Journal of Ubiquitous Systems & Pervasive Networks
Volume 6, No. 1 (2015) pp. 17-26

Analyzing On-demand Mobility Services by Agent-based Simulation

Michal Čertický*, Michal Jakob, Radek Píbil

Agent Technology Center, Faculty of Electrical Engineering
Czech Technical University, Prague, Czech Republic

Abstract
Widespread adoption of smartphones and ubiquitous internet connectivity gives rise to new markets for personalized and efficient on-demand mobility services. To rigorously analyze new control mechanisms for these services, we introduce an open-source agent-based simulation testbed that allows evaluating the performance of demand-responsive transport schemes. In particular, our testbed provides a framework to compare both centralized and decentralized, static and dynamic passenger allocation and vehicle routing mechanisms under various conditions; including varying vehicle fleets, road network topologies and passenger demands. The testbed supports all stages of the experimental process; from the implementation of control mechanisms and the definition of experiment scenarios, through to simulation execution, analysis, and interpretation of results. Ultimately, our testbed accelerates the development of control mechanisms for emerging on-demand mobility services and facilitates their comparison with well-defined benchmarks. We illustrate our approach on an example simulation study of standard taxi and taxi sharing services in the area of Sydney, Australia.

Keywords: *On-demand Transport, Mobility, Testbed, Agent-based Simulation, Resource allocation, Vehicle rout*

1. Introduction

On-demand mobility services have the potential to bring significant improvements to personalized transport via efficient utilization of transport vehicles. In on-demand mobility services, vehicle routes and schedules are not fixed a priori; instead, they are dynamically adapted to best serve continuously incoming transport requests. Traditionally, on-demand mobility approach was used mainly for providing small-scale specialized transport services that required prior-day booking. In the past years, enabled by the widespread adoption of smartphones and ubiquitous internet connectivity, on-demand mobility approach has been increasingly utilized for general-purpose real-time ridesharing, taxi, or bus-on-demand services. Looking to the future, new autonomous, driverless vehicles, are set to lead to further growth in both scale and scope of on-demand mobility services.
On-demand mobility services are inherently multi-agent. This is due to a large number of geographically distributed passengers, vehicles and service providers which, while having their individual interests, have to coordinate and agree on how the passenger demand is served by available transport resources. As such, agent-based techniques are beginning to play an important role in passenger allocation and vehicle coordination [5].
The performance of on-demand mobility services with multi-agent based allocation and coordination crucially depends on two key factors: (1) the transport *control mechanism* used to allocate vehicles to passengers and to determine vehicle routes and timings; and (2) parameters of the *deployment scenario*, in particular the topology of the underlying road network and spatio-temporal structure of passenger demand. Understanding how these factors affect the performance of the transport system is essential for a principled development and deployment of on-demand mobility services. Due to the complex nature of demand-responsive transport systems, however, gaining such understanding is difficult.
Simulation modelling is an established approach for analyzing the behaviour of complex socio-technical systems and is therefore also applicable for analyzing demand-responsive transport systems. Unfortunately, out of the number of transport simulation tools, none is specifically tailored and consequently particularly suitable for simulation modelling of on-demand mobility services.
In fact, within the broad family of *pickup and delivery problems*, in which on-demand services are a special case, benchmarking suites only exist for static freight transport vehicle routing problems (www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html). To the best of our knowledge, no benchmarking tools exist for dynamic, passenger-oriented variants of the pickup and delivery problem. The key reason for the lack of benchmarking tools is that a simulation engine is required to rigorously account for temporal dependencies. This means that benchmarking on-demand mobility services is a

* Corresponding author. Tel.: +420 608346566
E-mail: certicky@agents.fel.cvut.cz

DOI: 10.5383/juspn.06.01.003

significantly more challenging problem and cannot be solved by simple evaluation tools that are sufficient for static variants of the problem.

In this article, we detail our new simulation testbed - a rigorous and flexible benchmarking suite for on-demand mobility services. The testbed is based on our previous research in fully agent-based simulation modelling of transport systems [7]. It is built on top of a versatile transport simulation framework AgentPolis [8].

The testbed is designed for two main purposes. The first purpose is performance assessment of on-demand mobility services prior to their deployment in new locations or under different conditions. The second purpose is testing and evaluation of novel control mechanisms, algorithms and on-demand mobility schemes. As such, our testbed can speed up the development and deployment of on-demand mobility services.

The contributions of this article can be divided into three parts: First, we describe the testbed and its architecture in Section 3, then we explain the usage and experiment process in detail in Section 4 and finally, we demonstrate how the testbed can be used to evaluate and compare two examples of on-demand mobility service in Sydney, Australia, in Section 5.

2. Related Work

Since 1970s, we have seen numerous attempts to study mobility and transport systems in general by analytical modelling. An extensive overview of analytical modelling methodology, along with mathematical background can be found in a monograph by Ortuzar and Willumsen [15]. Early models of mobility systems were largely based on mathematical programming and continuous approximations. The former technique relied on detailed data and numerical methods, whereas the latter relied on concise summaries of data and analytic models. Geoffrion [16] advocated the use of simplified analytic models to gain insights into numerical mathematical programming models. In a similar spirit, Hall [17] illustrates applications of discrete and continuous approximations, and notes that continuous approximations are useful to develop models that are easy for humans to interpret and comprehend. Overview and classification of continuous approximation models can be found in [18].

In some of the more recent work, the attention was focused on demand-responsive mobility systems, which were formalized as mathematical abstractions, such as dial-a-ride problem (DARP) [19] or multiple depot vehicle scheduling problem (MDVS) [20] to allow further formal analysis. For example, Heuptmeier et al. [21] and Lipmann et al. [22] studied formal properties of certain algorithms solving DARP and its variant with restricted information.

Haghani and Banihashemi [23] addressed the influence of town size on the performance of algorithms for MDVS and its variant with route time constraints.

However, analytic models and theoretical algorithm analyses were often too abstract for expressing relevant aspects in the structure and dynamics of some transport systems. To deal with this shortcoming, the paradigm of simulation modelling was adopted by the transport research community and has been employed in parallel with the analytical approaches. In 1969, Wilson [24] conducted a pioneer simulation-based study of the influence of the service area, demand density and number of vehicles on the behaviour of transport system. Simulations have since then been extensively utilized in the research of transport and mobility, as a powerful tool for the analysis of system's behaviour. To mention a few more examples, Regan et al. \cite{regan1996dynamic} studied the performance of freight mobility system using different lead acceptance and assignment strategies, Fu [4] developed a simulation model of an urban paratransit system and Deflorio et al. [30] evaluated demand-responsive transport system under the influence of real-life aspects, such as customer delays and travel time variability.

Simulation modelling also allowed researchers to study DARP or MDVS control mechanisms empirically (in addition to formal algorithmic analysis). Bailey and Clark [31] investigated the performance of one of them in relation with varying vehicle fleet size. Jlassi et al. [10] simulated an ambulance service implemented as dial-a-ride system and Shinoda et al. [32] compared such systems to fixed-route systems under varying circumstances. Diana [33] assessed the effectiveness of scheduling algorithms under different percentages of real time requests and intervals between call-in time and requested pick-up time and Quadrifoglio and Dessouky [14] studied the insertion heuristic scheduling algorithm for Mobility Allowance Shuttle Transit systems, a hybrid transit solution that merges the flexibility of dial-a-ride systems and the low cost operability of fixed-route bus services. More recently, d'Orey et al. [3] used simulations to explore the trade-offs between the satisfaction of drivers and passengers.

In these simulations, the system's behaviour could only be centralized -- governed in a top-down manner by a single entity or mechanism. Also, any self-initiated interactions, communication or negotiation among individual actors (e.g. passengers) was impossible, severely limiting their level of autonomy. To overcome these limitations, a new paradigm called *Agent-based simulation* was introduced. Agent-based simulation has proven to be a highly valuable tool, especially when studying complex self-organizing systems in many domains [26]. Mobility systems modelled under this paradigm are implemented as multi-agent systems -- i.e. composed of autonomous entities termed *agents* situated in a shared environment which they perceive and act upon, in order to achieve their own goals. In the context of mobility, we usually distinguish between three relevant types of agents: *passengers* (announcing transport requests), *drivers* (serving passenger's requests) and *dispatchers* (optional kind of agents who can negotiate with passengers and coordinate the drivers).

Agent-based simulations have been used to study various aspects of mobility, as well as a number of different control mechanisms. Horn [6] employed a simulation, developed completely from scratch, to study operational characteristics of a multimodal transport system integrating conventional timetabled services (buses, trains, etc.) and flexible demand-responsive mobility services (single- and multiple-hire taxis). A combination of traditional and demand-responsive transit was also simulated by Edwards [27]. The impact of zoning vs. non-zoning strategies on demand-responsive mobility were studied by Quadrifoglio and Dessouky in [13]. Real-time taxi sharing schemes and ridesharing have been evaluated by Kamar and Horvitz [11], Lioris et al. [12], or Agatz et al. [28], while the efficiency of traditional taxi services have been studied by Cheng and Nguyen [29].

Control mechanisms that govern the behaviour of mobility systems are usually classified by the concentration of decision making into:

- *centralized* (all the agents controlled by a central entity, e.g. dispatcher)
- *distributed* (agents act based on their mutual, unorganized interactions)
- *hybrid* (combination of those two)

Alternatively, control mechanisms may also be divided based on the structure of transport demand they are dealing with into *static* (all transport requests are known in advance) or *dynamic* (future requests are unknown).
Since these control mechanisms represent a cornerstone of mobility system's behaviour and success, significant amount of research has been invested in them and more is still needed. It therefore makes sense to develop software tools that would assist in this research. The general idea of employing simulation testbeds to accelerate the development of multi-agent control mechanisms was put forward for example by [9].
A common attribute of simulations used in the works above is that they were developed from scratch using general-purpose programming languages (most often C++ or Java), in order to demonstrate only a single specific mechanism. This is because none of the existing general purpose (such as AnyLogic, http://www.anylogic.com) as well as transport-specific simulation tools (such as MATSIM, http://www.matsim.org or SUMO, http://www.sumo-sim.org) has proven suitable for simulation-based assessment of a wider variety of control mechanisms.
The agent-based simulation testbed described in this article was created to fill this gap and provide the researchers with a tool necessary to analyze and compare the control mechanisms of various classes without developing their own simulations.

3. Testbed Overview

Proposed simulation testbed is built upon the versatile transport simulation framework AgentPolis, which provides abstractions, code libraries and software tools for building and experimenting with fully agent-based models of interaction-rich transport systems.

3.1. Fully Agent-Based Simulation Approach

In fully agent-based simulations, individual entities of a transport system are represented as autonomous agents with continuous, asynchronous control modules and the ability to interact freely with their surrounding environment and other agents. Such an approach reduces coupling and allows modeling scenarios in which agents adjust their plans at any time during the day based on their observations of the environment and/or communication with other agents.
The AgentPolis framework, which implements the fully agent-based approach, provides several dozens of modelling elements that can be used to build a wide range of simulation models. The modelling elements provided by AgentPolis are organized in a modelling ontology and can be grouped to three high-level categories:

- **Agent modelling elements:**
  The concept of the *agent* in AgentPolis is defined rather loosely in order to support modelling a wide variety of agents (e.g. `DriverAgent`). The behavior of agents is defined in terms of *activities* - reactive control structures implementing the logic determining which actions or nested activities the agent executes at a certain point in time or in response to sensor information or messages received from other agents (e.g. `DriveVehicle` activity). As part of their behaviour, agents may need to make decisions that require executing complex algorithms, including the ones that comprise the control mechanisms we want to evaluate. In order to promote reusability, such algorithms are encapsulated into so-called *reasoning modules*. In practice, the reasoning modules are Java classes (e.g. `DriverLogic`) that can be easily rewritten to implement a wide variety of algorithms, or even call external tools or solvers.
- **Environment modelling elements:**
  The environment models the physical context in which the agents are situated and act. It is represented by a collection of *environment objects*, each representing a fragment of the modelled physical reality (e.g. `Vehicle`), and *queries* that allow agents to be informed about the state of the environment and about the events happening during simulation execution (e.g. `PositionQuery`).
- **Interaction modelling elements:**
  Modelling complex interactions among the agents or between the agents and the environment is crucial for the analysis of dynamic transport systems. In AgentPolis, agent-environment interactions are realized by *sensors*, which process the percepts from the environment and atomic *actions* that provide a low-level abstraction for modelling how agents actually manipulate the environment (e.g. `MoveVehicle`). Inter-agent interactions are realized by a collection of *communication protocols*. The testbed currently provides `1-to-1 messaging`, `1-to-many messaging` and `auction` protocols.

Detailed description of modelling abstractions and corresponding model elements can be found in [7].

3.2. Testbed Architecture

Although all the power and flexibility of the AgentPolis framework is accessible to the users of the testbed, it is hidden and only the relevant parts of it are exposed through a facade of APIs designed specifically for the simulation modelling of demand-responsive transport systems.
The components of the testbed can be broadly divided into three layers (see Figure 1):

- *AgentPolis Transport Simulation Model:* composed of the core simulation engine and the basic transport domain model. This model implements key elements comprising a transport system, such as road network and vehicles, and basic behavioral logic associated with them. It also provides routing algorithms and communication interfaces designed to simplify the implementation of higher-level simulation logic.
- *Testbed Core:* specializes the general AgentPolis simulator for the specific purpose of modelling demand-responsive transport systems. It implements the model of three types of agents (*Passengers*, *Drivers* and *Dispatchers*) and provides extensible abstractions for defining their behaviour.
- *Control Mechanism:* a user-supplied implementation of a specific control mechanism that is to be experimentally evaluated.

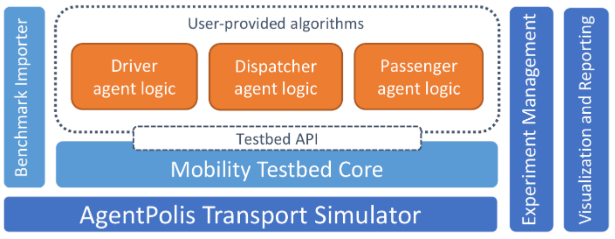


Fig. 1. Testbed's architecture overview.

In addition to these three layers, the testbed provides a suite of tools that facilitate creation, execution and evaluation of simulation experiments:

- *Benchmark importer* loads all the required input data (discussed in detail in Section 4, constructs the graph representation of a road network and creates the environment objects and agents accordingly. All the imported data is automatically checked for consistency in order to prevent hard-to-trace errors. Resulting internal representations are simplified by selectively removing redundant information in order to accelerate the reasoning without losing accuracy (see Figure 2).

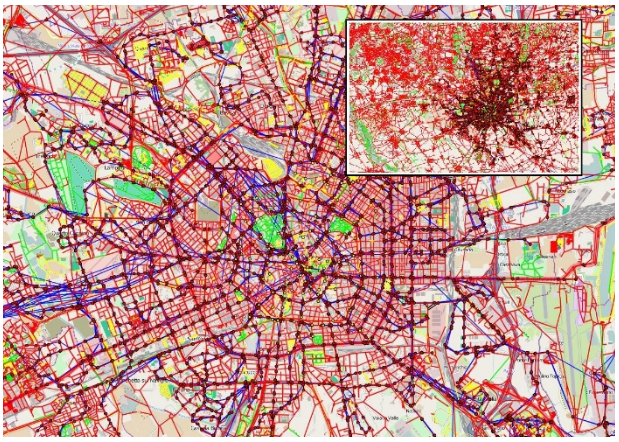


Fig. 2. An integrated simplified transport graph combining road and walkpath networks in Milan, Italy.

- *Experiment management* is supported by a benchmark generator and tools for the design of experiment. The *generator* allows users to build their own scenarios covering real-world or fictional locations with custom numbers and types of agents. Agents can be generated either based on real-world data or randomly, using various temporal and spatial distributions.
  Since a robust evaluation of the control mechanism under a sufficiently wide range of circumstances may require many simulation runs, the testbed provides tools for accelerating the evaluation process. In particular, it can use *design of experiments* methods to generate simulation configurations in a way so that maximum information about the behaviour of the control mechanism is obtained using a minimum number of simulation runs.
- The *Analysis and visualization* tools provide a way to interactively browse and review simulation execution and results at different spatial and temporal resolutions. This assists researchers during the development and debugging process, and allows them to find out how the tested mechanisms perform under different conditions. The aggregated results, as well as the visualizations, are generated based on the detailed low-level event log recorded during the simulation, containing all the important events related to passengers `passGotInVehicle`, `passGotOffVehicle`) and drivers with their vehicles (`vehicleMove`) and all the communication between the agents (e.g. `passSentRequest` or `requestConfirmed`).

3.3. Transport Control Mechanism

Unless the studied control mechanism has some special features, its incorporation into the testbed only requires implementing several classes and methods. For example, in the most simple case, the user only needs to extend the `DispatchingLogic` class and implement its `processNewRequest(Request r)` method.

The testbed allows us to incorporate and study a variety of control mechanisms. They can be divided into *centralized* or *decentralized* mechanisms, based on the degree of autonomy of the drivers. We also distinguish between *dynamic* and *static* control mechanisms, each suitable for the transport demand with different temporal structure.

Centralized vs. Decentralized: In a demand-responsive transport system, the behaviour of driver agents can be governed either centrally by (single or multiple) dispatcher agents, locally by the drivers themselves, or the combination of both. The reasoning logic for individual agents and central authorities is implemented by extending specific methods of abstract classes `PassengerLogic`, `DriverLogic` and `DispatchingLogic` (see Tables I, II and III).
Decentralized mechanisms are suitable in situations when communication capabilities are restricted, or when the agents are independent and self-interested but can still benefit from collaboration (e.g. ridesharing [11]).

Static vs. Dynamic: Dynamic control mechanisms (sometimes called "online") process the travel demand requests when they are announced. On the other hand, static (or "offline") mechanisms need to know all the requests in advance. Our testbed grants the driver or dispatcher agents the access to requests only after they are announced by the passengers. Nevertheless, to also cater for the requirements of static mechanisms, there are several benchmarks in which the travel demand is announced long time in advance.

`sendRequest(Request r)`: Called by the testbed whenever this passenger is supposed to announce a new travel request `r`, according to input data. The passenger should contact other agents (dispatcher or drivers) within this method.
`processProposal(Proposal p)`, `processRejection(RequestRejection r)`: Two methods that are called when the passenger receives a trip proposal `p` specifying details about the trip (e.g. price or arrival time) or rejection `r` of his older request from a driver or dispatcher.
`vehicleArrived(String driverId, vehicleId)`: Called when a driver arrives to pick the passenger up. Typically, the passenger just gets on board this driver's vehicle.

Table I. Abstract methods of `PassengerLogic` class, implementing the behaviour of passenger agents.

`processNewRequest(Request r)`: A method called whenever the driver receives a new travel request `r` from a passenger or dispatcher. Here, the driver should react by sending his trip proposal or request rejection.
`processNewAcceptance(Proposal p)`: Called when the driver's trip proposal `p` is accepted by the passenger. Here, the driver usually plans his route and starts driving.
`processNewRejection(Proposal p)`: If driver's proposal `p` is rejected, the testbed calls this method.

`processPassengerGotIn(String passengerId)`: Called when the passenger gets on board this driver's vehicle.

Table II. Abstract methods of `DriverLogic` class, implementing the behaviour of driver agents.

`processNewRequest(Request r)`, `processNewAcceptance(Proposal p)`, `processNewRejection(Proposal p)`: The methods with similar meaning as in `DriverLogic` class with the exception that the dispatcher usually negotiates with passengers and only sends instructions and routes to drivers.

Table III. Abstract methods of `DispatchingLogic` class, implementing the behaviour of dispatcher agents.

4. Experiment Process

After the tested mechanism is incorporated into the framework, the actual experimentation using the testbed follows a three-step process, as depicted in Figure 3.

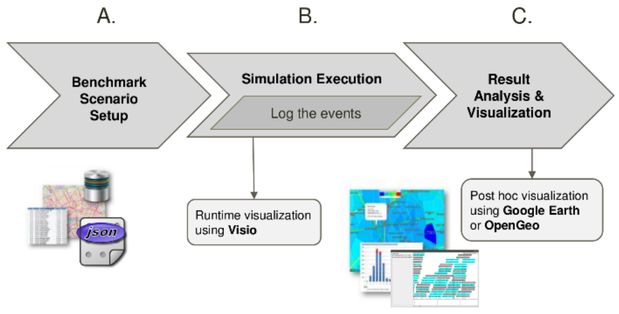

A.
B.
C.
Benchmark Scenario Setup
Simulation Execution
Log the events
Result Analysis & Visualization
Runtime visualization using Visio
Post hoc visualization using Google Earth or OpenGeo
json

Fig. 3. Three-step process of the experiment (setup, simulation, evaluation).

4.2. Scenario Definition and Setup

First of all, the user needs to set up and *configure the scenario* under which he wants the control mechanism to be evaluated. The scenario is described in terms of a benchmark package, which consists of the following files:

- *Road network* - The road network in the experiment area represented in the *OpenStreetMap (OSM)* format (http://openstreetmap.org/).
- *Driver agents* - Description (in JSON) of all the relevant drivers with their initial positions and the properties of their vehicles including the capacity, fuel consumption, CO2 emissions or non-standard equipment (e.g. wheelchair accessibility).
- *Travel demand* - The exact representation (in JSON) of travel demand containing all the passenger agents with their associated trip details: origin and destination coordinates, time windows, announcement time and special requirements.

4.2. Simulation Execution

Once the model is set up, the user invokes the simulation engine to execute the simulation. The AgentPolis engine employs the discrete event simulation approach [2] in which the operation of the target system is modelled as a discrete sequence of (possibly concurrent) events in time. Each event occurs at a particular time, with precision to milliseconds of the simulation time, and marks a change of state of the modelled system. Since there are no changes occurring between consecutive events, the simulation can directly jump in time from one event to the next, which, in most cases, makes it more computationally efficient than time-stepped approaches.

The simulation progress can be presented visually during run-time, using the internal visualization component of AgentPolis. It is capable of displaying the transport network and agents within the model, along with a convenient visualization of all the ongoing events (see Figure 4).

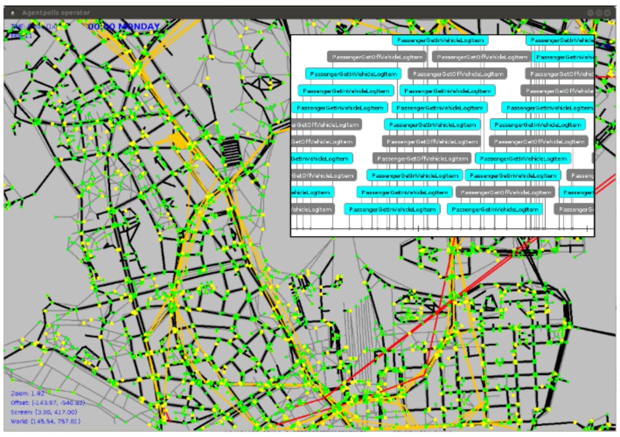


Fig. 4. Runtime view of a running simulation. Road network, Passenger and Driver agents are shown. Simulation events are depicted in the overlay window.

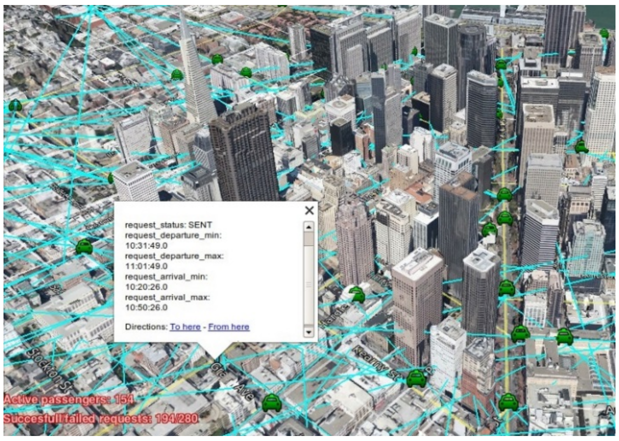
4.4. Result Analysis and Visualization

From the low-level event log recorded during the simulation run, the testbed calculates a range of higher-level, aggregate performance metrics. By default, these include:

- total vehicle distance driven,
- total fuel consumption,
- total values of pollutant emissions,
- average vehicle productivity (passengers per hour),
- passenger's total travel time statistics (average, median and maximum),
- passenger's on-board ride time statistics,
- passenger's waiting time statistics,
- total runtime of control algorithms.

Additional metrics can be defined. In addition to low-level event logs and highly aggregated metrics, the testbed also provides the means to visualize the simulation runs and results in the geospatial and temporal context, using external tools.

The interactive geobrowser *Google Earth* (http://earth.google.com/) can display the log of a simulation run exported in *Keyhole Markup Language* (KML, http://developers.google.com/kml/). It is capable of displaying a large number of agents, along with simple geometry and screen overlays, over a realistic satellite imagery and 3D model of the environment (see Figure 5).



```
request_status: SENT
request_departure_min:
10:31:49.0
request_departure_max:
11:01:49.0
request_arrival_min:
10:20:26.0
request_arrival_max:
10:50:26.0

Directions: To here - From here
```

```
Active passengers: 154
Succesfull/failed requests: 194/280
```

Fig. 5. Simulation run of on-demand transport coordination scenario exported in KML format and displayed by Google Earth. The input benchmark was based on historical traffic and demand data from San Francisco, 2008.

Google Earth can be further used to display the values of metrics as they vary across different areas. For example, Figures 9 and 10 show a heat maps representing the spatial distribution of successfully served and failed passenger trip requests.

5. Example Study

To demonstrate how the testbed can be used, we have implemented a control mechanism for dynamic multi-vehicle dial-a-ride problem, based on *parallel tabu search heuristic* [1]. We have run a series of experiments with it and evaluated a performance of two on-demand mobility services controlled by it: *single-passenger taxi* vs. *taxi-sharing service*.

5.1. Scenario Setup

Using our *benchmark generator*, we prepared 88 scenarios divided into two collections situated in a city of Sydney, Australia, covering the area of 2255.902km^2. In both collections, the travel requests of passenger agents were generated with realistic temporal and spatial distribution, taking into account peak/off-peak hours and spatial density of points of interest (restaurants, bars, stores, etc.) in the input OSM file. Passengers announced the travel requests 15 minutes before desired departure and accepted rides that would end less than 90 minutes from the announcement time. Each collection contained 44 scenarios - one for every combination of the number of vehicles (10 to 25 vehicles, increasing by 5) and request frequency (100, 200, 300, 400, 500, 750, 1000, 1250, 1500, 1750 and 2000 requests per day). In the first collection representing a single-passenger service, vehicles had the capacity of 2, while in the second, “taxi-sharing” collection, the vehicles had the capacity of 5 (in both cases, including the driver).
We were particularly interested in the following performance indicators for these two kinds of on-demand mobility services in the Sydney area:

- Success rate (percentage of successful vs. failed requests),
- Average distance driven per vehicle,
- Average passengers waiting and travel time.

5.2. Implementation of the Control Mechanism

Since *parallel tabu search* is a centralized control mechanism in a sense that the dispatcher agent has complete power over the behaviour of all the vehicles, we only needed to extend the abstract class `DispatchingLogic` and implement its method `processNewRequest(Request r)`, which is called every time the passenger agent announces a travel request.
This mechanism is considered a state of the art approach to the problem of dynamic multiple vehicle dial-a-ride problem. Details about it can be found in [1].

5.3. Simulation Results

We will divide the discussion of the simulation results into three parts – each related to one of the performance indicators of interest.
In first two cases, we will plot a series of charts depicting the values of service’s success rate and the average driven distance computed with different vehicle counts and under increasing number of requests per day (request frequency). In the third part, we will compare the average travel and waiting time in case of both services.
This will let us estimate how many vehicles would a service require in order to maintain a certain degree of efficiency under varying demand, or decide whether it is advantageous to introduce a taxi-sharing into this specific area.
Note that the testbed can also be used to gather a number of other indicators (see Section 4.4.) and outputs can be presented in different ways, according to the needs of the user.

5.3.1. Success Rate

Success rate is a key performance indicator - it tells us how many passengers the studied service is able to serve per day. The success rate is computed as a ratio of successfully served transport requests and all the announced requests.
Figures 6 and 7 depict the unsurprisingly decreasing trend of success rate values with the increasing request frequency. The charts show us how many requests per day we would manage to serve with different sizes of vehicle fleet. For example, 25 vehicles can successfully serve approximately 70 and 76% of 200 requests in case of single-passenger and taxi sharing case respectively.
This seemingly low efficiency is caused by the fact that the experiments cover very large urban area and the trips are quite long. Average passenger’s ride time among the successful requests from all the experiments is 1 hour, 17 minutes and 50 seconds and average distance driven to serve one passenger is 24.775km. This means that a single passenger occupies a vehicle for a relatively long time period, lowering the overall number of passengers that can be served.
Figure 8 compares the success rate of single-passenger vs. taxi-sharing service. It seems that the success rate of taxi-sharing service tends to be slightly higher. We suspect that the success rate difference between the two services would be higher if the requests were spread over a smaller area.

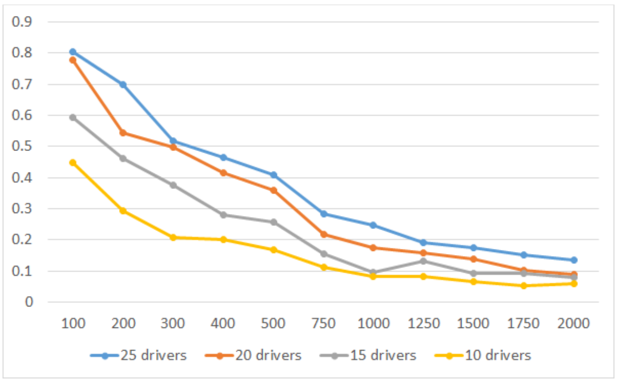


Fig. 6. *Success rate of single-passenger taxi* service with 10-25 vehicles. Horizontal axis corresponds to number of requests per day, while vertical depicts the success rate (success rate of 1 would mean that all the requests are served).

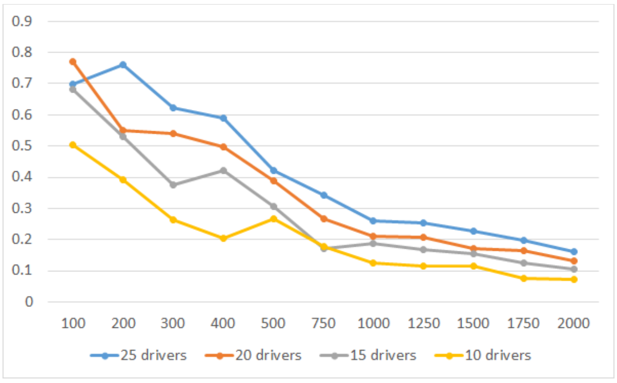


Fig. 7. *Success rate of shared taxi* service with 10-25 vehicles. Horizontal axis corresponds to number of requests per day, while vertical depicts the success rate (success rate of 1 would mean that all the requests are served).

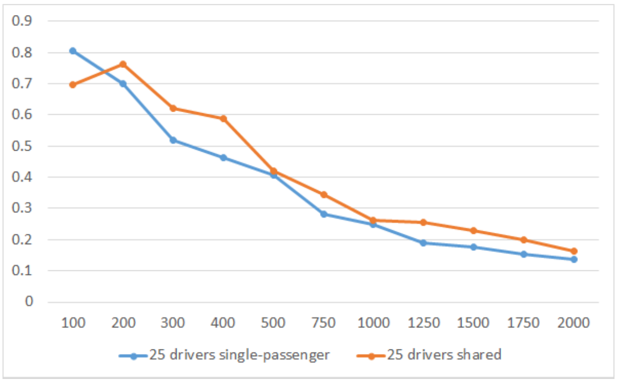


Fig. 8. *Success rate comparison* of single-passenger and shared taxi service with 25 vehicles. Horizontal axis corresponds to number of requests per day, while vertical depicts the success rate (success rate of 1 would mean that all the requests are served).

Spatial distribution of successful requests is depicted in Figure 9. Brighter areas represent the locations where more passengers successfully travelled. On the other hand, Figure 10 depicts a distribution of failed requests. It seems that the tested services are unable to deal with high number of trip requests with destinations around Bondi Beach.

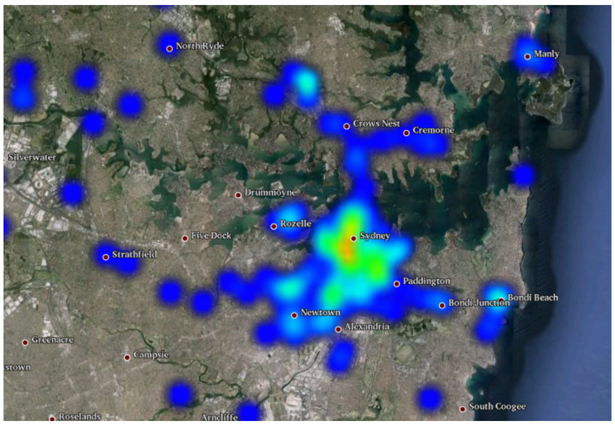


Fig. 9. *Successful requests of taxi-sharing* service with 25 vehicles depicted as a heatmap. The areas with higher density of successful request destinations are brighter.

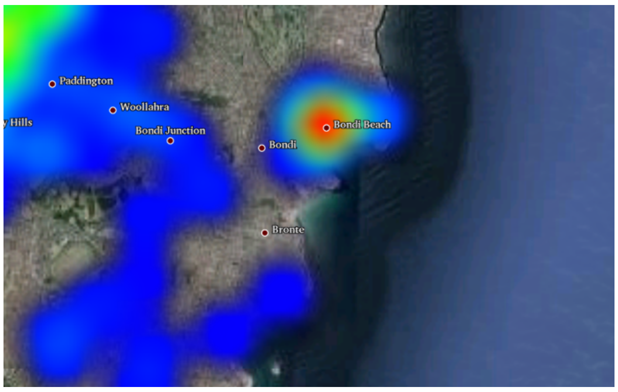


Fig. 10. *Failed requests of taxi-sharing* service with 25 vehicles depicted as a heatmap. The areas with higher density of failed request destinations are brighter.

5.3.2. Distance Driven per Vehicle

Next indicator is an average distance driven by a vehicle. This might be useful for users who need to estimate the operating cost of their service.

The testbed measures the length of all the simulated trips and divides it by the number of vehicles in the experiment. This also includes all the trips without any passengers on board. Measured values are in kilometers.

In figures 11 and 12, we can see that the distance driven by the vehicles seems to stabilize around the frequency of 1250-1500 requests per day. This suggests that the veicles are fully utilized (they are almost never idle) and an increase of request frequency beyond this point would not make them drive any more.

Figure 13 depicts slightly lower distance driven by the vehicles of a taxi-sharing service. Even though the difference is not very big, in combination with Figure 8, we can see that taxi-sharing service vehicles are able to serve more passengers while, driving smaller distances.

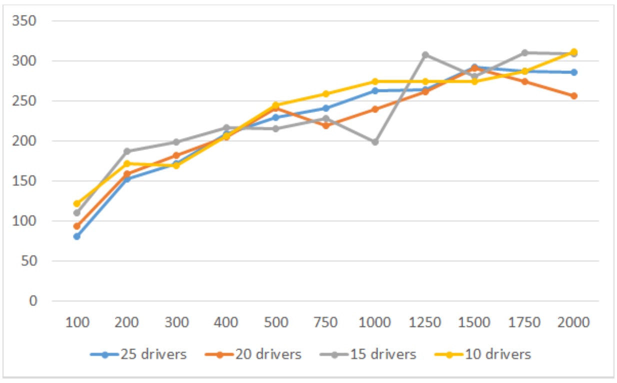


Fig. 11. *Average distance driven per vehicle* of a *single-passenger taxi* service with 10-25 vehicles. Horizontal axis corresponds to number of requests per day, while vertical represents the distance in km.

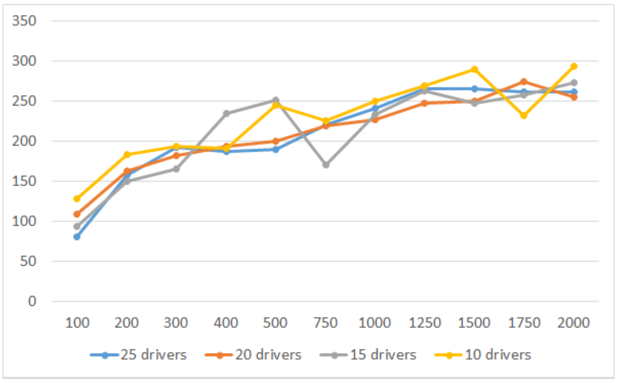


Fig. 12. *Average distance driven per vehicle* of a *taxi-sharing* service with 10-25 vehicles. Horizontal axis corresponds to number of requests per day, while vertical represents the distance in km.

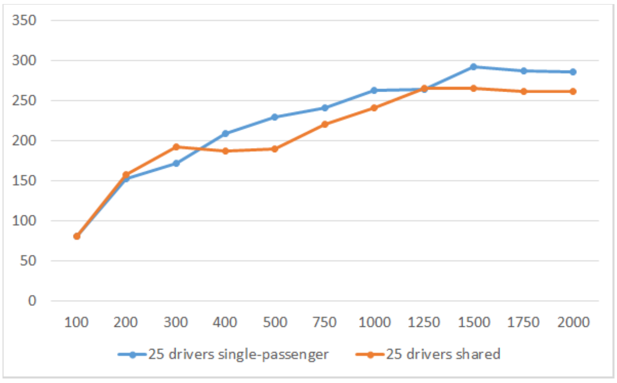


Fig. 13. *Comparison of average distance per vehicle* for a *single-passenger* and *taxi-sharing* service with 25 vehicles. Horizontal axis corresponds to number of requests per day, while vertical represents the distance in km.

5.3.3. Wait Time and Travel Time

Last indicators studied using the testbed deal with the comfort of the passengers. We are interested in the average time spent waiting between the request announcement and the moment when the passenger is picked up by a taxi, as well as an average time that a passenger spends on the actual trip.

As we can see in Figure 14, the average wait time of the passengers is 32:00 in case of single-passenger and 34:47 in case of taxi-sharing service. Relatively high waiting times are understandable, since the tabu search control mechanism does not explicitly minimize passenger's wait time. On the contrary, it tends to prolong the wait time (within acceptable boundaries) if it allows the vehicles to fulfill more requests.

We can also see that both the wait time and travel time are slightly higher in case of taxi-sharing service.

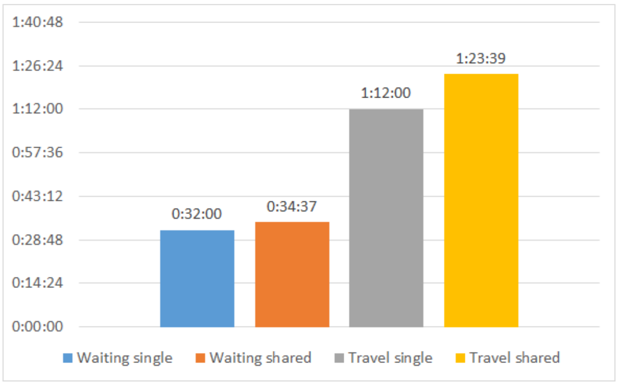


Fig. 14. *Average passenger's wait time* and *travel time* for a *single-passenger* and *taxi*-sharing services, computed over all 88 experiments. Horizontal axis depicts the time in H:M:S format.

5.4. Results Summary

The experiment results suggest that in the scenarios covering a large area of 2255.902km^2 the success rate of two on-demand services controlled by parallel tabu search heuristic decreases significantly with growing number of requests. According to expectations, the success rate is higher with a bigger vehicle fleet and and in case of taxi-sharing service.

Using heatmap visualizations, we identified the locations in Sydney area with highest density of successful requests and found one problematic area with very high density of failures. Insight like this might be useful for users planning to implement such services in real world (they could consider modifying their intended coverage area or control mechanisms).

Despite serving more passengers, the vehicles of taxi-sharing service seem to have slightly lower average distance driven per day. On the other hand, more effective utilization of vehicles in taxi-sharing service seems to negatively affect the waiting and travel times of the passengers.

6. Conclusion

We have presented a testbed for simulation-based evaluation of on-demand mobility services. The testbed allows its users to incorporate their own control mechanisms, to evaluate them with respect to a variety of performance metrics and to

compare their performance to alternative mechanisms under identical conditions using benchmark scenarios, based on realistic real-world or synthetic data. As such, the testbed can help policy makers and transport operators to assess on-demand mobility services prior to their deployment as well as it can assist researchers in developing new control mechanisms of on-demand mobility.
In the future, we aim to fully capitalize on the fact that the testbed is built on top of the versatile AgentPolis transport simulation framework. Two of the features that we plan to add in the near future are the incorporation of realistic time-dependent speed profiles for road network links and the use of activity-based models for passenger demand generation. In a longer term, we aim to combine the model of on-demand mobility services with the model of other transport modes supported by AgentPolis in order to allow studying properties of on-demand mobility within the context of fully integrated multimodal transport systems.

The testbed is freely available from:
http://github.com/agents4its/mobilitytestbed/.

Acknowledgments

This publication was supported by the European social fund within the framework of realizing the project „Support of inter-sectoral mobility and quality enhancement of research teams at Czech Technical University in Prague“, CZ.1.07/2.3.00/ 30.0034. Computational resources were provided by the MetaCentrum under the program LM2010005 and the CERIT-SC under the program Centre CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, Reg. no. CZ.1.05/3.2.00/08.0144.

References

[1] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Computing, 30(3):377–387, 2004. http://dx.doi.org/10.1016/j.parco.2003.12.001

[2] J. Banks, J. S. Carson, B. L. Nelson, D. M. Nicol, et al. Discrete-event system simulation. Pearson Prentice Hall Upper Saddle River, NJ, 2005.

[3] P. M. d’Orey, R. Fernandes, and M. Ferreira. Empirical evaluation of a dynamic and distributed taxi-sharing system. In Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems (ITSC), pages 140–146, 2012.

[4] L. Fu. A simulation model for evaluating advanced dial-a-ride paratransit systems. Transportation Research Part A: Policy and Practice, 36(4):291–307, 2002. http://dx.doi.org/10.1016/S0965-8564(01)00002-7

[5] A. Glaschenko, A. Ivaschenko, G. Rzevski, and P. Skobelev. Multi-agent real time scheduling system for taxi companies. In Proceedings of AAMAS 2009, pages 29–36, 2009.

[6] M. Horn. Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. Transportation Research Part A: Policy and Practice, 36(2):167–188, 2002. http://dx.doi.org/10.1016/S0965-8564(00)00043-4

[7] M. Jakob and Z. Moler. Modular framework for simulation modelling of interaction-rich transport systems. In Proceedings of 15th International IEEE Conference on Intelligent Transportation Systems. IEEE, 2013.

[8] M. Jakob, Z. Moler, A. Komenda, Z. Yin, A. X. Jiang, M. P. Johnson, M. Pěchouček, and M. Tambe. Agentpolis: towards a platform for fully agent-based modeling of multi-modal transportation. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3, pages 1501–1502. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[9] M. Jakob, M. Pěchouček, M. Čáp, P. Novák, and O. Vanek. Mixed-reality testbeds for incremental development of HART applications. IEEE Intelligent Systems, 27(2):19–25, 2012. http://dx.doi.org/10.1109/MIS.2012.2

[10] J. Jlassi, J. Euchi, and H. Chabchoub. Dial-a-ride and emergency transportation problems in ambulance services. Computer Science and Engineering, 2(3):17–23, 2012. http://dx.doi.org/10.5923/j.computer.20120203.03

[11] E. Kamar and E. Horvitz. Collaboration and shared plans in the open world: Studies of ridesharing. In IJCAI, volume 9, page 187, 2009.

[12] E. Lioris, G. Cohen, and A. de La Fortelle. Overview of a dynamic evaluation of collective taxi systems providing an optimal performance. In Intelligent Vehicles Symposium (IV), 2010 IEEE, pages 1110–1115. IEEE, 2010.

[13] L. Quadrifoglio, M. M. Dessouky, and F. Ordonez. A simulation study of demand responsive transit system design. Transportation Research Part A: Policy and Practice, 42(4):718–737, 2008. http://dx.doi.org/10.1016/j.tra.2008.01.018

[14] L. Quadrifoglio and M. Dessouky. Insertion heuristic for scheduling mobility allowance shuttle transit (mast) services: sensitivity to service area. Computer-Aided Systems in Public Transport, Springer Series: Lecture Notes in Economics and Mathematical Systems, 600, 2007.

[15] J. de Dios Ortuzar and L. G. Willumsen. Modelling transport. 1994.

[16] A. M. Geoffrion. The purpose of mathematical programming is insight, not numbers. Interfaces, 7(1):81–92, 1976. http://dx.doi.org/10.1287/inte.7.1.81

[17] R. W. Hall. Discrete models/continuous models. Omega, 14(3):213–220, 1986. http://dx.doi.org/10.1016/0305-0483(86)90040-X

[18] A. Langevin, P. Mbaraga, and J. F. Campbell. Continuous approximation models in freight distribution: An overview. Transportation Research Part B: Methodological, 30(3):163–188, 1996. http://dx.doi.org/10.1016/0191-2615(95)00035-6

[19] D. M. Stein. Scheduling dial-a-ride transportation systems. Transportation Science, 12(3):232–249, 1978. http://dx.doi.org/10.1287/trsc.12.3.232

[20] L. Bodin and B. Golden. Classification in vehicle routing and scheduling. Networks, 11(2):97–108, 1981. http://dx.doi.org/10.1002/net.3230110204

[21] D. Hauptmeier, S. O. Krumke, and J. Rambau. The online dial-a-ride problem under reasonable load. In Proceedings of the 4th Italian Conference on Algorithms and Complexity, CIAC '00, pages 125–136, London, UK, UK, 2000. Springer-Verlag.

[22] M. Lipmann, X. Lu, W. E. Paepe, R. A. Sitters, and L. Stougie. On-line dial-a-ride problems under a restricted information model. Algorithmica, 40(4):319–329, 2004. http://dx.doi.org/10.1007/s00453-004-1116-z

[23] A. Haghani and M. Banihashemi. Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints. Transportation Research Part A: Policy and Practice, 36(4):309 – 333, 2002. http://dx.doi.org/10.1016/S0965-8564(01)00004-0

[24] N. H. M. Wilson, J. Sussman, L. Goodman, and B. Hignnet. Simulation of a computer aided routing system (cars). In Proceedings of the third conference on applications of simulation, pages 171–183. Winter Simulation Conference, 1969.

[25] A. C. Regan, H. S. Mahmassani, and P. Jaillet. Dynamic decision making for commercial fleet operations using real-time information. Transportation Research Record: Journal of the Transportation Research Board, 1537(1):91–97, 1996. http://dx.doi.org/10.3141/1537-13

[26] F. Klügl. Agent-based simulation engineering. PhD thesis, Habilitation Thesis, University of Würzburg, 2009.

[27] D. Edwards, A. Elangovan, and K. Watkins. Reaching low-density urban areas with the network-inspired transportation system. In Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, pages 826–831. IEEE, 2012.

[28] N. Agatz, A. L. Erera, M. W. Savelsbergh, and X. Wang. Dynamic ride-sharing: a simulation study in metro atlanta. Procedia - Social and Behavioral Sciences, 17(0):532 – 550, 2011. Papers selected for the 19th International Symposium on Transportation and Traffic Theory.

[29] S. F. Cheng and T. D. Nguyen. Taxisim: A multiagent simulation platform for evaluating taxi fleet operations. In Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology Volume 02, pages 14–21, 2011.

[30] F. P. Deflorio, B. Dalla Chiara, A. Murro, and M. A. SpA. Simulation and performance of drts in a realistic environment. In Proceedings of the 9th Meeting EWGT on Intermodality, Sustainability and Intelligent Transportation Systems and 13th Mini EURO Conference on Handling Uncertainty in the Analysis of Traffic and Transportation Systems, 2002, 2002.

[31] W. A. Bailey Jr and T. D. Clark Jr. A simulation analysis of demand and fleet size effects on taxicab service rates. In Proceedings of the 19th conference on Winter simulation, pages 838–844. ACM, 1987. http://dx.doi.org/10.1145/318371.318705

[32] K. Shinoda, I. Noda, M. Ohta, Y. Kumada, and H. Nakashima. Is dial-a-ride bus reasonable in large scale towns? evaluation of usability of dial-a-ride systems by simulation. In Multi-agent for mass user support, pages 105–119. Springer, 2004. http://dx.doi.org/10.1007/978-3-540-24666-4_7

[33] M. Diana. The importance of information flows temporal attributes for the efficient scheduling of dynamic demand responsive transport services. Journal of advanced Transportation, 40(1):23–46, 2006.