Traffic Uncertainty in On-Demand High-Capacity Ride-Pooling

Sebastian Hörl, Felix Zwick

Traffic Uncertainty in On-Demand High-Capacity Ride-Pooling

Sebastian Hörl
Institut de Recherche Technologique SystemX
9 avenue de la Vauve, 91120 Palaiseau, France
ORCID number: 0000-0002-9018-432X
sebastian.horl@irt-systemx.fr

Felix Zwick
MOIA GmbH &
Institute for Transport Planning and Systems
ETH Zürich
Stefano-Franscini-Platz 5, 8093 Zürich (Switzerland)
ORCID number: 0000-0002-4067-4486
felix.zwick@ivt.baug.ethz.ch

Word Count: 6274 words + 0 table(s) $\times$ 250 = 6274 words

Submission Date: October 27, 2021

ABSTRACT

On-demand ride-pooling has been studied frequently in recent literature. However, real-world applications of large-scale ride-pooling systems are rare and the breakthrough of comprehensive systems is yet to come. In this study, we simulate a ride-pooling service based on real-world demand data from Hamburg, Germany and evaluate the impact of traffic uncertainty on the system performance using a state-of-the-art ride-pooling algorithm. We propose strategies to remedy customer constraint violations on wait and travel times due to wrongly estimated travel times. We show that uncertainty leads to a trade-off between violating customer constraints and rejecting requests after prior acceptance.

Keywords: ride-sharing, on-demand mobility, MATSim, simulation, uncertainty

INTRODUCTION

On-demand mobility services have been established for many years and play an essential role in urban mobility. Transport Network Companies (TNCs) such as Uber, Lyft or Didi offer convenient and personalized transportation and match drivers and riders via smartphone app.

In recent years, ride-pooling has gained attention in science and practice as it promises to increase system efficiency and vehicle occupancy and to relieve urban traffic. By transporting multiple riders with a similar route in one vehicle, the number of vehicles and vehicle kilometers traveled (VKT) can be drastically reduced (1, 2, 3) compared to on-demand ride-hailing where only one passenger is transported at a time. This way, negative externalities of traffic such as air pollutants, greenhouse gas emissions (4) or noise (1) can be reduced.

While on-demand services are already established today, they are assumed to gain even more attraction with automated vehicles in place, as operational costs are expected to drop substantially (5, 6). As the use of on-demand mobility is expected to grow, so will its impact and pooling potential. Therefore, it is critical to design an efficient system to get the most positive impact on the transportation system. Multiple pooling algorithms have been developed that dispatch passenger requests and vehicles and find matching trips (7, 8). A frequently used algorithm has been developed by Alonso-Mora et al. (3), which we use and adapt in this study.

We implement the pooling algorithm into the multi-agent transport simulation MATSim (9) and use historical demand from the ride-pooling operator MOIA in Hamburg as input for a trip-based simulation. We show the negative impact of traffic uncertainty on the on-demand ride-pooling system in terms of service quality and reliability, which are crucial for customer acceptance in real-world applications.

BACKGROUND

Ride-pooling, often also named *on-demand ride-sharing*, has caught enormous attention in recent years. Multiple pooling algorithms have been developed and applied in simulation studies to show its impact on the transport system (10, 11, 12, 13). Transport simulations are a common tool to mimic real-world mobility systems and evaluate the impact of changes in the system, such as new transport policies or the introduction of a new mobility mode.

Overall, it has been shown that ride-pooling on a large scale has great potential to decrease fleet sizes and VKT in urban areas compared to single-occupied mobility modes such as car or taxi (2, 3, 1). This makes the system so interesting for future mobility applications and policymakers. In reality, however, large-volume ride-sharing services still rarely exist and are mostly limited to small test fleets. Foljanty (14) analyzed the world-wide introduction of new on-demand ride-pooling services and determined an average fleet size of only five vehicles.

There are multiple reasons why the promising scientific results could not yet be transferred to reality on a large scale. One major reason is the lack of automated vehicles, which makes the operation of large ride-pooling fleets costly. Bösch et al. (5) identified the costs of (pooled) taxi operations that drastically decrease with automated vehicles, as the main costs of current taxi fleets are determined by the drivers' salaries. Another challenge of transferring the solutions presented in simulation studies to reality is that simulations always simplify reality and ignore certain aspects such as unexpected user behavior, operational challenges and travel time uncertainty, which highly influence the system. The latter is further explored here.

We focus on the dispatching algorithm developed by Alonso-Mora et al. (3), which has been applied and further developed in a wide range of publications. Ruch et al. (15) compared

the performance of four existing dispatching algorithms in two scenarios and showed that the one developed by Alonso-Mora et al. (3) overall develops best in terms of customer matching rate and VKT reduction. To further improve efficiency of the algorithm, Fielbaum et al. (16) consider dynamic pick-up locations while taking into account the time it takes for customers to walk to the meeting points. Engelhardt et al. (17) explore heuristic approaches for speeding up the customer-vehicle matching process by early filtering of feasible but improbable combinations.

Several studies have looked at dynamic aspects of the algorithm. Dandl et al. (18) study an approach for deciding early whether to accept or reject requests (rather than waiting until a maximum wait time is reached). Fielbaum and Alonso-Mora (19) study in detail the effect of re-assigning customers to new vehicles once the system has accepted their ride, and relate local delays to the average wait time of the system. Furthermore, they emphasize the link between higher rejection rates and increased reliability on planned pick-up and drop-off times if customer-vehicle assignments are binding. The paper lists a number of other sources of uncertainty, one of them uncertainty about travel times, which are examined in the present paper.

The contributions of this paper are the following:

- We implement the algorithm by Alonso-Mora et al. (3) as part of the transport simulation framework MATSim.
- We formally analyze the algorithm in presence of travel time uncertainty and propose strategies to operate it under such conditions.
- We underline the relation of customer constraint violations and rejections under traffic uncertainty, which is highly relevant for real-world applications.

METHODOLOGY

In the following section, we will give an overview of the algorithm introduced by Alonso-Mora et al. (3). While the specifics of the algorithm are presented in the given reference, we will focus on how varying travel times affect the functioning of the approach. Afterwards, we provide mitigation strategies for the problems that arise when considering travel time uncertainty.

Algorithm overview

The algorithm presented by Alonso-Mora et al. (3) traverses multiple steps to perform "any-time optimal" assignments between customers of an on-demand service and its vehicles. The assignments are optimal at a given time, but not necessarily when looked at over a longer time horizon. In every decision epoch (usually every 30 seconds), a request graph is constructed between all requests that have been submitted since the last epoch or which are currently assigned to but not yet picked up by a vehicle. To do so, each request is compared to each other in terms of their latest pick-up time and their latest drop-off time. These are calculated based on a maximum wait time and an allowed detour compared to the unshared ride time, which are configuration parameters of the algorithm. An edge is added to the request graph if two requests can be served by an imaginary vehicle that would be available at the present time, either at the origin of the first request, or at the origin of the second request, while adhering to the pick-up and drop-off constraints of both requests.

The second step of the algorithm makes heavy use of a *routing function*. Given the current location of a vehicle, its on-board requests, and a list of requests that may be served by the vehicle, the *routing function* will return a sequence of pick-up and drop-off tasks for the vehicle which minimizes the travel delay of all involved requests. The travel delay is defined as the difference

between the time at which the customer would have arrived on an unshared trip and the drop-off time that is predicted on the proposed route of the vehicle. In some cases, the routing function will indicate that no feasible route for the given list of requests can be found.

We consider a number of requests referenced by index k with their latest pick-up and drop-off times $t^{\text{latest}}_{k,\text{pickup}}$ and $t^{\text{latest}}_{k,\text{dropoff}}$. Furthermore, for each request, the unshared (direct) arrival time is known as $t_{k,\text{direct}}$. Another important concept is the "active" pick-up time. Initially, after submission of the request, we have $t^{\text{active}}_{k,\text{pickup}} = t^{\text{latest}}_{k,\text{pickup}}$. However, upon first assignment of the request, the pick-up constraint will be set to the time that is expected based on the assigned route. This resembles that the operator communicates the expected pick-up time to the customer and needs to stick to the offer in subsequent decision epochs.

Given a vehicle and a set of requests, whenever the *routing function* is called, it will try to find a sequence of pick-up and drop-off tasks for the vehicle which adheres to a number of constraints. While many approaches are possible to construct these sequences, without loss of generality we follow the approach of Alonso-Mora et al. (3) by performing an exhaustive search for the optimal sequence for vehicles that have up to four passengers on-board, and we perform an insertive search (keeping the order of on-board request tasks invariant) for higher occupancy. Given the network travel times, we can calculate expected pick-up and drop-off times for all involved requests as $\hat{t}_{k,\text{pickup}}$ and $\hat{t}_{k,\text{dropoff}}$. For every tested chain, apart from the capacity constraint, the following conditions need to be met for the sequence to be feasible:

$$\hat{t}_{k,\text{pickup}} \leq t^{\text{active}}_{k,\text{pickup}} \tag{1}$$

$$\hat{t}_{k,\text{dropoff}} \leq t^{\text{latest}}_{k,\text{dropoff}} \tag{2}$$

Note in order for a sequence to be feasible, these constraints need to be fulfilled for both requests that are to be assigned, and those which have already been picked up by the vehicle. Otherwise, their arrival time would be freely moved while integrating new requests.

If the sequence is feasible, the cost is calculated as:

$$C = \sum_k \max\{\ 0,\ \hat{t}_{k,\text{dropoff}} - t_{k,\text{direct}}\ \} \tag{3}$$

Using the routing function, a list of groups, each consisting of a vehicle, a list of assignable requests, and an associated cost (cumulative delay) is constructed. To do so, first, direct vehicle-request matches are evaluated. The routing function is evaluated for each vehicle and each request contained in the request graph. In case a feasible route can be found, the match between the respective vehicle and request is added to the list. Afterwards, all matches between one vehicle and two requests are evaluated. For that, all existing vehicle-request matches are considered, and for each group the request graph is used to find all other requests that may be combined with the already matched request. For each potential match, the routing function is evaluated again to check if a feasible solution exists. If this is the case, the match between a vehicle and two requests is added to the list with the respective cost. The algorithm continues, for each vehicle, by examining the existing matches and by finding feasible combinations of them to construct routes with three, four, and more assignable requests. Note that there is a limit to the size of the groups as longer sequences mean longer travel times which, eventually, will not allow adding more requests due to their constraints, but also due to the capacity of the vehicle which is considered in the *routing*

function. At the end, a list of potential groups with their costs is available.

The list of groups is then passed on to the selection stage, in which an Integer Linear Program (see exact definition in Alonso-Mora et al. (3)) is used to select a set of routes to assign to the vehicles in the current time steps. The problem ensures that:

- Each pending or previously assigned request is present in the selected list of routes *at most* once (otherwise, it would be assigned to multiple vehicles).
- Each vehicle is presented in the selected list of routes *at most* once (otherwise, there are multiple groups assigned to a vehicle).

The objective is to minimize the cumulative cost (delay) of the selected routes. On top, the objective contains a penalty term for every pending request that is not assigned, and a term for every currently assigned request that is not assigned again. These penalties are usually very high to avoid rejecting requests in the first place (*early reject*), but especially after they have been accepted already by the operator (*late reject*):

$$J = \sum_s x_s \cdot C_s + \xi_{\text{early/late}} \cdot (1 - y_k) \tag{4}$$

Here, x_s is a binary variable indicating whether group s is selected, and y_k tells whether request k has been assigned at all.

After solving the ILP, the selected routes are assigned to the vehicles. Vehicles, which are not assigned a route (anymore) are treated differently. As they may still have on-board requests, which need to be dropped off, the *routing function* is used to calculate a new minimum-delay route for the vehicle to drop off the remaining requests. In case no requests are assigned to a vehicle at all, no special steps need to be taken.

After the assignment process, Alonso-Mora et al. (3) propose to consider the requests that could not be assigned in the current epoch and to relocate idling vehicles towards those locations.

Varying travel times

The study presented by Alonso-Mora et al. (3) makes a strong assumption on travel times, i.e. they are not varying over the day. Hence, given an origin location and a destination location, the travel time is assumed to stay constant throughout the simulations. This has a couple of implications:

1. As long as two requests are not rejected or their constraints are violated by the bare progression of time, they remain in the request graph.
2. As long as requests remain in the vehicle graph, they remain in the group list for a vehicle that does not move.
3. A group of requests and its corresponding route that has been assigned to a vehicle will always remain feasible in the following decision epoch, and its cost will stay the same.

We now introduce varying travel times. In this context, the travel time between two distinct points in the network can change between two decision epochs. Such changes may be caused by targeted traffic control or congestion. All three corollaries stated above do not hold in such a setting:

1. If travel times increase between two epochs, two requests that may otherwise have remained as a poolable pair in the request graph, can disappear.
2. If the relevant requests were involved in the assignable group list, a range of groups may disappear in consequence. Note that this poses a filter even before a feasible route has been found for a group.

3. For the remaining groups, the routing function may now find that formerly feasible sequences are now infeasible and hence minimum-delay routes may change.

Note that the first two effects (1 and 2) strongly affect the rejection behavior of the algorithm. While in the context of invariable travel times, rejections happen before assignments, there is now a much higher chance that requests fall out of the request graph and the group list after assignment. This forces the algorithm to perform *late rejections*, which are highly undesirable from the customers' perspective.

The third effect, however, is more severe as it leads the algorithm into an undefined state. As mentioned before, the routing function only tags a route as feasible if the constraints for all assigned and on-board requests of a vehicle are met. While requests that would be picked up late can easily be late-rejected, on-board requests for which their drop-off constraints are violated may lead the algorithm to not finding *any* feasible sequence of drop-off tasks of the remaining requests. This is an undefined condition, as vehicles should not go into idle mode as long as they carry passengers.

The two major challenges when using varying travel times with the algorithm from Alonso-Mora et al. (3) are hence, (1) undefined system states when dropping off customers, and, (2) late rejections.

Solution strategy

To mitigate the problems arising with varying travel times, some exploration work has been performed, which has led to the following proposals.

To overcome the issue of undefined system states, a simple policy is proposed: (A) For the common assignment process, no adjustment is made. Based on the default logic, a vehicle, which cannot satisfy the drop-off constraints of its on-board requests will not appear in any assignable group in the selection stage, and hence, no assignable requests will be attached to it. However, when the "unassigned" route for the vehicle is to be calculated, drop-off constraints are now ignored. Note that the *routing function* still selects the route with the shortest cumulative delay. Hence, the vehicle will aim to drop off the on-board customers as quickly as possible and not engage in picking up any new passengers until they are all dropped off or their constraints can be fulfilled again.

While such a strategy will make the algorithm runnable, it can lead to a high number of late rejections. The second strategy (B) is less restrictive. At the beginning of each decision epoch, we examine the currently assigned routes of all vehicles. Although pick-up and drop-off times may have been planned differently in the previous epoch, we can now calculate an updated expectation of the future drop-off times by traversing the routes and summing up travel times. This allows us to calculate an *updated* drop-off time $t_{k,\text{dropoff}}^{\text{updated}}$ for each request, which is used instead of the latest drop-off time for on-board requests that would otherwise violate their constraints. Effectively, this means that the algorithm is allowed to construct routes in which new pick-up tasks are added to a vehicle as long as the on-board requests are not dropped off later than what is the absolute minimum, given current traffic conditions. To favor feasible solutions, a penalty is added for every second of violating a drop-off constraint.

Finally, (C) the concept is applied analogously to requests that have been assigned to a vehicle and are currently waiting to be picked up. In such a case, we can calculate $t_{k,\text{pickup}}^{\text{updated}}$ based on the current traffic conditions and used it instead of the active pick-up time of the request for all routes which re-assign the request to the vehicle it is currently assigned to. Less formally, this

means that increased traffic will not be a reason for performing a late rejection on the request. However, if another vehicle is able to pick up the request based on its originalactive pick-up time, the reassignment should happen. For that reason, again, a penalty is added to the cost of every route with violations.

Formally, we introduce two new quantities to the algorithm, $t^{\text{required}}_{k,\text{pickup}}$, and $t^{\text{required}}_{k,\text{dropoff}}$ which are calculated depending on the strategy. By default, we have:

$$t^{\text{required}}_{k,\text{pickup}} = t^{\text{active}}_{k,\text{pickup}} \tag{5}$$

$$t^{\text{required}}_{k,\text{dropoff}} = t^{\text{latest}}_{k,\text{dropoff}} \tag{6}$$

For strategy (B), for all requests that are assigned to the currently evaluated vehicle:

$$t^{\text{required}}_{k,\text{dropoff}} = \max\left\{t^{\text{updated}}_{k,\text{dropoff}}, t^{\text{latest}}_{k,\text{dropoff}}\right\} \tag{7}$$

Additionally, for (C), for all requests that are assigned to the currently evaluated vehicle:

$$t^{\text{required}}_{k,\text{pickup}} = \max\left\{t^{\text{updated}}_{k,\text{pickup}}, t^{\text{active}}_{k,\text{pickup}}\right\} \tag{8}$$

The constraints that are evaluated in the *routing function* are then:

$$\hat{t}_{k,\text{pickup}} \leq t^{\text{required}}_{k,\text{pickup}} \tag{9}$$

$$\hat{t}_{k,\text{dropoff}} \leq t^{\text{required}}_{k,\text{dropoff}} \tag{10}$$

The objective is modified such that it contains a penalty term for constraint violations:

$$C' = C + \alpha \cdot \left(\sum_k \max\{\, 0, \hat{t}_{k,\text{pickup}} - t^{\text{active}}_{k,\text{pickup}} \,\} + \sum_k \max\{\, 0, \hat{t}_{k,\text{dropoff}} - t^{\text{latest}}_{k,\text{dropoff}} \,\} \right) + \beta \tag{11}$$

While factor α penalizes the time of violation, β is a constant that is added *if* there have been *any* violations. The latter parameter should be put to a value which clearly distinguishes feasible sequences from violating sequences, such that when the routing function evaluates all possible configurations, it will favor those where no violations take place. Both values should be chosen in a way such that they generally do not exceed the penalties on unassigned requests in the ILP objective described in Equation 4.

Additionally, for (A), we modify the *routing function* such that it ignores drop-off violations when examining a potential route that *only* contains on-board requests.

Finally, note that requests may still be rejected late due to issue (1) noted above. To avoid that requests are rejected due to them disappearing from the request graph, we forcefully construct a potential route based on the requests that are currently assigned to a vehicle and add it, if not already present, to the list of potential routes with its respective cost.

SIMULATION SETUP

The following sections describe a simulation environment in which the strategies mentioned above will be tested. First, detailed on the simulation framework based on MATSim are given, second, the relevant data sources of the use case are introduced, and third, the experimental design is covered.

MATSim implementation

The algorithm by Alonso-Mora et al. (3) has been implemented as a new extension to the multi-agent transport simulation framework MATSim (9)[1]. The framework allows to flexibly combine numerous components around the simulation and analysis of transportation systems. Recently, the framework has been used to study the efficiency of on-demand systems (20, 21, 22).

The algorithm by Alonso-Mora et al. (3) is implemented directly as a replacement for the fleet optimizer in the DRT (Demand Response Transit) package, which is commonly used to simulate on-demand vehicle services in MATSim. Being part of MATSim leads to a range of aspects under which the algorithm has not been operated so far:

- Travel times can vary over the day. Such varying travel times can either be imposed directly on the network links, or they may emerge dynamically from the interactions of all travelling agents.
- Stop times are finite as it takes time for customers to enter or leave the vehicle. Although configurable, by default, a stop time of 60 seconds is used, which is in line with observations from MOIA.
- Vehicles are bound by physical constraints. If a vehicle is currently traversing a network link, it cannot simply perform a U-turn or change route at any time, but only once it arrives at the next network node.

While the first point has been discussed in detail above, the two other points mainly affect the calculation of stop departure and arrival times. Generally, passengers leave the vehicle at the beginning of a stop (i.e. directly after the vehicle has arrived at the destination), while passengers enter the vehicle after the finite stop duration. This duration hence needs to be taken into account when examining pick-up constraints and when calculating times for subsequent stops in a sequence.

The assignment based on (3) is executed in fixed intervals, here 30 seconds. Between these decision epochs, all incoming requests are collected in a queue and then processed jointly in the assignment step. Note that, theoretically, when varying travel times are used, we would need to route a large number of movements in the fleet to construct the group lists and estimate pick-up and drop-off times to verify the constraints. This quickly leads to millions of routing tasks per decision epoch and, hence, renders the algorithm computationally heavy. To reduce some of the computational load, we keep a cache of origin-destination relations. Once a travel time between an origin and a destination link is needed, it is first checked whether travel time information for this relation is already in the cache. If so, the respective value is returned. Along with the calculated travel time, we keep the last timestamp in simulation time at which the relation was routed. If the lifetime of a relation has exceeded a specific threshold, it is discarded and re-routed. It is hence possible to either have a rough approximation of the travel time (if the lifetime is high) or very detailed information (if the lifetime is low).

[1]The code will be contributed to the open source code base of MATSim.

Data preparation

For the simulation, we use the stop network and demand data from Europe's largest ride-pooling provider MOIA in Hamburg, Germany. MOIA operates with up to 500 vehicles in a 300 km^2 service area covering most populated areas of the city shown in Figure 1. Although the input data reflects the real-world service, it should be noted that the ride-pooling simulation, the used algorithms and the results only remotely resemble MOIA's real-world operation.

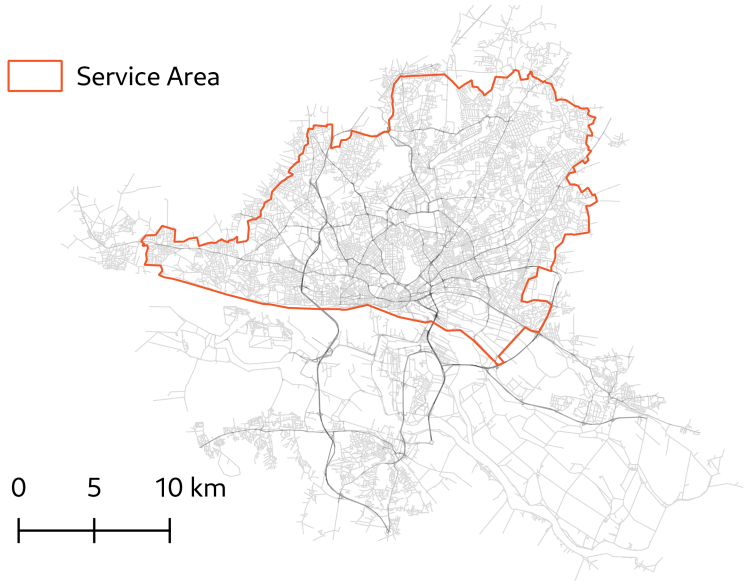

FIGURE 1: Study and service area for Hamburg.

The street network shown in Figure 1 is based on OpenStreetMap[2] data. MOIA's more than 10,000 virtual pick-up and drop-off stops are matched on it. We do not simulate transport modes other than ride-pooling and therefore do not observe congestion from external traffic in the system. To obtain realistic travel times that vary throughout the day, we use GPS-based speed data of all weekdays in November 2019 from TomTom and match it to the MATSim network with the help of a map-matching algorithm described by Yang and Gidófalvi (23). The network speeds are updated every 60 minutes.

We use 24,032 historical requests from MOIA, which have been collected on 4 typical weekend days between 19/09/2020 and 10/10/2020. A sample of one fourth of each day's requests is used to avoid outlying extreme demand scenarios of a single day. The requests are combined and assumed to occur on the same simulated day. Requests from the same person within 30 minutes after the first request are ignored to avoid unrealistically clustered requests.

Experimental design

To test the strategies presented above in the MATSim environment, two major experiments are performed.

First, we construct a synthetic test case. We impose (uncongested) free flow travel times that stay constant over the day and use 200 vehicles to serve the demand. In a series of tests, we

[2] `www.openstreetmap.org`

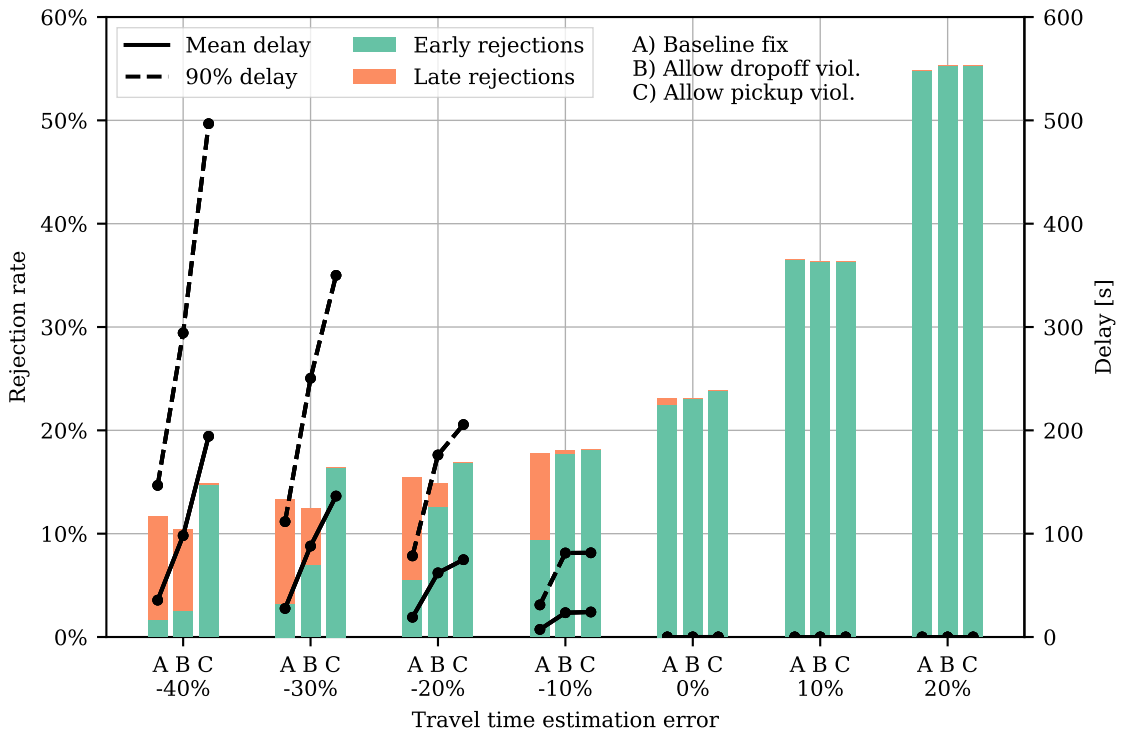

FIGURE 2: Simulation experiments with systematic estimation errors.

introduce a systematic bias to the travel times used by the algorithm in contrary to how the vehicle movements will be simulated. Travel times are scaled from -40% (underestimating arrival times, overestimating speeds) to 20% (overestimating arrival times, underestimating speeds). The goal is to see both the impacts of the estimation error for each strategy on the rejection rate (early and late), and on the additional delays beyond the latest arrival time of the requests.

In a second case, realistic travel times are imposed on the network and varied by time of day, as described above. The demand for Hamburg is served by 300 vehicles. The goals of these simulations is to see the impact of our proposed algorithm modifications in a realistic setting, based on real-world data. Uncertainty on travel times in this case stems from the varying travel times and the fact that a route calculated at one time during the simulation may overestimate or underestimate the travel time at a later time. Furthermore, we vary the routing interval (or "lifetime" of calculated routes) to understand its impact on system performance.

RESULTS

The following sections describe the simulation results for the synthetic and realistic cases.

Systematic changes

First, the cases with systematic estimation errors of travel time are examined. In Figure 2 we show a range of estimation errors (from -40% to +20%). For each estimation error, we show the share of rejections compared to all submitted requests and divide them in early rejections and late rejections. This is done for the three different strategies described above. On top, the mean and the 90% quantile of the distribution of delays of all served requests are shown. The delays quantify by

how much time requests are dropped off after their latest drop-off time constraint.

Starting at an exact estimation of travel times, Figure 2 shows that the system has a rejection rate of about 22%. If travel times are overestimated (right side of the plot), the rejection rate increases. This is due to the algorithm being too careful and calculating with too much of slack. Interestingly, an underestimation of travel times (left side) leads to a reduced number of rejections. This indicates that there is a general slack in the system. At most times when estimating travel times exactly, stop sequences for the vehicles have still room for inserting more requests, which are, however, not available due to the density and travel characteristics in the scenario. These capacities, however, are used when travel times are underestimated, because requests need to be reassigned frequently to ensure the promised level of service.

However, note that especially for case A, which is the minimum mitigation strategy to run the algorithm with varying travel times, almost all rejections are late rejections. This is neither desirable for the operator, nor for the customer. Likewise, as underestimation becomes more severe, the observed delays increase. For -40% estimation error, about 20 seconds delay can be observed on average, with 10% of trips having a delay of more than two minutes.

In terms of the three strategies, one can see that they have a strong impact on the system behavior. While, interestingly, the overall rejection rate is barely effected given a certain level of estimation error, the strategies have a clear effect on the ratio between early and late rejections. Allowing violations of drop-off constraints for the assigned vehicle (B) varies in impact depending on the estimation error. For -40% its introduction has only little effect compared to (A), while for lower estimation errors (e.g., -10%) it manages to avoid late rejections almost entirely. For more severe estimation errors, the avoidance of late rejections is only guaranteed by strategy (C) in which once assigned requests will always be served by their initially assigned vehicle. Note that for all mitigation strategies, reducing the number of late rejections is bought by the generation of additional delays, which can reach over 5 minutes if strategy (C) is used.

Realistic system

Clearly, the results from above are based on constructed scenarios to emphasize the effect of the mitigation strategies. It is hence interesting to have a look at a more realistic scenario. Figure 3 shows the results for the service with 300 vehicles and realistic travel times. As our aim is to avoid late rejections, only the simple case (A) is shown, along with allowing all violations for assigned vehicles (C). On the bottom axis, we vary between three different routing cache intervals, from rather detailed (15 minutes) to quite sparse (2 hours).

The results show that the rejection rate stays rather stable around 32% in all examined cases. However, the routing interval has an impact on the number of late rejections. As routing information becomes more rough, their number is increased. In all cases, introduction of strategy (C) reduces these late rejections. Interestingly, there is only little delay in the realistic scenario, where, even in the worst case, its mean value stays below 1 second on average.

DISCUSSION

From the results above, it becomes evident that one must be careful when applying the algorithm of Alonso-Mora et al. (3) to scenarios with varying travel times. To avoid undefined system states, measures must be taken, but the simplest approach (making sure that vehicles drop off late requests before taking on any additional task) can lead to a high rate of late rejections. The presented advanced strategies are able to mitigate this problem.

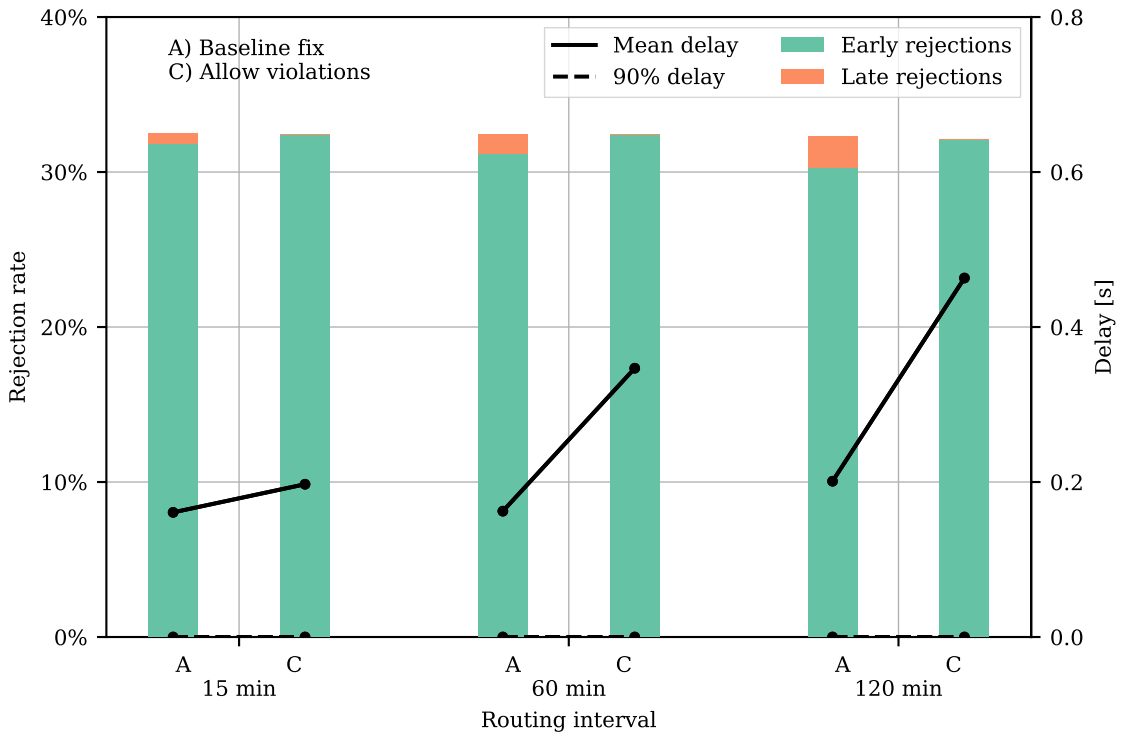

FIGURE 3: Simulation experiments with realistic demand and travel times.

Looking at the results from Figure 2, we can state that the strategies "buy" a better performance in terms of late rejections by "paying" with customer delays. And otherwise put, late rejections or delays are the "price" of having inaccurate estimates of travel time. Note that here we look at static demand simulations, but in reality excessive delays or late rejections would have a potentially strong impact on the attraction of the system and customer satisfaction. These findings are in line with (19) where (in the absence of travel time uncertainty) violations against the first pick-up time provided to the user, as well as binding customer-vehicle matching, are examined. Similar to our case, the paper's authors find that enforcing constraints on the stability of arrival times towards the user comes with the direct effect of increased rejection rates.

However, looking at Figure 2 the impact of delays seems low. First, it should be noted that the system does not seem to run at its capacity limit and results may differ if a more constrained system was simulated. This is a task that should be tackled in future research. Second, the simulations in Figure 2 are valuable as they tell us about the robustness of the algorithm towards travel times in a realistic context. While, here, the routing interval was varied, and we see that with fewer routing, system performance stays nearly constant, the results open up space for even further simplifications. For instance, instead of performing detailed routing, zone-based travel time matrices could be used in the future. These insights pose a large potential in speeding up simulation experiments which make use of the algorithm.

The experiments in this paper make use of extreme cases. Either, travel times are affected strongly by a fixed bias, or they are minor as they are only slightly affected by varying travel times along the day. Note that we do not cover an entirely realistic case, in which information on the variability of travel times at any point in time is known. We assume that operators would

work with upper bound estimates to provide a robust system, which, however, could be tested in simulation. The case is different for sudden shocks to the system, such as accidents. Both topics relate strongly to the robustness of the ride-pooling system and should be covered in future research. The integration of the algorithm into MATSim makes it possible to test it in a full-scale microscopic traffic simulation in which such phenomena can be replicated in detail.

CONCLUSION

To conclude, we examine the problem of travel time uncertainty in combination with the algorithm by Alonso-Mora et al. (3) which has been frequently used in literature. However, it has not been used in the context of uncertain travel times, which is a major challenge for real-world ride-pooling systems. We show that the original algorithm is undefined in certain situations when travel times have been estimated too optimistically.

Our results show that a trade-off arises necessarily when strategies are proposed to remedy the problem. If travel times are underestimated, the operator either must reject requests after they have been accepted, or allow violations of the latest pick-up and drop-off times that have been communicated to the customer.

However, we show that effects are minor in a scenario with realistic travel times, and we point out that there is a potential in time-slice based aggregation of travel times for speeding up simulation performance.

REFERENCES

1. Felix Zwick, Nico Kuehnel, Rolf Moeckel, and Kay W. Axhausen. Agent-based simulation of city-wide autonomous ride-pooling and the impact on traffic noise. *Transportation Research Part D: Transport and Environment*, 90:102673, 2021.
2. Luis M. Martinez and José Manuel Viegas. Assessing the impacts of deploying a shared self-driving urban mobility system: An agent-based model applied to the city of Lisbon, Portugal. *International Journal of Transportation Science and Technology*, 6(1):13–27, 2017.
3. Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences of the United States of America*, 114(3): 462–467, 2017.
4. Jeffery B. Greenblatt and Samveg Saxena. Autonomous taxis could greatly reduce greenhouse-gas emissions of US light-duty vehicles. *Nature Climate Change*, 5(9):860–863, 2015.
5. Patrick M. Bösch, Felix Becker, Henrik Becker, and Kay W. Axhausen. Cost-based analysis of autonomous mobility services. *Transport Policy*, 64:76–91, 2018.
6. Henrik Becker, Felix Becker, Ryosuke Abe, Shlomo Bekhor, Prawira F. Belgiawan, Junia Compostella, Emilio Frazzoli, Lewis M. Fulton, Davi Guggisberg Bicudo, Krishna Murthy Gurumurthy, David A. Hensher, Johan W. Joubert, Kara M. Kockelman, Lars Kröger, Scott Le Vine, Jai Malik, Katarzyna Marczuk, Reza Ashari Nasution, Jeppe Rich, Andrea Papu Carrone, Danqi Shen, Yoram Shiftan, Alejandro Tirachini, Yale Z. Wong, Mengmeng Zhang, Patrick M. Bösch, and Kay W. Axhausen. Impact of vehicle automation and electric propulsion on production costs for mobility services worldwide. *Transportation Research Part A: Policy and Practice*, 138:105–126, 2020.

7. Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Proceedings - International Conference on Data Engineering*, pages 410–421, 2013.
8. Joschka Bischoff, Michal Maciejewski, and Kai Nagel. City-wide shared taxis: A simulation study in Berlin. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2017.
9. Andreas Horni, Kai Nagel, and Kay W Axhausen. *The multi-agent transport simulation MATSim*. Ubiquity Press, London, 2016.
10. S. Hörl, C. Ruch, F. Becker, E. Frazzoli, and K.W. Axhausen. Fleet operational policies for automated mobility: A simulation assessment for Zurich. *Transportation Research Part C: Emerging Technologies*, 102:20–31, 2019.
11. Abood Mourad, Jakob Puchinger, and Chengbin Chu. A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological*, 123: 323–346, 2019.
12. Anna Pernestål and Ida Kristoffersson. Effects of driverless vehicles : Comparing simulations to get a broader picture. *European Journal of Transport and Infrastructure Research*, 1(19):1–23, 2019.
13. Peng Jing, Hanbin Hu, Fengping Zhan, Yuexia Chen, and Yuji Shi. Agent-Based Simulation of Autonomous Vehicles: A Systematic Literature Review. *IEEE Access*, 8:79089–79103, 2020.
14. Lukas Foljanty. On-Demand Ridepooling Market: 2020 Recap. 2020.
15. Claudio Ruch, ChengQi Lu, Lukas Sieber, and Emilio Frazzoli. Quantifying the Efficiency of Ride Sharing. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–6, 2020.
16. Andres Fielbaum, Xiaoshan Bai, and Javier Alonso-Mora. On-demand ridesharing with optimized pick-up and drop-off walking locations. *Transportation Research Part C: Emerging Technologies*, 126:103061, 2021.
17. Roman Engelhardt, Florian Dandl, and Klaus Bogenberger. Speed-up Heuristic for an On-Demand Ride-Pooling Algorithm. *arXiv:2007.14877 [cs, eess, math]*, 2020.
18. Florian Dandl, Roman Engelhardt, and Klaus Bogenberger. On the Dynamism of User Rejections in Mobility-on-Demand Systems. *arXiv:2104.01326 [cs, eess]*, 2021.
19. Andrés Fielbaum and Javier Alonso-Mora. Unreliability in ridesharing systems: Measuring changes in users' times due to new requests. *Transportation Research Part C: Emerging Technologies*, 121:102831, 2020.
20. Ihab Kaddoura, Joschka Bischoff, and Kai Nagel. Towards welfare optimal operation of innovative mobility concepts: External cost pricing in a world of shared autonomous vehicles. *Transportation Research Part A: Policy and Practice*, 136:48–63, 2020.
21. Reza Vosooghi, Jakob Puchinger, Marija Jankovic, and Anthony Vouillon. Shared autonomous vehicle simulation and service design. *Transportation Research Part C: Emerging Technologies*, 107:15–33, 2019.
22. Sebastian Hörl, Felix Becker, and Kay W. Axhausen. Simulation of price, customer behaviour and system impact for a cost-covering automated taxi system in Zurich. *Transportation Research Part C: Emerging Technologies*, 123:102974, 2021.

23. Can Yang and Győző Gidófalvi. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *International Journal of Geographical Information Science*, 32(3):547–570, 2018.