips in the same group while solving the optimization problem. The travel time for the passengers is the same as when served individually. But in this situation, the vehicle travel time can decrease if the travel time between the first destination and the second origin is less than the summation of the travel time between the first origin and the closest depot and the travel time between the start depot and the second origin. This means that the *SF* here is the difference between the passenger waiting times when the trips are in sequence and the passenger waiting times when the trips are individual.

The following equations show how we compute $SF_{i,j}$ for each pair of trips, considering the three situations (i, j show the origin and the destination of the trip (r_n)):

$$\begin{aligned} &(a): TT_i + TT_j = WT_i + DTT_{org_i,org_j} + WT_j + DTT_{org_j,des_i} + DTT_{des_i,des_j} \quad \forall i, j \in N \\ &SF^a_{i,j} = TT_i + TT_j - (DTT_{org_i,des_i} + DTT_{org_j,des_j} + WT_i^{'} + WT_j^{'}) \end{aligned} \tag{26}$$

$$\begin{aligned} &(b): TT_i + TT_j = WT_i + DTT_{org_i,org_j} + WT_j + DTT_{org_j,des_j} + DTT_{des_j,des_i} \quad \forall i, j \in N \\ &SF^b_{i,j} = TT_i + TT_j - (DTT_{org_i,des_i} + DTT_{org_j,des_j} + WT_i^{'} + WT_j^{'}) \end{aligned} \tag{27}$$

$$\begin{aligned} &(c): TT_i + TT_j = WT_i + DTT_{org_i,des_i} + WT_j + DTT_{org_j,des_j} \quad \forall i, j \in N \\ &SF^c_{i,j} = TT_i + TT_j - (DTT_{org_i,des_i} + DTT_{org_j,des_j} + WT_i^{'} + WT_j^{'}) \\ &SF^c_{i,j} = WT_i + WT_j - (WT_i^{'} + WT_j^{'}) \end{aligned} \tag{28}$$

Finally, the *SF* value for each pair of passengers is the minimum value among three different situations. It means that the algorithm chooses the condition that the additional travel time is minimum for sharing each pair of trips.

$$SF_{i,j} = minimum\{SF^a_{i,j}, SF^b_{i,j}, SF^c_{i,j}\} \tag{29}$$

Afterwards, we have the shareability function for each pair of requests and create the shareability matrix. The shareability matrix is a kind of similarity matrix for the requests received that can be used in the clustering process.

We can use both partitioned clustering algorithms and hierarchical clustering methods to cluster the trips based on *SF*. We use the multidimensional scaling method to convert the similarity matrix into a distance matrix which makes it possible to apply the appropriate clustering method (Wang and Boyer, 2013). After extracting the distance matrix, we use the modified k-mean clustering method to create the same size clusters for the data received at every assignment time step. The modified k-means algorithm can be used to obtain clusters in preferred sizes (Ganganath et al., 2014). Accordingly, we can find the best trade-off between cluster size and computation time, considering the objective function value. We favor a uniform distribution of requests among clusters to decrease computation times and facilitate parallel computations of each sub-problem.

3.4. Heuristic 3: Heuristic 2 + force the sharing method (FOSH method)

The optimizer works to minimize the objective function, which combines both passengers and operators objectives. Therefore, the algorithm may choose a branch which has less sharing, compared with other feasible branches in the tree built by the algorithm. We propose the third heuristic method to force the algorithm to favor the longest possible route, which is in favor of more sharing. When we increase the number of passengers assigned to a vehicle, the passengers waiting time and travel time increase, so we reduce the length of the trip time window to keep the passenger's objectives acceptable. The algorithm finds the longest possible routes, and then it chooses the path with the minimum objective. Algorithm 3 shows the modification in algorithm 1 to force the sharing. Thus, the command in line 10 of the algorithm is

Algorithm 3: FOSH method.

```
max − route − size = 0;
for feasible solution route ∈ Result do
    if number of stops on route > max − route − size then
        max − route − size = number of stops on route;
for feasible solution route ∈ Result do
    if number of stops on route = max − route − size then
        Put the route in re − Result;
Find the optimized route optimal − route ∈ Result;
```

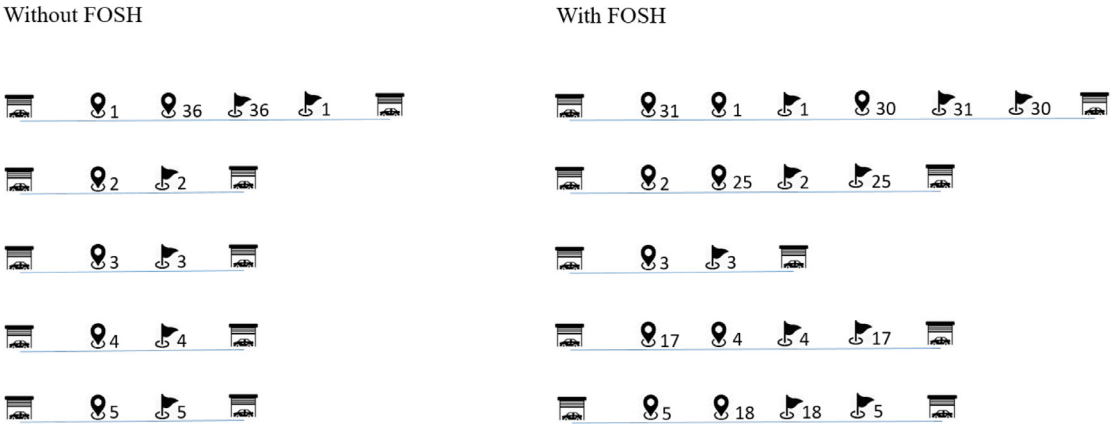


Fig. 6. The assigned routes for requests 1 to 5 with and without the FOSH method.

modified, as shown below:

When the algorithm assigns more trips to a vehicle, the number of all the trips decreases. Thus, with the FOSH method, we expect to use fewer trips and consequently fewer service vehicles. Fig. 6 shows the routes for requests 1 to 5 with and without the FOSH method when the number of sharing is 1. Without the FOSH method, five vehicles can serve just six passengers, but with the FOSH method, these five vehicles can serve ten passengers.

With the FOSH method, the algorithm can build fewer branches at each step. Because the number of trips on a single route is higher, it needs fewer branches to assign all the requests to the vehicles. Therefore, this method can also decrease computation time.

4. Experiments

In this section, we study the accuracy of the solution method steps regarding computation time and the size of the problem that can be addressed within each step.

4.1. Case study

In this study, the goal is to assess dynamic ride-sharing systems' performance in reducing congestion in both medium and large-scale cities. So, we implement the method on two networks. First, to assess the service in small and medium scales, we apply our method to a realistic O-D trip matrix for Lyon's northern half in France. Then, to assess the impact of ride-sharing on large trips set, we apply the method to the whole Lyon city network in France.

4.1.1. Medium-scale

In the research proposed, for the medium-scale, we apply our method to a realistic O-D trip matrix for the northern half of the city of Lyon in France (Lyon 6 + Villeurbanne). The network is loaded with travelers of all ODs with a given departure time to represent the morning peak hour (4 hours from 6:30 AM to 10:30 AM), based on the study of Krug et al. (2017). This network has 1883 nodes and 3383 links. The area is shown in Fig. 7. The origins set contains 94 points, the destinations set includes 227 points and the local depots (stop locations) set contains 237 points on the network. The number of trips during this period is 62,450. Some trips start from or end outside the network.

Only trips wholly inside the network can be assigned to the service depending on the market-share. Market-share is the percentage of the trips that will be served with the service vehicles. This corresponds to 11,235 trips and defines the maximal dimension of our optimization problem when the market-share is equal to 100%.

Table 2 shows the configuration of the simulations. The computations are carried out on a desktop with two Intel Xeon core E5-2620 processors, 64 GB RAM and the Windows 10 operating system running C++ Visual Studio 2013.

The time window for each trip is a fraction of the trip length. It is equal to 6 minutes plus one minute for each kilometer to be traveled.

Two kinds of depot are defined: local depots (237 depots) and the central depot. There are 237 depots for the service vehicles in this network. The central depot in the network can feed all the depots. Thus, there is no limitation on the fleet size. On the one hand, distributing vehicles over the depots will decrease the waiting time for passengers. However, on the other hand, in the peak hour, if many vehicles are circulating in the network, the congestion will increase, and it leads to more travel time for vehicles and passengers. We analyze the number of vehicles in depots over the network to decide about the best distribution for the vehicles. To locate the cars at the beginning of the simulations, we use the historical data for the network demand to estimate the demand distribution over the network. Then we specify the number of cars at

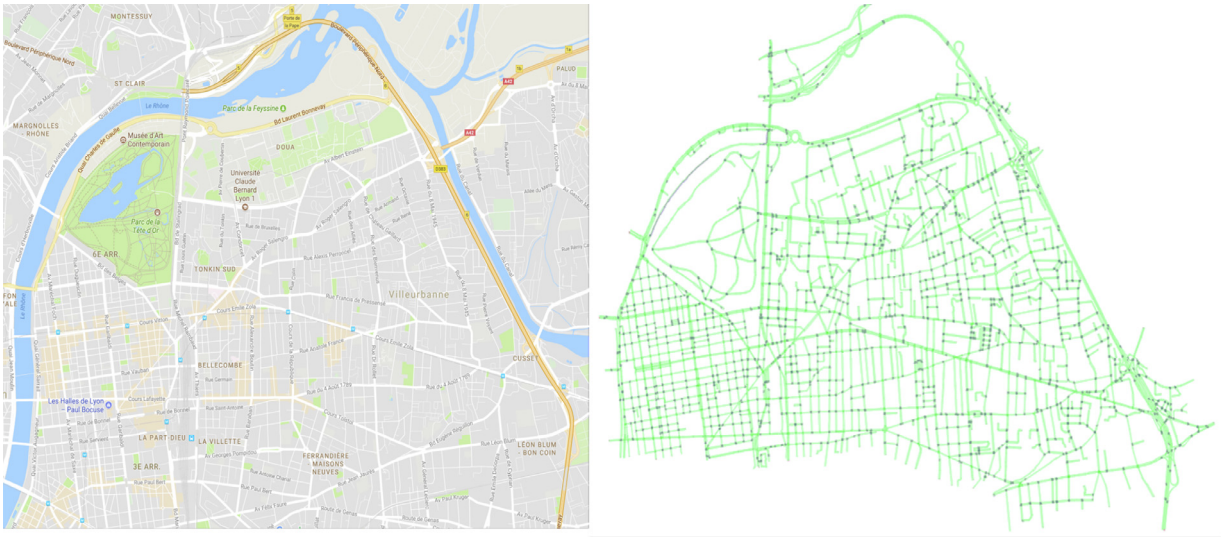


Fig. 7. Lyon 6e + Villeurbanne: Mapping data ©Google 2018 and the traffic network.

Table 2
Simulations configuration.

Parameters	Values
Rolling horizon	20 min
Optimization time step	10 min
Simulation time step	1 s
Fixed time window length	6 min
Number of sharing	0, 1, 2
Market-share	1 to 100%
Car capacity	4
Number of cars in local depots	142
Number of cars in central depot	1000
Service time for each passenger	1 min

each location based on the demand for the depot. So if the demand is high, we consider more cars on the depot, and if the demand is low, we put fewer vehicles at that location. Thus, considering the demand distribution, at the beginning of the simulations, we feed 14 depots with two vehicles, 114 depots with one vehicle, and put 14 empty depots. The central depot can feed each depot when the demand is high. Also, the system sends back the vehicles to the central depot when there exceed the required number of vehicles for this depot.

We consider a 1-minute service time for each passenger to get in and out of the service vehicle, which is computed in the total travel time.

Fig. 8 shows the temporal pattern of the demand for the private and service vehicles. We have 51,215 personal trips in the network. Besides that we have 11,235 demand for the service cars in the system. Based on the market-share, we select uniformly a part of this demand to be served with the service vehicles. Then we serve the rest of the trips with personal cars in the simulations.

One of the goals of this research is to figure out the performance of a dynamic ride-sharing system under the optimal situation. In this case study, we fully monitor traffic dynamics as we assess both service and personal trips in the network. In the rolling horizon approach, to find the near-optimal matching for the service vehicles in the dynamic traffic conditions, we choose a longer time step comparing with other dynamic methods to guarantee that we can find near-optimal solutions for the matching problem. Hence, to solve the problem dynamically, we apply the method every 10 minutes over the requests received in the next 20 minutes, considering we have a perfect knowledge of all requests over such a time horizon.

We use the trip-based MFD as the dynamic simulator and the predicted speed at the beginning of each horizon as the prediction model, which can be calibrated, to do the optimization. To predict the mean speed over the next time horizon $[t + TH]$, we use the current mean speed at time t weighted by a proportional function, which is different for the network loading and unloading phases. This function accounts for the speed decrease that usually happens during the network loading (an increase of congestion) and the speed decrease during the unloading. Fig. 9 shows the mean speed evaluation for a simulation with market-share = 100%. The results of the prediction with the current speed only show a lag between the prediction and actual speed. The introduction of the correlation factor provides accurate predictions in Fig. 10. The proportional function has been set to $\epsilon_l = 0.995$ (during the loading) and $\epsilon_u = 1.01$ (during he unloading).

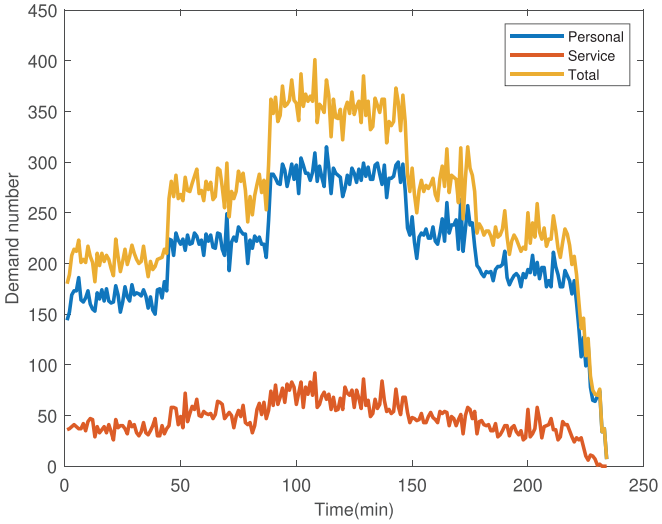


Fig. 8. Temporal demand pattern (When the market-share is 100%).

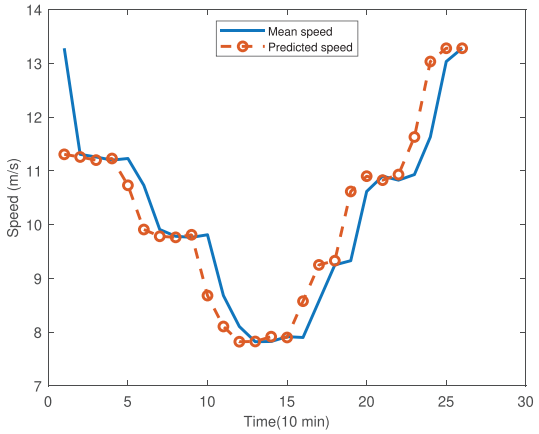


Fig. 9. Predicted speed and mean speed during the simulation horizon (before ϵ).

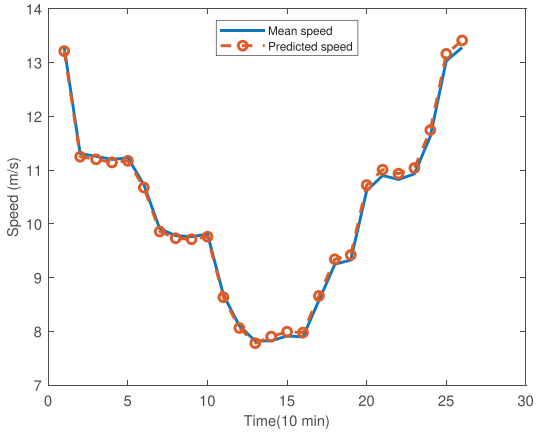


Fig. 10. Predicted speed and mean speed (after ϵ).

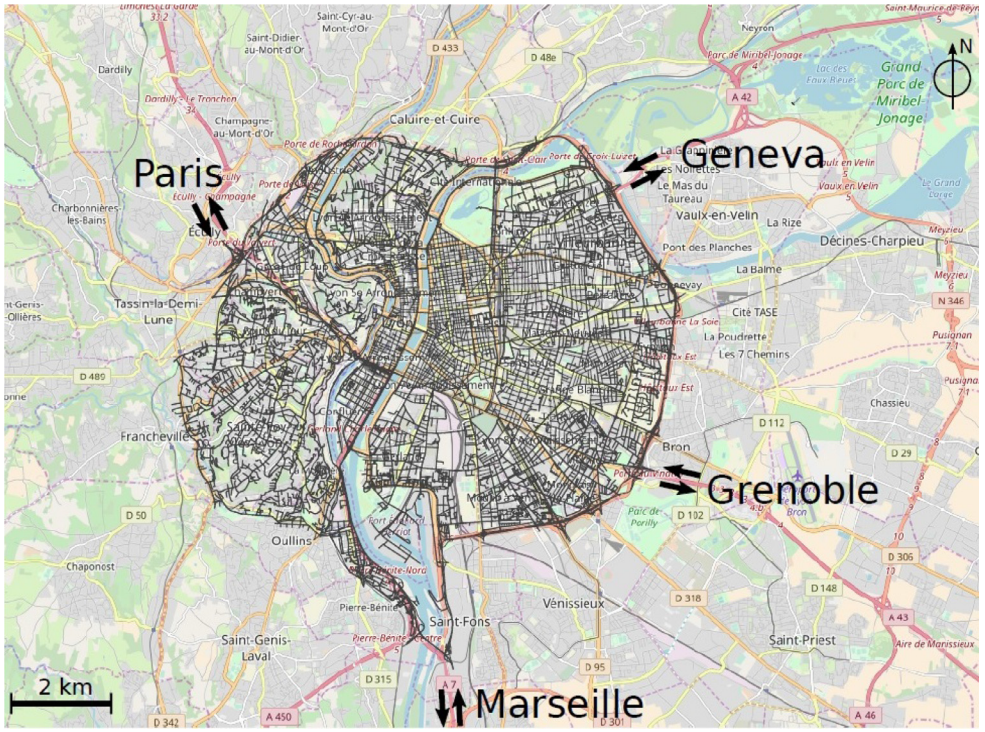


Fig. 11. Lyon city in France.

We assume that personal trips start at the origin precisely at the departure time without any waiting time and end at the destination. The travel time for the travelers of personal vehicles is equal to the vehicle travel time.

4.1.2. Large-scale

To assess the impact of dynamic ride-sharing on a large-scale trip set, we apply the proposed method on a realistic demand pattern for the full Lyon city. Each trip includes origin and destination, departure time and trip length.

Lyon is the second-largest urban area of France with an area of 80 km^2. The city is located in the south of Paris and is close to other mega-cities in France and Switzerland (Marseille, Grenoble, and Geneva). Fig. 11 shows the network of the city and its geographical position.

The trip set contains 11,314 different origin and destination locations. The local depots (stop locations) set contains 2272 points including 9 central depots on the network.

Fig. 12 shows the time evolution of the number of trips considering all OD pairs. This gives an overview of the demand dynamics, which is typical for a peak period from 6 AM to 10 AM. The number of trips during this period is 484,690. Among these trips, 279,382 trips have an origin or a destination outside the area and will be labeled as not-available for the mobility service. The service vehicles will then accommodate some trips among the 205,308 remaining ones depending on the market-share.

The proportional function has been set to $\epsilon_l = 0.993$ and $\epsilon_u = 1.020$ for Lyon network (Fig. 13 and 14).

In the network of Lyon city, we have defined nine central depots that are uniformly located in the network. The number of allowed stop locations is 2272 points on the network. As the main purpose of this research is to assess the impact of ride-sharing on reducing congestion, we put no limitation on the number of service vehicles, and the central depots can always feed the local depots.

4.2. Optimal system performance

Our first test case focuses on the exact global solution. It can be obtained only if the number of requests is low and no heuristics, including the rolling horizon, are running. The results are provided for a market-share of 4%, i.e. a total of 430 trips with the number of sharing 0, 1, 2 and 3. See Table 3. N_{trips} is the number of trips to serve all the requests, m is the number of vehicles used, $\sum_{k\in M} T_k$ and $\sum_{k\in M} TD_k$ are total travel time and distance for service vehicles, n is the number of passengers and $\sum_{i\in N} TT_i$ and $\sum_{i\in N} WT_i$ are total travel time and waiting time for passengers.

When the number of sharing is 0, each service vehicle serves only one passenger at a time, and the system does not share any ride. Then, with the number of sharing 1, the system is allowed to share a passenger ride with only one other passenger at the same time, as in e.g. Hyland and Mahmassani (2018), and in continuation with the number of sharing 2,

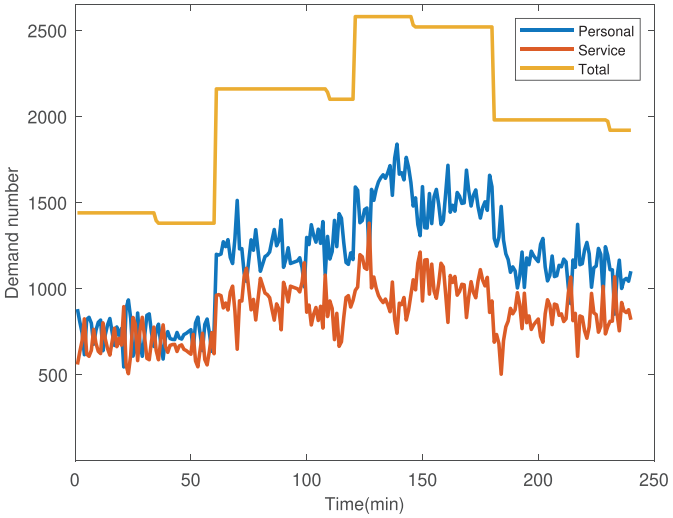


Fig. 12. Temporal demand pattern in Lyon (large-scale).

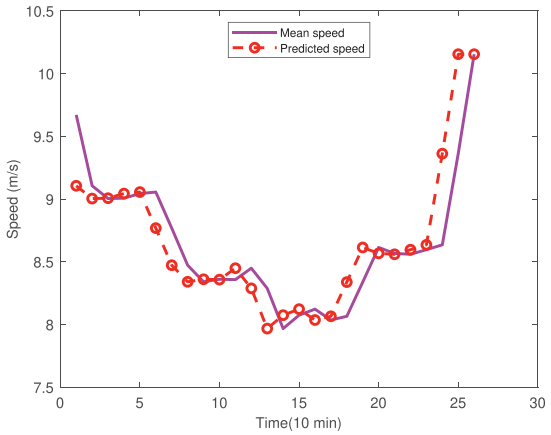


Fig. 13. Predicted speed and mean speed during the simulation horizon (before ϵ) in large-scale.

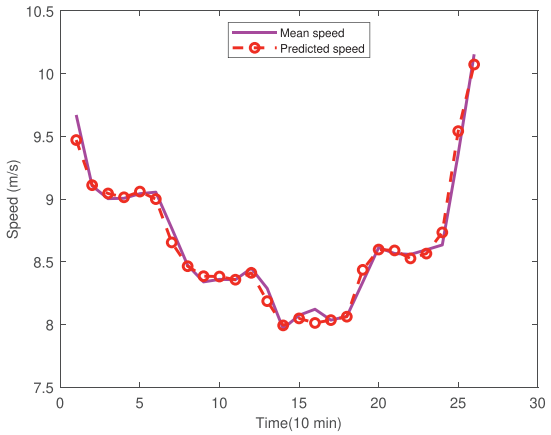


Fig. 14. Predicted speed and mean speed (after ϵ) in large-scale.

Table 3
Simulation results for optimal assignment.

Configuration	Shared vehicles				Passengers			Simulation time(h)
	N_{trips}	m	$\sum_{k\in M} T_k(h)$	$\sum_{k\in M} TD_k(km)$	n	$\sum_{i\in N} TT_i(h)$	$\sum_{i\in N} WT_i(h)$	
MS : 4%								
nshare = 0	429	32	46.36	1509.3	430	39.01	2.50	0.33
nshare = 1	419	32	46.09	1499.7	430	39.18	2.69	35.81
nshare = 2	417	32	46.08	1494.9	430	39.36	2.70	224.50
nshare = 3	416	32	46.08	1494.3	430	39.60	3.07	505.11

Table 4
Solution methods comparison.

Method	Number of requests	Objective function (normalized value)	Computation time (s)
CPLEX			
	4	0.54	0.40
	5	0.65	5.30
	6	0.74	43.50
	7	1.14	287.60
Exact solution Method			
	4	0.54	0.02
	5	0.65	0.05
	6	0.74	0.08
	7	1.14	0.19

three passengers can be in the same vehicle at the same time. With the number of sharing 3, the system uses all the vehicle capacity to serve the passengers.

When the number of sharing is 0, each vehicle serves just one passenger at a time. The algorithm also considers the trips that can be in sequence. When the travel time from the first destination to the second origin is shorter than the travel time between the first destination and the closest depot to this point, the algorithm puts these two trips in sequence. So, for example, in the first step when the algorithm is building the tree with branches, the number of branches does not go further than the number of requests. The results show that it takes just 0.33 hour to simulate our ride-sharing system in the morning peak hour. Then, when we increase the number of sharing to 1, we reduce the constraint on the number of passengers in the vehicle at the same time. Thus, the exploration space is expanded, and the algorithm can extract more branches at each step. The simulation time when we have an optimal system with the number of sharing 1 is 35.81 hours. When the number of sharing is 2, the algorithm can add any permutation of the other two trips to the first trips. Thus, the number of branches increases exponentially and, as can be seen the computation time for the number of sharing 2 is much longer than the computation time for the number of sharing 0 and 1. It takes almost 224 hours to simulate the system function with the number of sharing 2.

When the number of sharing is 3, the total travel distance is reduced by 600 meters while the total waiting time is increased by 22 minutes. Also, the computation time is 505 hours for the number of sharing 3.

To show the quality of our exact solution method, we have compared the computation time with a CPLEX solver for the same problem. Table 4 shows the computation time for different number of requests. As the problem is NP-hard, it is very expensive in terms of time to compute the exact solution with CPLEX for more than 7 requests. Only for 4 requests, the computation time for our presented algorithm is 26 times better than CPLEX.

Braekers and Kovacs (2016) propose an exact Branch-and-Cut algorithm for similar problem that can outperform the state-of-the-art solver CPLEX. The computation time of their proposed algorithm with 40 requests is 578 seconds. Our presented algorithm can solve the problem with 40 requests in 230 seconds. Then they propose a lean heuristic algorithm based on Large Neighborhood Search (LNS) to find near-optimal solutions. In another study on pick up and delivery problem for ride-sharing by Wang et al. (2018), the exact solution for the ride-sharing problem takes less than 8 seconds for 9 requests. Our method can find the exact solution for 9 requests in less than a second.

So our solution method outperforms some prevailing solvers when determining the exact solution for small instances. It allows us to compute the exact solution for medium-size instances, eg, market–share = 4%. But we can not solve larger problems without introducing heuristics. We will assess the performance and accuracy of our heuristics in the next sections.

4.3. The size of clusters

The clustering method proposed narrows the search for feasible solutions in the algorithm and makes it fast enough to assign requests to the shared vehicles in a short time and respond to the passengers quickly. Clustering can significantly improve the computation time with a very small increase in the objective function. This is because we carefully define the

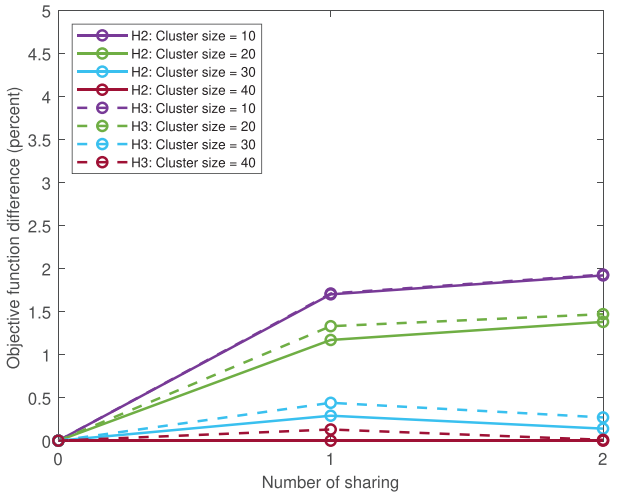


Fig. 15. Comparing different cluster sizes for the heuristic 2 and the heuristic 3 when market-share = 40%.

clusters by putting the shareable trips together. Therefore, the optimization is not deteriorated, even when the number of sharing is 1 and 2.

We expect a better objective function for bigger cluster sizes. However, increasing the size of clusters can increase the computation time. So we need to find the best trade-off between the quality of objective function and the computation time to find the appropriate size of clusters for heuristic 2 and 3.

To find the best size of cluster, we assess the objective function and the computation time for different sizes of cluster for heuristic 2 and 3 when the market-share is 40%. In this case we have enough number of requests at each time step to investigate different cluster sizes. We execute the simulations with cluster sizes of 10, 20, 30 and 40 to be sure that we have different number of clusters at each optimization time step and to be sure that the clustering method will be effective in terms of reducing the computation time.

When we privilege sharing in the algorithm in FOSH method, the assignment gives shorter travel times and distances for the vehicles and longer travel times and waiting times for the passengers. To keep the waiting time acceptable for passengers, we set the fixed time window length to 1 minute instead of 6 minutes in the simulations. As the time window for passenger pickup and drop off times is restricted, the decrease in the vehicle objective becomes dominant.

Fig. 15 shows the objective functions for heuristic 2 and 3 for different cluster sizes and numbers of sharing when the market-share is 40%. We expect a better objective function for bigger cluster sizes. Thus, here, we consider that the reference method is the clustering method (heuristic 2) with a cluster size of 40, and we compute the percentage increase of the objective function for other methods based on this reference method. The FOSH method (heuristic 3) increases the objective function by only 0.13% when the number of sharing is one and 0.01% when the number of sharing is two. The results are almost the same for the other cluster sizes. The objective function increases up to 2.5%. Also, the results show that cluster sizes of 30 and 40 can give very similar solutions. The difference between the cluster size of 30 and 40 is less than 0.5% for both heuristic 2 and heuristic 3.

Table 5 shows the computation time for different scenarios. FOSH method, with the cluster size of 30, can simulate the functioning of the system in less than 1 hour.

As the results show, a cluster size of 30 can give a reasonable trade-off between the computation time and the quality of the solution for this scale. Therefore, in the next experiments, we execute the simulations with a cluster size of 30 to ensure rapidity and also to keep the quality of the objective function at an adequate level.

4.4. Performance of heuristic methods

It is important to see the performance of the solution method comparing with the optimal situation when we increase the market-share.

Fig. 16 shows the percentage of difference for all the solution method steps compared to the optimal solution when the market-share is 10%, and the size of clusters is 30 and Table 6 shows the computation time. The number of requests, in this case, is 1092, and the computation time for the optimal solution is 80 hours when the number of sharing is 1. The figure shows that the final algorithm increases the objective function for 3.67% while the computation time decreases to 6 minutes (99.9% decrease) when the number of sharing is 1. The results prove the capability of the presented algorithm to provide fast and qualified solutions for the dynamic ride-sharing problem. The optimization time step is 10 minutes, so the rolling horizon method can find near-optimal solutions. Then in heuristic 2, the shareability index can effectively find the

Table 5
Simulation time for heuristic 2 and heuristic 3.

nshare	Simulation time (min)	
	H2	*H3*
cs = 10		
0	43.7	36.7
1	44.6	37.8
2	110.4	38.0
cs = 20		
0	43.7	43.8
1	56.4	46.1
2	598.5	49.7
cs = 30		
0	45.4	45.7
1	89.0	53.1
2	3091.3	94.3
cs = 40		
0	47.0	46.9
1	138.2	61.6
2	3333.3	238.2

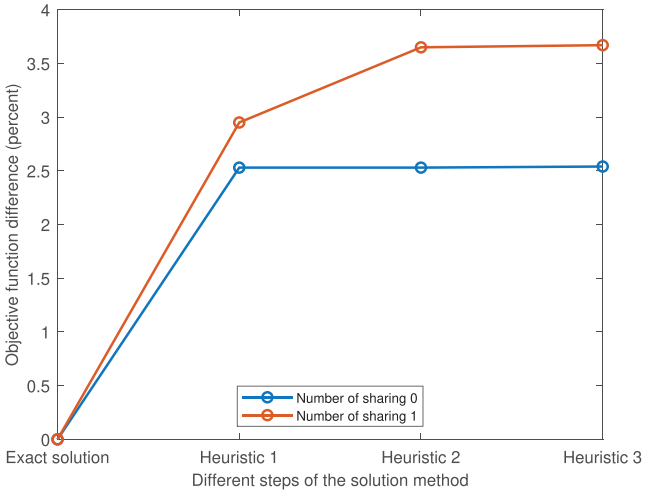


Fig. 16. Objective function difference for the solution method steps (market-share = 10%).

Table 6
Solution methods comparison (computation time for market-share = 10%).

Method	Simulation time (min)	
	nshare 0	nshare 1
Exact solution	54.1	4800.0
Heuristic 1	3.2	98.0
Heuristic 2	5.4	18.9
Heuristic 3	4.4	6.6

potential sharing opportunities and cluster the requests. Therefore, increasing the market-share will not affect the quality of the solution methods, and the results are very close to the optimal situation.

The heuristic method proposed here can also outperform the previous methods in the literature. In Braekers and Kovacs (2016) after an exact method, they propose a lean heuristic algorithm based on Large Neighborhood Search (LNS), to find near-optimal solutions. The lower bounds generated by their exact approaches are on average 3.68% better than the average LNS result and the average computation time is less than 90 s for instances with up to 40 requests, while large-scale instances with up to 100 requests are solved in about 10 min on average. Our heuristic method can solve the problem in less than 20 seconds for 40 requests and less than a minute for 112 requests while it increases the objective function only by 3.19% percent for 112 requests. For market-share = 10% (1092 requests) the heuristic method increases the objective func-

Table 7
Simulation results for heuristic methods (number of sharing = 1).

Configuration	Shared vehicles			Passengers	Personal vehicles	All vehicles
	N_{trips}	$\sum_{k\in M} T_k(h)$	$\sum_{k\in M} TD_k(km)$	$\sum_{i\in N} WT_i(h)$	Travel time (h)	Travel time (h)
MS : 0%					7978.4	7978.4
MS : 10%						
Exact solution	1015	117.6	3744.6	8.2	7904.1	8021.7
Heuristic 1	1041	119.5	3825.1	8.5	7907.5	8027.1
Heuristic 2	1067	120.5	3853.0	6.8	7906.5	8027.0
Heuristic 3	610	101.8	3279.4	20.2	7896.5	7998.3

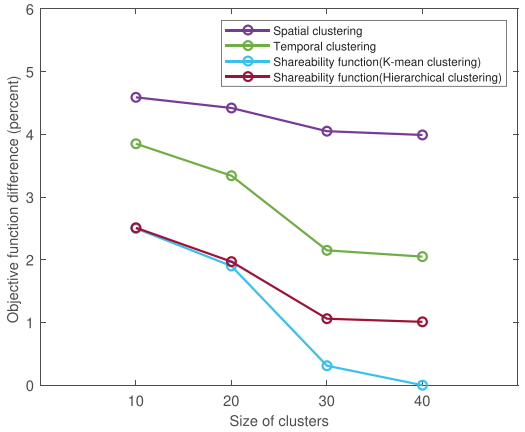


Fig. 17. Comparing clustering methods' objective function (market-share=50%).

tion for 3.67%. Wang et al. (2018) present a Tabu search heuristic for the pick and delivery problems for ride-sharing. They use a ratio (the objective value of the optimal solution divided by the output objective value of the heuristic method) to compare the heuristic method with optimal situation. For 9 requests, the ratio is 0.94 in average. The final heuristic method proposed in this study can find optimal solution for 9 requests. Increasing the number of requests keeps this ratio low. For market-share = 10%, this ratio is 0.96 in our method.

Table 7 shows the results for different solution methods when we serve 10% of the internal trips with service cars. The market-share = 0 is when only personal vehicles serve all the network demand. Using service vehicles increases the travel distance for the vehicles and consequently, the travel time. Serving 10% of the requests with service cars, increases the total travel time for vehicles by 43.3 hours in the optimal situation. The FOSH method reduces this value to 19.9 hours by favoring the sharing but it increases the passengers' waiting time from 8.2 hours in the optimal situation (27.0 seconds for each passenger on average) to 20.0 hours (66.6 seconds for each passenger on average).

Different clustering methods have been implemented in the literature for large-scale problems. They usually divide the space geographically and use a spatial clustering to downsize the problem. To show the quality of the proposed clustering method, we have compared the Shareability clustering with such a spatial clustering method.

For the spatial clustering, we put the two corresponding trips in the same cluster based on the distance between their origins. Also, we try to cluster the trips based on the time in a temporal clustering method. For the temporal clustering, we put two trips is the same cluster based on their departure time and their position. Finally, we compare these methods with our proposed method to show the quality of our proposition.

Figs. 17 and 18 show the comparison of different clustering methods considering the objective function and the computation time for four different cluster sizes when the market-share is 50%. The best objective function is provided by our k-means clustering method when the size of the cluster is 40. So we choose this objective function value as a base, and we compute the percentage of difference for other methods considering this basic scenario. As it is clear, the performance of spatial clustering is not acceptable compared to the other methods. In the best situation, the spatial clustering's objective function is 4% more than the k-means clustering. The temporal clustering can perform better than spatial clustering, but it can not outperform our clustering methods. With the cluster size of 40, the objective function for temporal clustering it is 2.02% more than k-means clustering. The computation time increases exponentially for the shareability clustering when the cluster size is 40. However, with the cluster size of 30, the algorithm can give a high-quality solution in a short time.

As explained in Section 4.3, we can do the transportation analysis for medium-scale with the k-means method when the cluster size is 30. For the large-scale network, we use hierarchical clustering method. Using the Sum of Squares method, we choose the cluster size of 125 for the large-scale.

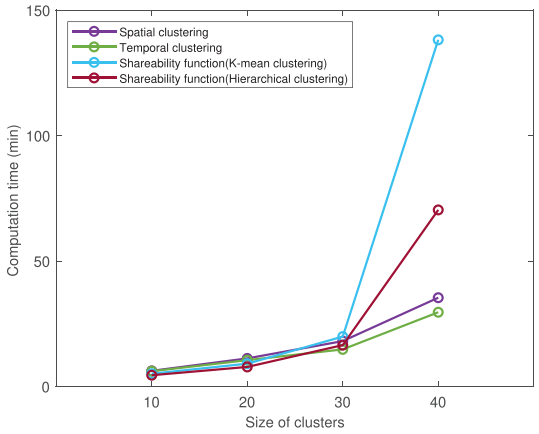


Fig. 18. Comparing clustering methods' computation time (market-share=50%).

Table 8
Simulations results for optimal assignment.

Method	Estimated objective function			Experienced objective function		
	nshare 0	nshare 1	nshare 2	nshare 0	nshare 1	nshare 2
Heuristic 1	2017	1980	1979	2063	2030	2029
Heuristic 2	2017	1994	1993	2063	2043	2039
Heuristic 3	2017	1994	1993	2063	2008	2007

4.5. Performance of simulation models

We use a trip-based MFD as the plant model to represent the real traffic situation. Mariotte et al. (2020); Mariotte and Leclercq (2019); Paipuri et al. (2019) prove that it is a fast and accurate framework to track congestion dynamics and mean speed at the network level. These studies prove the validity of MFD outputs vs. both micro-simulations results and real data.

Network congestion has impacts on the dynamic ride-sharing service. When the rides are executed, a gap can exist between the estimated travel times used by the optimization process at the beginning of the time horizon and the travel times experienced during the time horizon in the plant model. So, the objective function when solving the fleet allocation problem at the beginning of the time horizon can be different from the objective function experienced. Then, the current speed is updated based on the new traffic condition for the next step to take into account the impact of congestion and to minimize this gap.

Table 8 shows the estimated objective and the objective function values experienced (normalized values) for the different methods when the market-share is 10%, and the cluster size for heuristic 2 and 3 is 30. The objective function implemented is greater than the estimated objective function for all the methods because of the gap between the predicted and the real travel times. For example, when the number of sharing is 1, the estimated objective function for heuristic 1 is 1,980, but the objective function experienced is 2,030. The differences are small in all the scenarios showing that the prediction model is accurate enough.

5. Dynamic ride-sharing system performance in terms of traffic congestion

To assess the influence of the dynamic ride-sharing system on reducing traffic congestion, we compare the traffic condition for the dynamic ride-sharing system considering different market-shares and the numbers of sharing with the case where the market-share is zero, when only personal vehicles serve all the network demand (No service scenario in the figures).

5.1. System performance in medium-scale

Table 9 shows the simulation results for different market-shares and numbers of sharing. We discuss the results in detail in the next sections.

5.1.1. Influence of market-share

We use the vehicle accumulation in the network as a measure of traffic congestion. We compute the vehicle accumulation in the network every second in the simulations. Fig. 19 shows the vehicle accumulation in the network every 100 seconds for different market-shares when the number of sharing is 0. The service vehicles must pass a distance from the depot to

Table 9
Simulations results.

Configuration	Shared vehicles				Passengers			Personal vehicles	All vehicles	
	N_{trips}	m	$\sum_{k\in M} T_k(h)$	$\sum_{k\in M} TD_k(km)$	n	$\sum_{i\in N} TT_i(h)$	$\sum_{i\in N} WT_i(h)$	Travel time (h)	Total travel time (h)	Total travel distance (km)
MS : 0%	-	-	-	-	-	-	-	7978.4	7978.4	268481.0
MS : 20%										
nshare = 0	2235	158	251.7	7893.3	2236	204.0	7.5	7816.6	8068.3	269698.3
nshare = 1	1239	116	209.3	6676.8	2236	218.7	39.0	7797.1	8006.4	268481.8
MS : 40%										
nshare = 0	4482	307	501.7	15452.6	4482	415.7	9.9	7657.8	8159.5	270750.6
nshare = 1	2455	218	419.3	13235.8	4482	450.7	71.9	7603.5	8022.8	268533.8
MS : 60%										
nshare = 0	6731	422	754.7	23016.0	6732	632.2	11.1	7489.6	8244.4	271770.0
nshare = 1	3672	303	632.3	19779.8	6732	687.1	104.0	7417.9	8050.2	268533.8
MS : 80%										
nshare = 0	8963	515	1018.6	30867.6	8978	856.3	12.5	7308.3	8326.9	272887.6
nshare = 1	4880	404	852.4	26513.9	8978	937.5	133.5	7220.6	8073.0	268533.9
MS : 100%										
nshare = 0	11,213	578	1276.3	38434.0	11,235	1078.6	12.6	7140.6	8416.9	273891.0
nshare = 1	6072	477	1074.1	33099.4	11,235	1183.7	168.6	7032.6	8106.8	268556.4
nshare = 2	4993	474	1049.9	32605.2	11,235	1283.0	233.3	7021.4	8071.3	268062.2

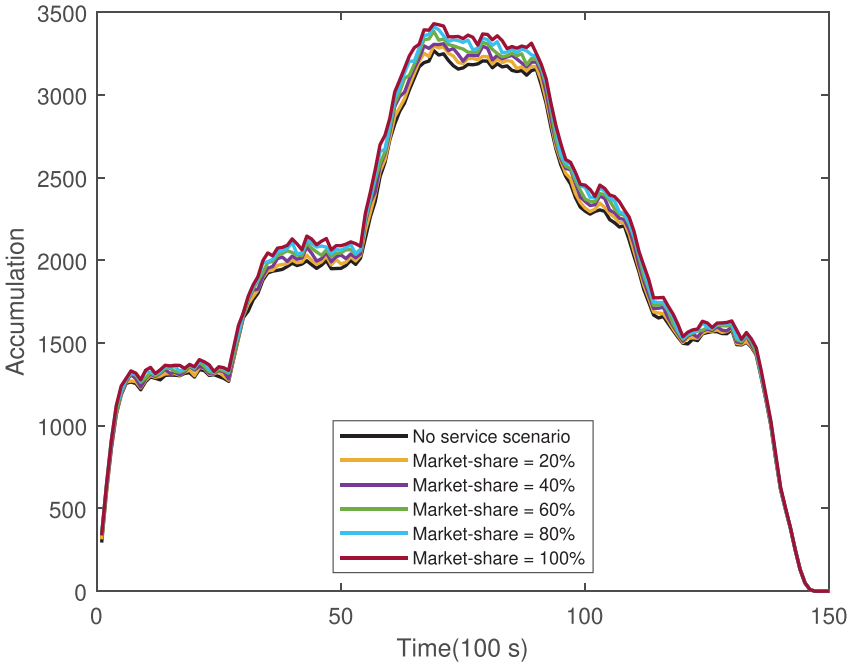


Fig. 19. Traffic situation for a number of sharing 0 with different market-shares.

the first origin and then from the last destination to the depot. This extra distance makes the car stay longer in the network and leads to more traffic. Hence, when the market-share increases, the accumulation of vehicles increases.

In Table 9, the travel time for the personal vehicles when the market-share is zero is 7,978.4 hours. Then, with a market-share of 20%, the total travel time for shared vehicles is 251.7 hours, and the total personal vehicle travel time is 7,816.6 hours. Therefore, the total travel time for all the vehicles in the network is 1.13% higher than the total travel time when there is no service vehicle in the network. Increasing the market-share will increase this extra travel time by 2.27%, 3.33%, 4.37% and 5.50% for market-shares of 40%, 60%, 80% and 100%.

As shown in Table 9, sharing decreases the travel distance and the travel time for service vehicles. Hence, the number of sharing 1 can reduce the accumulation of cars driving in the network. Fig. 20 shows the accumulation of all the vehicles in the network when the number of sharing is 1 for different market-shares. The results show that sharing can reduce traffic congestion for a given market-share. The total travel time for shared vehicles when the market-share is 80% is 1,018.6 hours,

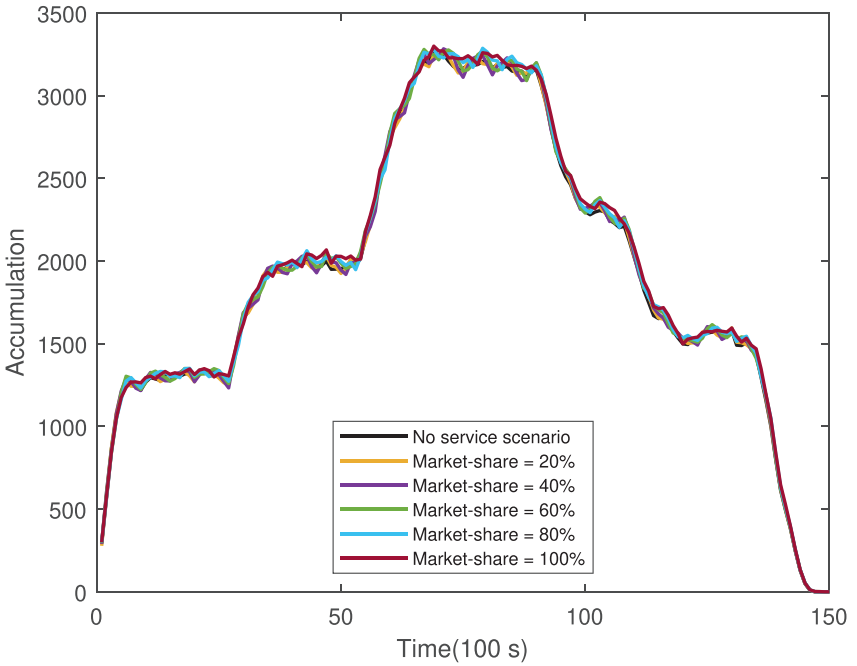


Fig. 20. Traffic situation for a number of sharing 1 with different market-shares.

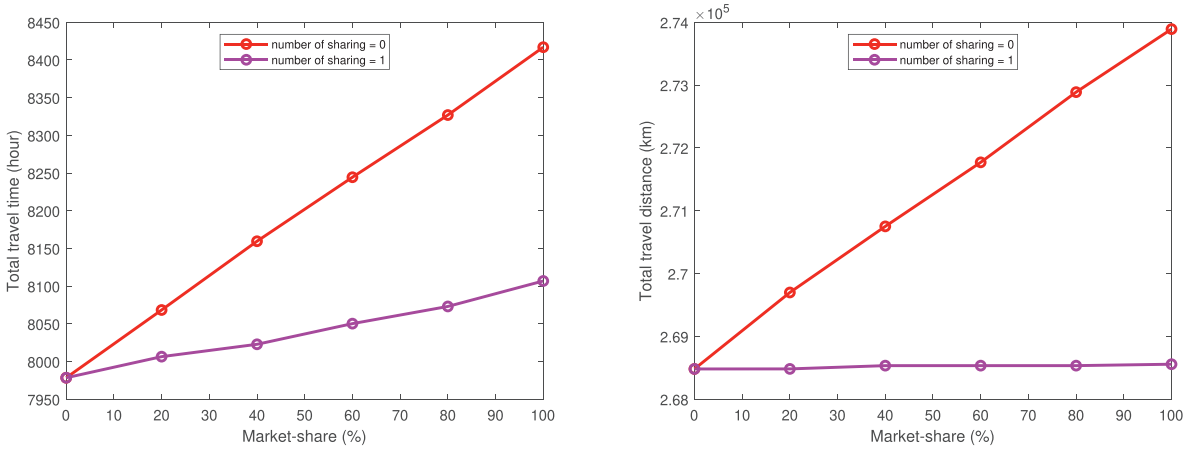


Fig. 21. Total travel time and distance for all the cars for the number of sharing 0 and 1 with different market-shares.

which is for 515 vehicles that make 8963 trips to serve 8978 requests when there is no sharing. But it falls to 852.4 hours with 404 vehicles via 4880 trips for the same number of requests when the number of sharing is 1.

However, sharing cannot improve the traffic situation significantly compared to the case when all the trips are made with personal cars. For example, for the market-share of 20 percent and the number of sharing 1, the total travel time for all the vehicles in the network is 8,006.4 hours, which is 0.77% better than the number of sharing 1 but still 0.35% worse than the no service scenario. The total travel time for all the vehicles in the network is 0.56%, 0.90%, 1.19% and 1.61% longer than the no service scenario for market-shares of 40%, 60%, 80%, and 100%.

Fig. 21 shows the total travel time and the total travel distance for all the vehicles in the network for the number of sharing 0 and 1 with different market-shares. It is clear that increasing the market-share increases the total travel time and distance for the number of sharing 0 when each passenger is served individually but then, when sharing the trip of just two passengers with the number of sharing 1, the slope of this increasing trend flattens considerably.

To consider the passengers' willingness to share the ride and their satisfaction, we optimize the passengers waiting time and travel time in addition to the vehicle objectives. Thus, increasing the market-share cannot increase the passengers' objectives so that this leads to their dissatisfaction. As we place a strict constraint on the passenger pickup and delivery time window, the average waiting time for each passenger is not more than 63 seconds. For the market-share of 20% the

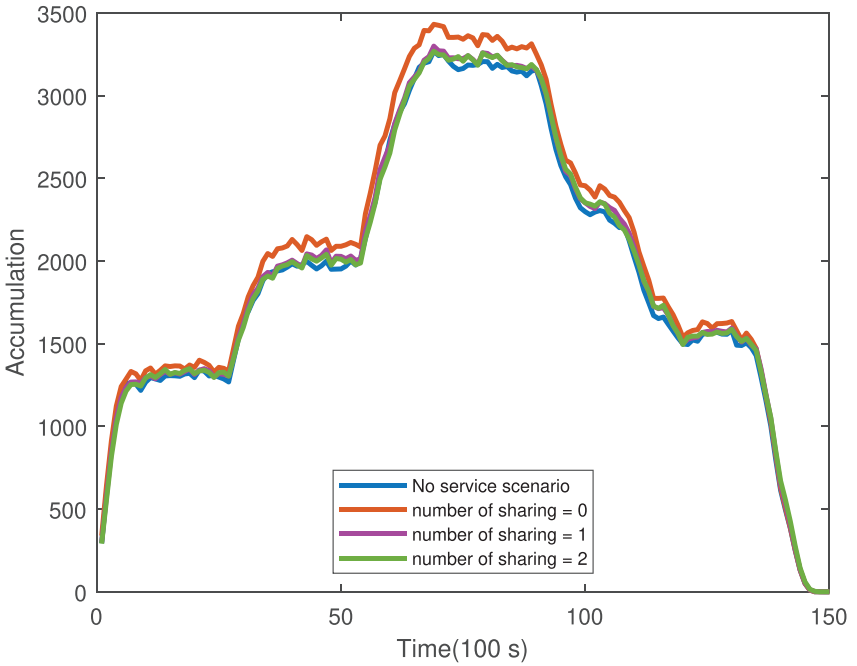


Fig. 22. Traffic situation for market-share 100% with different numbers of sharing.

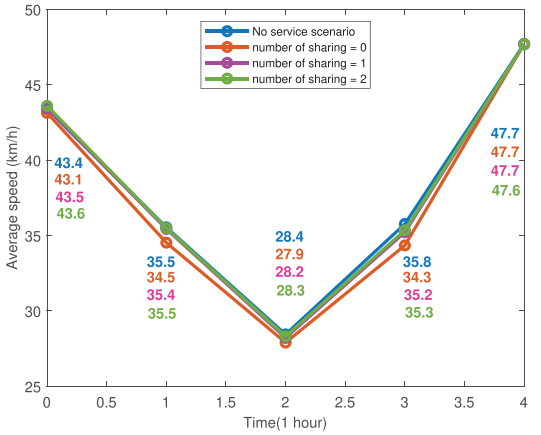


Fig. 23. Average vehicle speed.

average waiting time for each passenger is 62.8 seconds when the number of sharing is 1, and it is 54.0 seconds for the market-share of 100%. Also, the average travel time for passengers is 5.9 minutes when the market-share is 20% and the number of sharing is 1, and it increases by 24 seconds for the market-share of 100%.

5.1.2. Influence of the number of sharing

Increasing the number of sharing provides the system with greater leeway to decrease the travel distance by reducing the distance between stop points and depots. So, with more sharing, we expect a better traffic situation and fewer vehicles in the network.

Fig. 22 shows the network traffic when the market-share is 100% for the different numbers of sharing. The results show that increasing sharing can reduce congestion, but it still cannot compete with the no service scenario.

Fig. 23 shows the average speed every hour in the system for different numbers of sharing. The vehicle speed decreases using service cars without sharing. Sharing can increase the speed, but it still cannot be higher than the speed in the no service scenario. At the onset of congestion, with the number of sharing 2, the speed is 35.5 km/h which is 0.16% higher than the number of sharing 1.

Fig. 24 shows the accumulation differences with the baseline in peak hours. It is clear that sharing can improve congestion compared with the number of sharing 0 (systems like traditional taxis), but it is not better than the no service scenario.

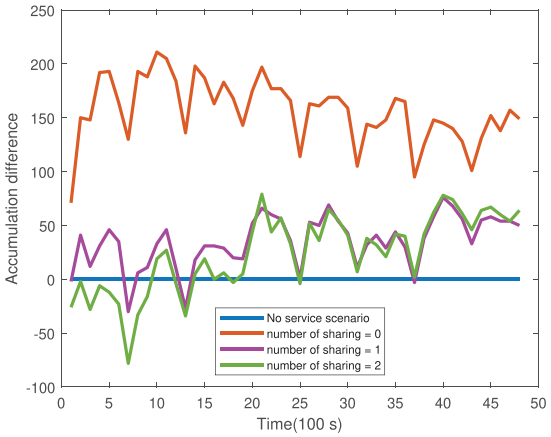


Fig. 24. Traffic situation at peak hours.

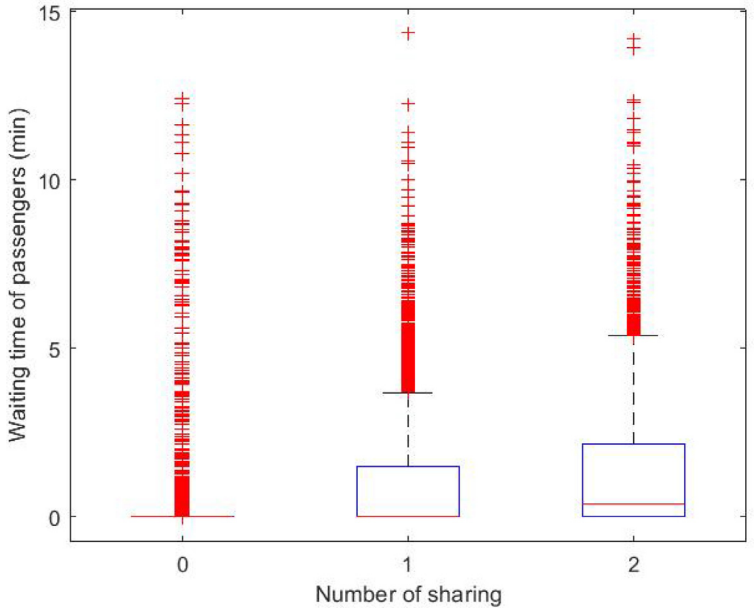


Fig. 25. Passengers' waiting time for different sharing scenarios when the market-share is 100%.

At the onset of congestion, sharing can prevail over the no service scenario, especially when the number of sharing is 2. Then, as congestion subsides, the sharing scenarios is better than the no service scenario. Finally, the number of sharing 2 is better than when the number of sharing is 1. This is because the vehicles have to travel a longer distance after the peak hour as they have more requests to serve.

Increasing the number of sharing will increase the passengers' travel time and waiting time. The average waiting time for the number of sharing at 0 (when there is no sharing) is 4 seconds. It increases to 54 seconds when the number of sharing is one and 74 seconds when the number of sharing is two. This increase is acceptable considering the last heuristic used to force the sharing. This means that the passenger must wait no more than 1 minute to be picked up at the origin when the number of sharing is 1. In the first heuristic, sometimes the algorithm can find a better assignment for the vehicles that are waiting in the depot, and it changes the first schedule. It can increase the passengers' waiting time. However, the waiting time will not be more than 10 minutes. Fig. 25 shows the variation of passengers' waiting time for different numbers of sharing when the market-share is 100%. For the number of sharing 0, the median for the waiting time is 0, and 50% of passengers depart at their defined pick up time. For the number of sharing one, the median of waiting time is still 0, and 50% of the passengers can start their trip at their desired time. The 75th percentile is 108 seconds, and the upper adjacent is 4 minutes. For the number of sharing 2, the median increases to 22 seconds, and the upper adjacent is 5 minutes.

In Section 4.4, we showed that the heuristic method gives an error of 3.67% comparing with the optimal solution. The total travel time for all the vehicles in the system is 7978.4 hours in "no service scenario". We can estimate the optimal

Table 10
Simulation results for different depot management scenarios.

Configuration	Shared vehicles				Passengers
	N_{trips}	m	$\sum_{k\in M} T_k(h)$	$\sum_{k\in M} TD_k(km)$	$\sum_{i\in N} WT_i(h)$
MS : 0%	11,213	578	1276.3	38434.0	4.03
MS : 100% nshare = 1					
Basic scenario	6072	477	1074.1	33099.4	54.04
237 depots and 1780 vehicles	6055	475	1068.2	32647.2	50.79
1067 depots and 1780 vehicles	6065	446	986.4	30118.0	52.01

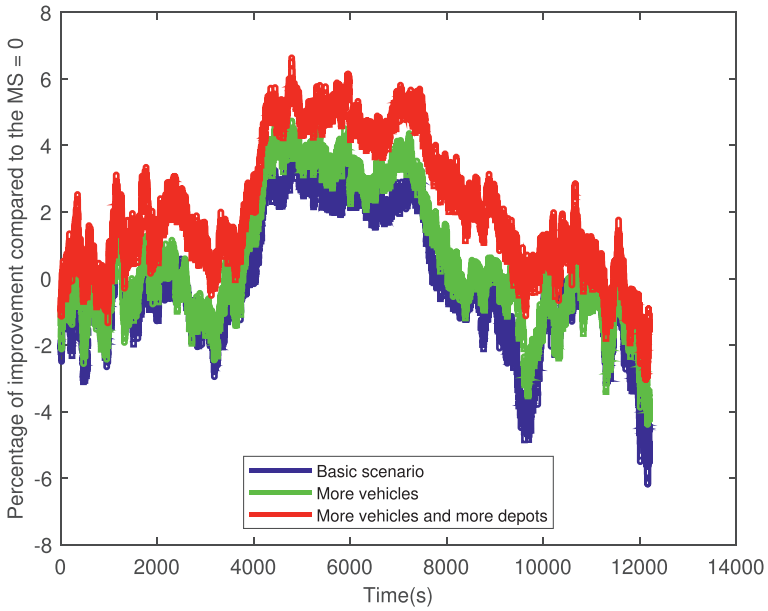


Fig. 26. Comparing different scenarios for depot management.

situation considering this error. If we reduce this error from the final results, the total travel time for the number of sharing one and market-share = 100% can be reduced to 8068.0, which is still more than the "no service scenario". The estimation of the heuristic error when the number of sharing is two is 4.66%. The total travel time for the vehicles when the number of sharing is 2 can be around 8022.4 hours in the optimal situation, and it is still more than "no service scenario".

5.1.3. Influence of the number of vehicles and local depots

Increasing the number of allowed stop locations in the network can increase the accessibility of cars to the closest passengers and reduce the passengers waiting time. It can also decrease the travel distance between the stop location and the first origin. To assess the impact of the number of local depots and the number of vehicles waiting in these locations, we define two different scenarios. We compare them with the basic scenario where we have 237 depots and 142 cars waiting in these depots. The first scenario increases the number of vehicles and puts 1780 vehicles in the stop locations. In the second scenario, we increase the number of depots to 1067 and distribute 1780 vehicles, considering the geographical demand pattern on these locations.

Table 10 shows the results for different scenarios. The second scenario can improve the total travel distance by 1.3% compared to the basic scenario, while the number of vehicles is increased by 1153%. In the third case, with a 350% increase in the number of allowed waiting locations for the vehicles, the total travel distance is improved by 9%. Increasing the number of vehicles in local depots can improve the total travel time by 16.3% compared to the no service scenario. Increasing the number of local depots in the third scenario can increase this improvement to 22.7%.

Fig. 26 shows the percentage of improvement in traffic conditions (accumulation of cars) for the three sharing scenarios compared to the scenario when we have just personal vehicles in the system. All the sharing scenarios decrease the congestion in peak hour but this improvement is not significant. In the best case, the basic scenario can improve the congestion by 4.1%. Increasing the number of vehicles can increase this improvement by 0.6% and increasing the number of stop locations by 5 times can make 2.5% improvement compared to the basic scenario.

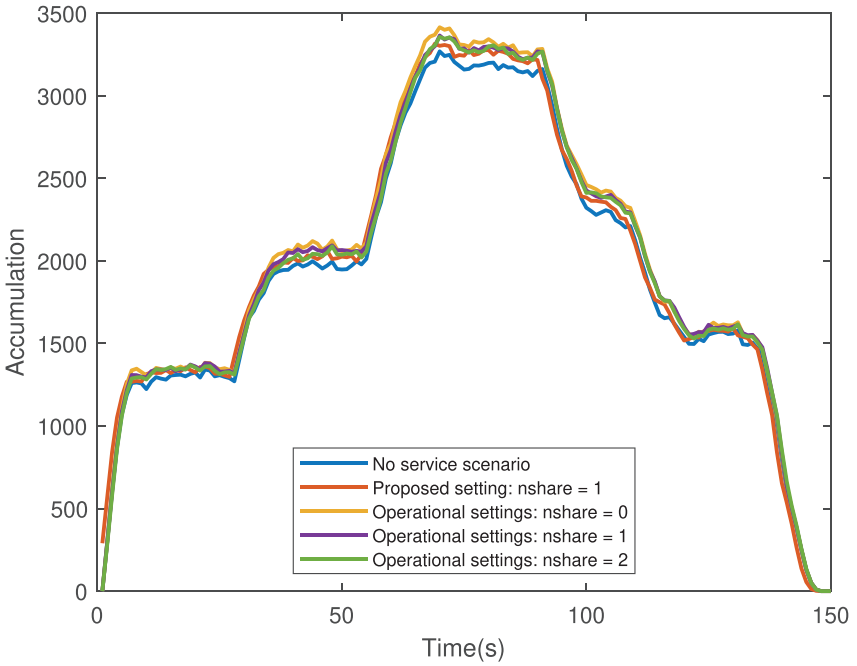


Fig. 27. Traffic situation for market-share 100% with operational settings.

Table 11
Simulation results for operational settings.

Configuration	Shared vehicles				Passengers	Personal vehicles
	N_{trips}	m	$\sum_{k\in M} T_k(h)$	$\sum_{k\in M} TD_k(km)$	$\sum_{i\in N} WT_i(h)$	Travel time (h)
MS : 100%						
nshare = 0	11,215	578	1277.9	39559.2	12.5	7141.6
nshare = 1	8243	510	1140.9	36875.9	206.4	7080.4
nshare = 2	7864	515	1140.6	36904.2	240.2	7076.3

5.1.4. Dynamic ride-sharing system performance under operational settings

The main goal of this study is to assess the performance of dynamic ride-sharing in terms of reducing congestion. To favor the ride-sharing service, we choose a longer time step comparing with other dynamic methods to guarantee that we can find near-optimal solutions for the matching problem. Hence, to solve the problem dynamically, we apply the method every 10 minutes over the requests received in the next 20 minutes, considering we have a perfect knowledge of all requests over such a time horizon. However, one can argue that today's real operational systems do not have this long horizon and that they answer service requests immediately. Hence, our conclusions would be valid only for the methods that we are using, i.e., with a rolling horizon.

To verify our conclusions in this context, we propose to mimic today's operational settings. To do so, we propose an insertion heuristic method that matches the passengers and vehicles in real-time. The chosen vehicle to insert the new request in its schedule is the one with the minimal marginal costs increase. If no existing vehicle can insert the request, a new vehicle is dynamically created, and the request is assigned to it. In this situation, the system has immediate response times and matches the received individual requests to the vehicles in real-time.

Fig. 27 shows the network traffic when the market-share is 100% for different numbers of sharing under operational settings comparing with the "no service scenario" and our proposed system when the number of sharing is 1. It is clear that the operational settings perform worse than the near-optimal situation proposed in this paper in terms of reducing congestion even with the number of sharing 2. Table 11 shows the results for operational system settings. The total travel time for service vehicles in the operational setting is 66.8 hours more than the proposed system when the number of sharing is 1. Also, the total travel time for personal vehicles in the network is 47.8 hours more in operational settings.

5.2. System performance in large-scale

In this section, we assess the impact of dynamic ride-sharing on traffic congestion for a large-scale trips set. This is much more challenging in terms of computational complexity for the optimization process. The Lyon network area is 220% larger

Table 12
Simulations results.

Configuration	Shared vehicles				Passengers			Personal vehicles	All vehicles	
	N_{trips}	m	$\sum_{k\in M} T_k(h)$	$\sum_{k\in M} TD_k(km)$	n	$\sum_{i\in N} TT_i(h)$	$\sum_{i\in N} WT_i(h)$	Travel time (h)	Total travel time (h)	Total travel distance (km)
MS : 0%	-	-	-	-	-	-	-	75271.1	75271.1	2290280.0
MS : 20%										
nshare = 0	41,011	3667	10373.2	312739.0	41,043	9439.9	265.7	65770.0	76143.2	2308249.0
nshare = 1	21,285	2089	6210.1	188979.0	41,043	9726.4	346.9	64850.3	71060.3	2184489.0
nshare = 2	14,809	1559	4579.2	139516.0	41,043	9894.5	498.4	64505.3	69084.5	2135026.0
MS : 40%										
nshare = 0	82,021	7261	20811.6	625745.0	82,104	19052.3	449.3	56176.4	76987.9	2325345.0
nshare = 1	42,374	4138	12377.0	380495.0	82,104	19406.1	661.7	54648.1	67025.0	2080095.0
nshare = 2	29,236	3034	9091.8	281067.0	82,104	19763.1	974.8	54088.3	63180.1	1980667.0
MS : 60%										
nshare = 0	123,053	10,632	31309.2	939129.0	123,166	28740.8	615.7	46520.6	77829.7	2342399.0
nshare = 1	63,566	6067	18456.4	573197.0	123,166	28921.7	970.5	44670.3	63126.6	1976467.0
nshare = 2	43,610	4368	13459.4	422194.0	123,166	29316.1	1476.1	44007.2	57466.6	1825464.0
MS : 80%										
nshare = 0	164,079	13,740	41708.9	1248210.0	164,227	38358.1	765.0	36951.7	78660.6	2359100.0
nshare = 1	84,686	7724	24309.3	762330.0	164,227	38110.8	1269.0	35037.5	59346.8	1873220.0
nshare = 2	57,968	5597	17745.97	564108.0	164,227	38577.8	1990.3	34383.1	52128.9	1674998.0
MS : 100%										
nshare = 0	205,124	17,102	52208.6	1558900.0	205,308	48068.1	913.2	27296.3	79504.9	2375940.0
nshare = 1	105,745	9489	30075.0	952139.0	205,308	47182.2	1564.4	25557.1	55632.1	1769179.0
nshare = 2	72,160	6826	21856.8	703947.0	205,308	47596.1	2519.0	24982.4	46839.3	1520987.0
nshare = 3	69,790	6595	19731.3	688755.0	205,308	49595.8	2812.4	24751.2	44482.5	1505795.0

than the medium-scale network, and the number of trips is 676% more. Also, there are longer trips in the large-scale test case.

In the previous section, we observed that dynamic ride-sharing can reduce the travel time and distance and improve the traffic congestion compared to the scenario when we use taxis in the system, and we do not share the trips. However, it can not overcome the basic scenario where all the trips are made with personal cars. The impact of ride-sharing is different when considering a larger scale, as the number of trips, and the trip length is much more than in the medium-scale, and the system has more opportunity to match the shareable trips.

5.2.1. Influence of market-share in large-scale

Table 12 shows the results for different market-shares and numbers of sharing in the network of Lyon (large-scale network).

We mentioned that we use the vehicle accumulation in the network as a measure of traffic congestion.

Fig. 28 shows the traffic situation in a large-scale from 6 AM to 10 AM, every 100 seconds for different market-shares when the number of sharing is zero compared with the no service scenario (when all the trips are made by personal cars). It is clear that increasing market-share increases the number of service vehicles in the system. For example, with market-share = 20%, the system has to serve 41,043 requests with the service vehicles. As we consider the "Sequential trips" in addition to the "Shared trips", even with the number of sharing zero, the system serves this number of requests with fewer trips. It makes 41,011 trips using 3667 service vehicles to serve the requests. With market-share = 40%, the number of requests increases to 82,104, and the number of required service vehicles is 7,261. Finally, with the market-share = 100%, the number of requests is 205,308. The system serves this number of requests, with 17,102 vehicles in 205,124 trips. Accordingly, the increase in the accumulation plot is not very huge. The total travel distance for all the trips increases by 0.8%, 1.5%, 2.3%, 3.0% and 3.7% for the number of sharing 20%, 40%, 60%, 80% and 100% respectively.

Sharing two passengers' trip with the number of sharing 1 with our proposed system can make big progress in reducing congestion in large-scale. Fig. 29 shows this fact. With market-share = 20%, the number of service trips decreases for 48.1%, and the system can make these trips with 1578 fewer cars. For all the market-shares, with the number of sharing 1, the number of trips to serve the same number of requests is almost 50% less than the number of sharing 0. The system serves the requests with 3123 fewer cars with market-share = 40%, 4565 fewer cars with market-share = 60%, 6016 fewer cars with market-share = 80% and 7613 fewer car with market-share = 100%.

Increasing the number of sharing to 2, can make a more remarkable improvement in the large-scale network traffic. Fig. 30 shows the traffic situation in large-scale when the number of sharing is 2 for different market-shares. The number of trips is reduced by 63.9% with the market-share 20%, 64.4% with the market-share 40%, 64.6% with the market-share 60%, 64.7% with the market-share 80% and 64.8% with the market-share 100%. Also, the number of sharing 2 can decrease the needed cars for 57.5%, 58.2%, 58.9%, 59.3%, 60.1% with the market-share 20%, 40%, 60%, 80% and 100% respectively.

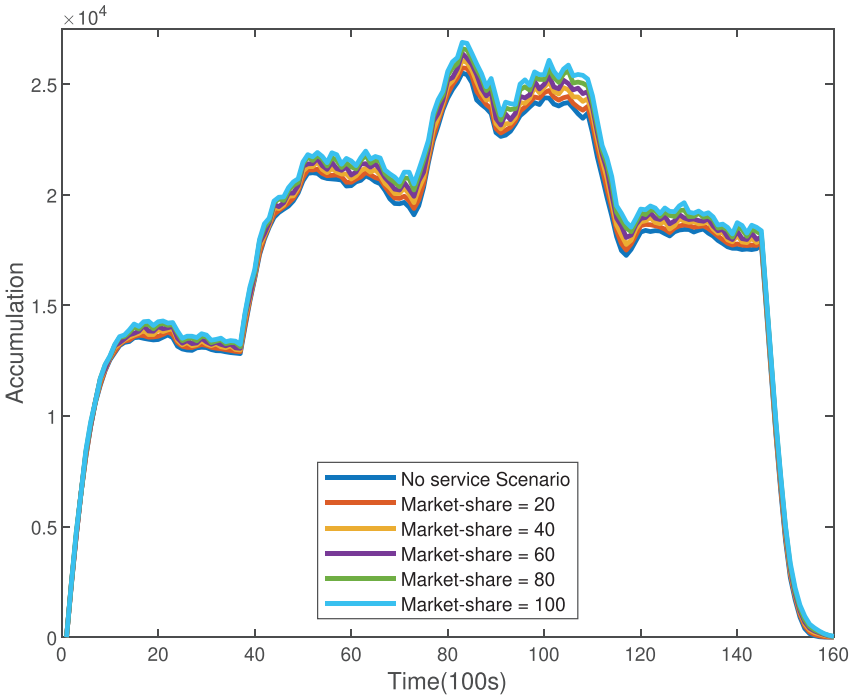


Fig. 28. Traffic situation for the number of sharing 0 with different market-shares (large-scale).

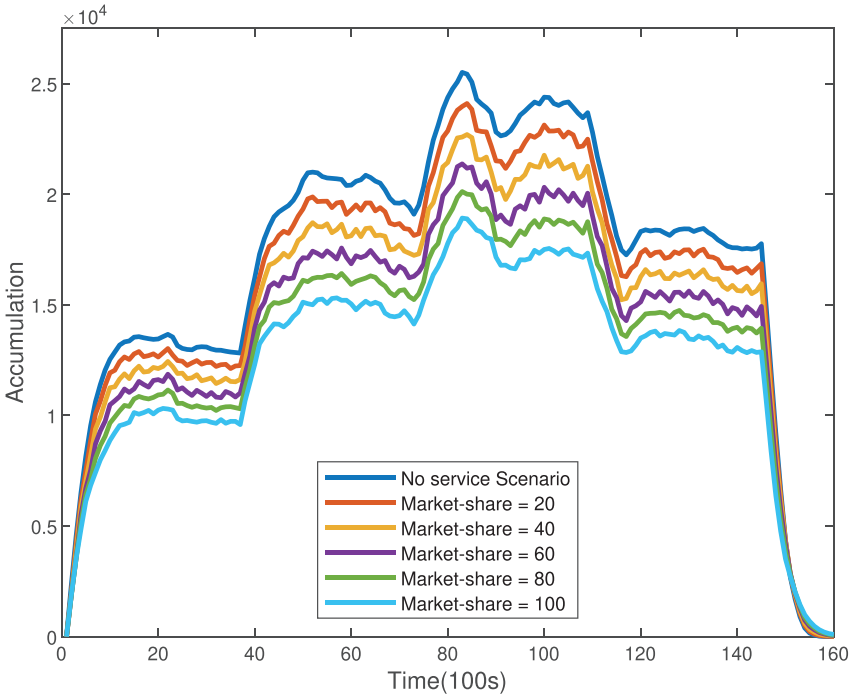


Fig. 29. Traffic situation for the number of sharing 1 with different market-shares (large-scale).

Table 12 shows that sharing can significantly decrease total travel distance and total travel time of the service vehicles and total travel time of the personal vehicles. Sharing the trips can reduce the number of moving vehicles in the network and, consequently, increase the vehicles' speed. So with sharing, even personal cars can move faster and have shorter travel times.

Fig. 31 shows the total travel time and distance for all the service and personal cars in the network for different market-shares when the number of sharing is 0, 1, and 2. It is clear that with the number of sharing zero, total travel time and distance increases with increasing the market-share. Market-share = 100% with the number of sharing zero can increase the

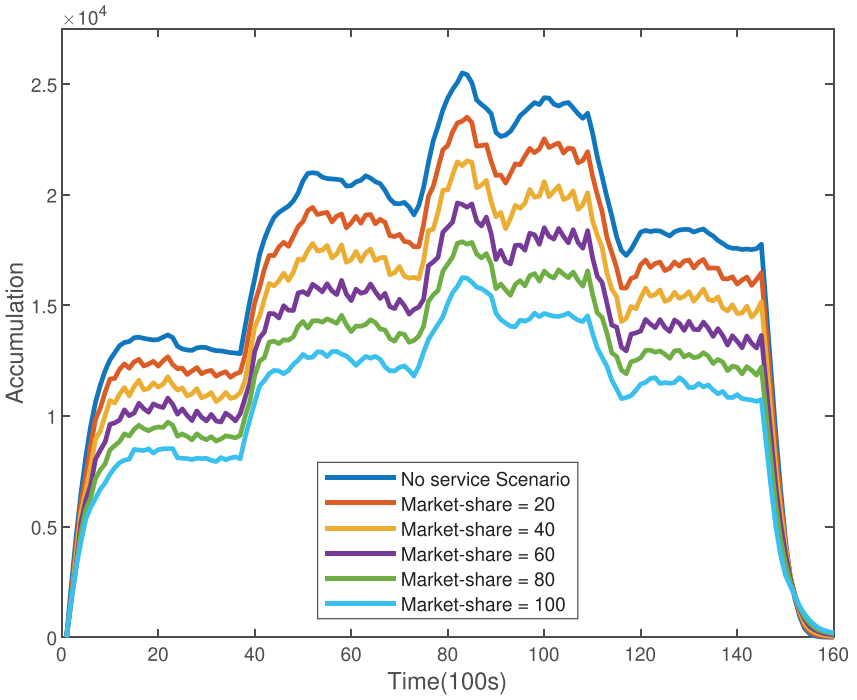


Fig. 30. Traffic situation for the number of sharing 2 with different market-shares (large-scale).

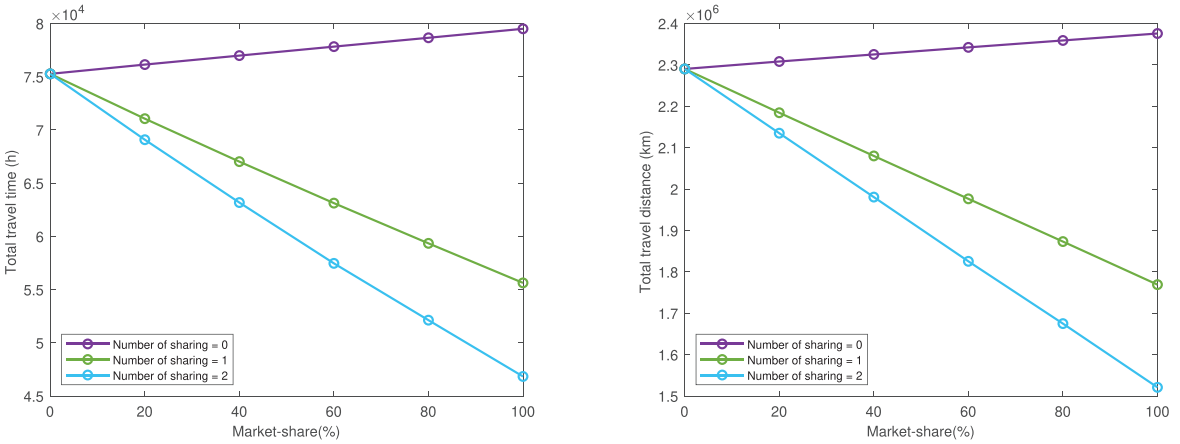


Fig. 31. Total travel time and distance for all the cars for the number of sharing 0, 1 and 2 with different market-shares (large-scale).

total travel time by 5.6% and the total travel distance by 3.7%. Then, sharing can fix this problem by reducing the total travel time by 30.0% with the number of sharing 1 and 41.1% with the number of sharing 2 compared to the number of sharing 0. Furthermore, the total travel distance is reduced by 25.5% with the number of sharing 1 and 36.0% with the number of sharing 2.

5.2.2. Influence of the number of sharing in large-scale

As we mentioned, increasing the number of sharing provides the system with greater leeway to decrease the travel distance by reducing the distance between stop points and depots. Fig. 32 shows how increasing the number of sharing can reduce congestion in large-scale (for market-share = 100%). The system serves 205,308 requests, with 205,124 trips using 17,102 vehicles when the number of sharing is zero. With the number of sharing 1, the requests are served with 105,745 trips using 9489 vehicles. The number of sharing 2 reduces the number of trips to 72,160 using 6826 vehicles. Finally, if we use all the car capacity and share each trip with a maximum of 3 other passengers, the system can serve the requests with 69,790 trips using 6595 vehicles.

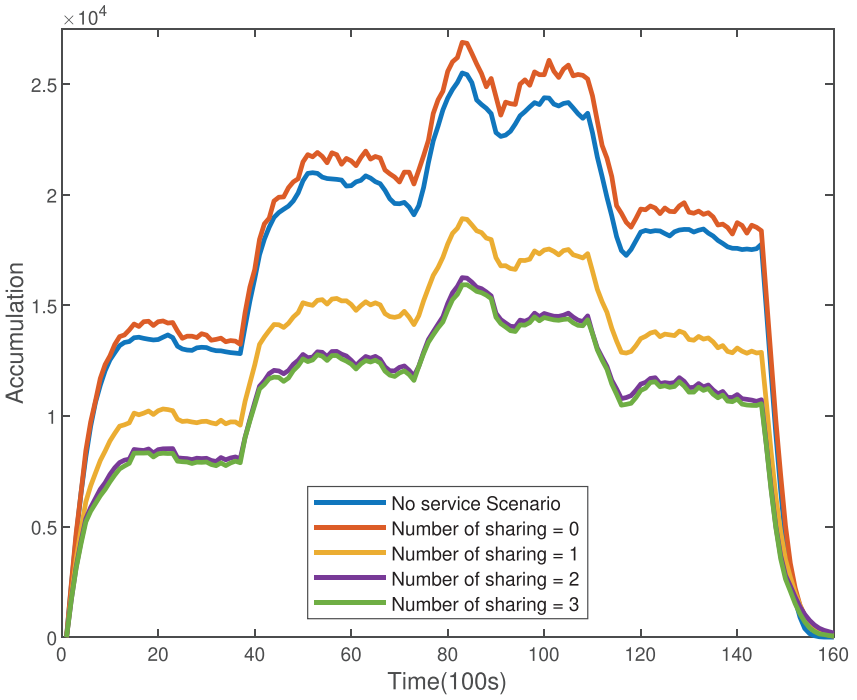


Fig. 32. Traffic situation for market-share 100% with different numbers of sharing (large-scale).

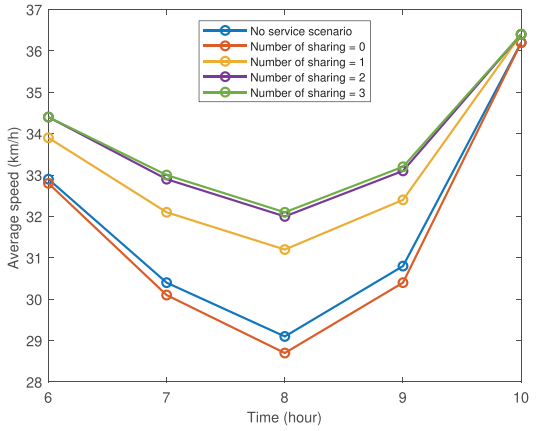


Fig. 33. Average vehicle speed (large-scale).

This reduction in the number of trips and service cars is significantly more effective in reducing congestion during peak hours. Fig. 34 shows how our proposed system can reduce traffic in the morning peak hour. The accumulation of cars at the busiest time of the day can be reduced by 14,908 with the number of sharing 2 compared to the number of sharing 0. The difference between the number of sharing 1 and 2 is more significant than the difference between sharing 2 and 3. With the number of sharing 3, the vehicles have longer travel distance and remain more in the system. So the increase in the vehicles' speed with the number of sharing 3 is not comparable with the number of sharing 1 and 2. Fig. 33 shows the average vehicle speed every hour in the network. In the peak hour, the average speed is 29.1 km/h for the "no service scenario" when we have just personal cars in the system. When we have service cars in the network without sharing, the average speed in peak hour is 28.7 km/h. The average speed increases to 31.2 km/h with the number of sharing 1, 32.0 km/h with the number of sharing 2 and 32.1 km/h with the number of sharing 3.

Increasing the number of sharing increases the passenger waiting time, but the waiting time remains acceptable for passengers with different numbers of sharing. This is because we are targeting to minimize the passengers' waiting time as an objective function in our proposed method. Also, in the large-scale, the accumulation of demands for the service is very high, and we have very close origin points. So the system can share these trips without deteriorating the passengers'

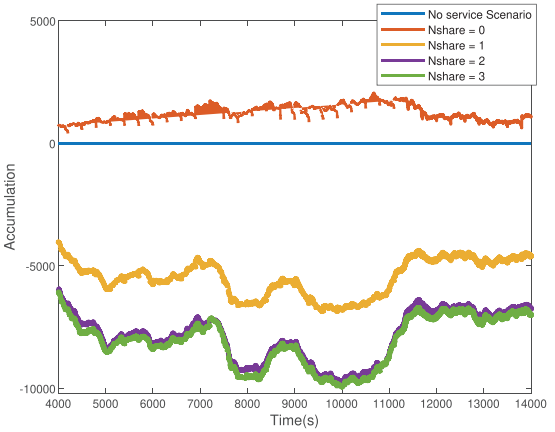


Fig. 34. Traffic situation at peak hour (large-scale).

waiting time. The average waiting time for the passengers is 16.0 seconds for the number of sharing 0. It increases to 27.4 seconds, 44.2 seconds, and 49.3 seconds for the number of sharing 1, 2, and 3.

6. Conclusions

A critical question about dynamic ride-sharing services is whether they can reduce traffic congestion. In this study, we aimed to find the answer. To this end, we first solved two sub-problems of the dynamic ride-sharing problem. The first sub-problem was to design a system which can serve demand and manage a fleet of vehicles accurately and in a short time for real-time services. The second sub-problem was to predict travel times precisely and take into account traffic conditions to determine vehicle availability and pick up/drop off times.

To design the fleet management system, we considered the providers and passengers objectives and proposed an optimization algorithm for the vehicle allocation problem based on the concept of the branch and bound algorithm. We introduced three heuristic methods to make the algorithm fast for large-size problems. Furthermore, we implemented a rolling horizon method in the first heuristic to reduce the number of variables in the algorithm and bring the problem expression more in line with common practice. Also, we assigned the new requests in priority to en-route vehicles. In the second heuristic, we proposed a clustering method based on the shareability index to place the most shareable trips in the same clusters. This is the core of our solving method which is both original and very effective. The main innovation is to cluster users/trips in groups when they are more likely to have interactions (sharing or chaining). In the third heuristic, we introduced the FOSH method to favor sharing opportunities. In the experiments section, we assessed the performance of all the heuristic methods and demonstrated that our heuristic approaches greatly improve computation time with few compromises on optimality.

To solve the second sub-problem, we define two models (plant model and prediction model) to deal with dynamic traffic conditions. Then, we performed an extensive simulation study (based on real-world traffic patterns) to assess the influence of dynamic ride-sharing systems on traffic congestion in medium-scale and large-scale networks. Different situations (five different market-shares and three numbers of sharing) were investigated in terms of traffic conditions. We compared these situations with a baseline traffic situation where all the trips are served with personal cars for both medium-scale and large-scale networks.

The results showed that ride-sharing cannot make a considerable improvement to the traffic situation in medium-scale, i.e when the network scale or the demand are rather limited preventing the system from massive trip reduction through sharing. In this scale, ride-sharing can reduce congestion compared to traditional taxi services and dial-a-ride services. To reduce travel times significantly during peak hours, we expect a remarkable reduction in the number of vehicles on the road network. However, high levels of market-share add extra travel distance and travel time to the trips and lead to more traffic in the network. Thus, dynamic ride-sharing can not be a proper solution for reducing traffic in medium and small-scale cities.

When considering a large trips set, the results are entirely different. In the large-scale (Lyon) simulations, the proposed dynamic ride-sharing system can significantly improve traffic conditions, especially during peak hours. Increasing the market-share and the number of sharing can enhance this improvement. Also, in the large-scale, the accumulation of demand for the service is very high, and we have very close origin points. So the system has more sharing opportunities and can share these trips without deteriorating the passengers waiting time. Therefore, the proposed dynamic ride-sharing system is a viable option, alleviating stress on existing public transport, to reduce the network traffic in populated and large-scale cities.

So the efficiency of a ride-sharing system to reduce congestion looks clearly related to having a critical mass of shareable trips that can alleviate the extra distances generated by the service functioning. Defining such a critical mass in terms of specific factors like trip density, city scale, demand heterogeneity, and distribution would require an extensive sensitivity analysis that is left for future studies.

Credit Author Statement

We confirm that this work is original and has not been published elsewhere no is it currently under consideration for publication elsewhere.

NA, LL and MZ defined the research questions and set up the methodology. NA did the algorithm implementation. The results were reviewed by all authors. NA wrote the initial version of the paper, which was reviewed and edited by LL and MZ.

Acknowledgements

This project is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 646,592 - MAGnUM project).

References

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: a review. Eur. J. Oper. Res. 223 (2), 295–303.
Agatz, N.A., Erera, A.L., Savelsbergh, M.W., Wang, X., 2011. Dynamic ride-sharing: a simulation study in metro atlanta. Transportation Research Part B: Methodological 45 (9), 1450–1464.
Alisoltani, N., Zargayouna, M., Leclercq, L., 2020. A sequential clustering method for the taxi-dispatching problem considering traffic dynamics. IEEE Intell. Transp. Syst. Mag. 12 (4), 169–181.
Ameli, M., Lebacque, J.-P., Leclercq, L., 2020. Cross-comparison of convergence algorithms to solve trip-based dynamic traffic assignment problems. Comput.-Aided Civ. Infrastruct. Eng. 35 (3), 219–240.
Ban, X.J., Dessouky, M., Pang, J.-S., Fan, R., 2019. A general equilibrium model for transportation systems with e-hailing services and flow congestion. Transportation Research Part B: Methodological 129, 273–304.
Bard, J.F., Jarrah, A.I., 2009. Large-scale constrained clustering for rationalizing pickup and delivery operations. Transportation Research Part B: Methodological 43 (5), 542–561.
Braekers, K., Kovacs, A.A., 2016. A multi-period dial-a-ride problem with driver consistency. Transportation Research Part B: Methodological 94, 355–377.
Caulfield, B., 2009. Estimating the environmental benefits of ride-sharing: a case study of dublin. Transportation Research Part D: Transport and Environment 14 (7), 527–531.
Chen, S., Wang, H., Meng, Q., 2020. Solving the first-mile ridesharing problem using autonomous vehicles. Comput.-Aided Civ. Infrastruct. Eng. 35 (1), 45–60.
Davis, N., Raina, G., Jagannathan, K., 2018. Taxi demand forecasting: a hedge-based tessellation strategy for improved accuracy. IEEE Trans. Intell. Transp. Syst. 19 (11), 3686–3697.
d'Orey, P.M., Ferreira, M., 2014. Can ride-sharing become attractive? a case study of taxi-sharing employing a simulation modelling approach. IET Intel. Transport Syst. 9 (2), 210–220.
Ganganath, N., Cheng, C.-T., Chi, K.T., 2014. Data clustering with cluster size constraints using a modified k-means algorithm. In: 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. IEEE, pp. 158–161.
Ghilas, V., Cordeau, J.-F., Demir, E., Woensel, T.V., 2018. Branch-and-price for the pickup and delivery problem with time windows and scheduled lines. Transportation Science 52 (5), 1191–1210.
Goel, P., Kulik, L., Ramamohanarao, K., 2017. Optimal pick up point selection for effective ride sharing. IEEE Trans. Big Data 3 (2), 154–168.
Gonzalez, M.C., Hidalgo, C.A., Barabasi, A.-L., 2008. Understanding individual human mobility patterns. Nature 453 (7196), 779.
Herbawi, W., Weber, M., 2012. The ridematching problem with time windows in dynamic ridesharing: A model and a genetic algorithm. In: Evolutionary Computation (CEC), 2012 IEEE Congress on. IEEE, pp. 1–8.
Herbawi, W.M., Weber, M., 2012. A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation. ACM, pp. 385–392.
Hyland, M.F., Mahmassani, H.S., 2018. Sharing Is Caring: Dynamic Autonomous Vehicle Fleet Operations Under Demand Surges. Technical Report.
Jia, Y., Xu, W., Liu, X., 2017. An optimization framework for online ride-sharing markets. In: Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on. IEEE, pp. 826–835.
Krug, J., Burianne, A., Leclercq, L., 2017. Reconstituting demand patterns of the city of Lyon by using multiple GIS data sources. Technical Report. University of Lyon, ENTPE, LICIT.
Lamotte, R., Geroliminis, N., 2016. The morning commute in urban areas: Insights from theory and simulation. Technical Report.
Leclercq, L., Chiabaut, N., Trinquier, B., 2014. Macroscopic fundamental diagrams: a cross-comparison of estimation methods. Transportation Research Part B: Methodological 62, 1–12.
Leclercq, L., Sénécat, A., Mariotte, G., 2017. Dynamic macroscopic simulation of on-street parking search: a trip-based approach. Transportation Research Part B: Methodological 101, 268–282.
Li, Y., Chung, S.H., 2020. Ride-sharing under travel time uncertainty: robust optimization and clustering approaches. Computers & Industrial Engineering 106601.
Linares, M.P., Montero, L., Barceló, J., Carmona, C., 2016. A simulation framework for real-time assessment of dynamic ride sharing demand responsive transportation models. In: Winter Simulation Conference (WSC), 2016. IEEE, pp. 2216–2227.
Ma, S., Zheng, Y., Wolfson, O., 2013. T-share: A large-scale dynamic taxi ridesharing service. In: Data Engineering (ICDE), 2013 IEEE 29th International Conference on. IEEE, pp. 410–421.
Ma, S., Zheng, Y., Wolfson, O., 2015. Real-time city-scale taxi ridesharing. IEEE Trans. Knowl. Data Eng. 27 (7), 1782–1795.
Mahmoudi, M., Zhou, X., 2016. Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: a dynamic programming approach based on state–space–time network representations. Transportation Research Part B: Methodological 89, 19–42.
Mariotte, G., Leclercq, L., 2019. Flow exchanges in multi-reservoir systems with spillbacks. Transportation Research Part B: Methodological 122, 327–349.
Mariotte, G., Leclercq, L., Batista, S., Krug, J., Paipuri, M., 2020. Calibration and validation of multi-reservoir mfd models: a case study in lyon. Transportation Research Part B: Methodological 136, 62–86.
Mariotte, G., Leclercq, L., Laval, J.A., 2017. Macroscopic urban dynamics: analytical and numerical comparisons of existing models. Transportation Research Part B: Methodological 101, 245–267.
Mourad, A., Puchinger, J., Chu, C., 2019. A survey of models and algorithms for optimizing shared mobility. Transportation Research Part B: Methodological.

Naoum-Sawaya, J., Cogill, R., Ghaddar, B., Sajja, S., Shorten, R., Taheri, N., Tommasi, P., Verago, R., Wirth, F., 2015. Stochastic optimization approach for the car placement problem in ridesharing systems. Transportation Research Part B: Methodological 80, 173–184.
Ordóñez, F., Dessouky, M.M., 2017. Dynamic Ridesharing. In: Leading Developments from INFORMS Communities. INFORMS, pp. 212–236.
Ota, M., Vo, H., Silva, C., Freire, J., 2017. Stars: simulating taxi ride sharing at scale. IEEE Trans. Big Data 3 (3), 349–361.
Özdamar, L., Demir, O., 2012. A hierarchical clustering and routing procedure for large scale disaster relief logistics planning. Transportation Research Part E: Logistics and Transportation Review 48 (3), 591–602.
Paipuri, M., Leclercq, L., Krug, J., 2019. Validation of mfd-based models with microscopic simulations on real networks: Importance of production hysteresis and trip lengths estimation.
Qi, H., Liu, P., 2018. Mining taxi pick-up hotspots based on spatial clustering. In: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, pp. 1711–1717.
Qian, X., Zhang, W., Ukkusuri, S.V., Yang, C., 2017. Optimal assignment and incentive design in the taxi group ride problem. Transportation Research Part B: Methodological 103, 208–226.
Ross, G.T., Soland, R.M., 1975. A branch and bound algorithm for the generalized assignment problem. Math Program 8 (1), 91–103.
Sáez, D., Cortés, C.E., Núñez, A., 2008. Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. Computers & Operations Research 35 (11), 3412–3438.
Stiglic, M., Agatz, N., Savelsbergh, M., Gradisar, M., 2016. Making dynamic ride-sharing work: the impact of driver and rider flexibility. Transportation Research Part E: Logistics and Transportation Review 91, 190–207.
Tahmasseby, S., Kattan, L., Barbour, B., 2014. Dynamic Real-Time Ridesharing: A Literature Review and Early Findings from a Market Demand Study of a Dynamic Transportation Trading Platform for the University of Calgary's Main Campus. Technical Report.
Wang, Q., Boyer, K.L., 2013. Feature learning by multidimensional scaling and its applications in object recognition. In: 2013 XXVI Conference on Graphics, Patterns and Images. IEEE, pp. 8–15.
Wang, S., Li, L., Ma, W., Chen, X., 2019. Trajectory analysis for on-demand services: a survey focusing on spatial-temporal demand and supply patterns. Transportation Research Part C: Emerging Technologies 108, 74–99.
Wang, X., Agatz, N., Erera, A., 2018. Stable matching for dynamic ride-sharing systems. Transportation Science 52 (4), 850–867.
Wang, X., Dessouky, M., Ordonez, F., 2016. A pickup and delivery problem for ridesharing considering congestion. Transportation letters 8 (5), 259–269.
Yuan, N.J., Zheng, Y., Zhang, L., Xie, X., 2012. T-Finder: a recommender system for finding passengers and vacant taxis. IEEE Trans. Knowl. Data Eng. 25 (10), 2390–2403.
Zargayouna, M., Zeddini, B., 2012. Fleet Organization Models for Online Vehicle Routing Problems. In: Transactions on Computational Collective Intelligence VII. Springer, pp. 82–102.
Zou, H., Dessouky, M.M., 2018. A look-ahead partial routing framework for the stochastic and dynamic vehicle routing problem. Journal on Vehicle Routing Algorithms 1 (2–4), 73–88.