SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.100

Published: 29 Jun. 2022

Simulation of Demand Responsive Transport using a dynamic scheduling tool with SUMO

Maria Giuliana Armellini

German Aerospace Center (DLR), Berlin, Germany
maria.armellini@dlr.de

Abstract

Demand responsive transport (DRT) has been increasingly tested and applied in recent years as a new form of transportation that seeks to address mobility problems in cities and rural areas. The planning of DRT systems is a challenging task for transport planners since the performance of the service depends significantly on the demand, how the scheduling is made, and how the routes are computed. Transport simulations are a useful option to evaluate these systems. The paper presents a Python tool, which aims to simulate di-verse DRT services using the software package for microscopic simulations Eclipse SUMO (Simulation of Urban MObility) as a framework. The fleet and requests of the DRT are handled dynamically by the scheduling module of the tool. This module is also responsible for calling a solver algorithm for the Dial-a-Ride-Problem (DARP), processing its results, and dispatching the DRT vehicles according to them. The tool also enables easier imple-mentation of other methods to solve the DARP. To demonstrate the use of the tool, a DRT service operating in two central neighborhoods of the city of Brunswick (Germany) is presented. The tool is called *drtOnline.py* and is included in SUMO since version 1.9.0.

Contents

1 Introduction

Demand responsive transport (DRT) has been increasingly tested and applied in recent years as a new form of transportation that seeks to address mobility problems in cities and rural areas. DRT seeks to serve trip requests from passengers using a fleet of vehicles without fixed routes. Depending on how the system operates and which constraints are considered, different types of DRT services can be given. For example, DRT can operate as a shared system, allowing the passengers to share their trips. DRT has also been studied to support public transport, being used as a transportation mode for the first or last mile. Depending on how the requests are made and managed by the service, the DRT system can be online (requests arrive and are managed in real-time), offline (requests are known in advance), or a mix of both [9].

Regarding the different forms of the DRT services, its planning is a demanding task for transport planners. The use of micro-simulations, like the open-source Eclipse SUMO (Simulation of Urban MObility) [2], constitutes an important tool for this aim. Since version 1.5.0 SUMO supports the simulation of demand responsive transport (DRT) via the taxi device. Vehicles with this device can receive trip reservations from persons in the simulation. The requests are processed by a dispatch algorithm that manages the route of each vehicle to serve the larger amount of requests under certain constraints. This problem statement is referred to in the literature as the Dial-a-Ride-Problem (DARP). SUMO counts with four different dispatch algorithms (greedy, greedyClosest, greedyShared, and routeExtension). However, due to the vast diversity of DRT services, these are sometimes not sufficient. The user has then the option to implement their dispatch algorithms using the TraCI API.

The taxi device is under continuous development. Since this year in version 1.9.0, the TraCI API enables the re-dispatching of taxis or DRT vehicles. This was an important step since it allows simulating shared DRT with a dynamic dispatcher. If the dispatcher finds a better route that can serve more requests, the vehicles can now change their route while driving.

In the literature, different methods to solve the DARP have been investigated. In [5] a study of different models and algorithms used in the literature to solve the DARP is presented. For large scenarios, finding an exact solution for the DARP can lead to long calculation times. For these cases, approximate solution methods, like metaheuristics are advisable. In [8], three different metaheuristics (Adaptive Large Neighborhood Search, Hybrid Bees Algorithm with Simulated Annealing, and Hybrid Bees Algorithm with Deterministic Annealing) were compared to solve a multi-depot and multi-trip heterogeneous DARP. [6] applies a multi-objective optimization to solve the DARP with time windows, whereas [7] implements an online rejected-reinsertion heuristics to solve a multi-vehicle DARP. Finally, other heuristics methods to solve the static and dynamic DARP are presented in [10]. The performance results of the DRT can depend significantly on the method used to solve the DARP, for which its election plays an important role.

To simulate DRT not only a DARP solver is needed, but also a scheduling module for the management of the fleet and reservations.

Based on this, a Python tool to simulate different types of DRT in SUMO was developed. The tool uses the taxi device and TraCI to implement a scheduling module that calls a DARP solver to simulate DRT services. The same scheduling module can be used with different DARP solvers to simulate different DRT services. It also allows users to implement their dispatching algorithms more easily and faster, and to compare their results with other methods. Since the dispatcher algorithms that are currently included in SUMO are not well suited to simulate shared DRT modes, a DARP solver to simulate these services is implemented as a first option.

In the next section, the tool and its modules are explained. After this, a user example is shown and finally, the conclusion and further work are presented.

2 Methodology

The aim of the proposed Python tool is not only to simulate shared DRT services but to allow the users to compare the performance of different DARP solvers and to enable easier implementation of its solvers. The tool was written in Python3 and uses the TraCI API to control the DRT requests and vehicles in real-time.

The tool consists of two modules: the scheduling module and the DARP solver. The inputs are the standard SUMO files. The tool needs at least information about the DRT vehicles, which are the ones equipped with a taxi device, and information about the requests, which are

modeled as persons having a ride element with the DRT lines. The scheduling module starts the simulation via TraCI and manages the DRT requests and fleet. This means that the module detects when a new request arrives or requests are waiting to be served and it calls the DARP solver to find the best route for each DRT vehicle. What makes a route the best, depends on the model of the DARP solver. The solver returns the routes to the scheduling module and dispatches the DRT vehicles with them. The tool has with a default DARP solver, which allows it to simulate shared DRT services (persons can share a trip).

The tool also accepts adding the surrounding traffic and other SUMO elements to the simulation. This allows obtaining more realistic results for the performance of the DRT service.

In the following sections, the scheduling module and the DARP solver of the tool are explained in detail.

2.1 Scheduling module

Figure 1 shows a flowchart of the scheduling module. The first step is to call TraCI with the given inputs. The required inputs are the SUMO network, and the DRT vehicles and requests. As mentioned before, other SUMO inputs (e.g. surrounding traffic) can be given. The tool also supports the use of SUMO *configuration-files* to manage the simulation options.

Once TraCI has been called, the simulation starts. For each simulation step, the tool checks if new reservations have arrived or if some reservations from the last steps have not been yet assigned to a vehicle. If this is not true, then TraCI calls the next step.

The new reservations are retrieved using the TraCI call '*traci.person.getTaxiReservations(1)*'. If new reservations have been made, the tool processes these and adds them to the reservation pool. The process of a reservation consists not only in reading the minimal required parameters (departure time and origin and destination edges) but also in adding new parameters that are required for the simulation. At least four new attributes are added to each reservation element:

- direct attribute: the travel time from origin to destination taking a direct route,
- the vehicle attribute: includes the ID of the vehicle, when the reservation is assigned to one, and
- the time windows for pick up and drop off: set the allowed times for the DRT vehicle to pick up and drop off the person.

Depending on the DRT service, the definition of the requests may require other attributes. This can be defined using SUMO *GenericParameters*. After a reservation is processed, it is added to the reservation pool, which contains all current reservation elements.

Next, the DRT fleet is retrieved using '*traci.vehicle.getTaxiFleet()*'. This retrieves the ID of all vehicles in the simulation that are equipped with a '*taxi.device*'.

As a further step, the module checks if reservations have been served or rejected during the last step of the simulation and if so, these are removed from the reservation pool. The rejected reservations or persons are then removed from the simulation. The remaining reservations are then classified by:

- new: if the reservation has been made during the current simulation step,
- unassigned: if the reservation has not been yet assigned to a vehicle,
- assigned: if the reservation is included in the route of a DRT vehicle, and

- picked up: if the reservation has already been picked up by a vehicle and it is on the route to its destination.

The reservations that have been already picked up but not yet dropped off are retrieved using the TraCI call '*traci.person.getTaxiReservations(8)*'.

This classification plays an important role for the DARP solver, which is called in the next step. The DARP solver finds the best routes for each vehicle, based on the DRT fleet, the reservations, and the defined constraints.

It is possible that the DARP solver only finds all possible routes, with their respective costs, for each DRT vehicle and then, the best routes should be found using an ILP (integer linear programming). For this purpose, the Python LP solver PuLP [1] is included in the scheduling

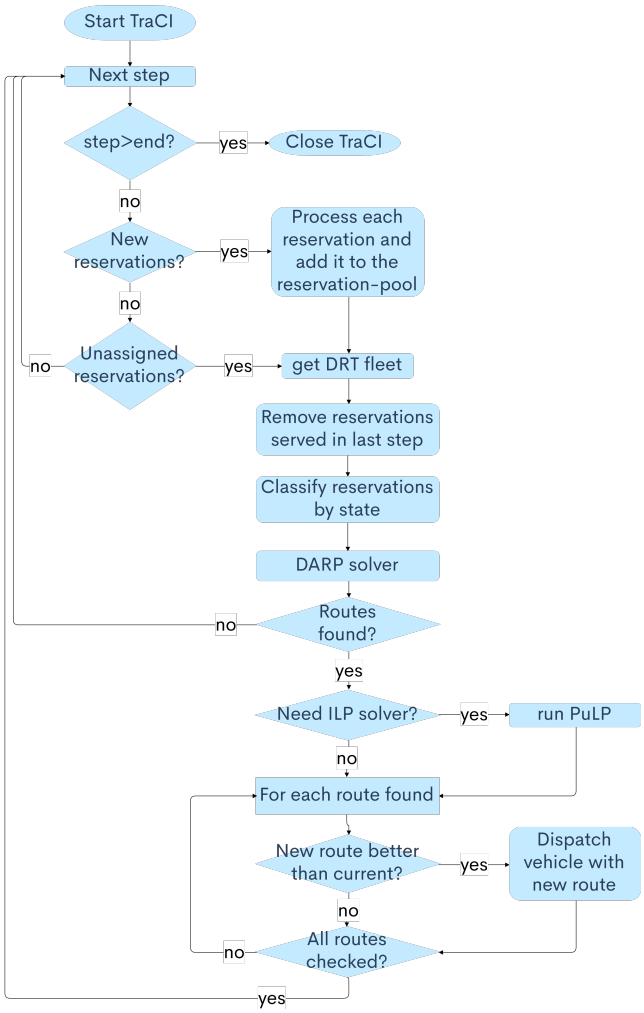

Figure 1: Flowchart of the scheduling module

[1] https://coin-or.github.io/pulp/ last access 17. May 2021

module. After calling the DARP solver, the module checks if an ILP is needed and if so, the PuLP solver is called.

Once the best routes have been found, the module will compare them to the current ones. This step is relevant since many DARPs are not solved with exact methods. As a result, the new best route of a vehicle can contain the same reservations but with a different order, which may not be better than the current one. To avoid this, the module compares the current route of each vehicle with the new best route found. If the new route is better, then the vehicle is rerouted using the TraCI call '*traci.vehicle.dispatchTaxi()*'.

When a vehicle is dispatched with a new route, the ID of the vehicle is added to the attribute '*vehicle*' of the reservations that are on the route.

After all the found routes are checked, TraCI calls for the next step and the loop starts again.

When the simulation step reaches the given end-time, the simulation is ended and TraCI is closed.

2.2 DARP solver

Having a group of reservations and vehicles that should serve them, the DARP solver will find the best route for each vehicle under certain constraints. There are different methods to solve a DARP, starting with the differentiation of an exact or approximate method, to which constraints should be considered (e.g., time windows constraints) and which objective should be optimized (e.g. passenger waiting times and vehicles idle times). Depending on the solver, the simulation results of the same DRT service can be different.

The goal of this tool is therefore not only to offer certain DARP solvers but also to allow the user to implement its algorithms. To solve the multiple vehicle DARP, many methods find as a first step all possible routes with their costs and then call an ILP solver to find the best ones. For this, the ILP solver PulP has been included as an option in the scheduling module.

The tool counts with a default DARP solver that simulates shared DRT. The solver applies the method of [1], which was also implemented in previous DRT studies [4, 3], but for solving the static or offline DARP case. This means that all reservations were known in advance and are not being processed in real-time or online, as is the case for the present tool.

The solver is called from the scheduling module and as input, the reservations, the DRT fleet, and the constraints are given. The solver searches first for all possible combinations between a vehicle and a reservation and between two reservations. A combination is possible if the pick-up and drop-off time windows can be regarded. For example, if the origin of reservation 2 wants to be paired with the destination of reservation 1, but the latest drop-off time of reservation 1 is at 10:00 and the earliest pick-up time of reservation 2 is at 10:05, then the pair is not possible. Reservations that have already been assigned or picked up by a vehicle can no longer be combined with another vehicle.

After searching all possible pairs, a pair-wise graph with each pair and its travel time is saved. The travel time is calculated using the TraCI call '*traci.simulation.findRoute()*', which considers the surrounding traffic for the calculation.

Next, the pairs of the mentioned graph are explored to find feasible trips for each vehicle. This proceeds incrementally in trip size (i.e. number of stops), starting from the vehicle-request pair. For each size, an exhaustive search is conducted. The time complexity of the algorithm depends on the number of vehicles and requests, and their shareability. If all vehicles can serve all requests and all requests can be combined with each other, the complexity will be $O(mn^v)$[1]. To allow for a faster (but not exact) solution, a timeout for the search of trips of each size can

be set.

A trip is primarily feasible if the requests are picked up and dropped off between the specified time windows and if the capacity of the vehicle is not surpassed. Since the simulation of the DRT is dynamic, this means that vehicles can be rerouted, other considerations have to be made. If some reservations have been assigned to a vehicle in a previous step, then these reservations have to be included in the new possible routes. Nevertheless, the order of the stops may be changed and new reservations may be added.

Idle vehicles with the same current edge and capacity will have the same possible trips. To avoid searching for the same trips multiple times, the exhaustive search will be performed only once and then the trips will be transferred to the other vehicles. In scenarios with large fleets and repeated origin or destination edges, this will speed up the simulation time notably.

Once all possible routes with their travel times are found, the best route for each vehicle that optimizes the entire DRT service has to be found. This optimization problem can be solved with an ILP solver. As the last step, the routes are written in a format that can be processed by the ILP solver module PuLP and are returned to the scheduling module.

3 Application example

As a study case, a DRT service operating in the city center (Innenstadt) and in the Nordstadt neighborhood of the city of Brunswick (Germany) was simulated. The operating area of the DRT service is shown in figure 2. The demand and sumo network were adapted from the publicly available Brunswick scenario [2], which simulates the motorized individual traffic in the city for a typical working day. For the DRT simulation scenario, a simulation time window between 6 a.m. and 10 a.m. was considered. An extra warm-up and cool-down time of 30 minutes each was adopted.

The requests to the DRT service stem from the demand of the Brunswick scenario. First, only persons that have made all their daily trips inside the DRT operating area were considered to use the DRT service instead of the private car in the simulation. For this, each of its trips was written as a DRT request. Between 6:00 and 10:00 in the morning, 758 persons made 956 trip requests. Once the reservation is made, the person should be picked up in the next 15 minutes. If there is no DRT vehicle available for this, the reservation is rejected and the person is removed from the simulation. In addition, to offer a trip with the DRT service, the estimated travel time should not be longer than twice the direct travel time with the private car (equivalent to a detour factor of 2) or surpass it by 10 minutes.

The DRT fleet consists of 50 vehicles with a capacity of 6 passengers each, allocated in three different depots (see figure 2). When the simulation starts, the vehicles wait for the requests at the given depot. The simulation step was set to 10 seconds, which means that the reservations are also processed within this time-step. Figure 3 shows the simulation. The red vehicles represent the DRT fleet and the blue points are the persons making the requests. The surrounding traffic was hidden in the figure for a better overview.

The described simulation scenario was run 5 times. Since the proposed DRT tool only controls the routing of the DRT vehicles and passengers via TraCI, all standard SUMO outputs (i.e. trip-information, stop-information, and emissions output) can be generated. In the following, some examples of the analysis that can be carried out with the tool are shown.

[2] https://github.com/DLR-TS/sumo-scenarios/tree/master/brunswick/miv last access 17. May 2021

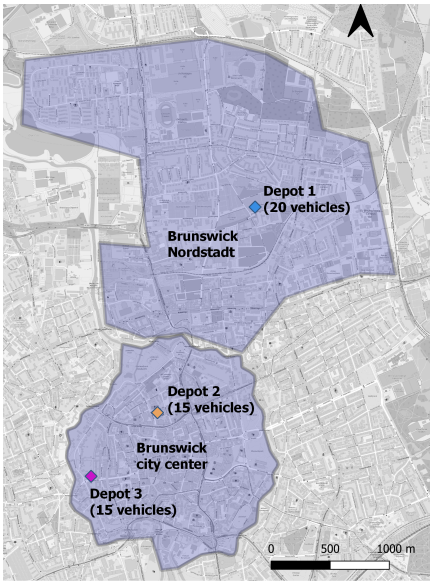

Figure 2: DRT operating area (background image from OSM)

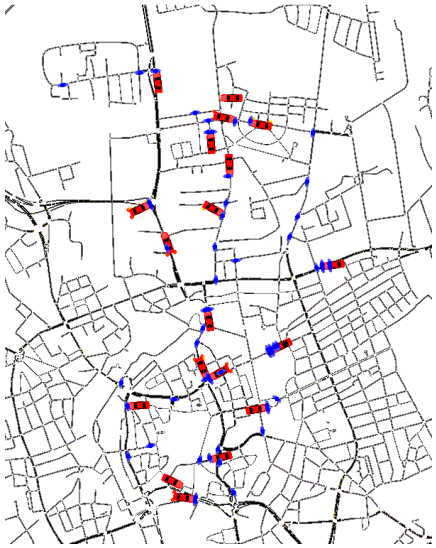

Figure 3: DRT sumo-simulation

3.1 Results

The results presented below were obtained from the simulation-log, trip-information, and stop-information outputs. Table 1 shows the results regarding the performance of the simulation and the DRT fleet for each time the scenario was run.

The five hours simulated (from 5:30 to 10:30) had an actual average duration of 95 minutes (1.58 hours), which is equivalent to a real-time factor of 3.20. According to the results, not all 50 DRT vehicles were used. The maximum number of vehicles used was 46 vehicles for the third simulation run. The mileage of all DRT vehicles was on average 2765.78 km. The total mileage of the first simulation run was 2757.73 km. The sum of the direct route lengths of the passengers served in this run was 1534.19 km. Hence the mileage of the DRT service is equivalent to 1.79 the mileage of the private cars. But the number of vehicles used is reduced from 758 in the case of all passengers using the private vehicle to only 43 DRT vehicles. In the five simulation runs, not all 956 trip requests were able to be served. This can be related to

Simulation run	1	2	3	4	5
Duration [min]	113.96	91.56	87.00	98.20	82.18
Real time factor	2.63	3.27	3.44	3.05	3.65
DRT vehicles used	43	42	46	43	41
Mileage DRT fleet [km]	2757.73	2709.89	2811.28	2799.67	2750.33
Max. passengers at same time	4	3	3	4	3
Requests served	871	835	869	876	884
Requests rejected	85	121	87	80	72

Table 1: Simulation performance and DRT fleet

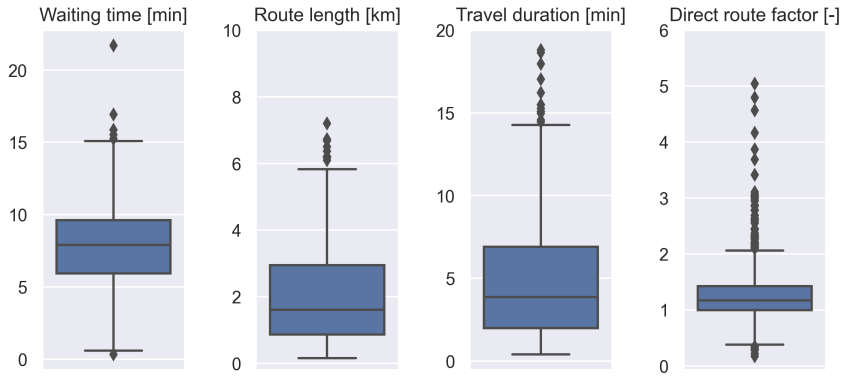

Figure 4: Simulation results for passengers as mean value of simulation runs

a longer than 15 minutes travel time from the depots to the pick-up of these passengers. But in the case of simulation run 3, where more requests were rejected, these extra rejections are most probably related to the ability of the proposed tool to find a compatible route (solve the DARP problem). This can be improve by setting a higher timeout to solve the DARP. The DRT vehicles have a capacity of 6 passengers, but as it is shown in the table, the maximum number of passengers at the same time was only 4.

Figure 4 shows the waiting time, route length, travel duration, and the direct route factor (relation between the travel duration with the DRT and the private car) for each passenger as box plots. The results for each passenger represent the mean value of the 5 simulation runs. The first plot shows the waiting time for the pick-up. As it was mentioned previously, this time should not exceed 15 minutes. Only in a few cases, this was not the case and it can be related to delays and congestion. On average the waiting time for pick up was 7.8 minutes. The direct route factor (last plot) was limited to a value of 2 or for the case of short routes, to 10 minutes longer travel time than with the private car. On average, the direct route factor was 1.28. But in some cases, it was greater than 2, which is related to the cases with short travel duration.

4 Conclusion and future work

This paper presents a SUMO Python tool to simulate Demand Responsive Transport (DRT) with a dynamic dispatcher using the SUMO taxi device and TraCI. The tool consists of two modules: a scheduling module, where the fleet and requests are managed and vehicles are dispatched, and a DARP solver module. The latter calculates the possible routes for the DRT vehicles by solving a Dial a Ride Problem (DARP).

The tool aims to simulate complex DRT systems and to make it easier for the users to implement their methods for solving the DARP.

The tool includes a DARP solver that enables the simulation of shared DRT. As an application example, a DRT service operating in the Innenstadt and Nordstadt neighborhoods of the city of Brunswick was simulated. The DRT service consists of a fleet of 50 vehicles with a capacity of 6 passengers each. A total of 758 persons made 956 trip requests between 6:00 and 10:00. The tool supports the global configurations and outputs that SUMO allows for persons and vehicles with taxi devices. For the application example, the surrounding traffic was considered in the simulation for a more realistic result. Only 43 of the 50 DRT vehicles available were

used to serve 871 requests. 85 requests of the 956 (8.89%) had been rejected by the service. The reason for the rejections is most probably related to the limited time of 15 minutes for the pick-up. These results are only an example of the analysis that can be made with the proposed DRT tool. The simulation time was on average 1.58 hours, which is equivalent to a real-time factor of 3.20.

The current DARP solver is not a good solution when dealing with large scenarios, for example, for the simulation of DRT services at a city scale. According to the literature, other heuristics methods like Variable Neighborhood Search (VNS) or Adaptive Large Neighborhood Search (ALNS) are a better solution for this purpose. As further work, the implementation of such methods, as well as the extension of the tool to manage other DRT options (e.g. first/last mile DRT, multiple person booking) are planned.

References

[1] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. January 2017.

[2] P. Alvarez Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[3] Maria Giuliana Armellini. A tool for simulating demand responsive transport systems in sumo. In *7th International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2021*, 2021.

[4] Maria Giuliana Armellini and Laura Bieker-Walz. Simulation of a demand responsive transport feeder system: A case study of brunswick. 2020.

[5] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46, 2007.

[6] Francesca Guerriero, Ferdinando Pezzella, Ornella Pisacane, and Luigi Trollini. Multi-objective optimization in dial-a-ride public transportation. *Transportation Research Procedia*, 3:299–308, 2014. 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain.

[7] Ying Luo and Paul Schonfeld. Online rejected-reinsertion heuristics for dynamic multivehicle dial-a-ride problem. *Transportation research record*, 2218(1):59–67, 2011.

[8] Mohamed Amine Masmoudi, Manar Hosny, Kris Braekers, and Abdelaziz Dammak. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, 96:60–80, 2016.

[9] Yves Molenbruch, Kris Braekers, and An Caris. Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, 259(1-2):295–325, 2017.

[10] André Luyde S Souza, Jonatas BC Chagas, Puca HV Penna, and Marcone JF Souza. A hybrid heuristic algorithm for the dial-a-ride problem. In *International Conference on Variable Neighborhood Search*, pages 53–66. Springer, 2019.