On-demand ridesharing with optimized pick-up and drop-off walking locations

Fielbaum, Andres; Bai, Xiaoshan; Alonso-Mora, Javier

DOI
10.1016/j.trc.2021.103061

Publication date
2021

Document Version
Final published version

Published in
Transportation Research Part C: Emerging Technologies

Transportation Research Part C 126 (2021) 103061

Contents lists available at ScienceDirect

Transportation Research Part C

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

On-demand ridesharing with optimized pick-up and drop-off walking locations

Andres Fielbaum [a,*], Xiaoshan Bai [b], Javier Alonso-Mora [a]

[a] Department of Cognitive Robotics, TU Delft, the Netherlands
[b] College of Mechatronics and Control Engineering, Shenzhen University, China

ARTICLE INFO

Keywords:
Ridesharing
On-demand
Pick-up and drop-off points
Ridepooling

ABSTRACT

On-demand systems in which passengers with similar routes can share a vehicle are expected to become a relevant part of future mobility, thanks to their flexibility and their potential impact on reducing congestion. Nevertheless, due to the long detours required by a door-to-door scheme, they induce extra costs to the users in terms of delay. In this paper, we face the design of such a system in which users might be requested online to walk towards/from nearby pick-up/drop-off points if this improves overall efficiency. We show theoretically that the general problem becomes more complex (as it contains two sub-problems that extend set-cover), analyze the trade-offs that emerge, and provide a general formulation and specific heuristics that are able to solve it over large instances. We test this formulation over a real dataset of Manhattan taxi trips (9970 requests during one hour), finding that (a) average walks of about one minute can reduce the number of rejections in more than 80% and Vehicles-Hour-Traveled in more than 10%, (b) users who depart or arrive at the most demanded areas are more likely to be required to walk, and (c) the performance improvement of the service is larger when the system receives more trip requests.

1. Introduction

Mobility on-demand (MoD) systems have started to change mobility systems worldwide, thanks to the ability of massive coordination between vehicles and users. Although these systems (such as Uber, Cabify, Didi or Lyft) can coordinate efficiently by reducing the idle time in comparison with traditional taxis (Cramer and Krueger, 2016), research has shown that they are most likely increasing congestion as some trips that could be done in public transport are now carried in these smaller and non-shared vehicles (Tirachini and Gomez-Lobo, 2020; Henao and Marshall, 2019; Agarwal et al., 2019).

In order to take advantage of the new technologies and reduce congestion, Shared[1] Mobility on-Demand (SMoD, also known as "ridesharing") systems have been proposed, in which different passengers whose routes are somewhat similar share a vehicle (Santi et al., 2014). However, trip sharing induces extra costs to the users in terms of delay, because vehicles may need to make long detours to visit all the origins and destinations that they are serving (Čáp and Alonso-Mora, 2018). Such a problem does not exist in traditional public transport systems (that are also shared), in which passengers walk towards the vehicle's route instead of proceeding the other way around. A similar idea can also be applied to a SMoD system, if the system decides for each user pick-up and drop-off (PUDO)

* Corresponding author.
E-mail address: a.s.fielbaumschnitzler@tudelft.nl (A. Fielbaum).

[1] Throughout the paper, we will use the words "shared" and "ridesharing" when passengers (might) use the same vehicle at the same time, also referred to as "ridepooling".

https://doi.org/10.1016/j.trc.2021.103061
Received 17 September 2020; Received in revised form 10 February 2021; Accepted 21 February 2021
Available online 13 March 2021

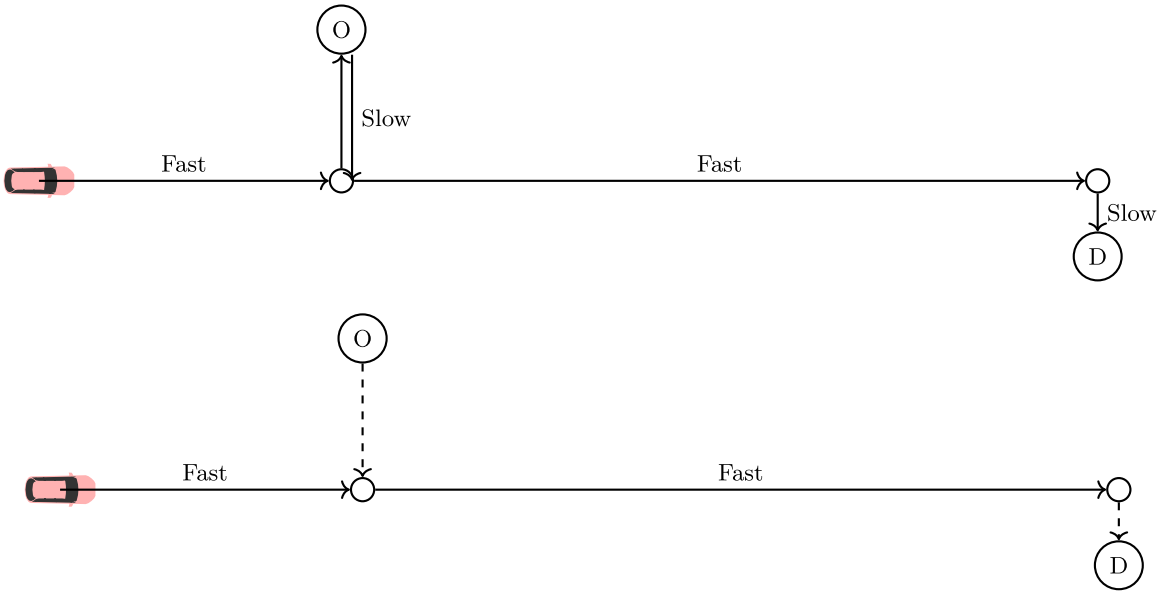

Fig. 1. A vehicle from a ridesharing system that saves significant driving time by requiring the user to walk to/from the fast street in which the vehicle is currently driving. In the top row, the service is door-to-door, i.e., the vehicle must visit the exact origin (O) and destination (D) of a passenger, whereas in the bottom row, walking is admitted (dashed lines).

points that can differ from their actual origins and destinations.

As we detail in the forthcoming Section 1.1, asking some users to walk might generate relevant benefits to the system. The usual door-to-door scheme often forces the vehicle to drive through very slow streets, or to deviate from the shortest and fastest paths, situations that could be avoided if users are willing to take short walks. However, optimizing the PUDO points together with the assignment between users and vehicles can be quite challenging from an algorithmic point of view. Even when passengers do not walk, the efficient operation of a ridesharing system is a complex mathematical task. The main reason is that it extends two classical NP-Hard problems: Dynamic VRP, because it is on-demand, and Dial-A-Ride, due to its sharing aspect. From an intuitive point of view, the complexity emerges due to a large number of feasible combinations between travelers and vehicles. Recently, some papers have proposed techniques that permit operating such systems at the scale of thousands of users per hour (Alonso-Mora et al., 2017; Simonetto et al., 2019; Tsao et al., 2019). When PUDO points are optimized together with the rest of the system, the scale of the problem increases significantly, as each feasible requests-vehicle combination presents now several solutions (by changing the PUDO points for each passenger), and new feasible combinations might emerge (walking can bring some passengers/vehicles closer).

In this paper, we extend the model proposed by Alonso-Mora et al. (2017) (that matches requests and vehicles and can be applied over large instances) by admitting walks. We show that the problem gains complexity and propose several heuristics to solve it. We then apply our model to the real trips that were performed during one representative hour in Manhattan, and compare the results obtained admitting walking or not, finding that when walks are considered, the system improves strongly, inducing short average walking times.

The paper is organized as follows. The remaining of this section explains in detail how walking might enhance mobility systems in general, emphasizing the potential improvements on ridesharing systems, and synthesizes previous papers that deal with similar ideas; Section 2 explains the algorithm that matches groups of requests (with optimized PUDO points) together and with the available vehicles, describes its complexity and proposes heuristics that permit running it over large instances; Section 3 shows the results over a simple network and a real dataset in Manhattan; finally, Section 4 concludes and proposes some ideas for future research.

1.1. Motivation: the role of walking in mobility systems

Most mobility systems that do not rely on private ownership of a vehicle require their users to walk. This is particularly clear in public transport, where passengers are collected in predefined stops or stations for the pick-up and the drop-off. Three main reasons explain why this is convenient. The first one is that public transport systems need to be easy to understand, achieved by having regular and fixed operation rules. The second reason is that gathering passengers together triggers scale economies, allowing the system to provide higher frequencies (Mohring, 1972; Jansson, 1980) and more direct routes (Fielbaum et al., 2020; Cats et al., 2020). These first and second reasons do not apply to ridesharing systems, as there are no such things as frequencies or routes. The third reason deals with diminishing the traveling times for the vehicles and users, which is achieved in three different ways:

1. Some streets have special infrastructure that prioritizes transit vehicles. Transit lines and stops tend to accumulate in those streets. An extreme case for this is the so-called BRT (Bus-Rapid-Transit) system that have been built in many cities worldwide (Deng and

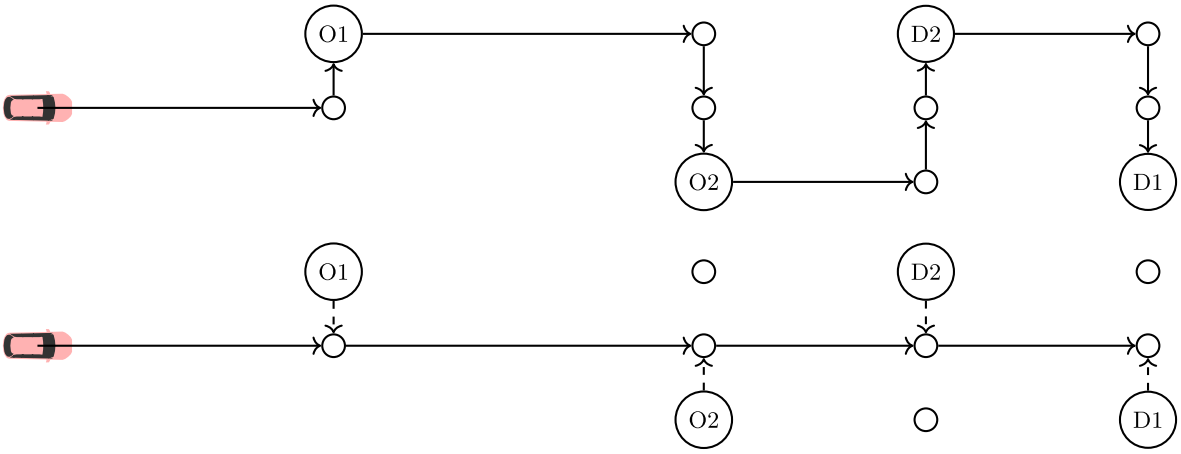

Fig. 2. A vehicle from a ridesharing system that avoids a long detour by requiring the users to walk to/from the fast street in which the vehicle is currently driving. In the top row, the vehicle must visit the exact origins (O1, O2) and destinations (D1, D2) of two passengers, whereas in the bottom row walking is admitted (dashed lines).

Nelson, 2011; Hensher and Golob, 2008; Levinson et al., 2003). Allocated lanes on the streets, in which private vehicles cannot enter, have also proved quite efficient (Basso and Silva, 2014; Basso et al., 2019). Ridesharing systems, if integrated into the general urban mobility system, might also take advantage of this last type of infrastructure.

2. Even without specialized infrastructure, cities naturally evolve with streets of different hierarchies. Large avenues are not only faster than the rest, but they are also the ones with more intersections (Fielbaum et al., 2017; Hu et al., 2008; Figueiredo and Amorim, 2007), meaning that using them requires fewer turns (which usually decreases speed). Therefore, transit lines are typically designed to avoid small streets. A flexible ridesharing system, in which lines are not fixed, might also avoid these slow streets to be more efficient.
3. Even if all streets were homogeneous, visiting the exact origins and destinations might impose significant extra detours to the route of a transit vehicle (Li et al., 2015). By gathering users together, the system ensures that short routes can be followed. As we discuss below, this is a central challenge for ridesharing systems that allow for walks.

Traditional taxi services also require users to walk, but only to the pick-up point. In this case, the coordination is implicit: nobody agrees an exact point, but there are some streets (usually the large avenues) that idle taxis cruise more often so that passengers know that it is easier to find a taxi quickly if walking there instead of waiting at their exact origin. Moreover, empirical studies show that under high-demand conditions, traditional taxi street-hailing (that requires walking) might be more efficient than e-hailing (door-to-door) (Nie, 2017).

How can walking enhance ridesharing systems? As discussed above, there are two main sources of improvement: prioritizing high-speed and highly-connected streets, and avoiding detours. The former is illustrated in Fig. 1, in which walking (bottom row) allows the vehicle to stay in the main avenue, keeping away from three slow segments: going to pick-up the request and back, and the path from the avenue to the request's destination (top row). Vehicles can get close to most points within a city using mainly medium or large streets, but reaching an exact point usually requires driving through some slow streets, i.e., the case illustrated in Fig. 1 can be usual[2].

Fig. 2 shows how avoiding detours can be very significant when several users share flexible routes. In this example, with only two sharing passengers, the vehicle needs to turn a corner seven times and cross the street from where it came three times (top row), inducing a severe extra time that is wholly saved when short walks are admitted (bottom row).

Although most flexible systems provide door-to-door services, there are real-life examples that do ask passengers to walk. The most usual ones are some airport shuttles that pick-up passengers in a predefined meeting point; this is the case, for instance, in Madrid, Melbourne, and Sidney. Another one is Uber's "Express pool" that operates in some American cities, in which passengers are told to walk to some pick-up points different from their origins.

Finally, it is worth explaining that both users and operators might benefit from this alternative. Operators' efficiency is directly improved when following shorter routes that diminish the required average driving time/distance to serve each passenger. On the other hand, users seem to be negatively affected because walking is less comfortable than waiting at home (or whatever the origin is). However, this can be compensated by lower waiting and in-vehicle times (as we show in our simulations), and even if that is not the case, operators' gains can be shared with users employing reduced fares. Moreover, SMoD systems might be integrated into transit networks (as suggested by Luo and Nie, 2019; Fielbaum, 2020), where passengers are already used to walk.

[2] This ubiquity of medium and large streets has been shown in many real cities, using the so-called urban *dual graph* (Porta et al., 2006).

1.2. Related work

New technologies have induced a boom in the research that deals with transport systems where cars are shared (but rides are not), which is surveyed in Narayanan et al. (2020), Wang and Yang (2019). In that context, ridesharing systems have also been studied with increasing intensity. The optimization problem that underlies these systems is much more complex than in the non-sharing case: when new requests have to be assigned to the fleet, the set of feasible solutions now also includes every possible matching between requests. Methods to operate such systems include agent-based models (Fagnant and Kockelman, 2018; Martinez and Viegas, 2017; Vosooghi et al., 2019) and algorithmic approaches. Let us describe some of the algorithmic ones in more detail, as this is the approach we follow in this paper: Alonso-Mora et al. (2017) identify the feasible sets of requests that can be matched together and which vehicles can carry them, to then solve an ILP that decides how to assign them; Simonetto et al. (2019) modify that model by assuming that at most one new request can be assigned to each vehicle at a time, which reduces the number of feasible combinations and turns the ILP into an LP, decreasing the computational load; Tsao et al. (2019) focus on edge-flows induced by the vehicles, which makes a problem that can be computed fast, but managing only vehicles of capacity 2; and Ota et al. (2016) use a greedy algorithm to assign requests to vehicles sequentially. As this is a dynamic problem (future requests are not known beforehand but affect the efficiency of the solutions), several techniques have been proposed to include future expectations into current assignment decisions (Wallar et al., 2018; Wen et al., 2017; van Engelen et al., 2018; Alonso-Mora et al., 2017).

However, the literature that deals with users that move towards on-demand vehicles is not so extensive. Such papers can be classified into two groups: those that integrate this idea into a general Mobility on-Demand (MoD) system that includes some assignment model, and those that study simpler problems to obtain theoretical insights.

Walking users have been integrated into non-shared MoD systems by Zhao et al. (2018), using time–space windows to model feasible solutions. Meeting points, in which several users board or exit the system, have been proposed for "flex-route" systems, i.e., traditional transit lines with some flexibility to adjust their routes on-demand (Zheng et al., 2019; Pei et al., 2019).

Regarding more flexible shared systems, Stiglic et al. (2015), Li et al. (2018) also consider meeting points, which are selected from a set that does not contain all of the nodes in the city, for systems in which private drivers seek for travelers with whom to share their routes; Kaan and Olinick (2013) study meeting points reachable by car, in a scheme in which different commuters share a common destination; and Fielbaum (2020) considers a feeder SMoD system, in which passengers are also gathered together, over a simplified scheme in which it is not necessary to model how to assign the requests together, thanks to its regularity assumptions. Note that all these papers decide meeting points, where all the co-travelers sharing a vehicle have to move, instead of designating one pick-up and one drop-off location per passenger, as we do in this paper. This restriction can be quite limiting, as it can only be applied when the co-travelers' origins (or destinations) are very close to each other, while an efficient ridesharing system might also match passengers when the origin of one of them is close to any point of the route of the other (like in Fig. 2 above).

From a theoretical point of view, the mathematical problem that emerges for a single vehicle that can choose PUDO points for its passengers is studied by Li et al. (2015) and Fielbaum (2021): the former considers that the order in which the requests are served can be optimized, showing that this makes an NP-Hard problem and proposing a dynamic-programming approach to solve it; while the latter assumes that the said order is fixed, where the problem can be solved in polynomial time. On a slightly different note, Gambella et al. (2018) propose a VRP (vehicle routing problem) version in which walking is allowed, studying ways to solve the induced linear relaxation.

Finally, the optimal amount of walking in public transport networks has been investigated extensively. Seminal works on this topic were proposed by Hurdle (1973), Kocur and Hendrickson (1982), Chang and Schonfeld (1991), which introduced "access costs" when optimizing transit lines' frequencies and spatial separation jointly. These access costs are proportional to users' walking times, an idea that we also follow in this paper. These ideas have been extended more recently by Daganzo (2010), Badia et al. (2014), Fielbaum et al. (2020) to include lines' density in the design of a bus transit network, and by Tirachini et al. (2010) when comparing different transit modes' costs. A similar approach was followed by Tirachini (2014) to calculate the optimal distance between consecutive stops within a single bus line.

1.3. Contribution

The main contributions of this paper are threefold: Firstly, we theoretically analyze the advantages and potential of optimizing the PUDO points in a ridesharing system, and we formulate the associated optimization problem, showing that it contains two sub-problems that extend set cover. Secondly, we propose a methodology able to solve this relevant problem over large instances and run simulations over a real-life study case, showing that the proposed method leads to relevant improvements over the original system without walking. Finally, we analyze how the quality of service changes in space to understand these emerging SMoD systems better.

2. An assignment method that allows walking

2.1. The optimization problem

Consider a strongly connected digraph[3] $G = (\mathcal{V}, E)$. Each arc e has two positive functions $t_V(e)$ and $t_W(e)$, representing the time required to cross it over a vehicle and walking, respectively. Walking is assumed to be possible in both directions of each edge with the same cost. Throughout the paper, we use $t_V(u_1, u_2)$ and $t_W(u_1, u_2)$ to denote the shortest time by vehicle and walking between nodes u_1 and u_2. Function $t_V(e)$ is assumed to be fixed (i.e., not dependant on the flow over e), meaning that congestion is not included in the model.

The requests of the SMoD are not known beforehand. The system's aim should be to provide the best level of service throughout the whole period of operation. However, as the system has to decide how to serve the users as they emerge, the mathematical problem that needs to be solved has to be posed with partial information. In this paper, we follow the approach of Alonso-Mora et al. (2017), Simonetto et al. (2019), which assigns a *batch*[4] of users to vehicles at constant times δ (here we use $\delta =$ one minute). Requests are accumulated and removed when picked up by a vehicle or rejected by the system. We aim for an optimal assignment for that set of requests, although heuristics will be needed in practice, as we show.

Formally, we have a set of requests $R = \{r_1, \ldots, r_n\}$, each one being a 4-tuple $r = (o_r, d_r, tr_r, k_r)$, representing its origin, destination, exact time of request, and number of passengers, respectively. Origins and destinations are assumed to be placed on the nodes of the graph[5]. These requests have to be assigned to a set of vehicles $V = \{v_1, \ldots, v_m\}$, each one characterized by its current position P_v, a capacity ν_v, a set of requests currently being served by the vehicle N_v, and a set of future stops S_v that the vehicle must visit, corresponding to either pick-ups or drop-offs of the requests in N_v. The objective is to match requests together and with the vehicles, providing good quality of service while restraining operators' costs. That is to say, to minimize the number of rejected requests and the expected waiting, walking, and travel times for the accepted ones, with low VHT (vehicles-hours-traveled). Note that VHT can also be interpreted as the impact on congestion.

Operating the SMoD system throughout the day requires solving this assignment problem (AP) many consecutive iterations. AP inputs are the graph, the requests R (that can include some previous requests that are not on-board yet so that they might be reassigned), and the fleet V.

AP outputs an **assignment**, i.e., a set of tuples (v, T, π), meaning that the set of requests N_v assigned to vehicle v now includes those in T (hence $T \subseteq R$) and its updated route is π. A feasible assignment must never exceed a vehicle's capacity and cannot allocate a single vehicle to more than one tuple. Additional constraints (e.g., dealing with the quality of service) might be considered.

Let us denote by $\mathcal{A}$ the set of all feasible assignments. The set of previous requests currently being served by the system is denoted by R_0. We aim to find an assignment that minimizes the resulting total costs, defined as the sum of users' costs c_U for new passengers (including the costs of being rejected), the extra users' costs Δc_U for previous passengers induced by new detours, and the operators' costs c_O due to the new routes. Walking costs are included in c_U.

$$\min_{A \in \mathcal{A}} \sum_{r \in R} c_U(r, A) + \sum_{r_0 \in R_0} \Delta c_U(r_0, A) + c_O(A) \tag{1}$$

2.2. The solution method

We now explain how do we solve AP (Problem 1), together with the specific model we consider, i.e., which are the operational constraints and how do we compute users' and operators' costs. We build upon the method proposed by Alonso-Mora et al. (2017) and adapt it to include the possibility of walking. We now describe the main changes, which we explain in detail in the upcoming subsections.

The most relevant changes relate to the identification of feasible assignments between groups of users and vehicles. When deciding whether a vehicle can serve a group of users and which route to follow, one needs to consider all the possible PUDO points for each user in the group, as explained in Section 2.2.1, and determine which PUDO points are feasible. We do this with additional constraints: namely, bounds on the maximum walking time per user, and ensuring that users have enough time to walk towards the pick-up point. These constraints are detailed in Section 2.2.2 together with the changes in the objective function (Eq. (1)) that now reflects that walking is uncomfortable for the users. The rest of the formulation of AP remains the same as in Alonso-Mora et al. (2017), although some heuristics are now required to solve the problem efficiently, which are explained in Section 2.4.

In the following we detail the full method, i.e., the original one with the modifications just stated.

[3] In the appendix we provide a glossary, synthesizing the abbreviations and mathematical symbols used throughout the paper.

[4] This approach is actually followed by some carsharing companies. See https://marketplace.uber.com/matching, accessed on 22/10/2020

[5] This implies that the possible PUDO points (i.e., the spots in which the vehicles can stop) depend on how one defines the set of nodes. A natural chance (that we use in the simulations over Manhattan in Section 3.2.2) is considering every corner and dead-end in the city, leading to a very granular graph. However, this might be different if one wants to consider specialized infrastructure, such as dedicated lanes, where nodes might better represent the only places the vehicle can stop.

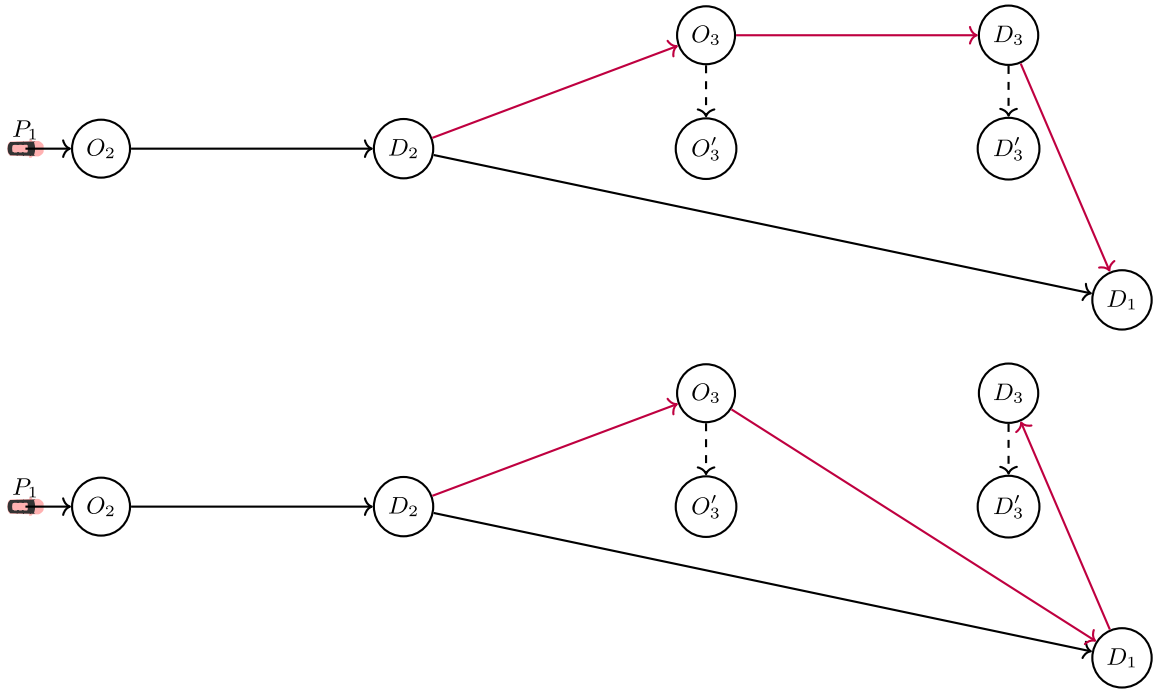

Fig. 3. A vehicle that is currently carrying passenger P_1 and has already been assigned to serve P_2, following the solid black line.There are two feasible updated sequences to serve the new request P_3, that are marked in purple: $(O_2, D_2, O_3, D_3, D_1)$ (top row) or $(O_2, D_2, O_3, D_1, D_3)$ (bottom row). Furthermore, P_3 can be picked up at his origin O_3 or at $O_3{}'$, and dropped off at D_3 or $D_3{}'$. Dashed lines represent the chance of walking. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2.2.1. *Definition of the GV-Graph*

Solving AP rests on building the "Groups-Vehicles" *GV*-graph defined in[6] Alonso-Mora et al. (2017). This graph has as nodes the set $T \cup V$. Each $\tau \in T$ is a set of requests, which forms a *group*, and $T \subseteq \mathcal{P}(R)$ is the set of feasible groups, where $\mathcal{P}(R)$ stands for the power set of R.

Each arc τv (defined as a *trip*) in the graph represents a potential matching between a group and a vehicle, and is characterized by the cost induced to the system when v serves τ. To compute this cost, note that there might be several feasible ways to match τ and v, depending on the *route* followed by the vehicle and the PUDO points for the requests in the group. Note that, for given PUDO points, the route can be fully defined by the *sequence* in which requests are served, as shortest paths are followed between two consecutive stops. To optimize the sequence and PUDO points, one needs to take into account only the status of v and the characteristics of the requests in τ, regardless of the rest of the system, because we assume no congestion. The arc's cost $cost(\tau, v)$ is the one that results after optimizing the sequence and the PUDO points.

The calculation of an arc's cost is exemplified in Fig. 3, when including the group formed only by P_3 to a vehicle v already assigned to two previous requests P_1, P_2, following the sequence (O_2, D_2, D_1). Assume that, according to some exogenous constraints, there are two feasible new sequences $(O_2, D_2, O_3, D_3, D_1)$ or $(O_2, D_2, O_3, D_1, D_3)$, and two possible pick-up and drop-off locations for P_3. This makes a total of eight possible ways to insert P_3 into v's route, such that the cheapest one will define $cost(\{P_3\}, v)$.

When deciding which are the feasible sequences, a decision needs to be taken regarding an inevitable trade-off between the costs explained so far and the so-called *unreliability* of the system. In fact, Fielbaum and Alonso-Mora (2020) define the "one-time unreliability" as the discomfort faced by users when the realization of their trips differ from what was originally predicted: each time a vehicle's route is updated, some of its previous passengers might face an additional delay, which can be annoying. If these changes are limited in order to make the system more reliable, the number of feasible updated routes decreases, leading to less efficient solutions. This issue can be quite relevant: for instance, numerical simulations by Fielbaum and Alonso-Mora (2020) show that more than 30% of the passengers face some change before completing their trips, with some users facing up to six changes, and that forbidding any route's update that increases a previous passenger's waiting time might yield a relative increase on the rejection rate of about 45%.

There are different ways to face this trade-off, all of them compatible with the general method we are explaining here. When we run the numerical simulations (Section 3), we explain the specific rules that we assume.

Throughout the paper, we use $t_w(r, v, i, j, o, d)$ to denote the waiting time experienced by request r if it is inserted in the vehicle v,

[6] In this reference, they call *trip* what we define below as a group, and they include the requests in the graph, which implies that the respective graph is denoted *RTV* instead of *GV*

being picked up in position i (this refers to the route sequence, for instance, $i = 2$ means that v is picking up this passenger right after the first stop on its current list) and node o, and being dropped off in position j and node d. As explained above, we shall select optimal values for all these decision variables i,j,o,d, i.e., we require finding the optimal sequence for v when including r and the PUDO points for r. To simplify the notation, we will also use $t_w(r,\tau,v)$ as the resulting waiting time faced by r, when the whole group τ is inserted in v, after optimizing the new sequence and the PUDO nodes for each of the requests in τ. Analogous notations will be used for the time over the vehicle t_v, the walking (access) time t_a, and total delay D. D is defined as the difference between the arrival time provided by the assignment and the arrival time t_r^* that would be achieved if the same request was performed by private car, which is given by $t_r^* = tr_r + t_V(o_r,d_r)$.

2.2.2. Building the GV-Graph

An edge τv exists in the GV-graph iff the group τ can be served by the vehicle v meeting the following conditions: (i) All users in τ have enough time to walk towards their pick-up node before the vehicle arrives, (ii) the vehicle's capacity is never exceeded, and (iii) the following constraints related to the quality of service are satisfied:

$$t_w(r,\tau,v) \leqslant \Omega_w,\ \forall r \in \tau \cup N_v \tag{2}$$

$$D(r,\tau,v) \leqslant \Omega_D,\ \forall r \in \tau \cup N_v \tag{3}$$

$$t_a^i(r,\tau,v) \leqslant \Omega_a,\ \forall r \in \tau, i = O,D \tag{4}$$

Where Ω_h are some exogenous parameters ($h = w,D,a$) defining the maximum waiting time, delay and access time, respectively. Note that Eqs. (2) and (3) are applied to the requests in τ, but also to the requests that are currently being served by v, as new requests might impose extra detour to the vehicle. Eq. (4) only applies to the new requests, as in our model, to avoid extra-annoyance, the drop-off point remains unchanged once a passenger has boarded a vehicle (this assumption is discussed and justified further in Section 2.6). The super-index i in Eq. (4) means that we are constraining walking times at the origin and the destination separately.

The cost of each arc τv is calculated as follows:

$$cost(\tau,v) = \sum_{r\in\tau} k_r[p_w t_w(r,\tau,v) + p_a t_a(r,\tau,v) + p_v(t_v(r,\tau,v) - t_V(o_r,d_r))] \tag{5}$$

$$+ \sum_{r\in v} k_r[p_w \Delta t_w(r,\tau,v) + p_v \Delta t_v(r,\tau,v)] \tag{6}$$

$$+ c_0 \cdot \Delta L(S_v) \tag{7}$$

Where p_w, p_a and p_v are the unitary values of waiting, walking and traveling over the vehicle, respectively, and c_0 is the unitary cost of operating a vehicle. The first term in the sum (Eq. (5)) represents the waiting (t_w), access (t_a) and in-vehicle (t_v) costs faced by the new users (c_U in Eq. (1)): we subtract the fixed distance between the origin and destination $t_V(o_r,d_r)$ from the in-vehicle time because it does not depend on the system's optimization. The second term (Eq. (6)) represents the extra waiting and in-vehicle time induced to passengers that were already being served by the vehicle (Δc_U in Eq. (1)), either waiting for it to arrive or riding it; note that the resulting waiting and in-vehicle times still need to respect the bounds given by Eqs. (2) and (3). The third term (Eq. (7)) represents the additional operating costs (c_O in Eq. (1)), caused by the increase in the time-length of the vehicle's route $L(S_v)$. We will say that $cost(\tau, v) = \infty$ when this is an infeasible assignment.

Eqs. (5)–(7) do not include operators' capital costs, because the optimization process is performed for a fixed fleet of vehicles. This is not a minor thing, because operators might be strongly favored by allowing walks if the fleet was also optimized, so our results will likely underestimate the optimal amount of walking. Fleet sizing can be faced by trying different fleets to find the Pareto curves between capital costs and the other costs, or by using analytical approximated functions and giving weights to each component of the cost function. These strategies are studied in more detail in Fagnant and Kockelman (2018), Daganzo and Ouyang (2019), Čáp and Alonso-Mora (2018), among others.

The number of possible trips can be large. In principle, it can be as high as $O(m \cdot n^\rho)$, with ρ the size of the largest feasible group, which might be larger than vehicles' capacities. Therefore, the computation of the GV-graph can represent a burden for the whole solution method. In order to face this issue, we take advantage of the intuitive Lemma 1, which is also used (and proven) in Alonso-Mora et al. (2017), Čáp and Alonso-Mora (2018). As they explain, the set of trips has a structure that can be exploited to make an efficient computation of the GV-graph.

Lemma 1. Let $\tau' \subseteq \tau$. If τv is an arc in the GV-graph, then $\tau' v$ also is.

Lemma 1 entails that for a trip to be feasible, each "subtrip" must be feasible as well. This idea can be seen the other way around: For a given vehicle v, when we know which groups of size q can be feasibly matched to v, we do not need to look at every possible group of size $q + 1$, but only at those whose subsets of size q have already proved feasible (if served by v). With this idea,we can search for trips of increasing size, using as a basis the trips that have already been found. The precise procedure for a fixed vehicle v is described in Algorithm 1, which has been shown to reduce the computational time effectively (Alonso-Mora et al., 2017; Čáp and Alonso-Mora, 2018).

Algorithm 1. Algorithm to calculate feasible trips for vehicle v.

$F_v = \{\varnothing\}$ %The set of feasible groups that can be served by v.
for $q = 1 \ldots \Gamma$bf do
%We analyze trips of increasing size q. Γ is any large number.
Compute the set $G_{v,q} = \{\tau \subseteq R, |\tau| = q : \forall r' \in \tau, \tau \backslash \{r'\} \in F_v\}$ %$G_{v,q}$ contains the groups τ that might be feasible according to the feasible groups of size $q-1$.
if $G_{v,q} = \varnothing$**then**
Break; %Stop when no new trips are found.
end if
for all $\tau \in G_{v,q}$**do**
if $cost(\tau, v) < \infty$**then** $F_v \leftarrow F_v \cup \tau$
end if
end for
end for

2.2.3. Exact ILP formulation of the Assignment Problem AP

Once the *GV*-graph has been obtained, we proceed exactly as Alonso-Mora et al. (2017), i.e., we solve the integer linear problem described by Eqs. (8)–(11), minimizing the sum of the costs of the selected τv arcs (trips) plus a penalization p_{KO} for each unassigned request. This is represented by Eq. (8), in which a binary variable $x_{\tau v}$ is equal to 1 if the group τ is assigned to the vehicle v, and a binary variable z_r is equal to 1 if r is rejected. Each unassigned request is marked as rejected (Eq. (9), that will be an active constraint for any optimal solution), and each vehicle is assigned to not more than one trip (Eq. (10)). If the *GV*-graph contains all feasible assignments (which is the case unless some heuristics are used, as discussed in Section 2.4), this ILP represents exactly the optimization problem underlying AP. Solving the ILP can be more complex than in Alonso-Mora et al. (2017) because allowing walks increases the number of feasible assignments, i.e., the number of binary variables. In large instances, this is tackled with a heuristic explained in Section 2.4.

$$\min \sum_{\tau v \in E(GV)} x_{\tau v}\, cost(\tau, v) + \sum_{r \in R} z_r p_{KO} \tag{8}$$

$$\text{s.t.} \sum_{\tau : r \in \tau} \sum_{v} x_{\tau v} + z_r \geqslant 1, \forall r \in R \tag{9}$$

$$\sum_{\tau} x_{\tau v} \leqslant 1, \forall v \in V \tag{10}$$

$$x_{\tau v}, z_r \in \{0, 1\}, \forall \tau v \in E(GV), \forall r \in R \tag{11}$$

After deciding the assignments, i.e. after solving AP, we perform a re-balancing process in order to leave the system better prepared for the next iteration: Those vehicles that are not moving (i.e., that had no assignments before and neither received one now), are assigned to the rejected requests, with maximum one request per vehicle. These vehicles are not actually picking up those requests (which have already been removed from the system). They are being directed from a zone in which there was oversupply of vehicles -which is why they were idle- towards zones in which there was lack of available vehicles -which is why there were rejected requests-. To do this, we solve the same linear program proposed in Alonso-Mora et al. (2017). Note that walking plays no role here, which is why we can use the rebalancing process from Alonso-Mora et al. (2017) directly.

2.3. Mathematical complexity

In this subsection, we analyze the complexity that is entailed by optimizing the assignments jointly with the PUDO nodes, by showing that the respective optimization problem contains two sub-problems that extend set-cover. Set-cover is a classical combinatorial problem, and it is a well-known fact that, unless $P = NP$, it is impossible to find a polynomial algorithm whose result is guaranteed to be not greater than η times the optimal result, for any $\eta > 0$ (Raz and Safra, 1997). We also analyze the more concrete ways in which this is an obstacle when solving AP.

2.3.1. Complexity of calculating $cost(\tau, v)$

Let us first analyze the problem of calculating $cost(\tau, v)$, i.e., for a given v and τ optimizing the sequence in which the requests in τ are served together with their PUDO nodes. We will show that the *Generalized Traveling Salesman Problem* (GTSP) can be reduced to this problem. GTSP is an extension of TSP (Laporte et al., 1987), in which the nodes of the graph are divided into (possibly disjoint) subsets, such that the traveling salesman has to visit at least one node per subset (instead of each node, as in the classical TSP). It is a known result that GTSP is an extension of set-cover (Garg et al., 2000; Slavik, 1998).

Consider an instance of GTSP, i.e., a graph $G' = (V', E')$, with costs c_e over each arc, $V' = V_1 V_2 \ldots V_q$, and starting node v'; for each $u \in V'$, let us denote $f(u) = i$ iff $u \in V_i$. Define then an instance of calculating $cost(\tau, v)$ over the same graph with an extra drop-off node w; the vehicle v is located in v', has capacity q and is currently carrying no passengers: in-vehicle costs c_V equal to $c_e, \forall e \in E$, and $c_V(u, w) = 0, \forall u \in V$. The extra node w has only one departing edge e' that goes to v', with $c_V(e') = 0; \Omega_a = 1$ (the other Ω's are not relevant

for the reduction) and walking costs are given by:

$$c_W(e = uv) = \begin{cases} 0 & \text{if } f(u) = f(v), \\ 2 & \text{otherwise.} \end{cases} \tag{12}$$

Furthermore, we have q requests, one per subset of nodes, such that the origin of r_i is any node in V_i, and all the destinations are w. What Eq. (12) ensures is that the pick-up node of each request has to be in the same group as its actual origin, and it does not matter which node within that group, exactly as in GTSP. The drop-off plays no role because it is costless to go by vehicle to w, and the vehicle is then forced to return to its initial point. Hence, it is straightforward to see that each feasible solution for GTSP is a feasible solution for our problem and vice versa, with equal costs.

This result does not apply when walks are not admitted, because in that case calculating $cost(\tau, v)$ is much more similar to (asymmetric) TSP, which does admit constant-approximation algorithms (Svensson et al., 2018). Hence, if one wants to run this model with high-capacity vehicles, optimizing both the sequence and the PUDO points might be out of reach. Moreover, even if we are using low-capacity vehicles, we might optimize but at the cost of looking at each feasible solution, which can also be unrealistic if the number of requests and vehicles is large, due to the huge possible number of trips.

2.3.2. Complexity of the assignment problem

The assignment problem (described by the ILP in Eqs. (8)–(11)) is also an extension of set-cover. In the set-cover problem, we are given a universe set U and some subsets $S_1, \ldots, S_p$ such that $\bigcup_{i=1}^{p} S_i = U$. Each set has a cost $c(S_i)$, and the aim of the problem is selecting some of these subsets such that they still cover U minimizing the total cost. Given an instance of the set-cover problem, let us define an instance of the assignment problem:

- The set of requests is U.
- Each $(S_i)_{i=1,\ldots,p}$ is a trip.
- There are p vehicles. Each v_i can serve S_i only (making inequality 10 trivially fulfilled).
- p_{KO} is high enough such that it is never optimal to reject a request (which can be done because we know that the original set-cover instance is feasible).

Therefore, choosing which trips to serve is equivalent to chose which subsets to select. It is straightforward to see that this is indeed a reduction that preserves the costs of the solutions.

2.3.3. Analysis

The two sub-problems mentioned above that extend set-cover affect the solution procedure in different ways.

- The calculation of $cost(\tau, v)$ has to be performed for each feasible trip τv. The number of such trips can increase exponentially with the size of the input (number of requests and vehicles). Therefore, the lack of an efficient optimal algorithm is relevant, and some heuristics are needed when solving large instances, which is true even if each trip presents a small size (recall that, in theory, it could be possible to have very large trips, but this is not the usual case and the size of a trip rarely exceeds the capacity of the vehicles). Noteworthy is that these calculations can be easily parallelized, for instance, if each vehicle carries its own computer and computes the trips in which it participates.
- The optimal assignment has to be found just once, when solving the ILP described by Eqs. (8)–(11), but the instance is the whole GV-graph, which might be large. Nevertheless, standard solvers are able to find the optimal solution (up to a small tolerance) in just a few seconds.

It is worth commenting about the optimality of this approach. If no heuristics are used and enough computational resources and time are available, the method we propose in this paper is optimal for AP as the one we take as a base, by Alonso-Mora et al. (2017). However, in real-life cases this might be impossible. The crucial difference is given by the number of possible PUDO points for each trip. For example, if each node has ten nodes within the maximum walking-distance, then a group formed by a single request could be served in a hundred different ways and a group formed by q users in 100^q ways. This increase precludes an exhaustive search, as performed by Alonso-Mora et al. (2017).

2.4. Heuristics

In the previous subsection, we conclude that heuristics are needed to apply the solution method over real-life instances. Moreover, we have identified the computation of the GV-graph as the method's bottleneck, because each of its many arcs requires optimizing the route and the PUDO points for the corresponding group of requests. With this in mind, we propose four heuristics that aim precisely at accelerating the computation of the GV-graph, by means of both reducing the number of trips (τ, v), and hastening the calculation of $cost(\tau, v)$.

Table 1
An example of the "Filtering Vehicles" heuristic. In the top matrix, request A can be served by four vehicles. In the bottom matrix, the trip (A, v_4) is erased (marked as red) because it is too costly, which discards automatically any assignment between v_4 and any group containing A.

	Groups			
Vehicles	$\{A\}$	$\{A,B\}$	$\{A,C\}$	$\{A,B,C\}$
v_1	1	1.5	1.7	2
v_2	1.5	–	–	–
v_3	1	–	1.8	–
v_4	2	2.2	2.1	2.6

	Groups			
Vehicles	$\{A\}$	$\{A,B\}$	$\{A,C\}$	$\{A,B,C\}$
v_1	1	1.5	1.7	2
v_2	1.5	–	–	–
v_3	1	–	1.8	–
v_4	2	–	–	–

2.4.1. Optimization of the sequence and the PUDO points

In order to reduce the time required to compute $cost(\tau, v)$, we do not perform an exhaustive search to calculate the sequence, but we use a so-called insertion heuristic, that adapts to the allowing-walks case the heuristic used by Alonso-Mora et al. (2017) when solving larger instances. An insertion heuristic means that the requests in τ will be inserted one by one into the vehicle (in some arbitrary order). As this idea has been previously used and shown to work properly, this heuristic will be taken as a basis for the other ones.

Assume that we have already decided where to include the first q requests in τ, and we are now inserting the $q+1$-th one, i.e., we search for $(i_{q+1}, j_{q+1}, o_{q+1}, d_{q+1})$: How to alter the vehicle's sequence and the PUDO points for the $q+1$-th request in τ. To do so, we optimize over a subset S_{q+1} of possible sequences and PUDO points. S_{q+1} can be exhaustive or might be built using the heuristics "Limiting sequences" and "Searching for PUDO nodes" explained below in this section.

Nevertheless, such a procedure, that considers the previous requests' positions and PUDO points as fixed, would have a relevant drawback: the optimal PUDO points selected for the previous requests in the trip might be inefficient when the following requests are inserted. To solve this problem, we keep in memory the best $L > 1$ solutions when searching in S_p for every $p = 1, \ldots, q$, and we search in S_{q+1} using all the possible combinations for the previous requests, meaning that we have to compare at most $|S_{q+1}| \cdot L^q$ possible combinations. For details, see Algorithm 2. In practice, we use $L \in \{2, 3\}$.

Algorithm 2. Optimization of the PUDO points and the vehicle's route.

1: $Z_1 = \ldots = Z_{|\tau|-1} = \varnothing$ %Z will store the L best solutions.
2: **for** $q = 1 \ldots |\tau|$ **do**
3: Compute S_q %The set of feasible sequence and PUDO points for r_q, that might be exhaustive or obtained with some heuristic procedures.
4: Define the feasible set of solutions as $F = Z_1 \times \ldots \times Z_{q-1} \times S_q$;
5: **if** $q = |\tau|$ **then**
6: Return the best solution in F;
7: **else**
8: Find the best L solutions in F. Define Z_q as the variables corresponding to S_q;
9: **end if**
10: **end for**

2.4.2. Additional heuristics

Besides the previous heuristic (adapted from (Alonso-Mora et al. (2017)), we propose three additional ones, that are used in different steps of the optimization process and pursuing specific targets: reducing the number of trips ("Filtering vehicles"), hastening the definition of the sequence in which the requests are served ("Limiting sequences"), and hastening the definition of the PUDO points ("Searching for PUDO nodes"), respectively. The last two heuristics build the set S_{q+1} just explained in Algorithm 2. The performance of these heuristics is shown in Section 3.1.

- **Filtering vehicles**: The GV-Graph is supposed to include all feasible trips, but this could be a huge set, requiring a very long time to be computed. To face this, we discard some feasible arcs in the GV-graph that link single-request groups with vehicles. More precisely, for each request r we consider the set of vehicles that can serve it, denoted V_r, and we discard rv arcs for those $v \in V_r$ that fulfill $cost(r, v) > mean\{cost(r, u) : u \in V_r\} + \beta \cdot std\{cost(r, u) : u \in V_r\}$ (std stands for standard deviation, and β is a parameter of the heuristic), unless v cannot serve any other request, to prevent that discarding these arcs increases the number of rejected requests. Recall that in Algorithm 1 we search for feasible groups of increasing size, so limiting those groups of size 1 has a relevant impact: if the arc rv is discarded, then no group containing r will be linked to v.

To understand this heuristic better, it is useful to consider the example given by Table 1, that depicts the costs of all the trips including passenger A without (top matrix) or with (bottom matrix) using this heuristic, using $\beta = 1$. Passenger A by itself could be feasibly served by four vehicles, and some of them can also serve larger groups, including passengers B and C. The heuristic only looks at the costs of transporting A alone (second column), leading to erase (marked as red in Table 1) v_4 because its cost exceeds $mean(\{(1,1.5,1,2)\}) + 1 \cdot std(\{(1,1.5,1,2)\})$. Note that in the last row of the bottom matrix, the costs of the matching v_4 with any other group containing A are not even computed, which is why this heuristic saves computational time.

- **Limiting sequences**: When calculating $cost(\tau, v)$, we limit the number of sequences even further (besides the insertion heuristic). To decide the sequence, recall that we are inserting the requests one by one, so we should decide a position i for the pick-up in which the vehicle is not full and a position $j \geqslant i+1$ for the drop-off. Instead of trying all possible (i,j) combinations, we only look at one j_i for each i, where j_i is calculated through a local search: assuming no walking for this search, we start from $j_i = i+1$ (i.e., dropping-off right after picking up), and we increase j_i only if it is feasible and if it makes the total cost to decrease.

  To understand this heuristic better, it is worth noting that we can express j_i as follows:

$$j_i = \max_j \{j > i : \forall j' = i+1, \ldots, j, cost(r, v, i, j', o_r, d_r) < cost(r, v, i, j' - 1, o_r, d_r)\}$$

  This procedure is detailed in Algorithm 3.
- **Searching for PUDO nodes**: We want an algorithm that can be used over very granular graphs, i.e., where nodes can be placed close to their neighboring nodes. Although Eq. (4) limits the number of nodes that can be used as pick-up or drop-off points, this number will still be high in detailed graphs. Instead of an exhaustive computation over all the possible PUDO points, we consider a limited set of "candidates", found through a local search from the exact origins and destinations: in simple words, we consider as candidates those neighbours of the exact origin (respectively, exact destination) that decrease the total costs, to then analyze their neighbours; we continue doing so until no new neighbour is included as a candidate (i.e., total costs cannot be decreased with any candidate's neighbour). This is explained in detail in Algorithm 4, where $X(u)$ stands for the walking-neighbors of u, $X(u) = \{u' \in \mathcal{V} : uu' \in E \vee u'u \in E\}$.

  To understand this heuristic better, consider a given order in which to insert the request in the vehicle's route, defined by (i,j). Without the heuristic, if the origin of the request is o_r, the set of pick-up candidates is $\{u \in \mathcal{V} : t_w(o_r, u) \leqslant \Omega_a\}$. This heuristic reduces the set of pick-up candidates to the nodes u that, besides being walking-reachable ($t_w(o_r, u) \leqslant \Omega_a$), verify the following condition: there exists a path $P = x_0, \ldots, x_\kappa$, with $x_0 = o_r$ and $x_\kappa = u$, such that

$$c(r, v, i, j, x_{i+1}, d) \leqslant c(r, v, i, j, x_i, d),\ \forall i = 0, \ldots, \kappa - 1$$

  I.e., there is a walking path that, assuming no walking in the drop-off, diminishes the induced cost in every step. The set of candidates for the drop-off can be expressed analogously.

Algorithm 3. Local search to decide drop-off's position given pick-up's position.

$j_i = i + 1$
for $j' = i + 2 \ldots l + 2$ **do**
%l is the number of stops prior to this insertion
if Vehicle v was not full between $j' - 1$ and j' $\wedge$ $cost(r, v, i, j', o_r, d_r) \leqslant cost(r, v, i, j' - 1, o_r, d_r)$
$j_i = j'$;
else
break;
end if
end for

Algorithm 4. Local search to decide PUDO points.

$C_O = C_D = \varnothing$; $o' = o_r, d' = d_r$;%C_O and C_D will be the candidates set for the pick-up and the drop-off point, respectively.
while true **do**
for all $x \in X(o') \setminus C_O$ **do**
if $cost(r, v, i, j, x, d_r) \leqslant cost(r, v, i, j, o', d_r)$ **then**
$C_O \leftarrow C_O \cup \{x\}$;
end if
Every element in C_O has had its neighbors checked
break;
end if
Update o' to a new element in C_O;
end for
end while
Build C_D in an analogous way

2.5. Complete formulation of the assignment algorithm

We can now put everything together to describe the algorithm that solves AP during each iteration. This is written in pseudo-code in Algorithm 5, assuming that all the heuristics are used. We show explicitly when the heuristics are called, so the algorithm without one or more heuristics is obtained discarding those calls.

Algorithm 5. Solution of the Assignment Problem.

1: Inputs: Set of request $(o_r, d_r, tr_r, k_r)_{r \in R}$, Set of vehicles (P_v, ν_v, N_v, S_v)
2: $E(GV) \leftarrow \varnothing$
3:**for all** $r \in R, v \in V$**do**
4: Compute $cost(r, v)$ using Algorithms 2–4; %Algorithms 3 and 4 are "Limiting sequences" and "Searching for PUDO nodes", respectively.
5: **if** $cost(r, v) < \infty$**do**
6: $E(GV) \leftarrow E(GV) \cup (r, v)$;
7: **end if**
8: **end for**
9: **for all** $r \in R, v \in V_r$**do**
10: **if** $cost(r, v) > mean\{cost(r, u) : u \in V_r\} + std\{cost(r, u) : u \in V_r\}$**do**
11: $E(GV) = E(GV) \backslash (r, v)$; %The "filtering vehicles" heuristic
12: **end if** 13: **end for**
14: Use Algorithm 1 to compute the arcs of the GV-Graph for trips of size $q \geqslant 2$; %Algorithms 2–4 are implicit in this step, when calculating $cost(\tau, v)$ (line 5 in Algorithm 1)
15: Solve the ILP defined by Eqs. (8)–(11);
16: Update P_v, N_v, S_v for vehicles v that were assigned to a trip;
17: Rebalance idle vehicles;

2.6. Chaining iterations, reassignments and the emerging trade-off

As explained at the beginning of this section, AP represents a single iteration of the problem. The daily operation of SMoD is achieved by solving AP many consecutive times. Two consecutive iterations are linked not only because the output of the first serves as part of the input of the second (including the rebalancing process explained at the end of Section 2.2.3), but also because we allow for *reassignments* that improve the efficiency of the system.

Reassigning a request means that, after accepting it and matching it with a group and vehicle, that decision might be changed later when the information of new requests appear. This reassignment is achieved by keeping those requests that have been assigned but not picked up yet as pending requests, meaning that they are not kept in the respective vehicle's plan, but included in the set of requests for the following iteration. Vehicles' positions are updated before removing such requests from their lists. Most requests will be assigned to the same vehicle over and over again until they are picked up, but some requests will be reassigned due to the changes induced by the new requests.

This reassignment process is related to the chance of walking, because any change regarding the pick-up time is more annoying if the user is waiting on the street rather than at his exact origin, so the system should try to avoid those reassignments. Therefore, the reassignment rules might depend on the request, posing some design decisions. Note that a trade-off emerges here: although the chance of walking should improve the system, as it increases the number of feasible solutions through new decision variables (the PUDO points), it might also have negative effects when reducing the flexibility to reassign.

How to limit reassignments due to walking is somewhat arbitrary. We will impose rules that stress out the said trade-off. By doing so, good results in the numerical simulations remain robust regarding this trade-off. Concretely, we make significant differences when a user has been told to walk:

1. Users waiting to be picked up at their exact origin are kept as pending requests for the following iteration. This means that when new requests arrive, they could be re-assigned to a different vehicle, or they can stay assigned to the same one but their waiting time might increase (always bounded by Ω_w) due to detours induced by new passengers. In the most extreme case, they could even become rejected if that reduces the total costs of the system.
2. Users that are told to walk towards a different pick-up node cannot be reassigned to a different vehicle, cannot be rejected, and their waiting time cannot increase.
3. Drop-off points cannot be changed once a user has boarded a vehicle.

The first of these rules is similar to the one used by Alonso-Mora et al. (2017) in the no-walking scheme, but admitting extra waiting time for passengers that don't walk at their origins. The second rule reduces the flexibility of the system to reassign each time someone walks. The third rule prevents a new source of flexibility to emerge thanks to walking. Therefore, we are considering an extreme version of the trade-off, in which walking reduces flexibility as much as possible. If any of these rules were modified, then walking

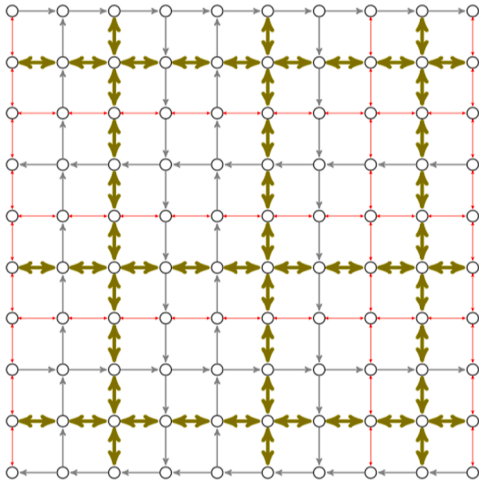

Fig. 4. Basic grid used to test the virtues of optimizing the PUDO points and the heuristics. Green, grey and red arcs are fast, mid-speed, and slow, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

would enhance the system's performance even further.

Note that these rules do not apply within a single iteration, so the system will always achieve a better assignment (i.e., it yields a lower value for the objective function) when walking is admitted because we are enlarging the feasible set of solutions. They are relevant when chaining consecutive iterations, because without walking there will be some extra degrees of freedom. Therefore, this design helps us to see if admitting walks improves the system even in such an environment.

3. Numerical simulations

In this section, we run the method over two different scenarios. The first one (Section 3.1) is a toy network, which is useful because it allows solving the problem without the heuristics, and compare that solution with the ones obtained when the heuristics are used. Therefore, we can study if the heuristics effectively reduce computational time and what is their effect on the quality of the results. We also take advantage of the short computational times required by this network to perform a sensitivity analysis regarding the network, the demand structure, and the two relevant parameters dealing with walking, i.e. Ω_a (the maximum walking time) and p_a (the unitary cost of walking).

The second scenario (Section 3.2) is a real-life case in Manhattan, in which the heuristics are necessary. Here we show that the method is able to deal with thousands of vehicles and requests, and analyze how the SMoD improves thanks to the optimization of the PUDO points in a realistic environment.

Recall that we need to decide on some rules to face the trade-off between which routes are admissible when updating the assignments (efficiency) and how many changes a user might face (reliability). Here we choose to allow the vehicle to serve new requests before previous ones, because the location of the new origins and destinations were not known when the previous route was decided. This includes the requests being reassigned (that enters together with the new requests as input of AP), as deciding the routes is part of the reassignment process. There could be significant losses on efficiency if the vehicle was forced to fulfill previous requests without any change. For instance, in Fig. 3, the vehicle would require to go to D_1 before going to O_3, inducing an unnecessary long detour. On the other hand, we do not allow changes in the relative positions of the users that are currently in the vehicle. Such changes would increase unreliability, and the benefits are expected to be lower, because those relative positions were decided with all the direct information and, moreover, the vehicle is already following the associated route. In Fig. 3, for instance, this means that when the vehicle has already visited O_2, it has to visit D_2 before D_1. Further, if we use an insertion heuristic (as explained in Section 2.4.1 and applied in Section 3.2.1) and we insert the requests in the same order they emerged, then such relative positions will always remain unaffected.

3.1. Results over a toy network

In order to analyze if the proposed heuristics help to solve this problem and perform different types of sensitivity analyses, we first study a small instance (also used by Fielbaum, 2021), in which we can optimize without using the heuristics, and compare the results with the ones obtained using them. The base graph we use is as follows: a 10×10 grid, in which streets can be slow (20 km/h), mid-speed (30 km/h), or fast (40 km/h). Mid-speed streets are unidirectional, and the others are bidirectional. All the arcs are assumed to have the same length (0.15 km), so walking times are the same everywhere; walking speed is 5 [km/h]. The network is depicted in Fig. 4.

Each minute, the system accumulates all the requests that have arrived in that lapse, and solves AP considering the current state of the vehicles. We consider two requests arriving every 15 s. In the base case, requests' origins and destinations are random (uniformly

Table 2
Results of SMoD with or without optimizing the PUDO points. All times are in minutes. Users' costs are shown in proportion to the cost of one minute over the vehicle.

Scenario	Walks?	Av. waiting	Av. walking	Av. delay	Rejections	Av. c_U	VHT
Base	Yes	2.15	2.29	3.38	5.8%	14.6	61.5
	No	2.18	–	3	15%	19.36	70.1
Non-uniform network	Yes	2.25	2.44	2.85	15%	21.8	58.3
	No	1.96	–	2.68	24.4%	26.1	67.1
Concentrated demand	Yes	1.9	2.23	3.03	2.92%	11.4	53.8
	No	2.06	–	3.06	4.58%	10.84	68.4
Non-uniform network & concentrated demand	Yes	1.92	2.34	1.74	7.92%	14.26	50.12
	No	1.85	–	1.4	12.1%	14.78	61.4

Table 3
Performance of the heuristics. All times are in minutes. Users' costs are in proportion to the cost of one minute over the vehicle, and are the result of the weighted sum of the rejection penalizations plus waiting, in-vehicle and walking times.

Heuristics	Running time	Av. waiting	Av. walking	Av. delay	Rejections	Av. c_U	VHT
No heuristics	2.62	2.39	3.68	3.94	7.92%	16.35	57.1
Filtering vehicles	1.57	2.21	3.81	3.94	8.13%	16.45	56.5
Limiting sequences	0.74	2.09	3.15	3.75	11.5%	18.17	62.2
PUDO nodes	2.41	2.28	3.16	3.9	9.79%	17.17	59.8
All heuristics	0.44	2.08	2.89	3.72	15.21%	20.86	61.7

distributed), each of them has only one passenger with probability 0.8, and two or three passengers with probability 0.1 each. The system is served during one hour by a fleet of 100 vehicles, that begin evenly distributed across the network, with capacity $\in \{3, 4, 5\}$ also evenly distributed.

For the cost parameters, we use $p_a = p_w = 2p_v$ (as in Chang and Schonfeld, 1991, a seminal paper in the topic of walking towards transit systems), $p_{KO} = 80p_V$ (a high value to give the acceptance rate a high priority) and $c_0 = 1.5p_v$ (approximated from Jara-Díaz et al., 2017 considering a vehicle of size 4). Regarding the restrictions, we take $\Omega_w = 5$ [min] (from Alonso-Mora et al., 2017), $\Omega_d = 10$ (from Ota et al., 2016), and we define $\Omega_a = 12$ [min]. We choose a high figure for this last parameter, which allows showing that the method can handle several feasible PUDO points. We include later in this subsection a sensitivity analysis for a lower value of Ω_a. The ILP is solved using Gurobi solver in Matlab. We first analyze the results of SMoD with or without optimizing the PUDO points, using the exact procedure (without the heuristics):

3.1.1. The benefits of admitting walks with different networks and demand structures

The virtues of enabling walks may depend on the structure of the network and the demand. Therefore, in this subsection we present simulation results for the base scenario and for three alternative ones in which the distances between consecutive nodes and the distribution of the requests are no longer homogeneous:

- It is usual that cities are more concentrated towards the center and dispersed at the peripheries, which might lessen the benefits of optimizing the PUDO points, as there are not many alternative PUDO points nearby for peripheral nodes. We represent this in the scenario **non-uniform network**, where we still use a 10×10 network, but in which the distance between consecutive nodes increases from 75[m] at the center of the network to 300[m] at the edges (recall that in the base scenario such distances are always equal to 150[m]).
- The optimal amount of walking may depend on the demand's dispersion. If origins and destinations are concentrated, a door-to-door scheme might reduce the detours' length, which was one of the main motivations to optimize the PUDO points. We simulate this situation in the scenario **concentrated demand**, by moving each origin and destination in the diagonal towards the center of the base network with a probability of 0.5.

Results are shown in Table 2 for the base scenario, the two alternatives just described, and a fourth scenario that merges those two modifications. All times are in minutes in Table 2. Several conclusions follow:

- Requesting users to walk improves SMoD significantly. Regarding operators, VHT is reduced by more than 10% for every scenario. For users, the number of rejections can be reduced by at least one third (the two last scenarios), and up to almost two thirds (the first scenario). Average users' costs diminish in three of the four scenarios.
- The reduced rejection rate reveals that diminishing the detours makes more vehicles available to transport the emerging users.
- Average delay might increase or decrease depending on the scenario, which results from two opposing forces: shorter detours reduce in-vehicle time, but walking the first or final legs is slower than riding a vehicle. Waiting times can also increase or decrease when PUDO points are optimized.

Table 4
Sensitivity analysis. The second and third rows show the results of the system when one parameter related to walking is changed.

Scenario	Running time	Av. waiting	Av. walking	Av. delay	Rejections	Av. c_U	VHT
Base	2.62	2.39	3.68	3.94	7.92%	16.35	57.1
$\Omega_a = 5$	2.57	2.29	2.72	4	9.31%	16.45	61
$p_a = 3p_v$	2.4	2.15	2.29	3.78	11.5%	16.8	61.5

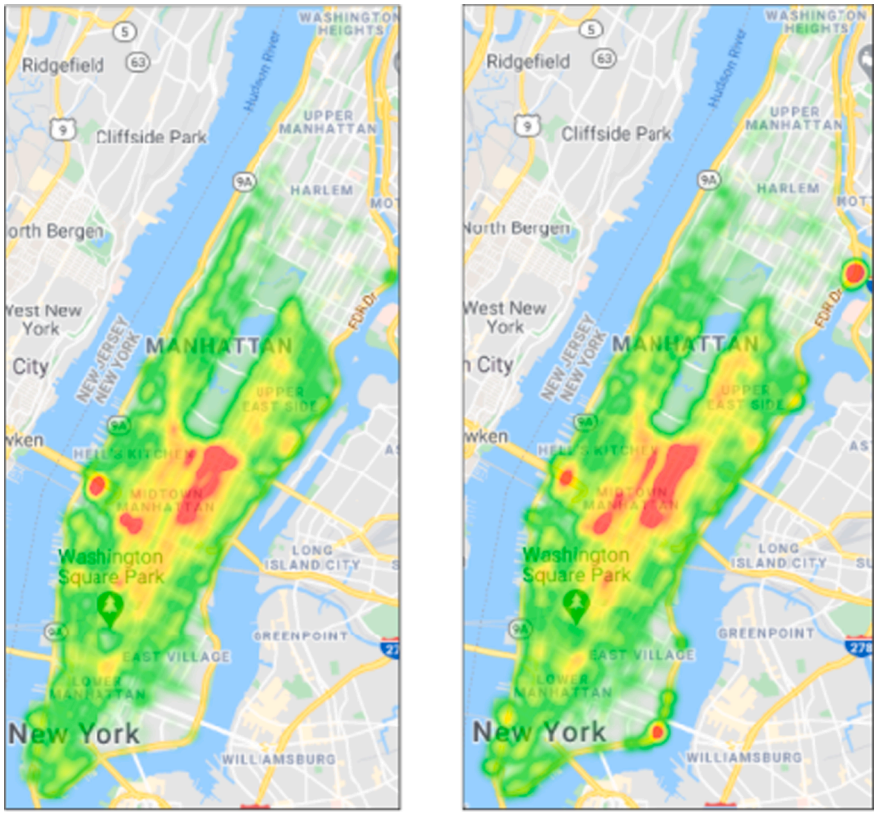

Fig. 5. Departure (left) and arrival (right) rates per node. Green zones represent lower values. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- When the demand is concentrated, average users' costs do not diminish, which happens because the rejection rate was already quite low. It is worth noting that average walks are similar to the other scenarios, and operators' savings become even more significant. This suggests that walking is always relevant to reduce detours, and such reductions might be utilized to increase the service rate, if there are many rejections, or to directly decrease VHT by allowing some vehicles to remain idle for some time.
- The impacts of optimizing PUDO points in the non-uniform network are similar to the base case.

In all, Table 2 reveals that avoiding a door-to-door scheme can be very beneficial for the system as a whole, regardless of the specific conditions of the demand and the network.

3.1.2. Analysis of the heuristics

We now assess the impact of using the heuristics instead of the optimal procedure. To do this, we use again the base scenario and compute a new set of (random) requests. We obtain the results described in Table 3, where all times are in minutes. The following conclusions and comments follow:

- Using all the heuristics reduce the running time in almost six times, which can be crucial when working over larger networks. This is achieved at the cost of higher users' and operators' costs.
- Filtering vehicles decreases operators' costs and increases users' costs only in a small amount. Limiting sequences is the most saving-time heuristic.

Table 5
Description of the Manhattan graph and requests.

Index	Value
Max in-vehicle time between nodes	53.7 [min]
Max walking time between nodes	4.7 [h]
Av. in-vehicle time between contiguous nodes	34 [s]
Max in-vehicle time between contiguous nodes	349 [s]
Av. walking time between contiguous nodes	89 [s]
Max walking time between contiguous nodes	31 [min]
Total number of requests	9970
Av. in-vehicle time between origin and destination	11.61 [min]
Max in-vehicle between origin and destination	43.5 [min]
Min in-vehicle between origin and destination	3 [min]
Mean number of passengers per request	2.1
Max number of passengers per request	6
Median $\{\|WD(u)\| : u \in \mathcal{V}\}$	116 nodes
Max $\{\|WD(u)\| : u \in \mathcal{V}\}$	246 nodes

- Searching for the PUDO nodes does not reduce the running time much, and it increases both users' and operators' costs: in this simple graph, the number of feasible nodes is small, so an exhaustive search is fast and better. It should be recalled, however that this heuristic is meant for very granular graphs, as explained in Section 2.4.

All in all, the heuristics prove useful. When we deal with a large instance of the problem in the following subsections, we are using all the heuristics.

3.1.3. Sensitivity analysis

Two parameters might play a key role in how much walking is introduced in the system, so we change them and show the respective results in Table 4:

- The maximum walking time affects the feasible PUDO points. Although the average walking time in the base case is far from its upper bound, some specific requests might face long walks. We now study the more prohibitive case in which no user can walk more than $\Omega_a = 5$ [min] at the origin or destination. Running time diminishes, because there are fewer feasible PUDO points, but the change is not very relevant. As expected, results get worse, because we reduce the number of feasible solutions. Users' reduction in walking times is outbalanced by the increase in the number of rejections and total delay. Operators' costs also increase.
- There are some papers (such as Fielbaum et al., 2020) that assume that walking is more annoying than waiting, i.e., $p_a > p_w$. If we raise p_a to $3p_v$, the average walk diminishes together with waiting and delay, at the cost of increasing rejections and VHT. Resulting costs should not be compared directly, because increasing one of the cost parameters will obviously yield higher costs.

As a synthesis, both parameters affect the system in the expected directions: they reduce the amount of walking, which increases VHT because vehicles require higher detours, which on turn increases the rejection rate because vehicles require longer times to become available.

3.2. Results over Manhattan

3.2.1. Description of the Manhattan dataset

As in Alonso-Mora et al. (2017), we use the publicly available dataset of taxi travels in Manhattan, New York City (Donovan and Work, 2017); with this dataset, we have the exact coordinates of the origin and destination, together with the time when the passengers boarded the taxi and how many, for each taxi travel in Manhattan during 2013. We will consider one hour of the operation of the system, just after 1 pm of January 15th, 2013, to have a large number of requests but avoiding the peak, in which trips have a quite specific direction; we filter those requests in which the origin and destination are too close (less than 3 min by car), as they rarely occur in real-world on-demand systems (Liu et al., 2019). The system is analyzed over a graph that reproduces the road network of Manhattan with 9453 edges and 4092 nodes, that represent the intersections and dead-ends of the streets. Fig. 5 shows heatmaps corresponding where each request departs (left) and arrives (right): the concentration of origins and destinations is much higher in the center of the network.

In-vehicle times represent the real ones at 1 pm. Walking times are calculated proportional to the length of each arc, which are known as the Manhattan graph provides the geographical coordinates of each node. Table 5 synthesizes the most relevant characteristics of the network and the requests. The last two rows refer to $WD(u)\,(u \in \mathcal{V})$, which is the set of nodes that are at "walking distance" from the node u, i.e., that can be visited walking in less than $\Omega_A = 12[\text{min}]$. Such rows show that the set of feasible PUDO points for each request is large, which justifies searching for the best ones with the heuristic described in subSection 2.4.

Table 6
Results over Manhattan. All times are in minutes. Users' costs are shown in proportion to the cost of one minute over the vehicle.

Scenario	Walks?	Av. wait	Av. walk	Av. delay	% rejections	Av. c_U	% walks	VHT
(i)	Yes	1.74	0.9	2.55	0.78%	5.81	34.11%	0.94
	No	1.68	–	2.97	4.44%	8.2	–	1.05
(ii)	Yes	2	1.2	3.12	1.1%	7.18	40.4%	0.75
	No	2.11	–	4.17	4.93%	10.22	–	0.9
(iii)	Yes	1.18	0.1	1.68	0.25%	3.16	9.79%	1.36
	No	1.22	–	1.83	2.1%	4.7	–	1.38

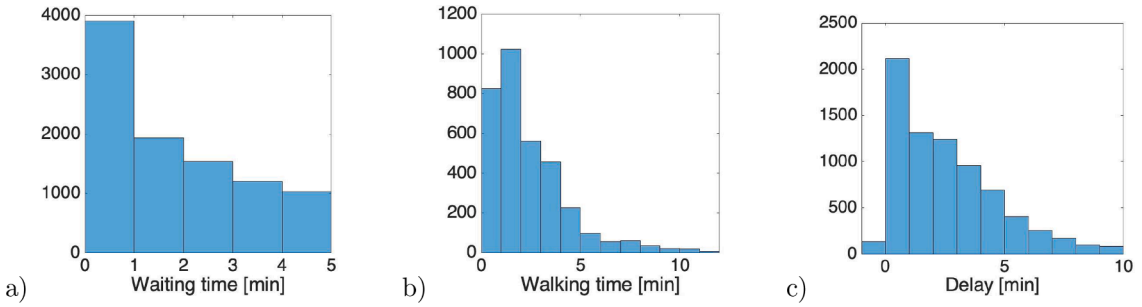

Fig. 6. Histogram of the distributions of (a) Waiting times (b) Walking times (c) Total delay.

3.2.2. *Global results over Manhattan*

Let us first focus on the direct effects of introducing walks. To do so, we compare in Table 6 the results obtained by the system when walking is admitted or not. Three scenarios are considered, which allows us to analyze whether the impact of allowing walks depends on different conditions. The three scenarios are:

(i) 3000 vehicles of capacity 6 (considered as the base scenario for the posterior analyses).
(ii) 2000 vehicles of capacity 9.
(iii) Same fleet as in scenario (i) but considering users' costs only (i.e. $c_0 = 0$, as in Alonso-Mora et al., 2017).

We consider again $p_w = p_a = 2p_v, p_{KO} = 80p_v$, and $c_0 = 1.5p_v$ in the first two scenarios.

Table 6 also shows average users' costs, the percentage of non-rejected requests that are actually requested to walk, and VHT in thousands. Waiting and walking times' units, as well as delays', are minutes.

Results in Table 6 reveal that in this real-life case the on-demand ridesharing system improves significantly when PUDO points are optimized jointly with assignments and routes. In all the three scenarios, both users and operators are better-off (on average) due to the walks. Again, the most relevant impact is on rejections, which are reduced in more than four fifths in the first two scenarios, and in almost nine tenths in the last one. Average waiting time can increase or decrease and in small amounts, while average delay decreases. The reduction in the number of rejections and in the total delay is expected, as some relevant detours can now be eliminated, which also induces a strong decrease in VHT (with the exception of the last scenario, in which VHT is not part of the objective function). The comparison of the first two scenarios suggests that allowing walks has a larger impact when the fleet is smaller (and the quality of service is worse): while users' costs are reduced in about 30% in both scenarios, the reduction of VHT is larger in the second one.

As discussed above, a central aspect of the optimization of PUDO points is the possible increase in computational times. In our simulations, the experiments that include walking take approximately ten times longer (from about one hour per iteration to ten hours). Nevertheless, in both cases, a centralized calculation is too slow, but a distributed one is fast enough. In fact, each vehicle can determine which trips it might serve: in the case we are studying, this would reduce the computational time required to compute the *GV*-graph by thousands (because the fleet sizes are 2,000 or 3,000 in the different scenarios). Admitting walks increases the number of edges in the *GV*-Graph (i.e. the number of feasible trips) from about 6,000 to 40,000. Computational time can be further reduced by tightening β in the **Filtering vehicles** heuristic.

Let us analyze the results in more detail, focusing on the base scenario admitting walks. Fig. 6 shows histograms corresponding to the waiting time, walking time and delay for each request. In Fig. 6b), we only include requests that do walk. There are a few requests with negative delay, which happens because walking the first or last leg might reduce total traveling time in some very specific conditions, e.g., when an arc can be toured faster walking than by car. Fig. 6a) shows that some passengers have negligible waiting time, and that the histogram then decreases towards the maximum waiting time (5 [min]) but slowly, such that there are about one thousand of requests in the highest categories. Figs. 6b), on the other hand, reveal that walking time and total delays tend to concentrate much more around lower values. This suggests that Ω_w plays the most relevant role determining the rejected requests.

Despite of the high value of Ω_a, the resulting walking times are realistic compared to the willingness to walk towards a public transport network: according to Besser and Dannenberg (2005), only 8.4% of the people in USA walk less than five minutes, and 22.4%

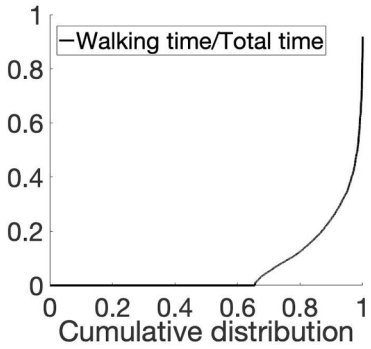

Fig. 7. Distribution of the ratio between each user's walking and total times.

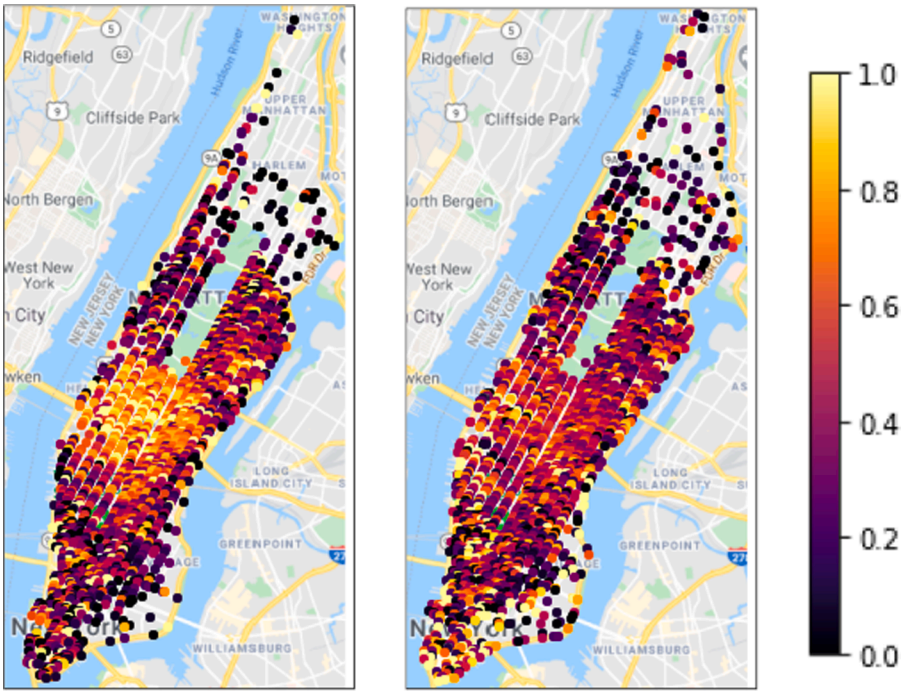

Fig. 8. Relative average delay for requests departing (left) and arriving (center) at each node. Dark dots represent lower values, as shown in the colorbar (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)(For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

walk less than ten minutes (on a daily basis), i.e., people tend to walk much more than what would be required in SMoD.

The proportion of the total time that is spent walking, depicted in Fig. 7, is also consistent with (and lower than) real-life cases: most of the travellers spend a minor part of the time walking, but in some rare cases the access and egress times are very high in comparison with the in-vehicle time. Previous analyses of this same comparison reveal that:

- In Sidney, Australia, out of the transit users that spend less than 15 min in the vehicle, the 75th percentile of walking distances is 775[m] (Daniels and Mulley, 2013), resulting in a 75th percentile of the ratio between walking and total travel times of at least 0.38 (using a walking speed of 5[km/h]), whereas in our simulations this number is 0.08.
- In Santiago, Chile, experiments ran by the local authority (studied by Durán-Hormazábal and Tirachini, 2016) show that the same 75th percentile is 0.18. The maximum value for the same ratio is 0.85, which is slightly lower than the maximum we obtain: 0.92.

3.2.3. Distribution of the results in space

Average values are informative, as they reflect the overall performance of the system. However, we know that the demand is not homogeneously distributed across the network, so the quality of service provided by SMoD might be heterogeneous as well. We now

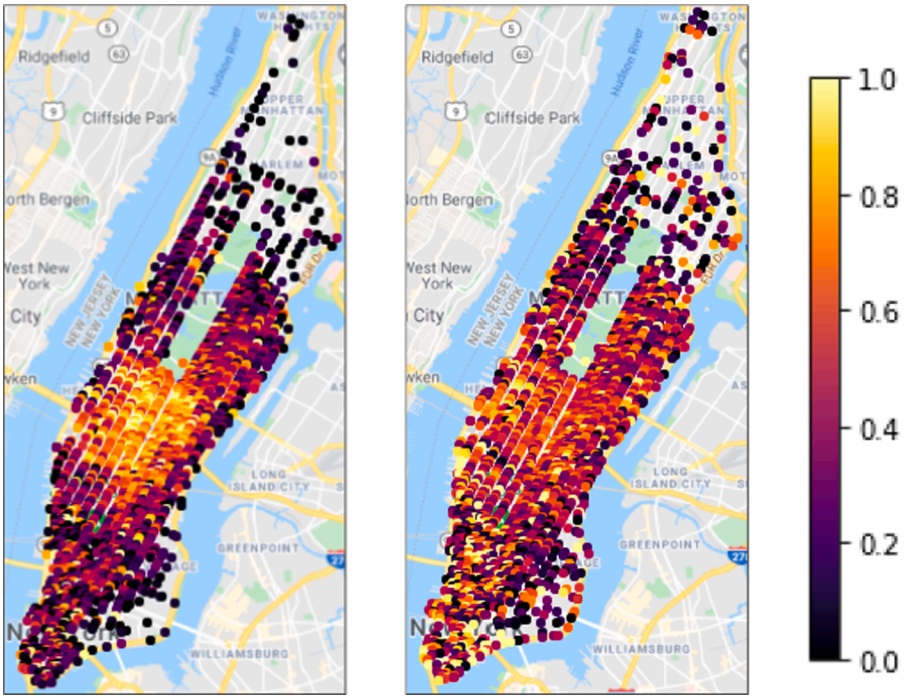

Fig. 9. Relative average waiting for requests departing (left) and arriving (center) at each node. Dark dots represent lower values, as shown in the colorbar (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

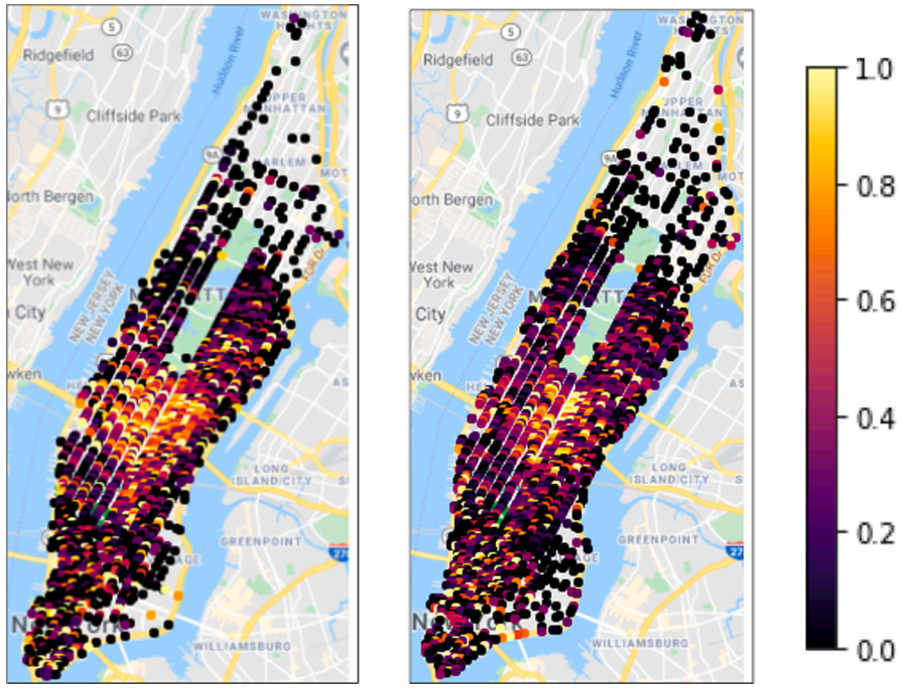

Fig. 10. Relative average walking for requests departing (left) and arriving (center) at each node. Dark dots represent lower values, as shown in the colorbar (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

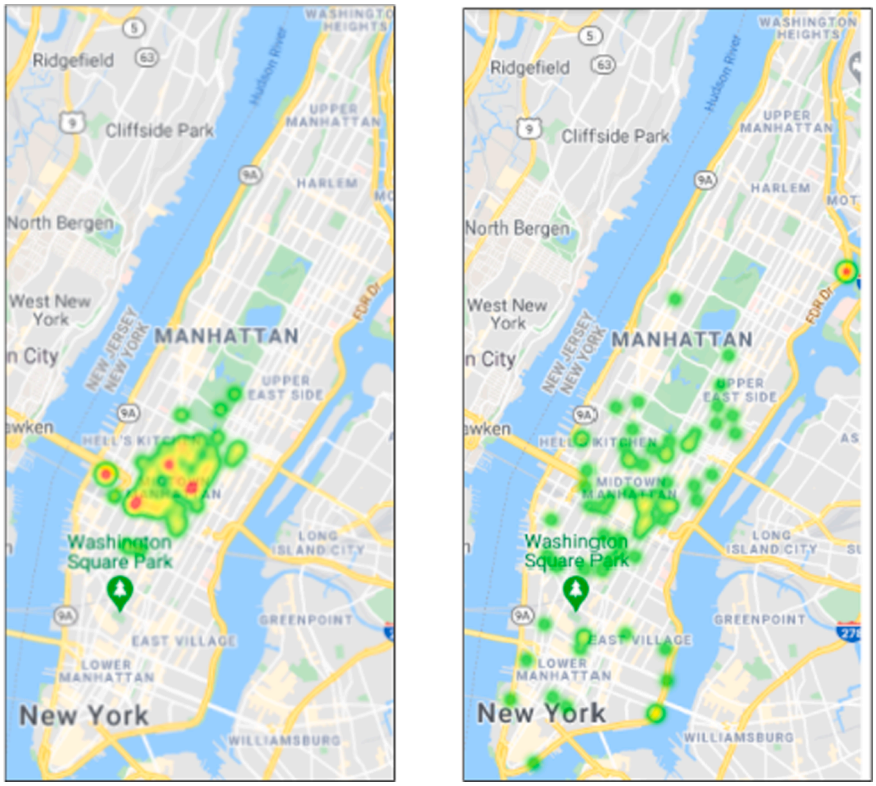

Fig. 11. Departure (left) and arrival (right) of each rejected request. Red represents more concentration. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 7
Correlation between average indices and the distance from the Origin (O) or Destination (D) of the request to Times Square.

Variable	Waiting O	Waiting D	Walking O	Walking D	Delay O	Delay D	Rejections O	Rejections D
Correlation	−0.49	−0.18	−0.32	−0.21	−0.37	−0.16	−0.13	−0.088

proceed to study the spatial distribution of the quality of service. In Figs. 8–11 we show the spatial distribution of the different indices. In order to emphasize the differences in space, for each index we show the relative values, i.e., we focus on depicting where do each index take its maximum, minimum and intermediate numbers (rather than the absolute numbers, already discussed in Table 6). We now explain in detail each of these Figures.

Fig. 8 shows the spatial distribution of total delay, by calculating for each node the average corresponding to the requests departing (Fig. 8 left) or arriving (Fig. 8 right) there. Values are normalized and truncated to the 40th percentile. It is apparent that both images are clearer at the center of the network (specially the left one). This means that requests departing from the most demanded zone present, in general, the highest total delay, which remains true (although softened) for the trips that arrive there. Intuitively, requests that depart or arrive in the center have a higher chance of sharing (which requires detours), but with requests that are nearby (which shortens such detours). Figs. 8 reveal that the first effect dominates. The respective colorbar (also used in the following Figures) is included as well.

Fig. 9 shows the analogous measures regarding waiting time, truncated to the second tercile. It reveals that waiting time depends on the origin but not on the destination, which fits intuition as one waits at the origin only. Requests departing from the center wait longer (on average), which is caused by the higher competition for the vehicles.

Average walking times (truncated to the first quintile) are shown in Fig. 10. They are concentrated towards the center of the network (specially with respect to the origins). As vehicles are more shared in the center, when a user walks there it can benefit (through reduced detours) more co-travellers than in the outer zones. This effect prevails over the longer detours that are expected in the low-demand areas, because it is more difficult to find passengers whose routes are compatible.

Finally, Fig. 11 shows heatmaps of the rejected requests, which is a better representation because they are just a few. Almost all the rejected requests would have departed from the center of the network, whereas their destinations are much more distributed. This suggests that the rebalancing process is not enough to allocate a sufficient number of vehicles to pick-up new passengers in the center, which reinforces the need for researching better rebalancing or predictive techniques (Wallar et al., 2018; Wen et al., 2017; Spieser et al., 2016; Alonso-Mora et al., 2017).

In all, Figs. 8–11 suggest that users departing from or arriving at the most demanded zones from the network expect a worse quality of service. This is verified by Table 7, where we study how nodes' measures change as they move away from the center of the network. To do this, we consider Times Square as the center of Manhattan, we calculate its distance to each node in the network, and we compute the respective linear correlations between such distances and the measures shown in Figs. 8–10 (the average delay, waiting time and walking time for the requests departing/arriving in each node), and also with respect to the percentage of rejected requests. All correlations are negative, meaning that requests that begin or finish close to the center face a worse quality of service, which is particularly true for requests whose origin is there. The fact that rejections present the lowest correlations can be explained because most of the nodes present no rejections at all.

4. Conclusions

Departing from existing methods to assign vehicles to sets of requests in large instances of a SMoD, we have included the chance of walking the first or last legs in order to avoid extra detours that might increase traveling times and the chances of being rejected. We theoretically discussed why walking could improve ridesharing systems, and we showed that doing so might reduce the flexibility of the system to reassign. As the problem becomes much more complex, we proposed some heuristics that allow solving the problem.

After showing that the heuristics are indeed helpful by applying them over a toy network, we applied this method over the real dataset of trips that took place by taxi during one representative hour in Manhattan, and compared the results with and without walking. We showed that short average walks improve the global figures of the system meaningfully, reducing mostly the number of rejections and VHT. We analyzed how results distribute in space, to conclude that passengers whose origins or destinations are located in the most demanded zones tend to have worse quality of service in general; in particular, they are more likely to be requested to walk. Sensitivity analyses over the toy network show that the improvements induced by walking are robust to changes in the most relevant numeric parameters, the demand structure or the network.

There are several relevant avenues to pursue future research. From the algorithmic point of view, herein we have used quite simple heuristics to face the increase in complexity induced by the need of optimizing the PUDO points. Although they were useful to reduce the required time and results were already good, it is possible to design more specific heuristics (for instance, based on the tools that already exist to solve TSP) that might yield even better results. Regarding the transport system itself, these shared systems still need better understanding in a number of topics, like how to determine the optimal fleet of vehicles (which can be directly affected by optimizing the PUDO points) or how does the demand respond to them.

Finally, the relationship between SMoD and public transport is a crucial aspect to investigate for the future of these mobility systems. On the one hand, the system we study here resembles public transport as some users need to walk, so comparing which of them performs better is a relevant question. Doing so is complex[7], because the optimal design of a public transport network is an NP-Hard problem for many specifications, so one usually rests on some model-specific heuristics and simplifying assumptions (Borndörfer et al., 2007; Fielbaum et al., 2018). On the other hand, integrating SMoD and public transport to create a unified transit network is a challenging and promising idea, in which it would be natural to admit walks within the on-demand sub-system.

CRediT authorship contribution statement

Andres Fielbaum: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Xiaoshan Bai:** Formal analysis, Investigation, Methodology, Writing - original draft, Writing - review & editing. **Javier Alonso-Mora:** Conceptualization, Formal analysis, Investigation, Methodology, Writing - original draft, Writing - review & editing, Funding acquisition, Resources, Supervision.

Acknowledgements

We want to thank Víctor Verdugo and Gonzalo Muñoz, both from Universidad de O'Higgins, Chile, for the fruitful discussions about the complexity of this problem, and two anonymous referees whose comments helped to improve this paper. This research was partially funded by Didi Udian Technology (Shenzhen) Co. Ltd.

[7] For instance, (Sörensen et al., 2020; Coutinho et al., 2020) study this comparison based on real-life situations.

Appendix A

Table 8.

Table 8
Glossary of the abbreviations and mathematical symbols used throughout the paper.

Symbol or abbreviation	Meaning
PUDO	"Pick-up and drop-off".
$G = (\mathcal{V}, E)$	The graph representing the streets layout.
$t_v(e), t_w(e)$	Time required to tour arc e over a vehicle or walking, respectively.
$R = \{r_1, \ldots, r_n\}$	Set of current requests.
o_r	Origin of request r.
d_r	Destination of request r.
tr_r	Time when request r emerged.
k_r	Number of passengers in request r.
$V = \{v_1, \ldots, v_m\}$	Set of vehicles.
P_v	Current position of vehicle v.
ν_v	Capacity of vehicle v.
N_v	Set of requests currently being served by vehicle v.
S_v	Set of future stops of vehicle v.
VHT	"Vehicles-Hour-Traveled"
AP	"Assignment Problem"
$\mathcal{A}$	Set of feasible assignments between the sets of requests and vehicles.
R_0	Previous requests that are currently being served by the system.
c_U	Users' costs.
c_O	Operators' costs.
GV	Requests-Groups-Vehicles graph.
T	Set of feasible groups.
t_w	Waiting time
t_v	In-vehicle time
t_a	Access time
$t_h(r, v, i, j, o, d)$	h-time of r if served by v, inserted in the sequence in (i, j), PUDO points (o, d) $(h = w, v, a)$.
$t_h(r, \tau, v)$	h-time of $r \in \tau$ if the group τ is served by v $(h = w, v, a)$.
D	Delay.
Ω_h	Maximum admissible h-time $(h = w, v, a)$.
p_h	Users' perceived cost of one unit of time in state h $(h = w, v, a)$.
Δt_h	Increase in h-time $(h = w, v, a)$.
c_0	Operators' cost of one vehicle moving during one minute of time.
$L(S_v)$	Length of the route followed by v to visit the points in S_v.
$x_{\tau v}$	Binary variable that determines if vehicle v is serving group τ.
z_r	Binary variable that determines if request r is rejected.
p_{KO}	Cost of rejecting a request.
β	Parameter of the "Filtering vehicles" heuristic.
$X(u)$	Set of walking-neighbours of node u
$WD(u)$	Set of nodes at walking distance from node u

References

Agarwal, S., Mani, D., Telang, R., 2019. The impact of ride-hailing services on congestion: Evidence from indian cities. Available at SSRN 3410623 (2019).

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D., 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proc. Natl. Acad. Sci. 114 (3), 462–467.

Alonso-Mora, J., Wallar, A., Rus, D., 2017. Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Sept. 2017), pp. 3583–3590.

Badia, H., Estrada, M., Robusté, F., 2014. Competitive transit network design in cities with radial street patterns. Transport. Res. Part B: Methodol. 59, 161–181.

Basso, L.J., Feres, F., Silva, H.E., 2019. The efficiency of bus rapid transit (BRT) systems: A dynamic congestion approach. Transport. Res. Part B: Methodol. 127, 47–71.

Basso, L.J., Silva, H.E., 2014. Efficiency and substitutability of transit subsidies and other urban transport policies. Am. Econ. J.: Econ. Policy 6 (4), 1–33.

Besser, L.M., Dannenberg, A.L., 2005. Walking to public transit: steps to help meet physical activity recommendations. Am. J. Prevent. Med. 29 (4), 273–280.

Borndörfer, R., Grötschel, M., Pfetsch, M.E., 2007. A column-generation approach to line planning in public transport. Transport. Sci. 41 (1), 123–132.

Čáp, M., Alonso-Mora, J., 2018. Multi-objective analysis of ridesharing in automated mobility-on-demand. In: Proceedings of Robotics: Science and Systems (RSS), 2018, vol. 14.

Cats, O., Vermeulen, A., Warnier, M., van Lint, H., 2020. Modelling growth principles of metropolitan public transport networks. J. Transp. Geogr. 82, 102567.
Chang, S.K., Schonfeld, P.M., 1991. Multiple period optimization of bus transit systems. Transport. Res. Part B: Methodol. 25 (6), 453–478.
Coutinho, F.M., van Oort, N., Christoforou, Z., Alonso-González, M.J., Cats, O., Hoogendoorn, S., 2020. Impacts of replacing a fixed public transport line by a demand responsive transport system: Case study of a rural area in amsterdam. Res. Transport. Econ. 83, 100910.
Cramer, J., Krueger, A.B., 2016. Disruptive change in the taxi business: The case of Uber. Am. Econ. Rev. 106 (5), 177–182.
Daganzo, C.F., 2010. Structure of competitive transit networks. Transport. Res. Part B: Methodol. 44 (4), 434–446.
Daganzo, C.F., Ouyang, Y., 2019. A general model of demand-responsive transportation services: From taxi to ridesharing to dial-a-ride. Transport. Res. Part B: Methodol. 126, 213–224.
Daniels, R., Mulley, C., 2013. Explaining walking distance to public transport: The dominance of public transport supply. J. Transport Land Use 6 (2), 5–20.
Deng, T., Nelson, J.D., 2011. Recent developments in bus rapid transit: a review of the literature. Transp. Rev. 31 (1), 69–96.
Donovan, B., Work, D.B., 2017. Empirically quantifying city-scale transportation system resilience to extreme events. Transport. Res. Part C: Emerg. Technol. 79, 333–346.
Durán-Hormazábal, E., Tirachini, A., 2016. Estimation of travel time variability for cars, buses, metro and door-to-door public transport trips in Santiago, Chile. Res. Transport. Econ. 59, 26–39.
Fagnant, D.J., Kockelman, K.M., 2018. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in Austin, Texas. Transportation 45 (1), 143–158.
Fielbaum, A., 2020. Strategic public transport design using autonomous vehicles and other new technologies. Int. J. Intell. Transport. Syst. Res. 18, 183–191.
Fielbaum, A., 2021. Optimizing a vehicle's route in an on-demand ridesharing system in which users might walk. J. Intell. Transp. Syst.: Technol. Plann. Oper. https://doi.org/10.1080/15472450.2021.1901225.
Fielbaum, A., Alonso-Mora, J., 2020. Unreliability in ridesharing systems: measuring changes in users' times due to new requests. Transport. Res. Part C: Emerg. Technol. 121, 102831 https://doi.org/10.1016/j.trc.2020.102831.
Fielbaum, A., Jara-Diaz, S., Gschwender, A., 2017. A parametric description of cities for the normative analysis of transport systems. Networks Spatial Econ. 17 (2), 343–365.
Fielbaum, A., Jara-Díaz, S., Gschwender, A., 2018. Transit line structures in a general parametric city: The role of heuristics. Transport. Sci. 52 (5), 1092–1105.
Fielbaum, A., Jara-Diaz, S., Gschwender, A., 2020. Beyond the Mohring effect: Scale economies induced by transit lines structures design. Econ. Transport. 22, 100163.
Fielbaum, A., Jara-Diaz, S., Gschwender, A., 2020. Lines spacing and scale economies in the strategic design of transit systems in a parametric city. Res. Transp. Econ.
Gambella, C., Naoum-Sawaya, J., Ghaddar, B., 2018. The vehicle routing problem with floating targets: Formulation and solution approaches. INFORMS J. Comput. 30 (3), 554–569.
Garg, N., Konjevod, G., Ravi, R., 2000. A polylogarithmic approximation algorithm for the group Steiner tree problem. J. Algorithms 37 (1), 66–84.
Henao, A., Marshall, W.E., 2019. The impact of ride-hailing on vehicle miles traveled. Transportation 46 (6), 2173–2194.
Hensher, D.A., Golob, T.F., 2008. Bus rapid transit systems: a comparative assessment. Transportation 35 (4), 501–518.
Hu, M.-B., Jiang, R., Wu, Y.-H., Wang, W.-X., Wu, Q.-S., 2008. Urban traffic from the perspective of dual graph. Eur. Phys. J. B 63 (1), 127–133.
Hurdle, V., 1973. Minimum cost locations for parallel public transit lines. Transport. Sci. 7 (4), 340–350.
Jansson, J.O., 1980. A simple bus line model for optimisation of service frequency and bus size. J. Transp. Econ. Policy 53–80.
Jara-Díaz, S., Fielbaum, A., Gschwender, A., 2017. Optimal fleet size, frequencies and vehicle capacities considering peak and off-peak periods in public transport. Transport. Res. Part A: Policy Pract. 106, 65–74.
Kaan, L., Olinick, E.V., 2013. The vanpool assignment problem: Optimization models and solution algorithms. Comput. Industr. Eng. 66 (1), 24–40.
Kocur, G., Hendrickson, C., 1982. Design of local bus service with demand equilibration. Transport. Sci. 16 (2), 149–170.
Laporte, G., Mercure, H., Nobert, Y., 1987. Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. Discr. Appl. Math. 18 (2), 185–197.
Levinson, H.S., Zimmerman, S., Clinger, J., Gast, J., 2003. Bus rapid transit: Synthesis of case studies. Transp. Res. Rec. 1841 (1), 1–11.
Li, R.-H., Qin, L., Yu, J.X., Mao, R., 2015. Optimal multi-meeting-point route search. IEEE Trans. Knowl. Data Eng. 28 (3), 770–784.
Li, X., Hu, S., Fan, W., Deng, K., 2018. Modeling an enhanced ridesharing system with meet points and time windows. PloS one 13 (5), e0195927.
Figueiredo, L., Amorim, L., 2007 Decoding the urban grid: or why cities are neither trees nor perfect grids. In: Sixth Int. space syntax symposium, 12–15 Jun 2007, Istanbul, Turkey.
Liu, T., Krishnakumari, P., Cats, O., 2019. Exploring demand patterns of a ride-sourcing service using spatial and temporal clustering. In: 2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS) (2019), IEEE, pp. 1–9.
Luo, S., Nie, Y.M., 2019. Impact of ride-pooling on the nature of transit network design. Transport. Res. Part B: Methodol. 129, 175–192.
Martinez, L.M., Viegas, J.M., 2017. Assessing the impacts of deploying a shared self-driving urban mobility system: An agent-based model applied to the city of Lisbon, Portugal. Int. J. Transport. Sci. Technol. 6 (1), 13–27.
Mohring, H., 1972. Optimization and scale economies in urban bus transportation. Am. Econ. Rev. 62 (4), 591–604.
Narayanan, S., Chaniotakis, E., Antoniou, C., 2020. Shared autonomous vehicle services: A comprehensive review. Transport. Res. Part C: Emerg. Technol. 111, 255–293.
Nie, Y.M., 2017. How can the taxi industry survive the tide of ridesourcing? evidence from Shenzhen, China. Transport. Res. Part C: Emerg. Technol. 79, 242–256.
Ota, M., Vo, H., Silva, C., Freire, J., 2016. Stars: Simulating taxi ride sharing at scale. IEEE Trans. Big Data 3 (3), 349–361.
Pei, M., Lin, P., Du, J., Li, X., 2019. Operational design for a real-time flexible transit system considering passenger demand and willingness to pay. IEEE Access 7, 180305–180315.
Porta, S., Crucitti, P., Latora, V., 2006. The network analysis of urban streets: a dual approach. Physica A 369 (2), 853–866.
Raz, R., Safra, S., 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In: STOC (1997), vol. 97, Citeseer, pp. 475–484.
Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S.H., Ratti, C., 2014. Quantifying the benefits of vehicle pooling with shareability networks. Proc. Natl. Acad. Sci. 111 (37), 13290–13294.
Simonetto, A., Monteil, J., Gambella, C., 2019. Real-time city-scale ridesharing via linear assignment problems. Transport. Res. Part C: Emerg. Technol. 101, 208–232.
Slavik, P., 1998. Approximation algorithms for set cover and related problems. Doctoral Thesis. University of New York at Buffalo.
Sörensen, L., Bossert, A., Jokinen, J.-P., Schlüter, J., 2020. How much flexibility does rural public transport need?–implications from a fully flexible drt system. Transp. Policy 100, 5–20.
Spieser, K., Samaranayake, S., Gruel, W., Frazzoli, E., 2016. Shared-vehicle mobility-on-demand systems: a fleet operator's guide to rebalancing empty vehicles. In: Transportation Research Board 95th Annual Meeting (2016), no. 16–5987, Transportation Research Board.
Stiglic, M., Agatz, N., Savelsbergh, M., Gradisar, M., 2015. The benefits of meeting points in ride-sharing systems. Transport. Res. Part B: Methodol. 82, 36–53.
Svensson, O., Tarnawski, J., Végh, L.A., 2018. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 204–213.
Tirachini, A., 2014. The economics and engineering of bus stops: Spacing, design and congestion. Transport. Res. Part A: Policy Pract. 59, 37–57.
Tirachini, A., Gomez-Lobo, A., 2020. Does ride-hailing increase or decrease vehicle kilometers traveled (VKT)? a simulation approach for Santiago de Chile. Int. J. Sustain. Transport. 14 (3), 187–204.
Tirachini, A., Hensher, D.A., Jara-Díaz, S.R., 2010. Comparing operator and users costs of light rail, heavy rail and bus rapid transit over a radial public transport network. Res. Transport. Econ. 29 (1), 231–242.
Tsao, M., Milojevic, D., Ruch, C., Salazar, M., Frazzoli, E., Pavone, M., 2019. Model predictive control of ride-sharing autonomous mobility on demand systems. In: Proc. IEEE Conf. on Robotics and Automation.

van Engelen, M., Cats, O., Post, H., Aardal, K., 2018. Enhancing flexible transport services with demand-anticipatory insertion heuristics. Transport. Res. Part E: Logist. Transport. Rev. 110, 110–121.

Vosooghi, R., Puchinger, J., Jankovic, M., Vouillon, A., 2019. Shared autonomous vehicle simulation and service design. Transport. Res. Part C: Emerg. Technol. 107, 15–33.

Wallar, A., Van Der Zee, M., Alonso-Mora, J., Rus, D., 2018. Vehicle rebalancing for mobility-on-demand systems with ride-sharing. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018), IEEE, pp. 4539–4546.

Wang, H., Yang, H., 2019. Ridesourcing systems: A framework and review. Transport. Res. Part B: Methodol. 129, 122–155.

Wen, J., Zhao, J., Jaillet, P., 2017. Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (2017), IEEE, pp. 220–225.

Zhao, M., Yin, J., An, S., Wang, J., Feng, D., 2018. Ridesharing problem with flexible pickup and delivery locations for app-based transportation service: Mathematical modeling and decomposition methods. J. Adv. Transport. 2018.

Zheng, Y., Li, W., Qiu, F., Wei, H., 2019. The benefits of introducing meeting points into flex-route transit services. Transport. Res. Part C: Emerg. Technol. 106, 98–112.