Implementation of an autonomous taxi service for the MATSim traffic simulation framework

Master thesis in Complex Adaptive Systems

Sebastian Hörl
9 May 2016

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Energy and Environment

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IVT Institut für Verkehrsplanung und Transportsysteme
Institute for Transport Planning and Systems

Contents

1 Introduction

Autonomous vehicles are expected to have a great impact on how the traffic situation in our cities will look like in the not so far future. During the last years, autonomous vehicle technology took great leaps forward, examples being Google's self-driving cars (Google, 2016) or Tesla's latest software updates on autonomous parking functionality (Tesla, 2016). Furthermore, renowned car manufacturers such as Audi recently joined the competition. Ambitious tests in real-world conditions are fostered all over the world, examples being Volvo's DriveMe project with 100 autonomously driving cars on the highways of Gothenburg (City of Gothenburg, 2016) and the LUTZ pathfinder project, establishing the use of three autonomous transport pods in the city center of Milton Keynes (Transport Systems Catapult, 2016).

Especially big cities like Singapore could gain a lot from autonomous car technology, be it from privately owned cars to publicly owned services. A floating fleet of autonomous vehicles (AVs) could have a drastic impact on land usage, reducing the area of parking space needed in the urban environment. At the same time, it has the potential improve road safety, access to transportation, congestion, and emissions (Kheong and Sheun, 2014). In fact, it is predicted that an AV fleet size of one-third the size of the number of private cars could serve the associated demand that is generated today (Spieser et al., 2014). From a user perspective, shared autonomous taxis could solve the classic "last mile problem" where AVs could bridge the transport gap from the closest public transport facility to the homes of the people (Litman, 2014). As soon as a particular supply of AVs is reached, prices are predicted to become highly competitive to if not even cheaper than private car ownership (Chen, 2016).

On the other side, with the adaptation of autonomous transport, fundamental moral problems (Hevelke and Nida-Rümelin, 2015) need to be discussed, and related questions in liability and ownership need to be answered (Anderson et al., 2014). In general, predicting how and when autonomous vehicles will be on the roads is a highly complex problem, and most of the proclaimed benefits can easily be defeated due to a lack of reliable data.

To give an example, one of these benefits is having less congestion (International Transport Forum, 2014). However, it is claimed that to make a passenger comfortable in a self-driving car, significantly smaller accelerations and decelerations are allowed, and thus the overall movements on the roads will be slowed down (Le Vine et al., 2015). Therefore, having only assumptions on how people would experience shared AV services, replacing a huge percentage of contemporary cars with autonomous ones could in fact increase overall

congestion.

In consequence, while the technological development in autonomous driving already came a long way, there is still much demand for research on an upper level, looking at the overall economic and societal picture (Silberg et al., 2013; Schoettle and Sivak, 2014). A valuable tool for doing this is the use of traffic simulation (Fagnant et al., 2014; International Transport Forum, 2014). Numerous efforts have been made in predicting the usefulness of AVs for a city regarding AV supply, i.e. answering the question how many shared AVs would be needed to serve the existing demand (Bösch, 2015). On the contrary, fewer results are available on how the new travel mode would interact with existing travel options and how customer preferences influence the mode choice.

The agent- and activity-based traffic simulation framework MATSim (Horni et al., 2015) allows for such research (Boesch and Ciari, 2015) by taking into account assumed customer preferences and letting people interact with a newly added means of transportation. The framework has been successfully used in a range of studies from autonomous taxi services in Berlin and Barcelona (Bischoff and Maciejewski, 2016) up to the simulation of the whole transport network of Singapore (Erath et al., 2012). While the framework allows for a quite precise prediction of traffic flows for scenarios involving car traffic and public transport, it yet does not allow the simulation of dynamically acting autonomous vehicles embedded in the overall traffic situation.

Therefore, the purpose of this thesis will be to implement means of simulating autonomous taxis within the MATSim framework. Subsequently, measurements on how people would switch to the new transport mode given certain supply levels, acceptance levels, and pricing schemes will be made. A major challenge will be to account for an acceptable simulation speed while keeping the dynamic detail, which is needed in order to simulate intelligently acting AV taxi fleets. At the same time the functionality should be kept as versatile and extendable as possible in order to make it possible to gain research results on a multitude of factors, which influence the adaptation of autonomous vehicles.

In the scope of this thesis, a basic model of autonomous vehicles, which are transporting one passenger, is developed. Many interesting aspects such as shared AVs, AVs for the purpose of feeding public transport facilities or simulating a refuel/recharging infrastructure can be subject of future research. Since the model relies on data on how likely people are to use autonomous technology, the model at the moment will be mainly pointed towards finding qualitative results on the interplay between factors that define the traffic situation. However, more and more data on customer preferences, actual experiences of AV technology and pricing information will become available over the next years, leading

the model to give a more accurate and reliable look into the future.

Given those data sets, the simulation developed in the thesis has the potential to act as a valuable tool in transport planning and urban development. It will give great aid in the implementation of the needed infrastructure, the development of pinpointed transport solutions and a restructuring of the present traffic network to account for the adaptation of autonomous vehicles.

The thesis at hand is structured in four main parts: Chapter 2 will introduce how traffic simulation is performed in the MATSim framework and highlight the aspects that are important to know for the implementation of autonomous vehicles. Chapter 3, as another prerequisite for setting up a meaningful simulation, covers the adaptation of a readily available MATSim traffic scenario to the increased network resolution, which is needed here. In chapters 4 and 5, the technical implementation of the AV simulation will be explained in detail on two abstraction layers, first introducing a new framework for simulating dynamically acting agents in MATSim and then creating a model of autonomous taxis based on that. Then, in chapter 6 the model will be tested with a variety of parameter configurations, giving insights on the general working of the model and predictions of AV usage in the artificial Sioux Falls test scenario. Finally, chapter 7 will give an outlook on a multitude of possible extensions of the model and further research questions that can be answered using the model at hand.

2 Simulation Framework

The investigations of autonomous vehicles in this thesis will be based on the agent-based transport simulation MATSim (Horni et al., 2015). The following sections will outline how the framework works and where it can be extended to shape it towards an autonomous taxi simulation. An overview will be given on which components need to be modified and how the final transport situation will result from all the different parts that are playing together in MATSim.

2.1 Agent-based transport simulation

The approach that is used in MATSim is to simulate a virtual population of a city on a per-agent timestep-based level. At the beginning of a simulation run, each person in the synthetic population has an initial plan of what it is supposed to do during the simulated day. Those those plans consist of two elements:

Activities have a start and an end time, as well as, depending on the respective scenario, certain constraints on when the earliest start or latest end time could be, i.e. how much the current values could be alternated to still give a realistic time frame. Furthermore, minimum durations can be defined for which and agent has to stay at a certain activity location. Those locations are given as facilities, which have specific coordinates on the scenario map. Usual activities are "home" (which usually is the first plan element of a day) and "work". More elaborate simulations can additionally use an arbitrary number of secondary activities.

Legs are the second type of plan elements. These describe connections between two activity locations and contain information like which mode of transport the agent will use (e.g. "car", "public transport", or "walk") and, depending on the selected mode, further data like the route that should be taken through the street network.

A typical day plan of a MATSim agent can be seen in fig. 1. The agent starts at home, then walks to his job, stays there for a certain time and then goes back home. Each agent in a popuation (which can range from several hundred to hundeds of thousands) has its individual plan that is executed when one day is simulated.

The network, on which the simulation is taking place, is described through nodes and connecting links. These are defined by a specific flow capacity which tells the simulation

Figure 1: A typical agent plan in MATSim

how many vehicles are able to pass the link within a certain time frame while the length and an average link speed determine how fast vehicles will travel to the next node.

The whole MATSim simulation is performed time-step by time-step. In a single simulation step (usually 1 second) the agents that are currently in an activity do not need to be taken into account unless their scheduled activity end is reached. The other agents, which are currently on a leg, are simulated in a dedicated traffic network simulation. This simulation moves the agents according to the capacity and current congestion from the start node to the end node of the respective links. Traffic is not simulated on a micro-level (i.e. the vehicles do not have distinct positions along the link) to increase the simulation speed. Consequently, congestion on the traffic network is emerging from the single travel plans of all the agents. If there are too many agents who try to use one route at the same time, the overall congestion will increase.

The actual routes that are being taken by the agents are generated based on the fastest path through the network for a certain leg, with a certain amount stochasticity added into the pathfinding process in order to avoid artificially created bottlenecks. This approach allows the optimization of the plans to be separated from the actual simulation of the plans, but is not suitable for the simulation of an AV taxi service, where routes through the network must be generated dynamically, based on the current demand. Therefore, to simulate dynamic agents for AVs, which are not statically residing in a fixed-timed activity or a predefined leg route, significant changes need to be made, and some of the computational advantages of the simulation approach need to be partly circumvented. This, however, mainly addresses implementational details and will be discussed later throughout the thesis in chapter 4.

All steps described above, i.e. the simulation of activities and legs during a whole simulation day, are summarized as the "Mobility Simulation" or, in short, Mobsim of the MATSim framework.

2.2 Utility-based scoring

As pointed out in the last section, individual travel decisions might lead to waiting times on the network, which then can also lead to late arrivals at the designated activity locations, which usually would be disadvantageous in the real world. Therefore, it might be beneficial for an agent to reconsider the time and route choices that had been made for the current day, just as a real person would do.

In order to "know" whether a plan worked out well or was disadvantageous, the single elements need to be weighed and quantified. This is done using the Charypar-Nagel scoring function (Horni et al., 2015):

$$S_{plan} = \sum_{q=0}^{N-1} S_{act,q} + \sum_{q=0}^{N-1} S_{trav,mode(q)} \tag{1}$$

It combines the marginal utilities of all activities ($S_{act,q}$) ranging over the N activities in a plan and the marginal utilities for all the legs in between.

The marginal utility for the activities, among other factors, depends on how long the activity has been performed, whether the agent needed to wait to start it (due to an early arrival) or whether it was forced to leave early. For more details the complete computation is shown in (Horni et al., 2015).

For the scope of this thesis the travel utility is more interesting. A basic version for a single leg q can look as follows:

$$S_{trav,mode(q)} = C_{mode(q)} + \beta_{trav,mode(q)} \cdot t_{trav,q} + \gamma_{d,mode(q)} \cdot \beta_m \cdot d_{trav,q} \tag{2}$$

Mode Choice The first term, $C_{mode(q)}$, describes a constant (dis)utility for the choosing a certain mode for the leg. It can be interpreted as how "favorable" going on a trip in the specific mode is and is negative.

Travel Time The parameter $\beta_{trav,mode(q)}$ is the marginal utility of traveling, which is multiplied by the time spent on the leg $t_{trav,q}$. It signifies how favorable it is to spend time on such a leg, i.e. the longer one needs to stay in a car or public transport, the larger the disutility gets (and therefore the parameter is usually negative). Values which are smaller, therefore, stand for travel modes where time is spent less useful or comfortably. The unit is "utility per time".

Travel Cost The third element involves the (positive) marginal utility of money β_m,

which is a universal simulation parameter and describes how the utility of money can be weighed against e.g. time., the unit being "utility per monetary unit", e.g. EUR. It is multiplied by the (negative) monetary distance rate $\gamma_{d,mode(q)}$, which states, to how much disutility per spanned distance the leg will lead. For the given example it would be stated as "EUR per meter". The parameter is useful for imposing distance-based fares in a transport mode and thus making it monetarily attractive or unattractive compared to other ways of traveling.

Beyond these parameters, which are usually used for all travel modes, there are a number of additions, e.g. for public transport, or yet unused options, such as a direct marginal utility of distance traveled.

For the purpose of simulation autonomous taxi services, one addition is made:

$$S_{av} = C + \beta_{trav,av} \cdot t_{trav} + \gamma_{d,av} \cdot \beta_m \cdot d_{trav} + \beta_{wait,av} \cdot t_{wait} \tag{3}$$

Here, β_{wait} is the marginal utility of waiting time, quantifying how disadvantageous it is to wait for an AV to arrive.

The utility computations presented here are used in the "scoring phase", which is taking place in MATSim after one simulation day (in fact, usually 30h are used) has been performed. Then all experienced legs and activities are scored, and each agent is assigned a final score, depending on which further replanning is done, as described in the next part.

2.3 Evolutionary replanning

The last step is to make all the agents replan their day in order to "learn" more optimal plans which make sure that they arrive on time at the activity locations and avoid congestion. This is done using an evolutionary algorithm in the following way:

Usually, an agent will start out with one quite random plan, go through it during one whole day and then get a score for it. Afterward, in some iterations, this plan is copied and modified slightly. Those modifications can happen with respect to start and end times of activities, mode choices for certain legs, etc. So after one iteration the agent might already have two plans to select.

Before the next day starts, one of the available plans is selected according to a certain strategy. The standard approach is to do a multinomial selection with respect to the previously obtained plan scores. So one after another plans, will be created, scored,

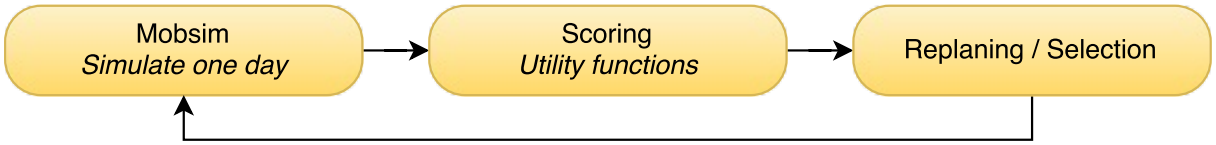

Figure 2: The basic co-evolutionary algorithm of MATSim, showing the three main stages: Mobsim, Scoring and Replanning/Selection

modified, rescored and so on. Because of the selection process, which favors high scores, better and better choices will be made.

However, this is done for each and every agent, so while improving the performance of one agent's plans, this might effect the performance of other agents negatively, which is especially true if one thinks about the example of highly congested roads due to too many agents choosing the same route. Finally, though, the algorithm reach at a quasi-equilibrium, which in MATSim is usually referred to as the "relaxed state", in which the average score of the used plans stabilizes within a reasonable variance.

Each "day" that is simulated in this manner is usually called an "iteration" of the simulation. It is common to divide these iterations into two parts. The first one is the "innovation phase", where plans have a certain probability to be modified while innovation is turned off in the second phase. This means that the agent will only choose among the present plans in his repertoire (usually around 5) and rescore them again and again, until the most favorable is selected.

All of the above is known as the "replanning" in MATSim. Putting everything together, a whole cycle in a MATSim simulation can be seen in fig. 2. Since the final traffic situation evolves from the evolutionary choice in the replanning and selection, as well as from the emergent congestion in the Mobsim, this whole cycle is usually referred to as a "co-evolutionary" algorithm.

Figure 3 shows a typical progression of the population-wide score in a MATSim simulation. What one can see there is an average of the worst and best plans of each agent. Additionally the mean of the per-agent average scores in their list of plans is displayed. Finally, one can see the average of all the plans that have been executed by the agents in a particular iteration.

The first phase until iteration 100 is the innovation phase, where time and mode choices can be made, while it is turned off at iteration 100. There, because now the best plans must adapt to the overall situation, the best plans are loosing in utility. Finally, a quite stable population-wide relaxed state is reached.

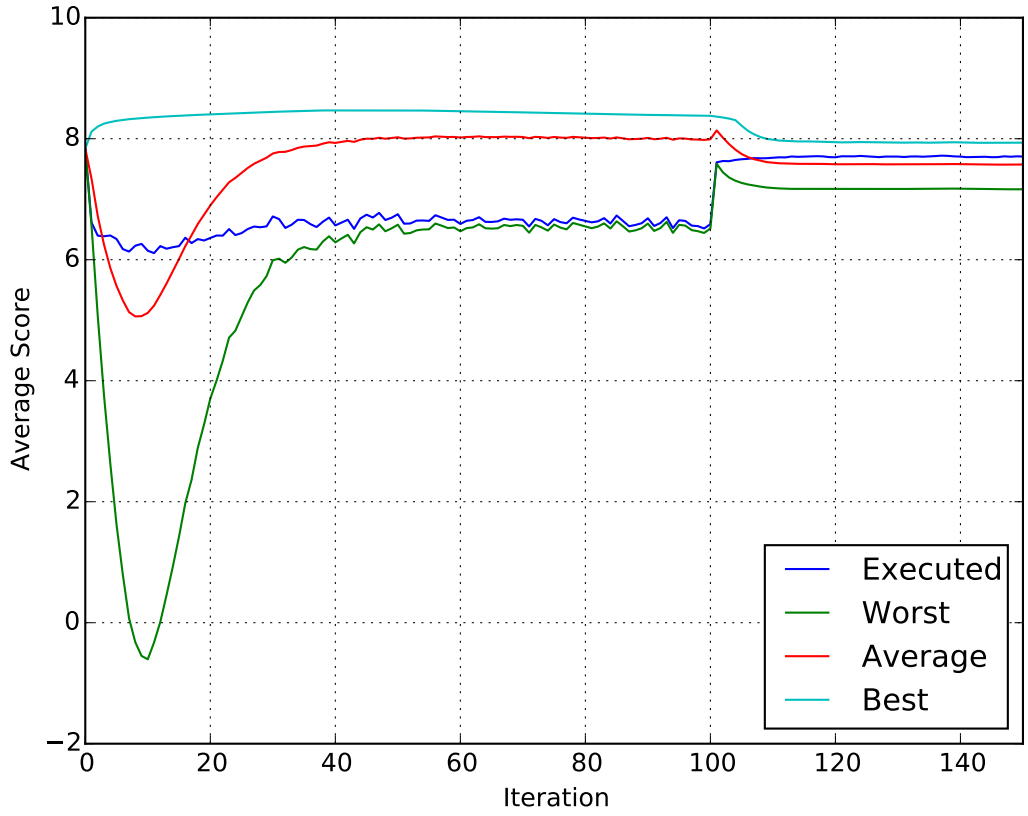

Figure 3: Typical progression of the agent scores throughout a MATSim simulation. "Worst" means that the plans with the worst scores from all the agents are taken and averaged, the same applies for the other graphs.

2.4 Queue-based simulation

The heart of the simulation in MATSim is the queue simulation, or more briefly, the QSim. This part of the framework is iterating through all the agents that need to take action in the current simulation step.

Itself, the QSim is split into the handling of agents which are currently performing an activity (i.e. are in an idle state in terms of the traffic simulation) and another part, the Netsim, which is simulating the traffic network.

When running the Netsim, the agents are moved on the network according to their current route and dependent on congestion. After moving an agent, the Netsim checks whether the agent should end its trip at the current link to which he has been moved. If this is the case, the agent is removed from the Netsim and the next agent state is computed.

The result of this computation is either that the agent wants to start a leg, in which case he is reinserted into the Netsim, or that he wants to start an activity, which will make the agent be added to the activity queue.

This activity queue is the other main component of the QSim. In fact, it is a priority queue, where all agents are sorted according to the time at which they want to end the next activity. So when adding an agent to the activity queue, it first is checked when the activity should be ended and then the agent is inserted at the corresponding position in the queue.

The processing of this queue in the QSim for each simulation step then works as follows: The first element of the queue is looked at and it is checked, whether the agent should already end the activity. If this is not the case, the simulation step is already finished. On the other hand, if the activity should be ended now (or previously if the time resolution of the simulation is quite high), the agent is removed from the top of the queue and the next state is computed as described above. Then the new top element of the queue is examined. The whole QSim simulation is schematically rendered in fig. 4.

The big advantage of this simulation architecture is as follows: When an agent is in an activity, no computation needs to be performed. So instead of polling all the agents in every simulation step to check if they want to end an activity, the computational demand is much lower when using the queue, since many agents can be skipped. The sequential processing of the priority queue is very fast in terms of computational complexity ($\mathcal{O}(1)$ for checking the top element and $\mathcal{O}(\log n)$ for fetching it) (Java API, 2016).

Furthermore, the same concept is used within the Netsim to speed up the computation of

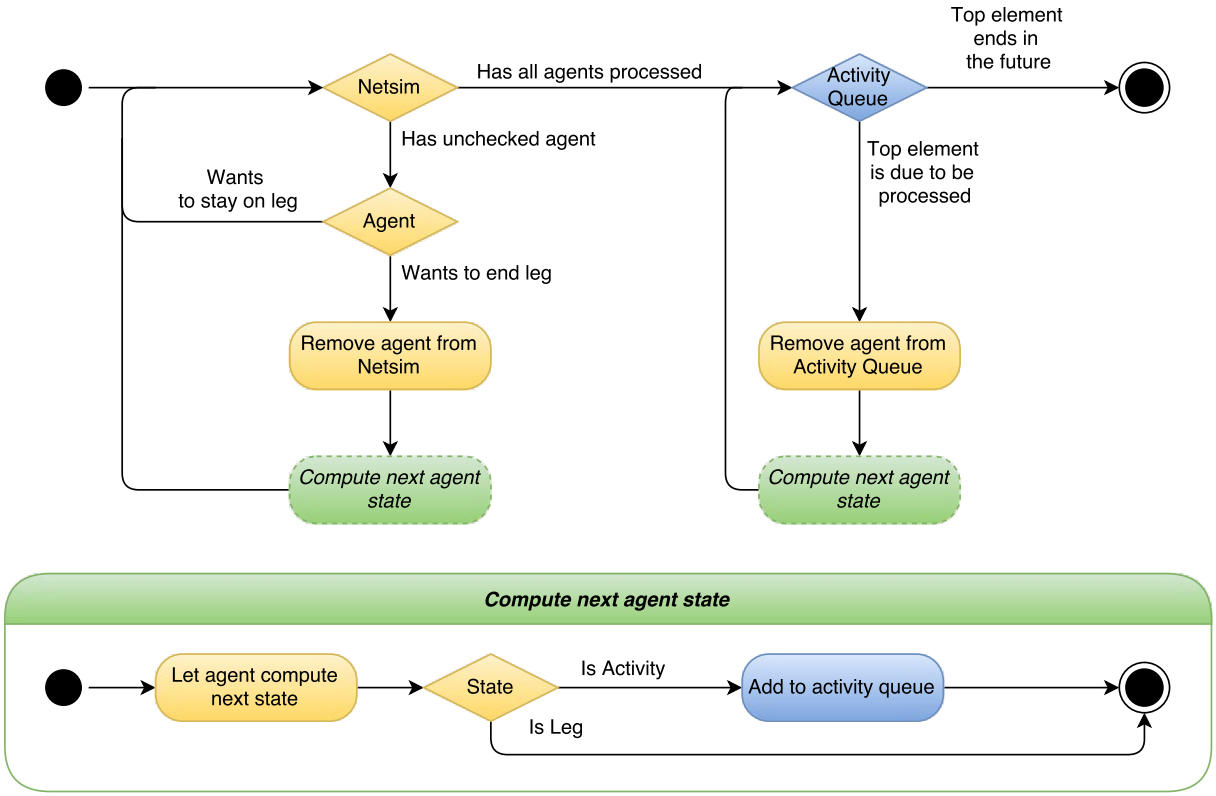

Figure 4: Simplified structure of the QSim in MATSim.

the traffic situation. In both cases, for the QSim and Netsim, those savings in computation time naturally decrease the versatility of the simulation environment.

One major drawback is that if an agent, which is already queued, should abort its current activity, it needs to be removed from the queue and added at a new position. Both operations are quite costly for the priority queue (Java API, 2016), so if more and more reschedulings are needed, the computational overhead can become quite large.

However, for truly dynamic agents, like autonomous vehicles, it is necessary to adopt their plans frequently and thus some thought needs to be put into how to achieve this freedom while still keeping as many advantages from the existing simulation architecture. Chapter 4 will explain how this problem has been overcome in this thesis.

3 The Sioux Falls Scenario

The City of Sioux Falls in South Dakota (fig. 5) has been a classic test case in transport research for more than four decades, being first mentioned in this context in Morlok et al. (1973). While none of the numerous implementations of the network are intended to be accurate and realistic with respect to the actual City of Sioux Falls, they are merely aiming towards providing downscaled, computationally tractable test cases for transport planning and simulation problems.

In Chakirov and Fourie (2014), the scenario ("Sioux-14", fig. 6) has been adapted to the MATSim framework. A sparse street network, which is computationally easy to handle by the simulation, was introduced. It consists of 27 nodes and 76 links, representing the main arterial roads of the city, split further down into 282 nodes and 334 links in order to arrive at partial link sizes of less than $500m$. This is necessary because MATSim agents start their travels at the start node of a link and thus a high resolution is needed to avoid unrealistic clustering.

On the supply side, there is furthermore a public transport network, which consists of 5 lines with bus stops along the arterial roads in a distance of $600m$. The stops are placed at a distance of $5m$ perpendicular to the road and departures from the lines' respective start links take place every 5 minutes.

Much of effort has been put into the modeling of the demand side, covering the realistic generation of home locations, workplaces, secondary activity locations, as well as the distribution of socio-economic factors such as age, car ownership and gender across a synthetic population, as it is needed for the agent-based simulation.

In this regard, the scenario provides a lightweight example of a complete MATSim simulation, where it is easy to test different parameters and extensions to the framework on a near-realistic baseline scenario. However, while working with the original Sioux-14 network during the course of this thesis, it became evident that the simplified structure of the underlying traffic network does not provide enough resolution for the simulation of autonomous vehicles.

The main reason is that agents, who choose to take a car in the Sioux-14 network, probably living in the middle of the rectangular regions of the network would be teleported to the nearest link to start their travels. This would also be true for autonomous vehicles, which is not a realistic assumption in both cases, especially when comparing it with public transport, where agents in MATSim are explicitly penalized for covering the distance between home and bus stop by foot. Furthermore, the effect of people being

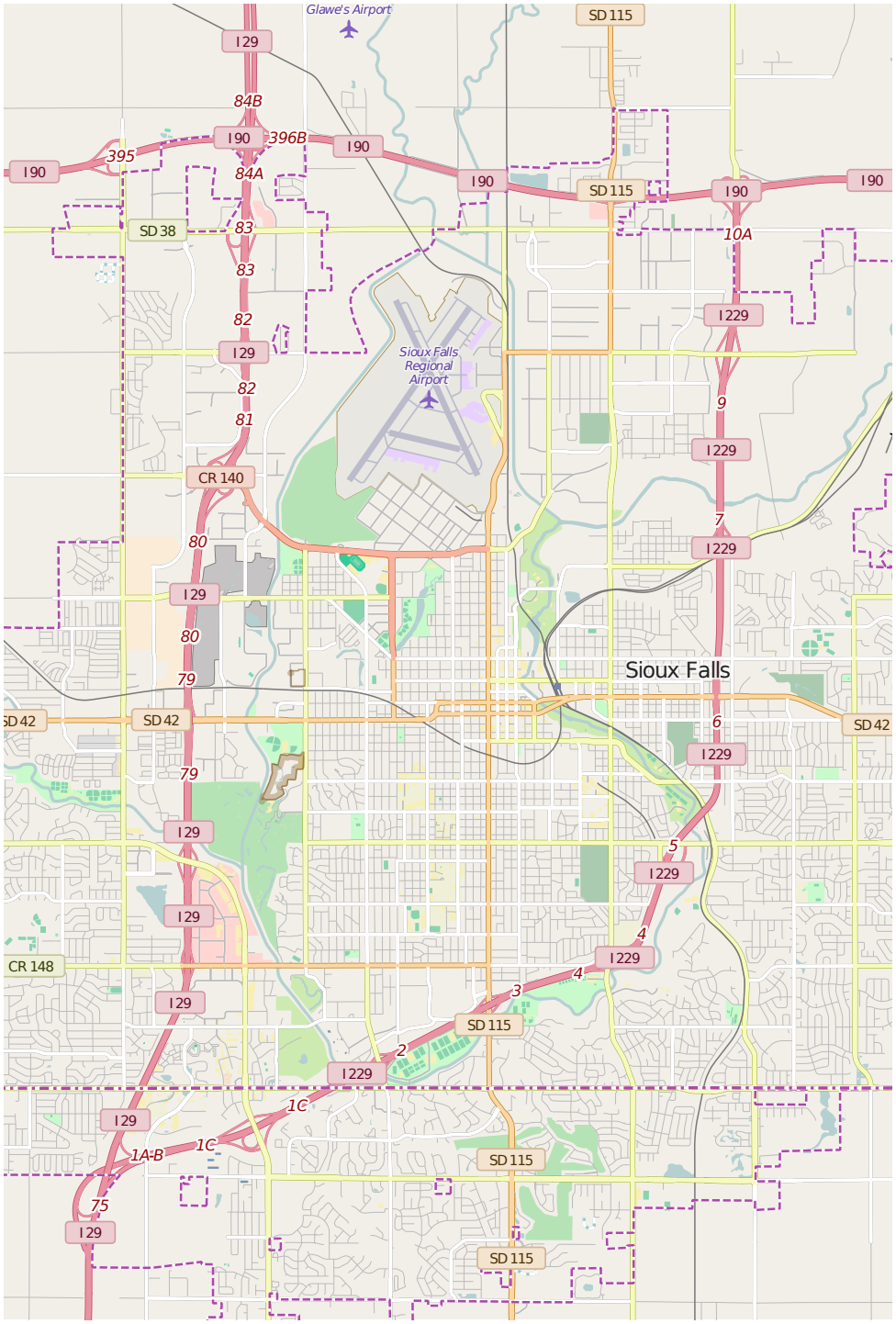

Figure 5: Map of Sioux Falls from OpenStreetMap. Longitude from −96.8105° to −96.6653° and latitude from 43.4729° to 43.6286°.

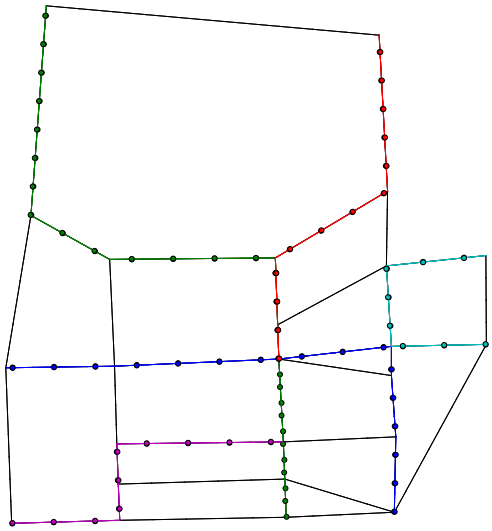

Figure 6: Sioux-14 road and public transport network

more inclined to opt for an autonomous taxi when living far from public transport can only be convincingly simulated on a finer network.

The following sections will describe how, starting from the initial Sioux-14 scenario, a new more fine-grained versatile test scenario for MATSim has been developed, which in turn has been used as the basis of the following investigations in this thesis.

The new Sioux-16 network, which has been developed in this thesis, is based on the demand model of Sioux-14, which means that all locations for homes, workplaces and secondary activities are kept equal, while it differs on the supply side, aiming to resemble the original scenario as closely as possible. The following sections will describe, how the Sioux Falls network from OpenStreetMap (with the state as of 18 Apr 2016) has been converted and adjusted to be compatible with MATSim and closely match the prior version of the test scenario. Furthermore, it will be explained, how the public transport network has been adapted to the fine-grained Sioux-16 scenario.

3.1 Network generation and adjustment

For the creation of the new scenario, an area covering Sioux Falls has been captured from OpenStreetMap, which can be seen in fig. 5. Using the MATSim exporter in the JOSM tool [1], a simplified, MATSim-compatible network of the selected region has been created. In this process, all primary, secondary and tertiary streets have been selected, while road types further down the hierarchy (for instance residential streets) have been omitted.

[1] https://josm.openstreetmap.de/

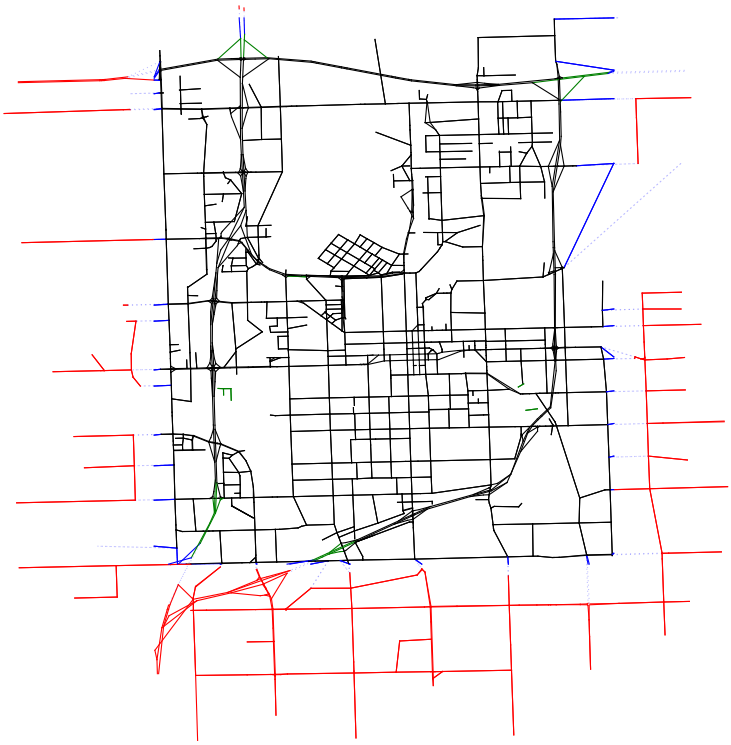

Figure 7: Automatic network modifications: Red links have been removed while blue links have been shrunk to the closest border points of a bounding box enclosing the facilities of the scenario. Green lines have been removed either for being detached from the main network being sinks/sources.

However, some adjustment was needed for the network to play nicely with the given facilities from the Sioux-14 scenario. Most importantly, the network from OSM was defined in the EPSG:3857 coordinate system, while Sioux-14 uses EPSG:26914. Therefore, in a first step, the coordinates of all the generated nodes needed to be converted to the old system.

In a second step, the positions of all facilities in Sioux-14 have been obtained and a bounding box with a margin of $500m$ around them has been computed. Subsequently, all links, which were located out of the bounding box, were removed from the network, while those who were crossing the borders have been cut to fit into the area. The initial network with all removed (red) and cut (blue) roads can be seen in fig. 7. In total 352 outside links have been removed, and 83 links have been adjusted.

After that, a cleanup up the network needed to be done, partly because of artifacts due to the filtering of road types, partly because of the removal of the external sections. This removal was done in a couple of steps:

1. The lengths of all the links have been updated to the L^2 distance of their respective

start and end nodes in the new coordinate system.

2. The network has been searched for sources (nodes that only have outgoing links) and sinks (nodes that only have incoming links). Those have been removed, because they make no practical sense. This proecdure lead to 53 removed links in total.

3. One seed node has been defined, which definitely belongs to to the street network and then by traversing the all paths from that node, it has been determined, which streets belong to the main network. All remaining nodes and links, which have become detached from this main network have been removed (11 nodes and 14 links).

4. Finally, the network has been searched for duplicate links with the exact same start and end node. Only the first of those links have been kept, which lead to the removal of 3 duplicates.

5. In Sioux-14, the links have been split such that there are no connections longer than $500m$. This procedure has also been applied to the network at hand, leading to 389 split links, which have been cut into a number of equal parts with less than $500m$ length depending on their overall length.

Regarding nodes, these procedures in total lead to an increase of nodes from 1392 to 1806 and in an increase of links from 2957 to 3335. The links that have been removed during the cleanup are colored in green in fig. 7.

3.2 Public Transport Adaptation

The adaptation of the public transport network to the new ("Sioux-16") scenario posed some challenges that needed to be solved:

- The links of the original network did not exist anymore, obviously the routes for the different public transport needed to be mapped to the new network.

- The stops from the original network could not be mapped easily to the new roads, partly because some streets in Sioux-14 were "invented", only approximating connections in the finer network on a very coarse level, but also because roads that consisted of two overimposed links for both directions were now split up into two spatially distinct lanes (as can be seen in the upper left part of fig. 8).

In order to get a rough routing for the bus lines, the main nodes of the Sioux-14 network have manually been mapped by hand to nodes in the Sioux-16 network. This made it possible to obtain new public transport roads in terms of those guide points: For each of the lines it has been defined, which guide points should be traversed and in which order. Then, the Dijkstra algorithm (Dijkstra, 1959) with travel time as the objective has been used to find the shortest path from waypoint to waypoint. After fitting those partial paths together, the whole routes in terms of the links of the Sioux-16 network have been obtained. As can be seen in the colored routes in fig. 8 this made it easy to obtain routes that take into account the specific map structure (for instance the highway on the upper left or the one-way streets, which are traversed by the blue line in the center of the map).

The generation of bus stop facilities was a more challenging problem. Given the location from Sioux-14, one could roughly match the stops to links along the new routes, which worked in principle. However, using this approach, bus stops were cluttered all along the lines. With the intention of having a rather realistic network some conditions needed to be taken into account:

- The bus stops of one line should be on opposite sides of a street and not have a large longitudial distance. While satisfiying results could obtained, for instance in the center with the blue line, the same approaches did not work well for the highway connectors on the left for the green line, and vice versa.

- The bus stops of parallel lines should be at the same locations, i.e. in the center where the green line uses the same roads as the red one, the same locations should be chosen. This constraint usually interfered with the approaches that took into account the first constraint (especially in the center for the blue line).

- Finally, some of the Sioux-14 locations where completely off the network in Sioux-16, for instance for the red line, where one can see a bend in the diagonal connection, which was modelled as a straight line in Sioux-14.

Given all those constraints the best approach seemed to put in manual work with some automated help. The final approach made use of a small program, written to manually choose the stop locations along all roads and then subsequently choose which stop locations should belong to which line. In this step one did not take into account the cases of two lanes, as just the general positions needed to be known (e.g. for the blue line in the center an approximate position between the two lines would be chosen).

In another step, the locations that had been assigned to each line have been assigned to the respective links along the line. So for the blue one, the average points in between would have been assigned to a link underneath for one direction, and another stop would be created for the link above. This resulted in a final set of stop locations.

Furthermore, some stop locations were located on one single link. This can happen if there is a link of roughly $500m$ and the stops are for instance located at $1m$ and $499m$ along the link. In those cases, the network needed to be broken up at those positions. In the current scenario, this leads to the splitting of only four links.

Finally, according to the setup of Sioux-14 the stop locations have been moved to a position normal to the link direction, with a distance of $5m$. Additionally a schedule with buses departing in $5min$ intervals has been created.

The final public transport network can be seen in fig. 8. It features 150 stops with an average distance of $520m$ and a median of $566m$. This is a result of not being able to put the stops as accurately in intervals of $600m$ as it is possible in Sioux-14. Moreover, the L^2 distance between two stops along the routes have been used instead of a measure along the actual path. The minimum and maximum distance between stops are $218m$ and $847m$, respectively.

All steps that have been described above have been implemented in a reusable and parametrized way. For instance, one could choose to allow a maximum link length of $5km$ and would still obtain a valid network at the end, probably just with a larger number of split links during the processing of the public transport.

3.3 Scenario Calibration

The Sioux-14 network features artificially chosen link capacities, which are based on a minimum of two and a maximum of three lanes per link, where three have only be chosen for a fraction of highway links (Chakirov and Fourie, 2014). While the added minor streets in Sioux-16 should not make too much of an impact on overall capacity, the highways and main arterials of the real network usually feature greater capacity (for instance by providing *four* lanes on the western highway).

Looking at fig. 9, it can be seen that the travel time (of cars) in the new network is lower (red), compared to Sioux-16 (black). This effect can partly be explained by the increased capacity, but also by the multitude of new options to choose the most effective route for a trip through the fine-grained network. In fact, the route choice might be the major

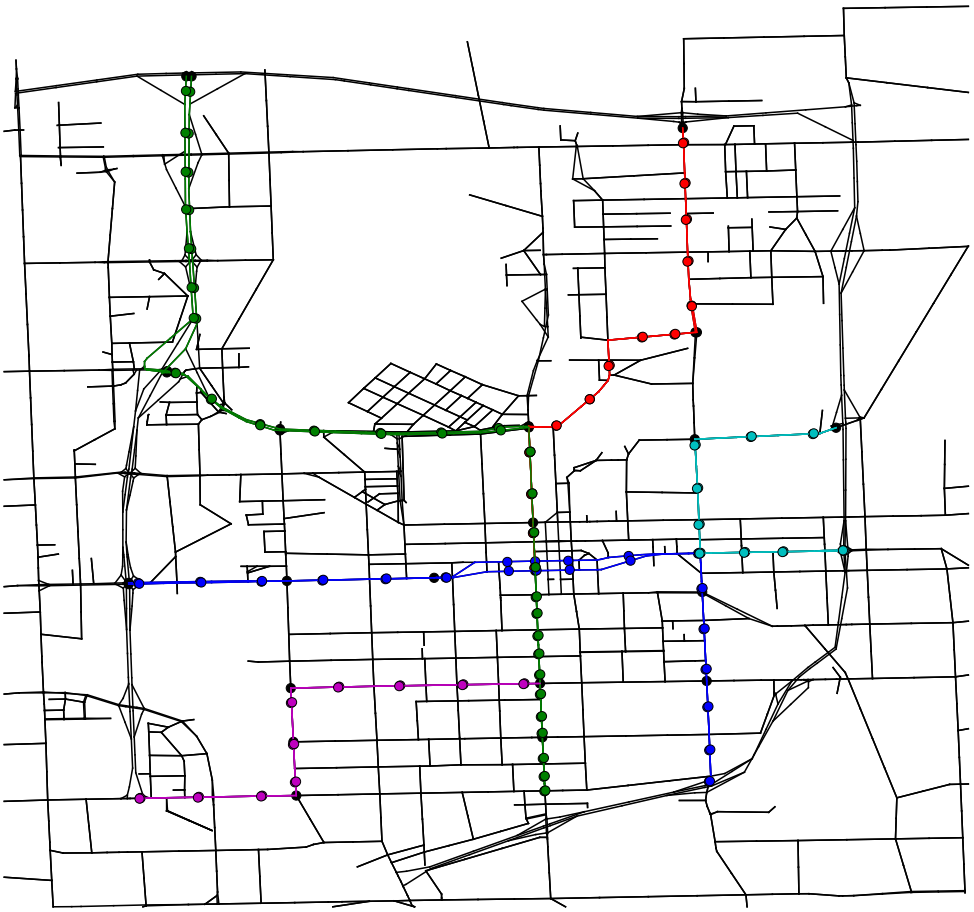

Figure 8: Sioux-16 network with five public transport lines and respective stop facilities

influence when comparing with the data from fig. 10. There the decrease in link speeds (relative to the freespeed) can be seen, which is quite similar, indicating a comparable amount of congestion in the network.

Depending on how important the comparability to Sioux-14 in a specific scenario is and what time of the day should be compared, it might be beneficial to adjust the flow capacity of the network[2]: In terms of travel times a scaling of 50% would resemble the afternoon peak way better than the 100% version, while the morning peak would best be recovered by a value in between (fig. 9). A similar situation arises for the link speeds in fig. 10, where a value of 50% is better suited for comparing the morning peak while the 100% scaling creates more comparable results in the evening.

Furthermore, in terms of link speeds, it can be seen that the Sioux-16 scenario features less congestion during the off-peak hours due to the possibility of distributing trips all over the network.

[2]In MATSim, this can be easily done in the Mobsim configuration, e.g. `qsim.flowCapacityFactor` for QSim

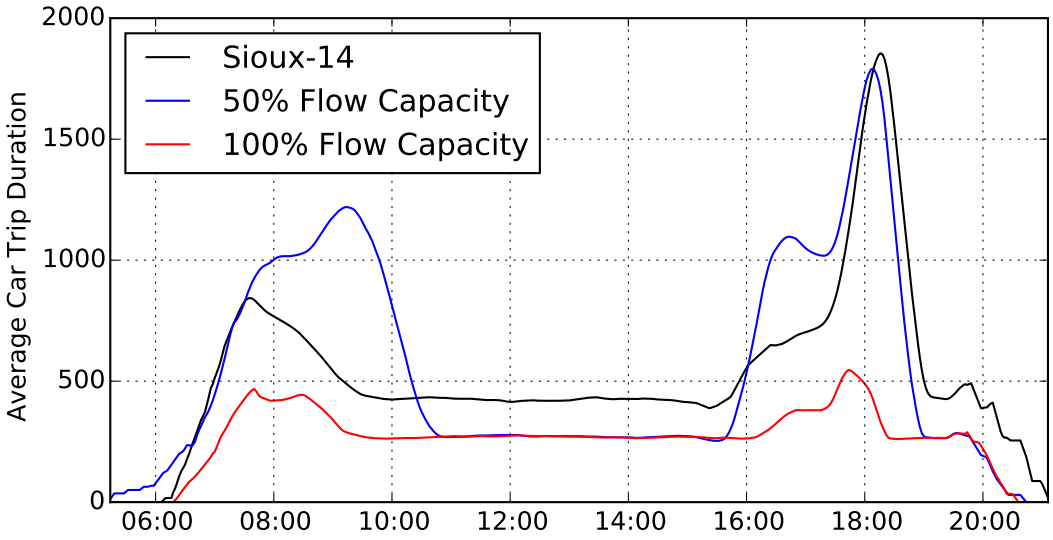

Figure 9: Comparison of the average duration of car trips by day time in Sioux-14 and Sioux-16 with 50% and 100% flow capacity.

In any case it has to be kept in mind that comparing the speed decreases is only a rough measure of network congestion. Most importantly, the values displayed are average values, which means that they biased towards outliers, i.e. capturing the changes in main arterials. That is a good comparison to Sioux-14, but on the other hand, the average is also taken over an increased number of links, of which some might rarely be used and therefore dragging the average down.

A comparison of the scenarios in terms of average travel times, distances and more shares can be found in table 1. What can be seen is that averaged over the whole day, values stay roughly the same, with the biggest differences being present in car traffic. There, especially the decrease in travel distance is noticeable but expected, since more direct routes can be taken. For public transport, the result of Sioux-16 is similar to Sioux-14, which is an indicator that the network is strongly resembling the former version.

Looking at the travel time and distance for the transit walk to and from public transport facilities, one can see that changes are quite small, which allows the conclusion that the fine-grained network does not incline agents to switch to the car mode on the new network. This is verified by a comparison of the mode shares, which stay roughly constant for the scenarios. A major difference can be seen in the mode shares of walking and public transport legs, where roughly two to three percent of the agents in the network switch from the walking mode to public transport.

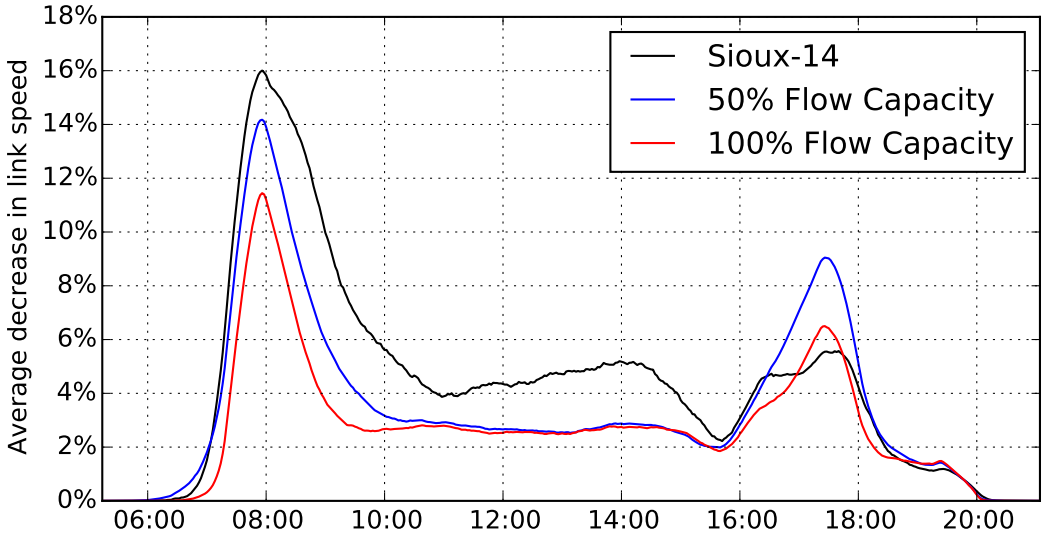

Figure 10: Comparison of the decrease of link speeds by day time in Sioux-14 and Sioux-16 with 50% and 100% flow capacity.

Table 1: Comparison of Sioux-14 and Sioux-16 in terms of travel measures. The respective distances and travel times of the walking legs to and from public transport facilities are included in the measures for public transport

Scenario Flow Capacity	Sioux-14	Sioux-16 50%	Sioux-16 70%	Sioux-16 100%
Travel Distances [km]				
Car	5.30	3.79	3.74	3.70
Walking	1.31	1.29	1.27	1.26
Public Transport	3.73	3.82	3.86	3.89
(Transit Walk)	1.31	0.53	1.28	1.28
Travel Times [mm:ss]				
Car	11:56	08:39	06:59	05:08
Walking	26:16	25:48	25:19	25:08
Public Transport	32:37	29:05	28:32	28:14
(Transit Walk)	24:05	21:50	21:55	21:59
Mode Shares				
Car	63.57%	63.23%	64.84%	65.71%
Walking	9.29%	7.50%	6.80%	6.56%
Public Transport	27.14%	29.27%	28.36%	27.72%

4 Dynamic Agents in MATSim

Several approaches have been proposed to allow for dynamic behavior in MATSim. The decision on which option is best depends mainly on what a level of complexity in the behavior is needed.

A simple version of dynamic behavior is implemented in the public transport (PT) extension, where passenger agents are saved in a list as soon as they reach the link in the network, where they should be picked up by a public transport agent. As soon as the PT agent (e.g. a bus) arrives at that link, persons from the link are deregistered from the list and "teleported" to the destination stop as soon as the vehicle arrives there. This setup fits very well into the queue based structure of MATSim.

An even more dynamic approach is used in the DVRP framework, which will be discussed in the first part of this section. At the time of this thesis and for the specific use case of autonomous vehicles, some drawbacks will be shown and finally a new abstraction layer for dynamically acting agents will be presented, which has been part of the thesis work.

4.1 DVRP

The DVRP extension (Maciejewski and Nagel, 2012; Horni et al., 2015) is designed to provide a level of abstraction to the simulation of dynamic transport services, such as taxis. Its general structure is quite flexible so that it is easy to implement for instance electric vehicles (Bischoff and Maciejewski, 2014) (which need to recharge at some point during the simulation) or taxis, which are roaming randomly through the city and serving requests when they are made by passengers (Maciejewski and Bischoff, 2015).

However, this flexibility comes at a cost: The architecture of DVRP circumvents the efficient queue simulation of MATSim for activities. While "normal" agents are simulated as described before, dynamic agents (DynAgents) have two modifications:

Legs are sent to the conventional Netsim. However, agents have the ability to change their paths dynamically, i.e. one can modify the route of the agent during the simulation steps and the next time the Netsim wants to move the agent or checks whether the agent should arrive at the current link, its response is calculated from the updated path.

Activities are simulated separately from the non-dynamic agents. As soon as a DynAgent starts an activity it is added to a list of active agents. In each simulation step a

specific callback for each of those agents is called and then it is checked whether the agent wants to end this activity or not. This polling approach, as depicted in fig. 11, makes sure that it is not necessary to know when an activity (for instance a taxi waiting for any requests) should end or how long it should take. On the other hand, this approach is much more computationally expensive than the efficient queue simulation. Given that the simulation is done on a second-by-second basis, an activity that would take one hour, would cost only one insert and one lookup on the acitivty queue. In the polling approach it costs 3600 calls to the simulation step callback (even if it does not actually compute anything, this adds a computational overhead) and 3600 checks whether the activity should be ended.

Using DVRP, Bischoff and Maciejewski (2016) ran a study on autonomous vehicles, where the demand in Berlin has been obtained using an ordinary MATSim simulation. Then the link speeds of the underlying network have been modified, to resemble the traffic situation in a congested situation and then *only* autonomous vehicles have been simulated. This means that the simulation could be run once, and the results were obtained because only the QSim was used and not the evolutionary learning of MATSim.

However, in the project of this thesis, autonomous vehicles should be tested in an existing multi-modal scenario, where the evolutionary learning is necessary for the agents to arrive at their quasi-optimal mode choices. So if Bischoff and Maciejewski (2016) mentions computational times of 20h for a large number of agents, it is a measure for one iteration of the evolutionary algorithm. For the scenarios that will be tested here, iterations still do not reach a satisfactory equilibrium, although having a theoretical computation time of 2000h already.

Measurements have been made to determine, how much the simulation of an idle agent, which always stays in one activity, would cost regarding execution time on a test machine[3]. The final results gave an average time of 25ns. So simulating for instance 8000 agents for a common simulation time of 30h, would lead to an execution time of:

$$T_{30h} = \underbrace{25}_{\text{Nomial}} \cdot \underbrace{3600 \cdot 30}_{\text{One day in seconds}} \cdot \underbrace{8000}_{\text{Agents}} = 21.6s \tag{4}$$

For the whole MATSim simulation, this needs to be multiplied by at least 100 iterations, so that the overhead of a simulation of 8000 agents doing nothing would already reach a (highly optimistic) computational overhead of

[3]Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz

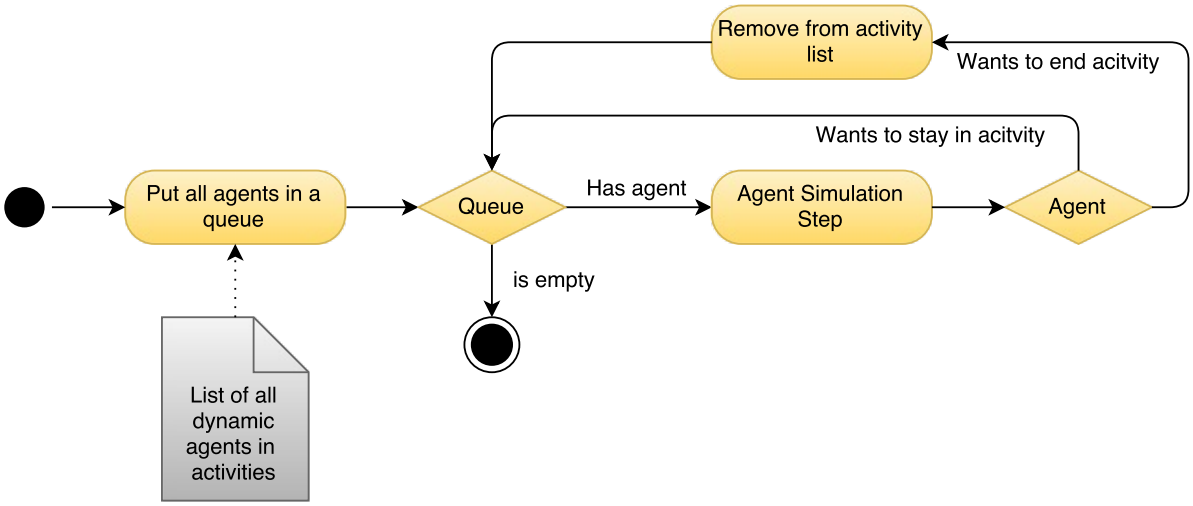

Figure 11: Polling approach of the DVRP framework

$$T_{sim} = 21.6s \cdot 100 = 36min \tag{5}$$

Furthermore, while working with DVRP, it has been found that one needs to put increased efforts into making the framework compatible with parallelized computation in the Netsim, which would have lead to a neglect of a good way to improve the computational performance of the simulation if ignored.

As a summary, one can say, that DVRP allows for a great freedom in simulating dynamic behavior and its structure and signaling flows are very straight-forward and easy to work with. However, for large numbers of iterations, it would be beneficial to find more efficient ways for the simulation.

4.2 The AgentLock framework

As an alternative to the simulation of dynamic agents using DVRP, the AgentLock framework has been developed as part of this thesis. It tries to combine the advantages of a queue simulation while providing as much flexibility as possible to create rich and dynamic agent behaviors in MATSim.

The basic idea is as follows: Usually agents in MATSim do not need much processing power, i.e. the per-agent simulation step of DVRP is hardly ever used and can usually replaced by some callbacks and event handlers outside of the actual agent simulation.

Furthermore, this means that an activity just means that an agent is residing at a certain position in the network and not taking part of the network. So an activity for an agent

is just keeping the agent back from joining the traffic network again after a certain time or event.

With this idea in mind, three different types of ways to "lock" an agent into an activity have been proposed:

Blocking activities will just let the agent reside in this activity until it is released through a call from outside.

Time-based activites are the ones from the basic MATSim simulation: They have a certain duration and therefore a fixed end time.

Event-based activities let the agent reside in the activity until a certain event happens.

The heart of the AgentLock extension is the LockEngine. Whenever it encounters an agent that wants to start a blocking or event-based activity, it is removed from the simulation and only added back manually or as soon as the event occurs. Then it either is passed to the ordinary Netsim or the next activity is started, just as requested by the agent logic.

For time-based activities, a similar approach to the activity queue for ordinary agents is taken. The first idea that would come to one's mind is to order the agents by the end time of their activity and as soon as there is a change in plans, remove the element from the priority queue and add it back at a certain position.

Compared to DVRP, where a change of plans would cost nothing, here this would lead to an overhead of $\mathcal{O}(\log n)$ and $\mathcal{O}(n)$ for these operations (Java API, 2016). So it is necessary to weigh the overhead of the polling in DVRP against the overhead of the rescheduling, which mainly depends on how often such reschedulings appear. In the concrete example of autonomous taxis, this itself depends directly on the travel demand.

If one sacrifices a slightly increased memory consumption for the sake of having a faster computation time, this setup can be improved further, as done for the AgentLock framework. Here, every time a time-based activity is started, a handle to the corresponding agent is saved into the priority queue, ordered by the end time of that activity. Furthermore, an indicator is saved, whether the handle is still valid. So if in the meantime the plan is changed (i.e. before the end of the activity has been processed), this handle will be invalidated, and it will simply be ignored when processing the priority queue.

The whole process (fig. 12) works as follows: In every simulation step, the top element of the priority queue is checked. If the top lock handle is scheduled for the current or a past time step, the associated agent is saved for further processing. If the handle

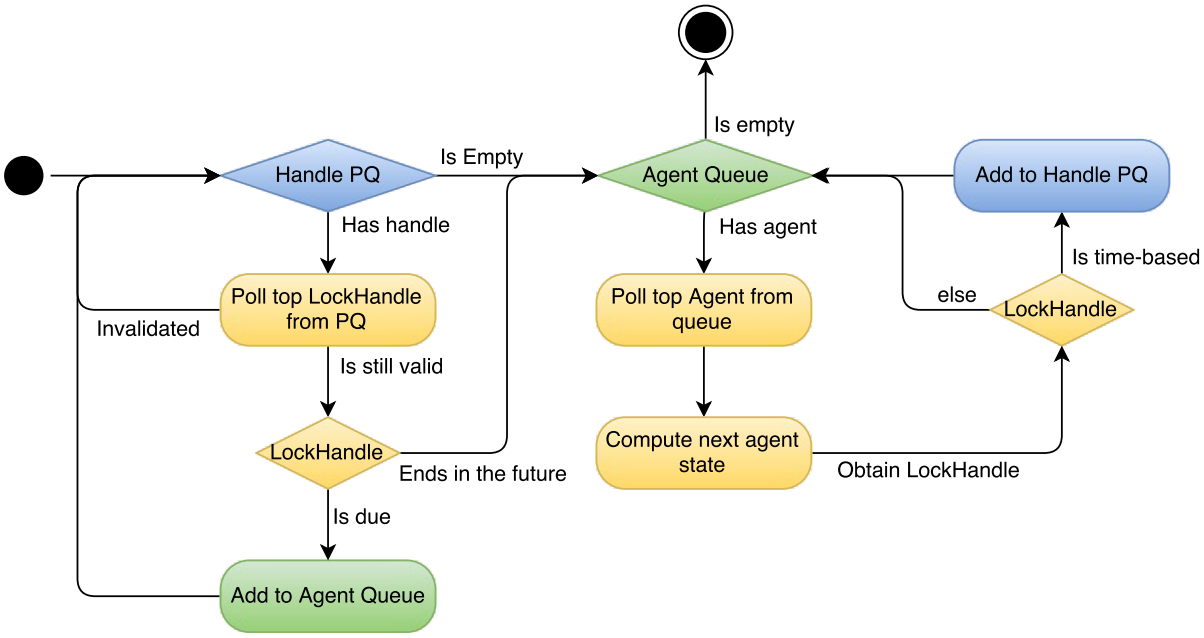

Figure 12: Per-Iteration procedure of the LockEngine in the AgentLock framework

already had been invalidated, the processing continues with the updated top element of the priority queue until a handle is found that has an expiry time which lies in the future. Subsequently, the new state (activity or leg) is computed for each saved agent, and a new lock handle is added to the handle queue if it is time-based.

The main advantage of this setup is that elements are only removed at the top of the queue, which is significantly less expensive than removing elements at arbitrary positions within the queue.

Furthermore, the AgentLock framework provides methods for dynamically ending legs, and it has been made sure that all the functionality has a high degree of compatibility with the existing parts of MATSim, such as the multi-threaded Netsim.

4.3 AgentFSM

While the AgentLock extension mainly provides an abstraction layer for the dynamic rescheduling of activities and legs, another layer has been developed, which is loosely resembling a finite state machine, tailored towards agents in MATSim.

In this framework all the activities and legs are predefined states in a finite state machine and the transitions from one step to another can either be triggered manually (which is the *blocking* lock from above), by time (*time-base lock*) or by an event (*event-based lock*).

The main task of the AgentFSM framework is to encapsulate common programming steps

when designing dynamic behavior in MATSim. This means that one usually just has to define which states the behavior is built of and how the transition from one to another works. All of this functionality is provided with a simple programming interface.

In more detail, each state is either a LegState or an ActivityState. For each state, an *enter* callback exists, which the programmer can use to execute arbitrary code. It needs to either return how this state is locked from the three options above or issue a direct switch to another state.

As soon as the state is ended due to the above conditions, the *leave* callback of a state is called, which must return to which next state the simulation should switch.

The concrete implementation for the autonomous vehicles will be discussed in chapter 5.

4.4 Comparison

An implementation of autonomous vehicles in DVRP has been compared to an implementation using the AgentLock framework. A specific number n of autonomous vehicles are created and put into a queue. As soon as an agent wants to start a leg using an AV, the top element of the queue will drive to that position, pick up the passenger, bring him to the final destination and drop him off. Then the AV will be added back to the end of the queue. This dispatching algorithm is quite inefficient but sufficient to do investigations in terms of computation time for the two implementations since the behavior is equal and easily understandable.

Simulations with $n = 2000, 4000, 8000$ have been done on the Sioux-14 scenario. What can be seen in fig. 13 is the number AV legs in the simulation and the associated computation time. For a large range, the implementation using AgentLock/AgentFSM (solid) is much faster than the corresponding DVRP implementation (dashed). In fact, for $n = 8000$ only at a (quite unrealistically big) AV share of around 60%, DVRP starts to be more efficient.

Important to notice is that the performance of DVRP stays roughly the same over the whole range of shares. This is because all 8000 AVs are simulated in every single time-step, no matter if they are active or not. In the AgentLock implementation, however, agents are only simulated if they are really in use. At $110k$ legs, though, when the reschedulings of used AVs are getting too frequent, the repeated queue operations get more expensive than the polling and therefore DVRP becomes more efficient.

As a result, one can say that the AgentLock implementation is usually more efficient than the DVRP version if there is a limited number of reschedulings. If they get too frequent,

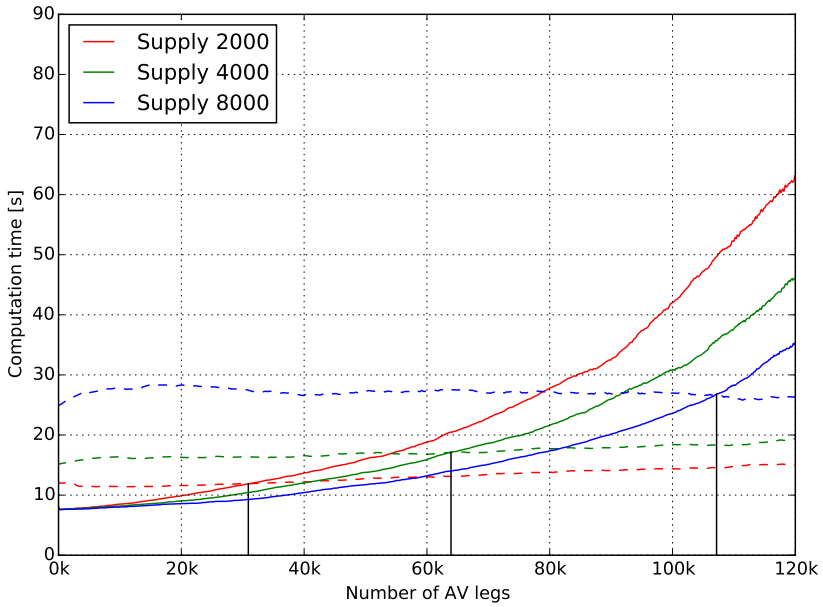

Figure 13: Comparison of Mobsim runtimes between DVRP (dashed) and AgentLock/AgentFSM (solid) implementation, dependent on the number of AV legs. The scenario is Sioux-14 with a total number of legs of around 180k.

i.e. the behavior is characterized by many alterations of some initial plan, the polling structure of DVRP gets more efficient. For the purpose of simulating autonomous taxis in this thesis, the number of such reschedulings should be minimal, as it will be described in chapter 5. Therefore, the development of the AgentLock framework indeed bears a computational advantage for this thesis.

Regarding a multithreaded Mobsim, only tests with the new AgentLock implementation could be done, as shown in fig. 14. Obviously, though the improvement is small, the simulation with two threads performed best. This shows that it is an advantage to be able to use the multithreaded Mobsim. Similar results have been found for other scenarios. For instance, the optimal number of threads for the Singapore scenario has been found to be four (Erath et al., 2012).

In conclusion, while the AgentLock framework is suited for the investigations in this thesis, a hybrid framework, offering both approaches, would be highly beneficial. While the AgentLock framework could be extended easily to provide this functionality, another idea would be to integrate the locking approach into DVRP to create a unified basis for dynamic agents in MATSim. Even more, it would be interesting to investigate adaptive algorithms, which could switch intelligently between the processing modes, dependent on the actual computation time and number of reschedulings in a specific simulation.

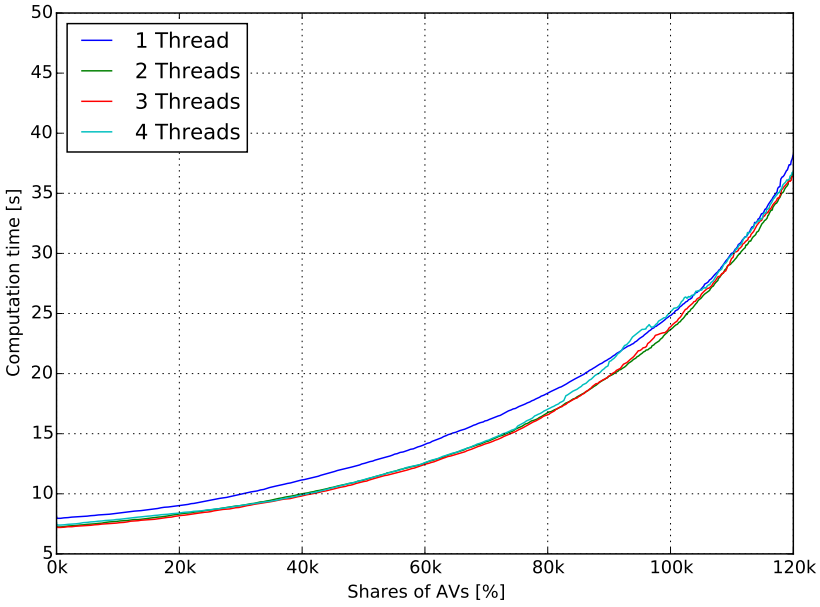

Figure 14: Comparison of different numbers of threads for the Mobsim with the AgentLock implementation

5 Autonomous Taxi Fleet Model

Simulating autonomous taxi services in MATSim requires the implementation, but even more so a concise planning and definition of how the exact behaviour of those agents should look like. This will be the first part of this section. Then, when it is known how those agents can be simulated the next question is how they should be dispatched for the requests of passengers and finally it has to be decided, where the autonomous taxis are placed on the network at the beginning of the simulation. These question will be answered in the second and third part of this section.

5.1 Agent Behaviour

The behaviour of an autonomous taxi is not a lot different from an ordinary one, except that there is no need to roam through the city to find new customers. Therefore, the AV behaviour presented here is mainly inspired by the model in **TODO** `cite`, where ordinary taxi services have been simulated.

The behaviour of an autonomous taxi agent can be described in terms of a task life cycle. When the agent is not in a task, he is in idle mode, basically (for the basic model) meaning that he will just stay at the current position and wait for further instructions. The following steps in the life cycle are depicted in fig. 15.

1. **Pickup Drive** is the phase where the AV has got a task and is moving to the requested pick up location. If the AV is already at the right location, this point can be skipped, otherwise it represents a leg driving from the current position to the pickup location.

2. **Waiting** is the phase in which the AV has arrived at the pickup location, but the passenger is not there yet. This can only happen if passengers request cars in advance, prior to the time when they actually want to be picked up. If the passenger is already present at the time of the arrival at the pickup location, this state can be skipped.

3. **Pickup** is the state in which the passenger is picked up. It is modelled as a fixed time, e.g. one minute and started as soon as both the AV and the passengers are present at the pickup location.

4. **Dropoff Drive** represents the leg going from the pickup location to the dropoff point.

5. **Dropoff** is the point where the passenger leaves the vehicle. Again, this is modelled as a fixed time interaction.

6. **Idle** is the final (and initial) state of the AV life cycle. At this point the AV will just wait until it receives a new task to pick up another passenger.

The states described above fit very well to the distinction of Activities and Legs in MATSim as well as to the structure of the AgentFSM framework, which has been developed for this purpose. Resulting from this behavioural model, a couple of parameters end implementational questions arise, that can be configured accordingly:

The Idle behaviour for the basic model just means that the AV will stay at its last position. Future extensions could make use of parking facilities or do an intelligent repositioning to improve the overall performance of the service.

Pickup and Dropoff duration need to be specified. In accordance with **TODO** cite, $t_{pickup} = 120s$ and $t_{dropoff} = 60s$ have been chosen. The time used for the pickup action will not be counted as waiting time (which, in turn, would be penalized through the marginal utility of waiting as described before).

5.2 Distribution Algorithm

At the beginning of a day simulation MATSim all the n available AVs must be placed somewhere in the network. Two simple distribution algorithms have been implemented for the purpose of testing in this thesis.

The first one is **Random Distribution**. Here n links are chosen randomly from all available network links. While this is an easy distribution strategy for testing, it create a quite unrealistic scenario. Obviously, in a real AV service, vehicles would be relcoated over night, such that they can serve as many passengers as possible during the morning peek.

Therefore a more elaborate distribution strategy has been implemented, which should lead to more realistic results. Basically one can define a density over the network, such that every link has a certain probability to get an AV assigned. Then, in n steps, the n available AVs are added to a link dependent on the assignment probability.

This probability density can be based on many factors. The ones that are implemented for this thesis are:

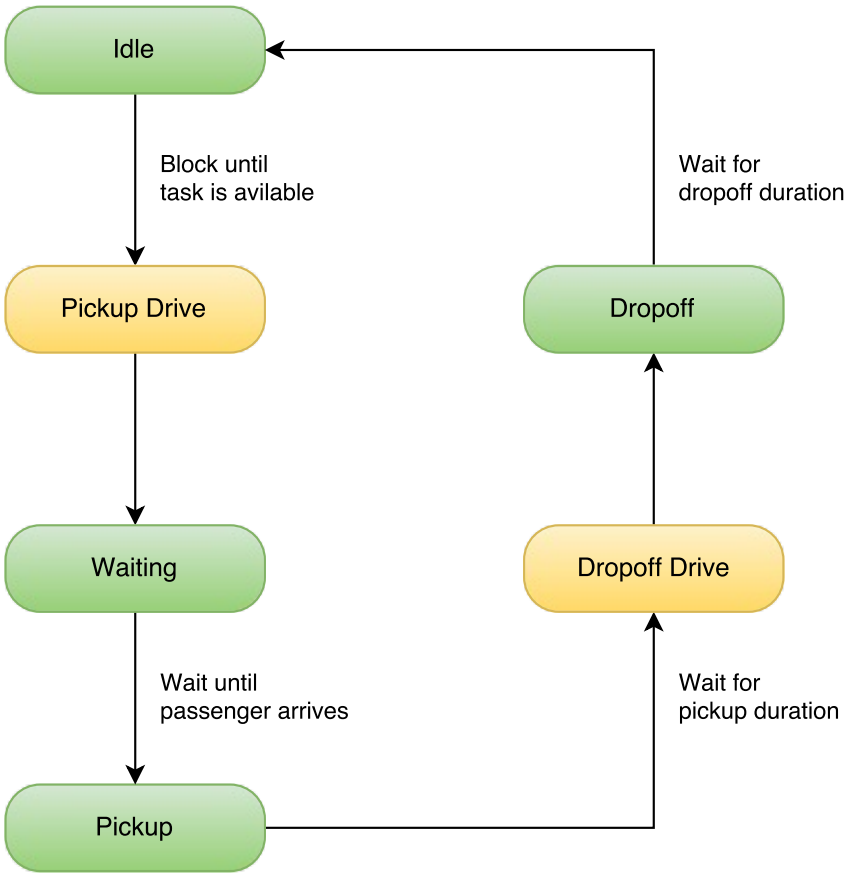

Figure 15: State chart of an AV agent. Activity states are displyed in green, while Leg states are yellow.

Population density What is measured here is the population density in terms of how many people are havingtheir "home" activity on a certain link.

AV commuter density Here for each link it is counted how many people will have chosen the AV mode for their first leg, i.e. the one that is starting from home.

While the prior one is static, the latter on is dynamic in the sense that for every iteration in the evolutionary learning, the density will change. So while the population-density based distribution is a fixed constraint, the latter one captures the factor of availability. For instance, if there are many agents using AVs in a certain area, the availability is very high, and thus more people might be inclined to use it. On the other hand, if availability is low in an area, it might lead to longer waiting times and people are less likely to use AVs.

While it might be interesting to observe different patterns of availability, for the first testing purposes the population density is better suited, since a detailed investigation would need to be done if any results from the AV leg density come from the mode choice of the agents or of feedback behaviour within the algorithm itself.

5.3 Dispatcher Algorithm

The dynamic dispatching of taxi vehicles is quite a complex scheduling problem, which is quite hard to solve or needs numerous asumptions to be feasible. In general, the problem will be a NP hard and heuristical solution strategies need to be applied if fast solutions are needed. An overview, as well as a proposed heuristic algorithm can be found in Maciejewski and Bischoff (2015); Bischoff and Maciejewski (2016).

The algorithm can be described in two different states:

Oversupply occurs when there are more available autonomous vehicles than requests. This means that each request can be assigned without delay. In that case, as soon as a request arrives at the dispatcher, the closest vehicle to this request is searched and assigned to serve the customer.

Undersupply is the case when all autonomous vehicles are occupied. In that case requests will stack up, which cannot be handled immediately. In this case the algorithm works the other way round: As soon as an autonomous vehicle gets available, it is dispatched to the closest request.

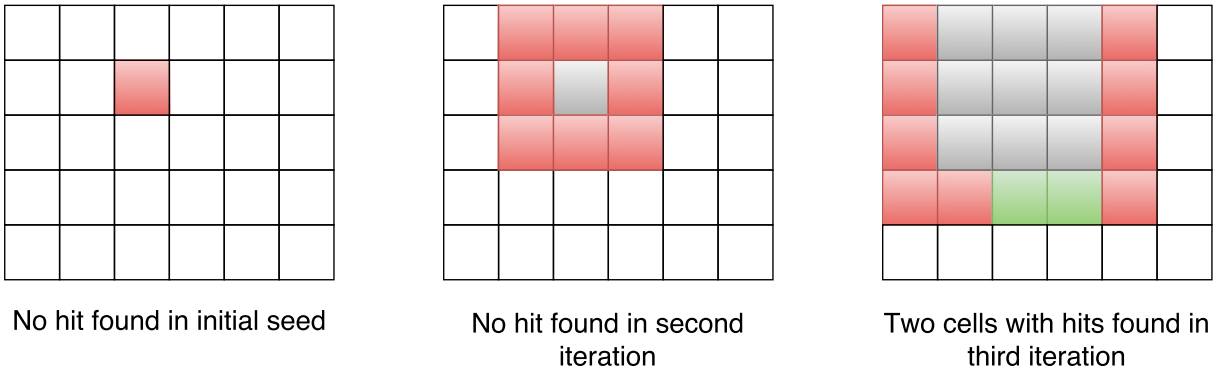

Figure 16: Schematic grid search algorithm for the dispatcher

According to the beforementioned papers this strategy gives a near-optimal solution, although providing fast computational times.

The implementation for this thesis is based on a spatial relaxation of the traffic network. This means that a grid with a specific resolution in x and y is fitted over the links. One of these grids saves the locations of all the available AVs, while another grid saves the locations of all open requests. Because the grids have a fixed structure, finding the closest AV or request is quite simple, since each position in x and y belongs to one specific cell of the grid. If no option is found in a certain cell, the search continues with all cells in its Moore neighborhood (all 8 surrounding cells). If this still gives no result, the radius is increased and so forth. When specifiying "find at least K hits" as a stop criterion for a search this resembles an approximate K-nearest-neighbor search based on a L_1 norm due to the grid character. The procedure is depictred in fig. 16.

This search algorithm imposes new parameters to the implementation, which basically are the cell counts in horizontal and vertical direction. Depending on the topology of the network, different parameters might be more efficient. If the grid is chosen to be too dense, many iterations are needed in the algorithm, while a too sparse one in the extreme case can lead to finding most of the items in only single cell.

In fact, for some topologies it might probably be more efficient to use different data structures like a binary tree or quadtree to improve the search procedure. Also more elaborate network search algorithms could be used, which make direct use of the topology.
TODO cite example maybe

TODO Explain how this has been parallelized! Show spoeed improvement!

For the Sioux Falls scenario it has been tested which impact the choice of the grid size has on the results. I fig. 17 one can see that very high resolution grids ($n = 200$) lead to a great increase in computational time, due to the necessary "expansion" of unoccupied

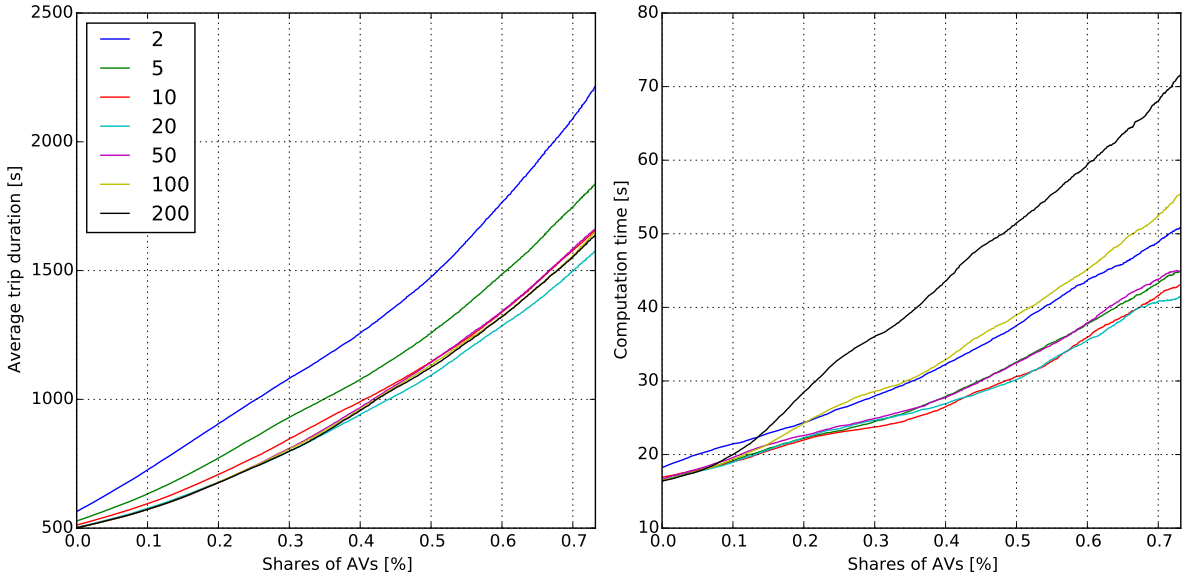

Figure 17: Performance of different grid sizes in the dispatching algorithm for Sioux Falls

cells, as described above. On the contrary, very low grid sizes lead to very poor results in terms of the overal average travel time of the agents, because just selecting a random agent from one of a few cells is basically just random assignment. A good value for the Sioux Falls network has been found to be $n = 20$, which is computed fastest over the whole range of shares and furthermore does a good job in decreasing the overall travel time.

Further improving the dispatching process, the routing has been parallelized. When the dispatcher is called one per iteration, all the assignments between autonomous taxi agents and passengers are cached to be processed while the network simulation of the next iteration is running. Only then, so after one iteration, the dispatcher waits for all running routing jobs to finish and notifies the agents. This way all the expensive routing through the network can be done in parallel to the Netsim. Figure 18 shows the impact of adding this feature: While only applying one parallel router there is already a huge increase in computational performance compared to the serial processing of the routing tasks. As soon as more routers work concurrently, this performance is even improved. However, as usual there is the point where managing the parallelization adds more overhead to the simulation. For the given scenario a number of routers of 2 seems to be a reasonable choice.

The parallelization of the dispatcher could be pushed even further. Next to the routing the actual assignment is the other expensive component in the algorithm. However, it would probably be hard to parallelize, because concurrent workers would need to access

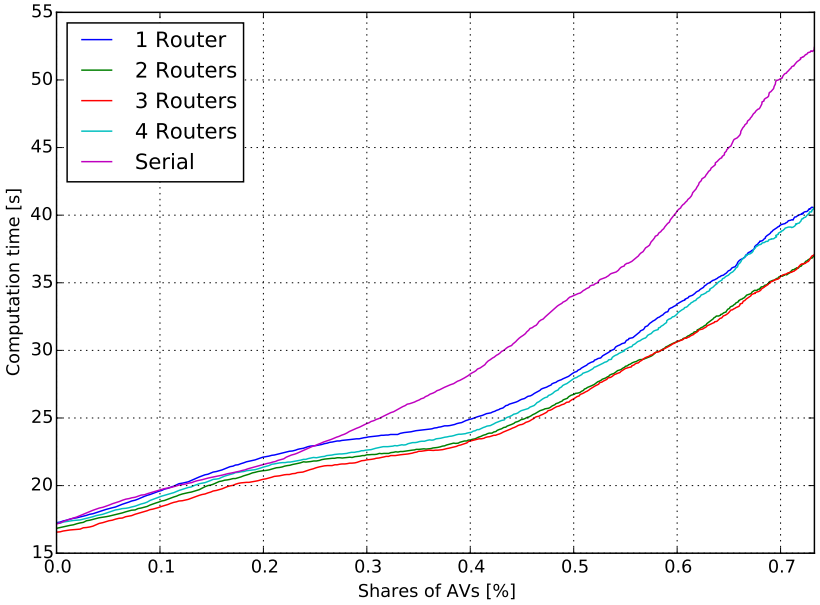

Figure 18: **TODO** todo

the same grid structures. The overhead of managing which worker has access is likely to very fast add more overhead than improvement to the simulation. One could, however, put the whole assignment process as a serial procedure in parallel to the Netsim, as done with the routers. This could be a promising approach of pushing the performance of the simulation even further.

5.4 Routing Algorithm

The easiest most simple option for routing autonomous vehicles in the model would be to solve a Dijkstra routing problem through the street network, while minimizing the expected travel time based on the freespeed values of the links. This strategy works well for small shares of AVs, because they have not a big impact on the overall traffic situation. However, when reaching high shares of AVs one (in simulation as in reality) faces the problem that the routing cannot be done on the fastest route for most of the people, as this generates too much congestion. By finding the shortest path in terms of freespeed, one artificially creates bottlenecks, because all AVs are routed through the same arterials. This is already an interesting result that can be moved directly to reality: As soon as an AV operator gains such a high share that he directly impacts the traffic situation, he must also make sure to relax congestion in the network, otherwise the service is creating too much of a delay.

In the simulation this got apparent when, for certain shares, huge numbers of agents could not finish their plans, because all arterials were cluttered by queueing autonomous taxis. While finding an "intelligent" congestion redistribution strategy is a whole different topic **TODO** `There are probably interesting publciations, have a look for that`, for the purpose of simulating AVs in this thesis a rather static approach is taken. The main idea is to add stochasticity to the Dijkstra process, thus creating slightly perturbed routes. What needs to be answered for using this approach is, which magnitude those perturbations should have. In order to solve this problem, the Sioux-16 baseline scenario has been examined. The 30h day has been divided into $N = 720$ bins ($2min$ each) and for each find the relative decrease in speed for all link passages starting in each bin has been computed:

$$r_{i,j} = \sum_{j=1}^{n_i} \frac{v_{i,j,freespeed} - v_{i,j,simulated}}{v_{i,j,freespeed}} \tag{6}$$

Here n_i is the number of link passages (all agents and all links) that occured in bin number i and $r_{i,j}$ is the relative decrease of speed in each bin for each passage.

As can be seen in fig. 19, the distribution of decreases in travel speeds of the network links can approximated using a Gamma distribution. The outliers, which can be seen on the right, are those cases where there is a lot congestion in peak hours. Because those are exactly the cases that one wants to avoid here, they are ommitted in the following steps. The assumption of a Gamma-distribution is wholly motivated by the shape of the histogram, performing an educated derivation of the distribution shape would be an interesting investigation, which is out of the scope of this thesis though.

A Gamma distribtion is defined by a shape parameter k and a scale parameter θ. For all bins in the baseline scenario, the respective parameters have been obtained using the MLE estimator for θ and a Newton approximation for k as described in **TODO** `add citation`. Figure 20 shows the parameter values for each bin on top, after they have been smoothed using a moving average filter of length 20. Also, for the times before roughly 5am and after 10pm, average parameter values over the off-peak hours in the middle of the day have been inserted due to a lack of sufficient data for fitting the speed decreases in these outer areas.

Also shown in fig. 20 (on the bottom) is the mean of the speed decrease at each time of the day, framed by the 10% and 90% quantiles, now instead of simulation results based on the statistical model. It can be seen that due to the Gamma model, the range of probable

decreases is high in peak hours, up to 70%, while the 10% stays quite constant during the day. The graph of the mean increase suggests that the presented model is a reasonable approximation, which qualitatively makes sense.

The final travel speed for the router can now be described as a random variable:

$$V_{i,j,routing} = v_{i,j,freespeed} \cdot (1 + R_i) \tag{7}$$

with $R_i \sim \Gamma(k_i, \theta_i)$ being Gamma-distributed for each bin.

The overall effect of this model is then as follows: There is a certain increase in travel time expected for each link, which depends on its freespeed and the time of the day. However, especially in peak hours, when the bottlenecks would be a critical problem, more variation is added to the expected increases in travel time and thus a better variation in the routing is reached. In each case, the expected increases are founded on the actual network characteristics of the Sioux-16 network.

What has to be kept in mind is, that this effectively puts a boundary on the possible number of AVs in the simulation. While the presented approach helps to avoid too much congestion for a wide range of AV shares, it doesn't perform an actual intelligent redistribution of congestion in the network. So because of too much congestion there will be a limit of AVs that can be used, since waiting times and travel times will get to expensive. This equilibrium generally should be lower than the actual "network congestion capacity" for the theoretical case of globally optimally routed AVs.

In the same sense it has to be acknowledged that no investigation has been done to find out how close the proposed model is operating to a theoretical optimum, so the following results in this thesis have to be understood under the assumption of this stochastic routing approach and might not represent the best possible solution.

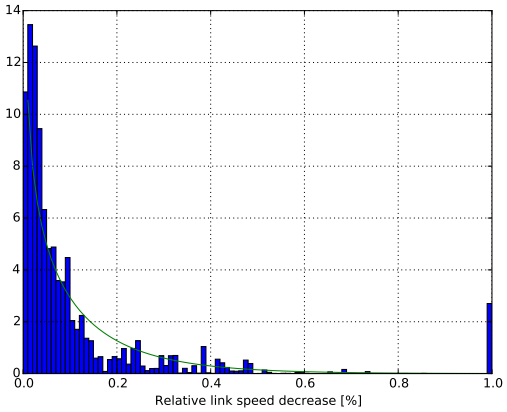

Figure 19: **TODO** todo

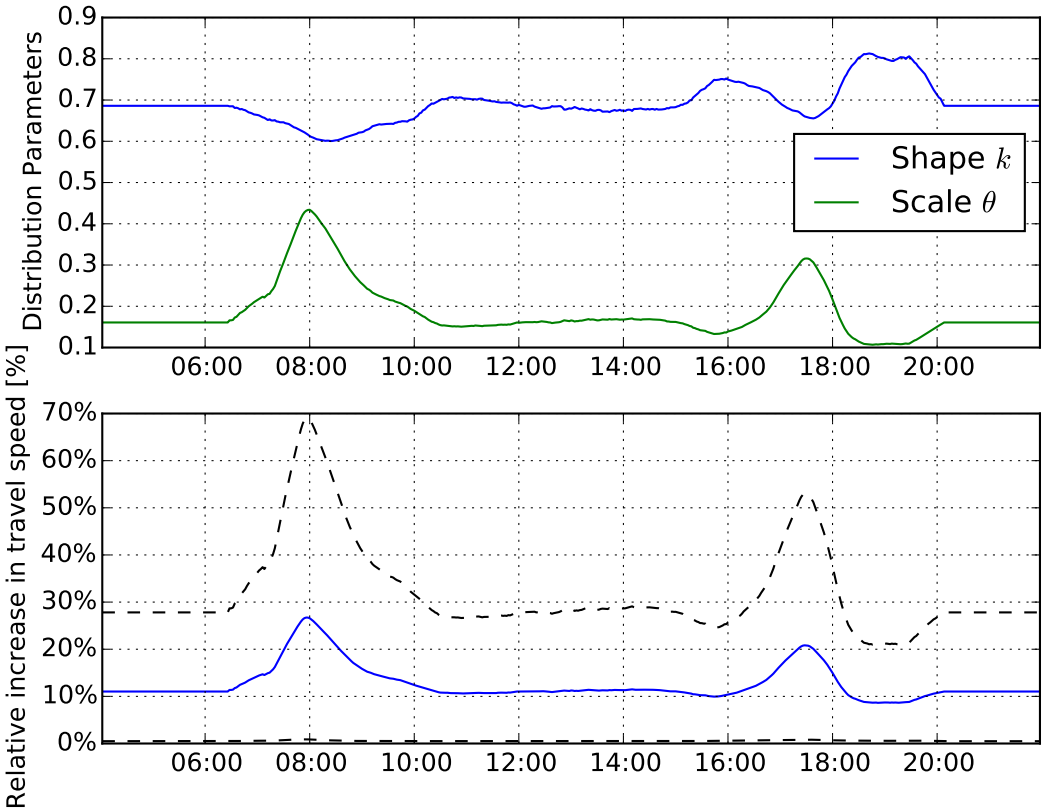

Figure 20: **TODO** todo

6 Simulation Results

6.1 Baseline Results

The model with the described implementation has been tested on the Sioux-16 network with a flow capacity scaling 0f 70% to allow for a reasonable amount of congestion.

The travel disutility parameter has been chosen to be zero, as is the travel disutility for taking a car in Sioux-16. The reason behind this is that there are numerous studies indicating positive and negative effects of autonomous vehicles, so it is hard to decide whether the perception of AVs will tend towards one side or the other compared with cars. To name just two examples, it has been stated that autonomous vehicles will increase the usefulness of time spent in the car, because people will be able to do work **TODO** cite sth. On the other hand it has been found that people are most likely to suffer from travel illness if they aren't concentrating on the road **TODO** cite sth and people subjectively might perceive the travlel in an AV as lasting longer compared to the car, due to the lack of activity.

The constant disutility for cars in the Sioux scenarios has been computed by combining the travel disutility for 10min walking (as to account for getting to and from a parking lot) and the monetary disutility for paying \$6 for parking. For the AVs in this simulation it has been assumed that there is no such additional cost, but a monetary fee per taking an AV trip. This assumes that a fictious operator already included the costs of parking into the pricing theme (which is reasonable taking the values form actual taxi services).

Since the values in Sioux-14 are based on measurements in Sydney, the current maximum charges for taxi trips there have been used as a reference (Transport for NSW). According to this source an initial charge of \$3.60 has been added to the constant disutility, while the monetary distance factor for AVs has been set to \$2.19 per km.

Finally, the disutility for waiting for an AV has been assumed to be the same as the waiting disutility for publci transport from Sioux, which itself is just a vague assumption, but at least allows for a systematic comarison.

Using these parameters, which are summarized in table 2, the scenario has been simulated until relaxation. The following paragraphs will show the respective results in terms of the traffic situation. A sufficiently high number of available AVs ($N = 8000$) to show how agents make a choice for taking an AV based on their utility evaluation.

Table table 3 shows the basic trip statistics of the case where AVs have been introduced

Table 2: Baseline Scenario Parameters

Parameter	Computation	Baseline Value
Constant Utility per Trip	$C_{av} = -\beta_m \cdot \$3.60$	-0.2232
Marginal Utility of Travel Time	$\beta_{trav,av} = \beta_{trav,car}$	0.0
Monetary Distance Rate	$\gamma_{car} = 2.19\$/km$	0.00219
Marginal Utility of Wait Time	$\beta_{wait,av} = \beta_{wait,pt}$	-0.18

to the baseline scenario. It can be seen that with the given utility parameters, the AVs reach a share of almost 30% averaged over the day, mainly decreasing the share of public transport and walking, while also attracting some of the former private car users.

Interesting to see is that for public transport and walking agents the travel distances decrease, because relatively long trips in those modes will be replaced by AVs, thus drawing the average down to shorter trips, while it increases for cars. Here, mainly the shorter car trips are replaced by AVs.

These results can also be in fig. 21, where the distribution of travel distances by mode is presented. Clearly, AVs act as a competitor towards public transport there, serving mainly the same range of trips with the assumed utility parameters.

In terms of travel times it can be seen that there is a slight increase for the car mode, which stems from the same argument as before. The decrease in travel time for public transport and walking agents is quite significant though. For the walking agents the change is obvious as described before. The decrease for public transport can be explained by the switch of agents, who needed to have long walking distances to the closest bus stop, which is included in the calculation. By only keeping those agents at using public transport, which live nearby a bus stop, the overall travel time decreases quite substantially.

Table table 4 shows how the mode choice took place after AVs have been introduced. The rows show the original modes, while the percentages indicate how many of the initial users were using the column mode after the introduction of AVs. What can be seen is that 46% of all initial public transport users and 60% of all walking people opted for taking an AV, while only 18% of car users switch modes. This again shows that with the baseline parameters, AVs rather work as a competitor against public transport while additionally drawing new adopters from the walking people. Therefore this scenario represents the rather unwanted case where AVs actually lead to a less optimal situation on the road, leading to more congestion and less use of collective transportation.

The waiting times for AVs are on average around 04:40 min in the morning peak and

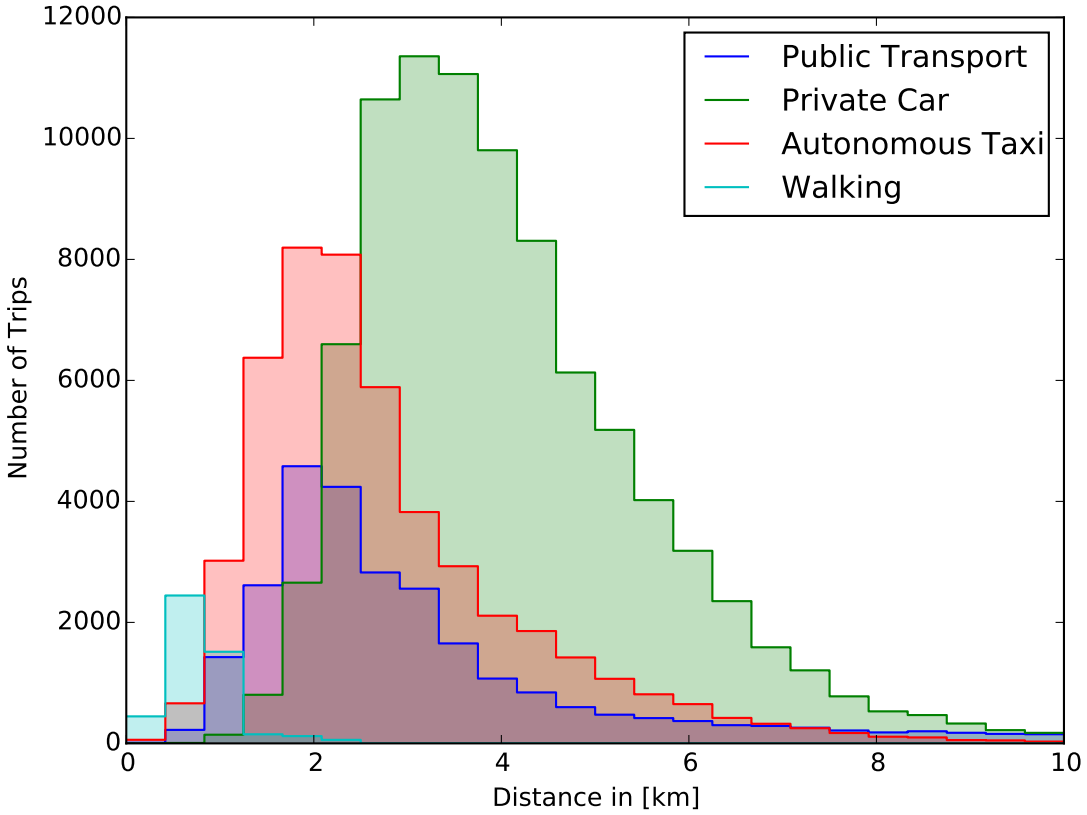

Figure 21: **TODO** todo

Table 3: **TODO** TODO, formatting

	Baseline	**With AV**
Travel Distances [km]		
Car	3.74	4.13
Walking	1.27	0.80
Public Transport	3.86	3.27
Autonomous Taxi		2.78
Travel Times [mm:ss]		
Car	06:59	08:26
Walking	25:19	16:02
Public Transport	28:32	19:56
Autonomous Taxi		10:08
Mode Shares		
Car	64.84%	52.54%
Walking	6.80%	2.81%
Public Transport	28.36%	15.79%
Autonomous Taxi		28.85%

Table 4: **TODO** TODO

	Autonomous	Car	Public Transport	Walking
Car	17.72%	80.65%	1.53%	0.10%
Public Transport	46.71%	0.80%	51.93%	0.56%
Walking	60.54%	0.30%	1.14%	38.02%

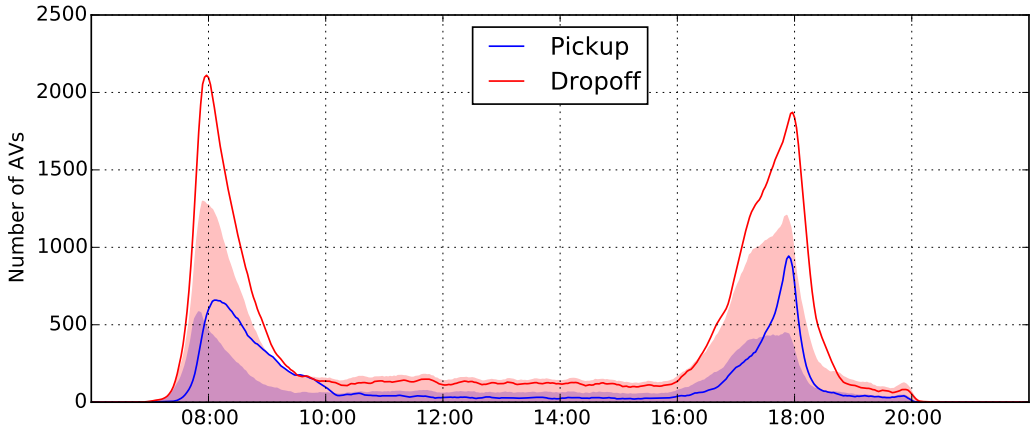

Figure 22: **TODO** todo

02:55 min in the afternoon, while the daily average lies at 01:40 min.

In terms of travel distances the total amount of kilometers driven increased from around 404,000 km in the baseline to 527,000 km in the AV scenario. The amount of kilometers driven by AVs is 161,000 km from which around 29,000 are for the purpose of picking up passengers, i.e. they are unoccupied during this time, which is roughly 18%. This is quite a small number compared to ordinary taxi services. This is quite small compared to ordinary taxis with 52% as stated in recent statistics for Oslo (Geir Martin Pilskog, 2015) or around 50% in Barcelona (Amat et al., 2014).

Finally, fig. 22 shows the states of the AVs during the day. While the lines show how many AVs are currently performing either a pickup or dropoff task, i.e. being "en tour", the shaded areas show how many passengers have been picked up or dropped off at at certain time of the day. In this baseline scenario, only around 3000 cars are actually active of the available 8000, so from the perspective from an AV operator the scenario also would not be an ideal case, because the usage of the AVs is not nearly close to saturation.

This following section will show how perturbations to the given parameters will change the system behaviour of the simulation.

6.2 Sensitivity Analysis

Intuitively, the behaviour of the utility parameters should be quite clear: If the utility is increased, the AV mode gets more favorable, if it is decreased, less people will use it. However, in such a complex traffic system there are secondary effects, which influence the adaptation of AVs. In the following first the general influence of the single parameters will be discussed and limiting cases will examined.

7 Outlook

During the course of this thesis a versatile and extentable basis model for the simulation of autonomous vehicles has been developed. Many topics in this field, which would be interesting to investigate, were not part of the scope of this thesis. The following sections will present the most important and interesting ones and give directions on how to simulate them with the framework. Likewise, these sections also give an overview about critical points, which are not taken into account in the previous results, but could change the overall picture significantly.

7.1 Recharging Infrastructure

Since autonomous vehicles in general will take great advantage of the ongoing electrification in the automotive industry, the availability of the respective infrastructure is one factor, which can determine how fast the adaptation of AVs will progress **TODO** `cite sth`.

Given the right amount of resources to create such an infrastructure, a big question that arises is how many recharging facilities are needed and where they should begin located in order to serve certain sizes of AV fleets (Chen, 2015). Furthermore, how can a combined infrastructure for ordinary owned EVs, private AVs and publicly provided AV services be designed?

As a first step one could assume that AVs recharge at dropoff locations, where they are artificially put to rest for a certain minimum idle time in order to simulate the recharging process. This would already lead to results on the customer acceptance side, but would ignore effects on congestion if AVs would actually need to take long trips to the next charging facility.

Previous studies using MATSim already gave interesting results on the implementation of private EVs **TODO** `cite EVs Berlin` in Berlin with distributed charging facilities over the network. Such research could be combined with the AV framework developed here. The modular AgentFSM component would it make easy to add specific new points in the state chains of an AV to drive to a charging facility.

Then again, the simulation framework could be used to get insights in which scheduling strategies would be optimal for an AV fleet if recharging has to be taken into account.

7.2 Shared Trips

One obvious advantage of AV fleets is, that up to five people could be transported in ordinarily shaped cars and even more in autonomous minibusses or full-sized busses. Intelligent routing and scheduling algorithms would make it possible to pick up passengers at arbitrary locations, not being bound to a fixed public transport schedule.

Again, due to the extensible structrue of the AV extension, it would be easy to add such behaviour in principle. However, the problem of optimizing the trips of more than one passenger, probably while already on a ride, is a highly complex problem and acceptable heuristics are still an important research subject.

Adding such functionality to the AV framework would make it possible to test such strategies in near-realistic scenarios and measure the performance of different heuristics.

TODO some citations here

7.3 The Last Mile Problem

Autonomous vehicles are likely to relax the last mile problem, where people are not motivated to opt for public transport, because the last mile from the transport facility to their home is too long. An AV could bridge this distance, maybe being prescheduled to pick up the passenger according on the current expected arrival times.

Implementing this behaviour would need only small changes in the MATSim framework. Generally, a public transport trip is generated as three legs: One `transit_walk` leg in order to get to the stop facility, one `pt` leg for the actual residual time in the bus or train and another `transit_walk`. A very easy first attempt would be to replace some of the `transit_walk` legs during the planning phase with `av` trips and a quite convincing simulation of AVs feeding public transport lines would be set up.

A probable requirement for this to work well would be to have prescheduled AVs, which are more or less likely to arrive at a prescheduled time.

7.4 Prescheduled AVs

Prescheduled autonomous vehicles would be easy to implement in the existing infrastructure. In fact, test have already be done, but abandoned due to the fact that in a first approximation delays from the scheduling can be modeled using worse utility values for the waiting times. At least this would lead to comparable results on the customer

acceptance side.

In the current state, the AV framework fully supports such prescheduled trips, though their impact has not been investigated in the scope of this thesis. As shown in figure **TODO** reference, where the state diagram of the AV was depicted, a "Waiting" state is already included, which would make an AV, which arrives early at a pickup location, wait for the passenger.

7.5 Intelligent Repositioning

A big field of research on taxi services in general is how to reposition available taxis while they are not in service. This means that before peak hours taxis could be intelligently moved to likely pickup positions and thus minimize the waiting time for customers, thus increasing the acceptance while at the same time reducing operator costs because less unoccupied miles might be travelled (if the algorithm is effective).

With the parallelization of customer trips the presented framework already shows by example how such an algorithm could be incorporated into the existing infrastructure without having too much impact on the computation times. In general, doing "some" intelligent repositioning should always be more beneficial than doing none. In this regard, the repositioning could be computed in parallel to the ongoing traffic simulation, while still offering the ability to restrict it if it slows down the general computation of MATSim.

Alternatively, repositioning could of course be run serially and thus arriving at complete heuristic or optimized solutions.

7.6 Parking

So far it has been assumed that autonomous vehicles will reside where the last dropoff has taken place. This assumption, especially in heavily packed cities, is quite optimistic, since parking space might be rare. In consequence, the driven kilometers per AV in the unoccupied state could be quite off the actual distance that would be needed to get to the next pickup location, but also to find a parking space in between.

Again, this rises a whole range of question on how an optimal AV scheduling would look like, maybe depending on the demand level, it might be even interesting to run preliminary studies on whether the search for parking space might be inferior to roaming (and this towards intelligently chosen locations).

7.7 Heterogeneity

Recently, efforts have been made to diversify the population in MATSim simulations. This means that people might be constrained due to age or income to use certain means of transport. Combining the simulation of AVs with the introduction on heterogeneity (as previously done for the Sioux Falls scenario) could give insights on how the availability and pricing of AVs affects the distribution of acceptance among diverse social groups in a city.

7.8 Adaptive Pricing Strategies

Related to this are adaptive pricing strategies. Once it is established, which disadvantages might be introduced by AVs (such as less accessibility to public transport for some social or spatially remotely located groups), research can be done on how to even out those differences by applying pinpointed pricing strategies. (Chen, 2016) suggests `TODO there are some interesting options, just name them here`

In order to "free" the city center from too much congestion one could for instance in the Sioux Falls scenario put high monetary fees on using the streets within the highway belt for private cars, while in parallel increasing the likelihood for AVs to use the highways. This way one could try to move traffic to the highways, then taking a direct trip to the workplaces in a perpendicular way.

Experiments like this are ready to be done with the existing simulation framework by adding custom scoring functions for private cars and autonomous vehicles.

7.9 Spatial Dependence

Another intereting point is the spatial dependence in the simulation. On one side, it would be interesting to investigate where AV users live, maybe indicating that on specific pricing strategies, people from the suburbs prefer cars, while other strategies might encourage people living in the center to use them.

An important point related to this is the initial distribution of AVs at the beginning of the daily simulation. As described above, here AVs are distributed dependent on the population density, though that might not be the best approach. Especially if one wants to encourage suburbians to use AVs, the density there should be higher.

In that regard it would also be interesting to investigate how the initial conditions in the

MATSim simulation influence the result on an abstract level. For the case of AVs, having initial plans to use AVs for a lot of center people, but not for people from the suburbs, the relaxed state might settle down in exactly this condition. However, if AVs are mainly distributed in the suburbs in the initial plans, the main user group might stay there, just because during the simulations the waiting times are shorter in either case and therefore these people might stick to their initial plan decisions.

8 References

C Amat, J Ortigosa, and M Estrada. Assessment of the taxi sector efficiency and profitability based on continuous monitoring and methodology to review fares. 2014. doi: 10.1016/j.pce.2013.02.001.

J M Anderson, K Nidhi, and K D Stanley. *Autonomous vehicle technology: A guide for policymakers*. 2014. ISBN 9780833083982.

Joschka Bischoff and Michal Maciejewski. Agent-based Simulation of Electric Taxicab Fleets. *Transportation Research Procedia*, 4:191–198, 2014. ISSN 23521465. doi: 10.1016/j.trpro.2014.11.015. URL http://www.sciencedirect.com/science/article/pii/S2352146514002981.

Joschka Bischoff and Michal Maciejewski. Simulation of city-wide replacement of private cars with autonomous taxis in Berlin. 00, 2016.

Patrick M. Boesch and Francesco Ciari. Agent-based simulation of autonomous cars. *Proceedings of the American Control Conference*, 2015-July:2588–2592, 2015. ISSN 07431619. doi: 10.1109/ACC.2015.7171123.

Patrick M. Bösch. Required autonomous vehicle fleet sizes to serve different levels of demand. *Arbeitsberichte Verkehrs- und Raumplanung*, 1089, 2015. ISSN 19454589. doi: 10.1017/CBO9781107415324.004. URL http://e-citations.ethbib.ethz.ch/view/pub:163885.

Artem Chakirov and Pieter J Fourie. Enriched Sioux Falls Scenario with Dynamic And Disaggregate Demand. (March):39, 2014.

T Donna Chen. MANAGEMENT OF A SHARED, AUTONOMOUS, ELECTRIC VEHICLE FLEET: IMPLICATIONS OF PRICING SCHEMES. *Proceedings of the 95th Annual Meeting of the Transportation Research Board*, 2016.

Tong Donna Chen. Management of a Shared , Autonomous , Electric Vehicle Fleet : Vehicle Choice , Charging Infrastructure & Pricing Strategies. 2015.

City of Gothenburg. DriveME self driving cars for sustainable mobility, 2016. URL http://international.goteborg.se/smart-cities-and-sustainable-solutions/driveme-self-driving-cars-sustainable-mobility.

E W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. ISSN 0945-3245. doi: 10.1007/BF01386390. URL http://dx.doi.org/10.1007/BF01386390.

Alex Erath, Pieter Fourie, Michael van Eggermond, Sergio Ordoñez, Artem Chakirov, and Kay W. Axhausen. Large-scale agent-based transport demand model for Singapore. *13th International Conference on Travel Behaviour Research*, (June), 2012.

Daniel Fagnant, Cockrell Jr Hall, and Kara M Kockelman. The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. pages 1–22, 2014.

Geir Martin Pilskog. Statistisk sentralbyrå: Taxi transport, Q4 2015, 2015. URL http://www.ssb.no/en/transport-og-reiseliv/statistikker/drosje/kvartal/2016-03-03.

Google. Google Self-Driving Car Project, 2016. URL https://www.google.com/selfdrivingcar/.

Alexander Hevelke and Julian Nida-Rümelin. Responsibility for Crashes of Autonomous Vehicles: An Ethical Analysis. *Science and Engineering Ethics*, 21(3):619–630, 2015. ISSN 14715546. doi: 10.1007/s11948-014-9565-5.

Andreas Horni, Kai Nagel, and Kay W Axhausen. The Multi-Agent Transport Simulation MATSim. 2015.

International Transport Forum. Mobility : System Upgrade. Technical Report October, 2014.

Java API. PriorityQueue, 2016. URL https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html.

T A N Cheon Kheong and Tham Kwang Sheun. Autonomous Vehicles, Next Stop: Singapore. *Journeys*, (November):5–11, 2014.

Scott Le Vine, Alireza Zolfaghari, and John Polak. Autonomous cars: The tension between occupant experience and intersection capacity. *Transportation Research Part C: Emerging Technologies*, 52:1–14, 2015. ISSN 0968090X. doi: 10.1016/j.trc.2015.01.002. URL http://dx.doi.org/10.1016/j.trc.2015.01.002.

Todd Litman. Autonomous Vehicle Implementation Predictions: Implications for Transport Planning. *Transportation Research Board Annual Meeting*, 42(January 2014): 36–42, 2014. ISSN 10769757. doi: 10.1613/jair.301.

Michal Maciejewski and Joschka Bischoff. Large-scale Microscopic Simulation of Taxi Services. *Procedia Computer Science*, 52:358–364, 2015. ISSN 18770509. doi: 10.1016/j.procs.2015.05.107. URL http://www.sciencedirect.com/science/article/pii/S1877050915009072.

Michał Maciejewski and Kai Nagel. Towards Multi-Agent Simulation of the Dynamic Vehicle Routing Problem in MATSim. 2012. doi: 10.1007/978-3-642-31500-8_57.

E.K. Morlok, J.L. Schofer, W.P. Pierskalla, R.E. Marsten, S.K. Agarwal, J.W. Stoner, J.L. Edwards, L.J. LeBlanc, and D.T. Spacek. Development and application of a highway network design model. Volumes 1(Final Report: FHWA Contract Number DOT-PH-11, Northwestern University, Evanston.), 1973.

Brandon Schoettle and Michael Sivak. Public Opinion About Self-Driving Vehicles in China, India, Japan, The U.S., The U.K. and Australia. (UMTRI-2014-30 (October)): 1–85, 2014. doi: UMTRI-2014-30.

Gary Silberg, Mitch Manassa, Kevin Everhart, Deepak Subramanian, Michael Corley, Hugh Fraser, Vivek Sinha, and Are We Ready. Self-Driving Cars: Are we Ready? *Kpmg Llp*, pages 1–36, 2013.

Kevin Spieser, Kyle Ballantyne, Rick Zhang Treleaven, Emilio Frazzoli, Daniel Morton, and Marco Pavone. Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems A Case Study in Singapore. In Gereon Meyer and Sven Beiker, editors, *Road Vehicle Automation (Lecture Notes in Mobility)*. Springer, 2014.

Tesla. Summon Your Tesla from Your Phone, 2016. URL https://www.teslamotors.com/blog/summon-your-tesla-your-phone.

Transport for NSW. Maximum taxi fares and charges. URL http://www.transport.nsw.gov.au/customers/taxis/maximum-taxi-fares-and-charges.

Transport Systems Catapult. Self-driving pods, 2016. URL https://ts.catapult.org.uk/pods.