Extending the DRT module to enable simulations of pre-booked MoD services

Tarek Chouaki, Sebastian Hörl

June 2023

1 Introduction

Mobility-on-Demand (MoD) services have been extensively investigated in the recent years. A MoD service consists in a fleet of vehicles that can be requested by travellers to perform trips. The design of a MoD service involves various strategic, tactical and operational decisions. Many of these questions are investigated using agent-based modelling and simulations. One prominent tool is MATSim [3] as its flexibility and modularity allows implementing various type of services and its model allows assessing the impact of different services on user choices [2, 4].

The simulation of MoD systems in MATSim is enabled by the DRT [1] and DVRP [5] modules which provide fleet management functionalities. Mainly, a vehicle assignment algorithm matches passenger travel requests to vehicles. An insertion heuristic is used to find the best way to insert a trip into a vehicle's schedule with the goal to minimize arrival time for the passenger without delaying the previously assigned requests. In the current implementation, travel requests are handled only in an online manner, with passengers that are able and willing to be picked up as soon as possible.

In this work, we extend the DRT and DVRP modules in order to support prebooking. In this setting, MoD trips can be booked and planned in advance.

2 Approach

The task at hand involves making methodological changes to the implementation in MATSim.

First, the algorithm, as soon as a new request arrives, determines where to insert it into the existing vehicle schedules. Figure 1 shows an example of such a schedule that already contains a dropoff task (D1, the passenger is on board) and another pickup and dropoff pair (P2, D2). The first step of the algorithm is to determine viable insertion locations, which are either at the beginning of the schedule (i.e., immediately) or after any other pickup or dropoff.

Example (a) in Figure 1 shows the standard case (without prebooking). For the sake of simplicity, we don't consider finite stop times of the pickup and dropoff tasks here. Each pickup has a latest time at which it should happen (departure time plus maximum wait time) and each dropoff has a latest arrival time that is based on the direct travel time and a reasonable detour factor. Hence, the tasks cannot be moved beyond a certain limit unless deteriorating the service level. In Figure 1 the current planned times of the tasks are indicated as black dots, while the available *slack* to the upper limit is indicated as a dashed blue line. In consequence, these blue lines indicate the time span that a task can be moved forward in time when inserting a new request without violating the constraints. They, hence, also indicate the potential time interval in which a pickup or dropoff can

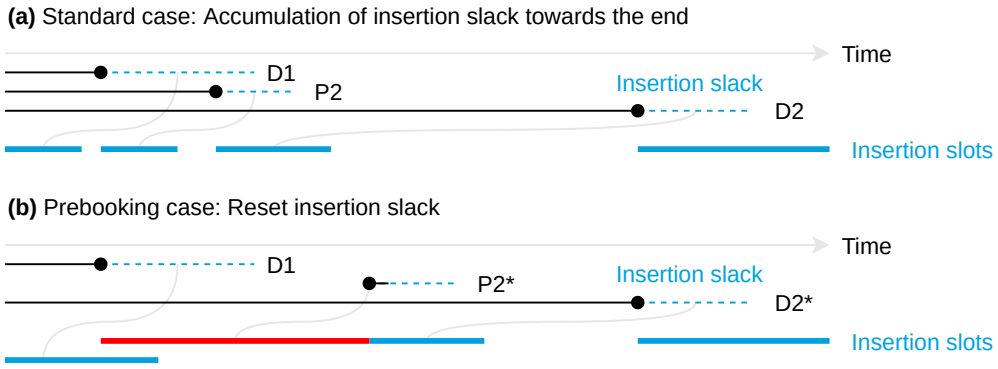

Figure 1: Insertion slack in the standard and prebooking cases

be inserted *before* a specific task (solid blue line). Note that the slack is monotonically increasing in the standard case because, for instance, the slack before D1 is limited by the slack of P2. Otherwise, one could move forward D1 but at the same time violate P2's constraint.

The situation is different in the prebooking case. Figure 1b shows P2* as a prebooked request with an additional *earliest departure time*. While this task also has an allowed pickup interval, the slack before the task is much larger as there is idle time between D1 and P2*. Hence, the slack is *reset* to the difference between the previous task's (D1's) currently planned end time and the *earliest start time*. Note that in the prebooked case, the slack is *not* monotonically increasing with time.

Without this change, the algorithm would not find the majority of viable insertion slots before prebooked requests. These slots are then used to insert the drive times to, between, and from the new pickup and dropoff locations.

Second, the evaluated insertions need to be inserted into the DVRP vehicle schedules. They are constructed by *Drive*, *Stop*, and *Stay* tasks. By default, since requests are immediate, the schedules are a sequence of *Stop* tasks that represent pickup and dropoff activities of the vehicle. Only after the last activity, the schedule is filled up with a *Stay* task where the vehicle idles as long as no new requests are inserted (Figure 2). *Stop* tasks have a fixed duration and once started, they can not be cancelled. The vehicle will wait infinitely for the passenger to arrive (which, however, is already present in the standard case).

In the prebooking case, pickup activities can be inserted *in advance*. This means that a vehicle drops off a passenger at 9:00, then drives to a pickup that has been prebooked for 10:30, but it should not start the blocking *Stop* task immediately. Instead, we need to insert a new *Wait* task that lets the vehicle idle (Figure 2b). Two options are possible: Either the vehicle drives to the pickup location and idles there, or it idles at the current location and leaves at the right time to arrive at the requested pickup time. The first case minimizes the potential delay for the customer, while the second option bears the risk of getting stuck in traffic and arriving late. However, the first option potentially reduces the overall drive distance of the fleet and thus the operating cost. Our implementation will allow benchmarking these two configurations.

Third, the stop simulation logic must be extended. In the standard configuration, customers are already present at the time the vehicle arrives to pick them up. MATSim defines a fixed stop time (for instance 60 seconds) for each *Stop* activity, regardless of the passenger count (see Figure 3a). It is triggered once the vehicle arrives.

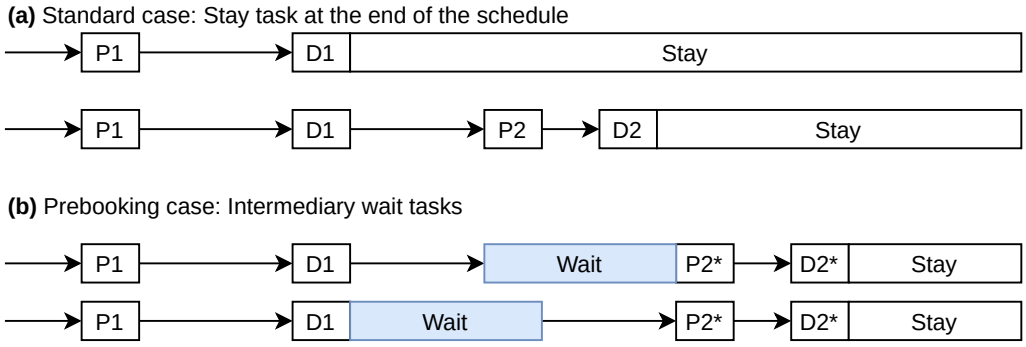

Figure 2: Vehicle schedules without and with prebooking

Introducing prebooking functionality requires an extension to the logic, but our goal is to include the current approach for immediate requests as a special case. Figure 3 shows the other case (b) that ca arise where the vehicle arrives in advance. After performing the *Wait* task, the vehicle may start the *Stop* task according to schedule while still waiting for the passenger. According to the standard logic, the 60s would start at this point and the vehicle would depart immediately after the customer's arrival, which is not the desired behaviour. The situation gets even more complex if there are multiple customers with different time properties. Example (c) shows two agents that arrive at different (prebooked) times at the same stop for the same vehicle.

We define an extended logic based on two rules:

- Every customer must spend 60 seconds (configurable) to be picked up by a vehicle.
- A vehicle leaves a stop once all customers have been picked up.

This logic covers the standard case and creates a timing scheme as shown in Figure 2. In example (c) the first passenger arrives, spends 60s and enters the vehicle. Partly in parallel, the second passenger arrives, spends also 60s and enters. Finally, the vehicle leaves with both customers on board. The overall stop duration of the vehicle is, hence, dependent on multiple requests, but its departure time can be calculated at any point in time as

$$T = s + \max_i \left\{ \max \left(t^p_{\min,i},\ t^a_i \right) \right\} \tag{1}$$

with $t^p_{\min,i}$ indicating the earliest possible departure time of request i, s the stop duration, and t^a_i is set to the time the agent has arrived at the stop or zero if it is not there yet.

3 Implementation and benchmark

On the technical side, our goal is to integrate prebooking in the DRT and DVRP modules in the least invasive way possible and by using the existing functionalities. Among the affected classes of the codebase, most will be subject only to minor changes.

The changes proposed above are in an experimental state and have partly been tested in ongoing projects. In the coming months, they will be backed up by appropriate unit tests to ensure stability of the code base.

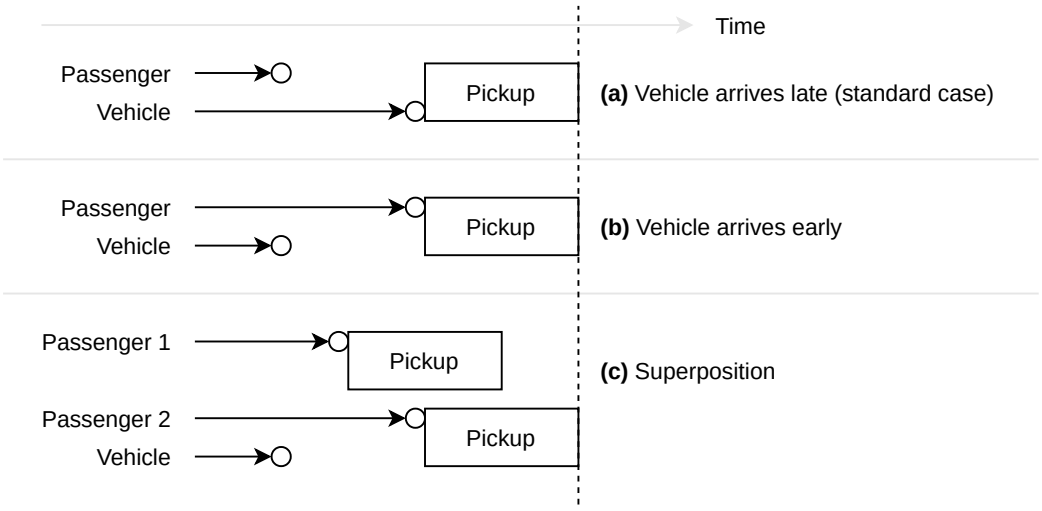

Figure 3: Stop logic without and with prebooking

Based on the proposed changes above, we are currently in the process of consolidating the code and preparing a benchmark study on existing DRT simulations for MATSim. Such a benchmark will reveal impacts on waiting times, total travel times, customer delays and other dimensions based on a number of parameters:

- *Wait* tasks: Idle at current location or drive to customer?
- Prebooking horizon: How much in advance are requests considered for insertion? All at the beginning of the day / a certain time before the end of an ongoing activity?
- Percentage of requests that are submitted as prebooked

4 Conclusion

While the first simulation results have been promising, a comprehensive analysis will be presented at the MATSim User Meeting. At the same time, we plan to present the technical contribution and how to make use of the new features.

References

[1] City-wide shared taxis: A simulation study in Berlin.

[2] T. Chouaki, S. Hörl, and J. Puchinger. Towards Reproducible Simulations of the Grand Paris Express and On-Demand Feeder Services. In *102nd Annual Meeting of the Transportation Research Board (TRB 2023)*, Washington D.C, United States, Jan. 2023.

[3] A. Horni, K. Nagel, and K. Axhausen. *The Multi-Agent Transport Simulation MATSim.* Apr. 2016.

[4] S. Hörl, C. Ruch, F. Becker, E. Frazzoli, and K. Axhausen. Fleet operational policies for automated mobility: A simulation assessment for zurich. 102:20–31, 2019.

[5] M. Maciejewski, J. Bischoff, S. Hörl, and K. Nagel. Towards a testbed for dynamic vehicle routing algorithms. In *Highlights of Practical Applications of Cyber-Physical Multi-Agent Systems: International Workshops of PAAMS 2017, Porto, Portugal, June 21-23, 2017, Proceedings 15*, pages 69–79. Springer, 2017.