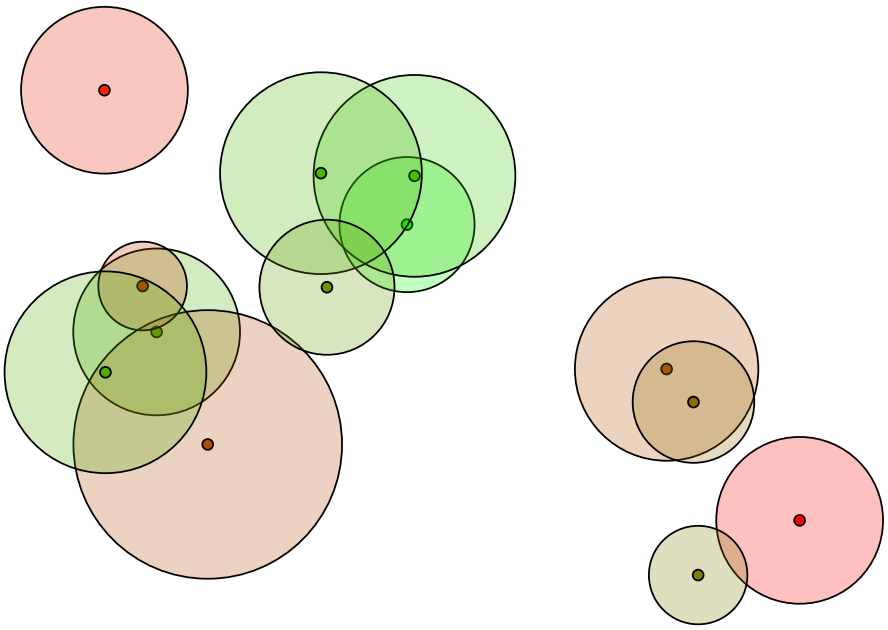

A Survey of Transport Simulation Tools

Comparing free tools for simulating autonomous vehicles in traffic and logistics networks

Master's thesis in Computer Science – Algorithms, Language, and Logic

JONAS A. HULTÉN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

A Survey of Transport Simulation Tools

Comparing free tools for simulating autonomous vehicles in traffic and logistics networks

JONAS A. HULTÉN

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2019

A Survey of Transport Simulation Tools
Comparing free tools for simulating autonomous vehicles in traffic and logistics networks
JONAS A. HULTÉN

Supervisor: Robin Adams, Dept. of Comp. Sci. and Eng.
Supervisor: Jonas Hellgren, Volvo Group Trucks Technology
Examiner: Devdatt Dubhashi, Dept. of Comp. Sci. and Eng.

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A number of communicating agents shown with communication radius and communication reliability.

Typeset in LaTeX
Gothenburg, Sweden 2019

A Survey of Transport Simulation Tools
Comparing free tools for simulating autonomous vehicles in traffic and logistics networks
JONAS A. HULTÉN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

With the emergence of autonomous transport vehicles it becomes desirable to simulate and evaluate their behavior before they begin operating on the open road. In this work we study three simulator platforms for traffic and logistics simulation and evaluate them over eleven criteria and three scenarios. We find that none of the simulators is strictly better than any of the others, instead excelling in handling different classes of problems, such as simulation of goods transport or fuel consumption estimation.

Keywords: Traffic, logistics, simulation, transportation, autonomous vehicles, survey.

Acknowledgements

I want to begin by thanking my supervisor Robin Adams for his unwavering support and inexhaustible encouragement. This thesis project would not have gotten off the ground, let alone finished, without your help.

I then want to extend my most heartfelt thanks to my friends and family for their support, encouragement, and help throughout this process. You all know who you are, and I could not have done this without you.

I also wish to thank Jonas Hellgren and Pernilla Sustovic at Volvo Group Trucks Technology for their help and for granting me the opportunity to work with them on their exciting projects.

Finally, I wish to thank Yoni Rabkin, licensing volunteer at the Free Software Foundation, for his help in understanding SimMobility's license.

Jonas A. Hultén, Gothenburg, February 2019

Thesis supervisors: Robin Adams, Dept. of Comp. Sci. and Eng.
Jonas Hellgren, Volvo Group Trucks Technology
Thesis examiner: Devdatt Dubhashi, Dept. of Comp. Sci. and Eng.

Contents

List of Abbreviations

AGV Automatic Guided Vehicle
AV autonomous vehicle
AV-HT autonomous vehicle for heavy transport

DARP Dial-a-Ride Problem

FOSS free and open source software
FSF Free Software Foundation

GPDP General Pickup and Delivery Problem
GPL GNU General Public License
GUI Graphical User Interface

MATSim Multi-Agent Transport Simulation

OSI Open Source Initiative

PDP Pickup and Delivery Problem
PDVRP Pickup and Delivery VRP

SUMO Simulation of Urban MObility

TDP Truck Dispatching Problem
TLS Traffic Light System
TraCI Traffic Control Interface
TSP Traveling Salesman Problem

V2I Vehicle-to-Infrastructure
V2V Vehicle-to-Vehicle
V2X Vehicle-to-Somewhere
Volvo GTT Volvo Group Trucks Technology
VRP Vehicle Routing Problem
VRPB VRP with Backhauls
VRPCB VRP with Clustered Backhauls
VRPDDP VRP with Divisible Deliveries and Pickups
VRPMB VRP with Mixed Backhauls
VRPPD VRP with Pickup and Deliveries
VRPSDP VRP with Simultaneous Deliveries and Pickups

1

Introduction

In the European Union, the transportation sector directly employs around 10 million people and comprises 5 % of GDP [1]. At the same time, the transportation sector accounts for 14 % of global greenhouse gas emissions [2], the vast majority of which — 72 %, or 10 % of all emissions — come from road transport [3]. In order to improve the efficiency of the transport sector while lessening the impact on the environment, researchers are looking to both the automation and electrification of transport vehicles.

Volvo Group Trucks Technology (Volvo GTT), a research and development division of Volvo Group, are making advances in automating short-range, high-demand, repetitive transport tasks. In 2016 Volvo presented a pilot autonomous vehicle for heavy transport (AV-HT) for use in underground mining operations [4] and recently presented an electric AV-HT intended for use in ports or similar logistics centers [5]. Further advances in the field of autonomous vehicles (AVs), broadening the scope of what AVs can do, may soon follow.

While these two examples are quite different, both in form and purpose, they are both AVs. Developing and testing such advanced vehicles, as well as designing the transport networks in which they operate, can be costly and — if the AVs travel on public road — dangerous. Indeed, as soon as the network exceeds one or two stretches of public road, practical trials become impossible. To work around this problem, engineers may instead seek to simulate the vehicles. Such simulations are substantial undertakings which necessarily must start by selecting the right tool for the task. In this thesis, we will survey a number of simulation tools in order to find what they are capable of, where they fall short, and the problems they are intended to solve.

1.1 Problem description

There are numerous scenarios in which one may wish to employ AV-HTs and there may be any number of properties one wishes to study before deploying them into the real world. Doing real-world studies is, of course, the most direct approach, but it is often prohibitively expensive and time consuming while also adding many

uncontrollable variables.

In order to allow engineers to study strictly controlled, simplified models of complex systems, we can use simulation. Traffic and logistics simulations are no exception — indeed, modeling infrastructure before roads are built and tracks are laid is essential, as both are extremely expensive to re-build later.

In this thesis we will be looking at such simulations from a different angle — instead of designing the roads, we are designing the vehicles and the fleets they operate in. We may also get to decide how many loading/unloading bays we have or where charging ports are, but we cannot re-draw roads.

The potential use-cases for a fleet of AV-HTs are many and they may not have a lot in common. In the case of an underground mine, we are moving through rough and hazardous terrain, but we also have absolute control over any other traffic along any given route and perhaps even where the routes are. In some other scenario, say a port-to-distribution center network, our AV-HT may need to use public roads and thus have to interact with other vehicles, both human-operated and autonomous, as well as pedestrians.

There are also any number of details we wish to study in these scenarios. Perhaps we wish to study different models for AV pathfinding, and don't want other vehicles to disturb. On the other end of the spectrum, we may want to study fuel consumption in a busy traffic network, looking at the effect of both road slope and load weight while ambient traffic causes interference. We may also want to study the optimal network configurations, regarding number of AVs, number of loading/unloading bays, and charging/fueling stations, seeking to provide robust service at minimal cost.

All in all, there are many problems engineers can encounter when building a fleet of AV-HTs and its auxiliary systems. The testing of the vehicles, their logic, and the supporting system can be done both faster and cheaper if simulated instead of tried in practice. This thesis sets out to discover the strengths and weaknesses of a few such simulators, so that these engineers can choose the right tool for their problem.

1.2 Goals and aims

Working from the problem description above, we studied three traffic- and logistics simulators. These simulators were selected since they were freely available, both as source and as binaries. Our three simulators are:

- Multi-Agent Transport Simulation (MATSim) [6]
- RinSim [7]
- Simulation of Urban MObility (SUMO) [8]

The simulators were subjected to a three-part analysis, described in chapter 3, then critically compared and contrasted against each other. The goal of this work is to show how suitable each simulator is to handle different types of problems.

1.3 Limitations

When seeking to model and simulate a complex system — such as the real world or the movement of a vehicle fleet — there are any number of aspects one can choose to focus on. For this reason, we have made a number of limitations which we will *not* consider or evaluate. These are:

- **Weather** The weather can have a profound effect on vehicles and traffic. Fog can limit max speed, wind can pose a hazard to high or long vehicles, and rain or snow may interfere with AV sensors. These are important details which should be taken into account, but they stray too far from the primary purpose of this thesis.
- **Road conditions** The condition of the road itself can impact how a vehicle behaves while driving on it. A cold, icy asphalt road will behave differently than a hot, wet, gravel road or, indeed, an uneven rock "road" underground. The surface that our vehicles drive on will impact their performance and fuel consumption, but we will not look into simulation of these different surfaces.
- **Platooning** A common topic in research regarding AVs is *platooning*, meaning that AVs drive close to each other and move as a unit in order to gain from shared aerodynamic properties. This can, for example, reduce fuel consumption of vehicles. Simulating platooning and, in particular, the gains from it, is dependent on an aerodynamic model of each vehicle. Like the previous limitations, this is an important aspect for a realistic simulation, but it strays too far from the primary purpose of this work. For that reason, we will not consider platooning.

1.4 Contribution

The work put forth in this thesis expands on the current understanding of traffic- and logistics simulators, seeking to critically evaluate and compare their capabilities. We will be focusing on freely available simulators, as they allow anyone, anywhere to pick it up and start working without needing to pay a license cost. However, the simulators are not created equal, so in order to find the one best suited for a particular problem, we provide this survey, analysis, and summary.

This thesis can be used as a starting point, both for a broader study of similar simulators, or for a deeper inspection of a subset of our simulators. We also foresee

this work to be of use to someone seeking to develop an entirely new traffic or logistics simulator, in order to understand common features and shortcomings. Finally, it should serve as a guide to the engineer who has a problem and needs to select a simulator in which to solve it.

1.5 Outline

This thesis will begin by presenting the extant research regarding traffic and logistics simulators as well as related theoretical problems in chapter 2, then proceed to describe how we intend to compare our set of simulators in chapter 3. In chapter 4 we present our results which are then discussed and analyzed in chapter 5. The thesis closes with our conclusions and suggestions for further work in chapter 6.

2

Background

In this chapter, we will introduce traffic and logistics simulation, along with literature discussing the construction of such simulators, in section 2.1. In section 2.2 we will focus particularly on the simulation on AVs, and then close out the chapter by introducing the theoretical Pickup and Delivery Problem and its variants in section 2.3.

2.1 Traffic simulation: a summary

Traffic simulation has been a topic of research since the 1950s, as noted by Krauß [9] and Behrisch et al. [10]. These simulations were initially concerned with modeling the ebbs and flows in traffic for the benefit of civil engineers designing the road networks. Since then, simulation has evolved. The simulation of traffic patterns is still relevant, but models to simulate even larger traffic networks at even grander timescales have emerged. This has given rise to a classification of simulator model scope: microscopic, mesoscopic, and macroscopic. These will be introduced below.

2.1.1 Scopes: micro, meso, and macro

In a microscopic model each individual element of traffic is modeled and simulated independently. An example is Krauß's model [9] wherein each vehicle exists as a separate and unique entity in the simulation and traffic flows happen as a result of the interactions between vehicles. Similarly, as noted by Behrisch et al. [10], each road segment — or even lane — is modeled individually. Indeed, the lane-changing behavior of vehicles is important in microscopic models.

In contrast, a macroscopic model is one where individual vehicles are less important than the flows they create. These are then modeled as statistical distributions, detailing traffic *density* rather than the position of specific vehicles. Such models often consider longer time steps than a microscopic model. For example, as detailed in Adnan et al. [11], a microscopic model might use a time step of a fraction of a second to a few seconds, while a macroscopic model can use steps of a day or more. At time steps of a day or longer modeling traffic no longer makes sense — such a

simulator will instead look at long-term movements in population density or land use.

A mesoscopic model is an in-between model — it is not a clearly defined class rather than a collective name for models which are neither fully microscopic or fully macroscopic. Such a model could, for example, have a macroscopic view of traffic flow in general, but allow for microscopic simulation of specific vehicles. Such vehicles would be slowed depending on the macroscopic traffic density around them, rather than by interacting with other microscopic vehicles.

In this work, all simulators considered use *microscopic* models.

2.1.2 The reason for simulating

As noted above, one of the key reasons behind the development of traffic simulators is to evaluate the performance of roads before they are built. In the same vein, an engineer could simulate and evaluate how a road network performed if traffic conditions changed.

One common task for microscopic traffic simulation is designing and evaluating Traffic Light System (TLS) programming, such as in Vaudrin, Erdmann, and Capus [12]. The interplay between individual intersections can have a far-reaching impact on the larger traffic system. For this reason, TLS-controlled intersection are often extensively simulated to find a programming which maximizes throughput for most traffic conditions.

Hernández-Paniagua et al. [13] exemplifies another problem which benefits from traffic simulation, namely emissions modeling. Hernández-Paniagua et al.'s study used microscopic simulation but emissions modeling could be done using any scope, as traffic density plays a more significant role than individual vehicle behavior.

Closely related to emissions modeling is fuel consumption modeling. Basso et al. [14] studied methods for optimal routing of electric buses such that they would use the minimum amount of power possible. Gallet, Massier, and Hamacher [15] did a similar study, looking at the energy demand of introducing electric buses in Singapore. Both studies employed microscopic simulation in order to accurately calculate the power consumption of each individual bus.

Studies such as Basso et al. and Gallet, Massier, and Hamacher are interesting for both traffic researchers and vehicle manufacturers. The former wants to study the effects on traffic — or auxiliary systems, as in Gallet, Massier, and Hamacher — of introducing new types of vehicles, while the latter wants to study how traffic affects their new vehicle models. In recent years microscopic traffic simulation has begun to incorporate AVs for the above mentioned reasons, among others. Such simulations are introduced in the next section.

2.2 Simulating autonomous vehicles

Autonomous vehicles do not behave like human-operated vehicles. They do not need to rest, nor stop for food or water. They respond faster than human drivers do, and can — generally — replan their routes faster than humans can when faced with unfavorable conditions. They are also able to share information with each other or the environment — Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I) communication, respectively — in ways that human drivers cannot.

As an example, Gora simulates the movement of communicating AVs and using the data gathered from them to predict and optimize traffic flow [16]. Gong, Shen, and Du similarly make use of communicating AVs to simulate a self-regulating platoon [17]. A platoon — or platooning — is a formation of vehicles arranged such that they benefit from each others' slipstream, minimizing aerodynamic drag throughout the platoon.

Independent AVs could be used to construct a distributed network of personal transports, as studied in Fagnant and Kockelman [18]. Such a network could lower traffic congestion and emissions while also increasing mobility for persons who otherwise have difficulties getting around, according to Litman [19].

The transport sector has gained interest in AV-HT in order to automate the movement of goods, rather than people. As mentioned in chapter 1 above, Volvo GTT has announced two such vehicles which will be used for short-distance, repetitive transport tasks. Here, simulation could play a role in the design of the vehicle, its control software, or the design of the transport network it will operate in. Optimizing the flow of goods in such transport networks is a difficult problem which can be described in terms of a Pickup and Delivery Problem. This will be described in the next section.

2.3 The Pickup and Delivery Problem

The following section is not directly relevant to the analysis of simulators carried out in this work. Nevertheless, it is presented below as relevant background to the general fields of traffic and logistics simulation.

Moving a unit of goods to the right recipient at the right time is not an easy task. In 1959 Dantzig and Ramser defined the Truck Dispatching Problem (TDP) as a generalization of the famous Traveling Salesman Problem (TSP) [20]. In the TDP, the goal is to find optimal loops going to and from a "depot" node, rather than finding the optimal full circuit, as in the TSP. This restriction can be interpreted as a limitation on either fuel or storage capacity. While the TDP is more realistic than the TSP it is still insufficient for some problems, such as handling customers who need to *send* goods, rather than just receive.

The General Pickup and Delivery Problem (GPDP) extends on the TSP and the TDP to handle such problems. Parragh, Doerner, and Hartl [21], [22] provide an excellent survey of the GPDP and its subclasses — as shown in figure 2.1 — as well as potential methods for finding solutions. In their survey, Parragh, Doerner, and Hartl focus on the Vehicle Routing Problem (VRP), which is the *multi-vehicle* form of the TSP. The VRP is relevant for our study as a transport network with AVs is likely to include more than one AV. Further, Parragh, Doerner, and Hartl distinguish between full-truck and less-than-full truck transport. This distinction is simple: in the full-truck case one unit of goods being transported fills an entire truck, while in the less-than-full case a truck can transport more than one unit of goods.

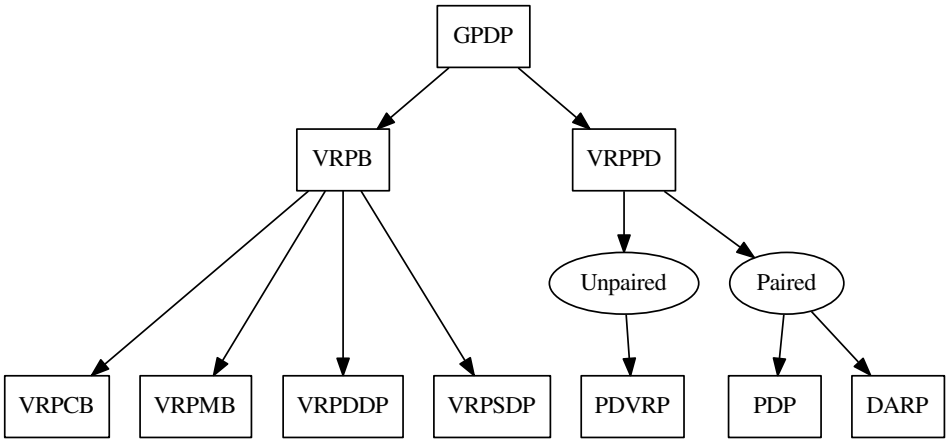

Figure 2.1: Structure of sub-classes of the General Pickup and Delivery Problem. Based on figure 1 in Parragh, Doerner, and Hartl [21].

The two subclasses of the GPDP will be described below.

2.3.1 The Vehicle Routing Problem with Backhauls

The Vehicle Routing Problem with Backhauls (VRPB) is similar to the TDP in the sense that a vehicle cannot visit all nodes within the network in one tour. The Vehicle Routing Problem with Backhauls (VRPB) extends the TDP, adding an additional type of transport — *backhauls* — in which goods are moved *from* customers *to* the depot. The conventional type of transport, moving goods from the depot to customers, is called *linehaul*. A VRPB also allows different starting and ending depots, which was not allowed in Dantzig and Ramser's definition of the TDP.

There are four subclasses of the VRPB which distinguish how we schedule our backhauls or model customer demands. These are briefly described below.

VRPCB The Vehicle Routing Problem with Clustered Backhauls (VRPCB) requires that, as the name suggests, the backhauls of each truck be clustered. In other words, all of its linehauls must be completed before it can start collecting backhauls.

VRPMB The Vehicle Routing Problem with Mixed Backhauls (VRPMB) is the complement problem of the VRPCB — instead of requiring that all backhauls happen after all linehauls, we allow them in any order. The VRPCB is therefore a special case of the more general VRPMB.

VRPDDP The Vehicle Routing Problem with Divisible Deliveries and Pickups (VRPDDP) introduces a new type of customer to our problem: one that both wants to receive and send goods. We will call such customers *double-haul* customers. In the definition of the VRP it is required that each customer is visited *exactly once*. For the VRPDDP, this restriction is relaxed, allowing double-haul customers to be visited exactly *twice* — once for linehaul and once for backhaul, though not necessarily in that order.

Parragh, Doerner, and Hartl notes that a VRPDDP can be considered a special case of the VRPMB [21] if each double-haul customer is defined as as two vertices in the graph — one with a linehaul demand, the other with a backhaul demand. This way, the *visit exactly once* restriction need not be relaxed, making the VRPDDP a special case of the VRPMB.

VRPSDP The Vehicle Routing Problem with Simultaneous Deliveries and Pickups (VRPSDP) is a limitation on the VRPDDP, reinstating the *visit exactly once* restriction and additionally requiring that pickup and delivery be done simultaneously, by the same truck. This is achieved by changing how demand is modeled, allowing a single customer — a single node in the graph — to have *both* linehaul and backhaul demand. Consequently, it is *not* a special case of the VRPMB — rather, the VRPMB is a special case of the VRPSDP, where a customer is allowed to have either a linehaul or a backhaul demand.

2.3.2 The Vehicle Routing Problem with Pickup and Deliveries

In the Vehicle Routing Problem with Pickup and Deliveries (VRPPD) goods do not need to pass through a depot before they are delivered to a customer. Instead, it allows direct customer-to-customer delivery. There are three subclasses of the Vehicle Routing Problem with Pickup and Deliveries (VRPPD), split into two types: paired and unpaired. The subclasses are described briefly below.

PDP The classic Pickup and Delivery Problem (PDP) is a *paired* form of the VRPPD. This means that nodes in the graph are paired — one producer and one consumer — which Parragh, Doerner, and Hartl call "requests." When a truck picks up a unit of goods from a producer, that good must be delivered to the paired consumer. While this transport mission is in progress the truck cannot pick up any other goods.

DARP The Dial-a-Ride Problem (DARP) is an extension of the Pickup and Delivery Problem (PDP) which considers transporting humans instead of inanimate parcels. Humans, in difference from parcels, can become impatient while waiting, adding additional restrictions on timeliness to the PDP.

PDVRP The Pickup and Delivery Vehicle Routing Problem (PDVRP) is an *unpaired* form of the VRPPD. In a PDVRP multiple producers may produce goods which is desired by multiple consumers, such that a unit of goods from one producer could satisfy the demand of several consumers. The PDVRP also allows a truck to pick up and deliver goods in any order, no longer requiring that a unit be delivered before another is picked up.

2.3.3 Additional constraints

Parragh, Doerner, and Hartl note that any of the above problems can, rather easily, be extended with additional constraints. The examples they give concern a time window for pickup or delivery and a maximum route duration. The demand model could also be changed to allow for heterogeneous goods. Similarly, truck and depot capacity could be changed to model size or weight restrictions in goods.

3

Method

As mentioned in section 1.2, the simulators were studied using a three part analysis. These three parts are:

1. Studying and summarizing the published literature related to the simulators.

2. Defining a set of common criteria and comparing the simulators using these.

3. Designing a small number of evaluation scenarios for the simulators, in order to test the capabilities of each simulator and how easy they are to work with. These scenarios were designed based on some problem which was explored in the published literature.

In this chapter, the three parts above will be described in detail.

3.1 Literature study

The literature study for each simulator began by searching Google Scholar, CiteseerX, and the Chalmers Library for articles related to "simulation" and "autonomous vehicles." Articles related to one or more of our simulators were considered particularly relevant. Having established an understanding of the simulators, we also searched for articles related to "network design," "routing," and "optimization" in the context of traffic or logistics simulation. Finally, we searched for "comparison" and "survey," again in the context of our simulators, seeking previous studies comparing and contrasting simulator platforms.

Articles were selected on the condition that they were relevant to this work, peer-reviewed, and published within the last fifteen years. Older publications were included for historical context when considering relevant theoretical problems which can be solved using traffic simulation. We also reviewed article quality — poorly written articles or articles from dubious sources were discarded, as we cannot guarantee that they have been thoroughly peer-reviewed.

3.2 Criteria

In this section, we present the eleven criteria, divided into two categories, which were selected to compare the simulators in a fair manner. The criteria evaluate not only the capabilities of the simulator itself, but also its extensibility and ease of use. Some criteria are vague, but they have been defined such that a binary yes-or-no answer can be given to each of them.

3.2.1 Primary criteria

Certain capabilities of a simulator are critical if it is to be easily included in a vehicle or fleet development project — it cannot take up large amounts of time and resources to set up. We make this limitation since our simulators have open source code, meaning a developer with a wealth of time and resources could mold any of them to fit any problem. However, such a scenario is unlikely to occur in any real development office. For this reason, three of the eleven criteria were selected as *primary* criteria, as they allow quick integration into an existing project. These are:

1. **External control**
2. **On-line interaction**
3. **GUI**

Below, these criteria will be thoroughly explained.

External control The external control criterion requires that the simulator can be *fully* controlled from some arbitrary external process *without* changing the simulator source code.

This would allow a simulator to be interfaced against a script running on the same computer as the simulator, such that the script can both read and write to the entire simulator state. This could also be done with a network interface, allowing the external process to be located on some other computer.

The motivation for this criterion is the fact that AVs seldom operate alone. There may be external systems such as fleet managers, routers, mission planners, cargo coordinators, or similar, which all need to communicate with an AV. In a simulation without external control, these systems would need to be built directly into the simulator source code, requiring a deep understanding of the simulator and increasing the likelihood of bugs. With external control, the simulator could instead be interfaced against the same systems that would coordinate the real-world AVs. This opens a new avenue for testing these auxiliary systems. Further, it reduces the work needed to do so since they do not need to be ported into the simulator code.

On-line interaction The on-line interaction criterion requires that the simulator state can be manipulated *while the simulator is running*.

A simulator which does not allow on-line interaction must be left alone to run — only once it has run its course can the result be studied and the simulation scenario be tweaked for the next run. On-line interaction instead allows the simulation state to be read and written to mid-simulation, allowing dynamic strategy changes as the simulation unfolds. *How* the state is manipulated is not important — it could be via an interface to an external process, through a GUI (Graphical User Interface), or through some other means.

The motivation for this criterion is the fact that a simulator which can be inspected and tuned on the fly will take less time to correctly configure than an offline simulator. They may both require the same number of full simulation runs to get right, but a simulator which supports on-line interaction allows a user or program to witness the simulation unfold and influence it as it does. This is contrasted against the log or result file *post mortem* which would be needed to understand what an offline simulator did during its run.

GUI The GUI criterion requires that the simulator provides some support for graphically visualizing the simulation, either while it is happening, or after the fact. A GUI to simply select files or change settings is not sufficient.

The motivation for this criterion is ease of use. While a simulator can be perfectly capable of simulating all necessary aspects of a system without also having a GUI, having one can make the system more intuitive to the user and make it easier to demonstrate a simulation to third parties.

3.2.2 Supplemental criteria

In addition to the primary criteria, there are a number of features which are *nice to have* more than they are critical for use. To cover some of these, the remaining eight criteria were designated as *supplemental* criteria. They are:

4. **Separate configuration**
5. **Goods**
6. **Communication**
7. **Third dimension**
8. **Fuel**
9. **Transparency**

10. **Documentation and Community**
11. **License**

This list is by no means exhaustive, as there are any number of features which could be simulated to add realism which are not considered in this project. Despite this, we believe that this set of criteria will give a good overview of the capabilities of each simulator within the scope of this thesis.

Separate configuration The separate configuration criterion requires that the simulator can be configured to simulate a scenario *without* editing the simulator source code.

The motivation for this criterion is again ease of use. Requiring a user to modify the simulator's source code in order to start simulating entails a steep learning curve, even if the user already is well versed in the language in which the simulator is written. The alternative, having the simulator load configuration files of some description, may lower this learning curve, depending on how complex these configuration files are themselves.

Goods The goods criterion requires that the simulator supports simulating the movement of inanimate objects independent of the vehicles that transport them.

The key part of this requirement is that each unit of goods is treated as an independent object in the network, existing outside of the vehicles, so that it does not "disappear" when not loaded on a vehicle. Given that, there are several possible extensions for this criterion. The simplest case, where each vehicle carries one unit of goods, is all that is required for fulfillment. In a more more realistic implementation, however, each vehicle can carry any number of goods. Going further, the goods themselves may be of different types, where each type might have a separate destination. The goods might have properties of their own, such as weight or dimensions, affecting how many can be loaded on a vehicle.

The motivation for this criterion is the fact that we will often be considering AV-HTs and the freight networks in which they operate. Considering a system where trucks deliver goods to and from depots, it is as important to track the goods currently loaded on trucks as it is to know what is waiting at the depot.

Communication The communication criterion requires that the simulated vehicles can communicate with each other or some simulated infrastructure.

There is a distinction between this type of communication and the communication mentioned in the external control criterion. In the latter, programs and scripts are communicating with each other in order to facilitate the simulation itself. In this criterion, we consider *simulated* communication, where simulated entities are communicating with each other within the simulation.

The motivation for this criterion is the fact that AVs rarely operate without at least communicating to some control center. They may also communicate with each other in order to exchange information.

Third dimension The third dimension criterion requires that the simulator supports positioning of all entities in 3D-space as well as modeling road slope.

The motivation for this criterion is fuel efficiency simulation. For heavy vehicles, an incline can drastically impact fuel consumption, while electric vehicles can regain battery charge through regenerative braking on declines.

Fuel The fuel criterion requires that the simulator supports modeling fuel consumption.

For this criterion, only support for conventional gasoline- or diesel-fueled vehicles is needed for fulfillment. It is desirable if the simulator, in addition to fuel consumption, also tracks fuel capacity and the volume of fuel remaining, thus capping a vehicle's range. It is also desired that the simulation supports electric vehicles.

The motivation for this criterion is, predictably, also fuel efficiency simulation, in particular ones where a refueling step is part of the simulation. While not a significant problem for conventional vehicles, electric vehicles — especially ones for heavy transport — require frequent charging. It is not reasonable to assume that an electric transport vehicle can complete a full day's work without needing to charge, so it becomes necessary to simulate its battery drain so that charging can be scheduled.

Transparency The transparency criterion requires that the simulator's internal models are transparent to the user, so that the user can review and understand its processing.

This is one of two vague criteria. A perfectly non-transparent simulator is a black box, revealing nothing about its inner workings save the results which a simulation produces. The opposite — a perfectly transparent simulator — is harder to define. Information on the simulator's processing should be available to the user, but not so much that the user becomes inundated. To pass this criterion, a simulator must, at least, be near the center of that hypothetical scale.

The motivation for this criterion is the need for realism and accuracy in many simulations. In particular, it is important to be able to review that the simulator operates in a predictable, unobscured manner, so that the user can refine their scenario or strategies in an informed way.

Documentation & Community The documentation & community criterion requires that the simulator has a well-written and complete documentation along with

an active community of users and developers. This criterion will also consider the wealth of published scientific literature which uses the simulator.

Akin to Transparency above, this is a vague criterion. A simulator which completely fails this criterion would be abandoned and undocumented, having no published literature and unavailable source code. The opposite is a ubiquitous program, known by everyone, and actively developed, which has exhaustive documentation and well-commented, open source code. Neither of these extremes are likely to apply to any of our simulators, so we will grade them on the spectrum in between.

The motivation for this criterion is ease of use. Having good documentation along with a community of users functions as a safety net for a new user or developer coming to grips with the simulator. Even better if the process of developing for or extending the simulator is documented.

License The license criterion requires that the software qualifies as free and open source software (FOSS).

For the purposes of this criterion, we define "free" software as software that is under a license which conforms to the Free Software Foundation (FSF)'s Free Software Definition [23]. Similarly, "open source" software must be under a license that conforms with the Open Source Initiative (OSI)'s Open Source Definition [24]. To be FOSS, it must conform to both.

The motivation for this criterion is ease of acquisition and ease of use. An advantage of FOSS is that there are no licensing costs or other fees which must be paid to acquire the software, benefiting both home-users, academics, and corporations. While many licenses are lenient to home-users, commercial use is a significant problem, such as forbidding commercial use altogether or forbidding the resale of the software or its modifications.

With FOSS, these problems are taken care of. The "free" in "free and open source software" not only means "free of charge" but also "free as in freedom" — a common expression describing FOSS. A FOSS license expressly allows a user, whoever they may be or represent, to use the software for any purpose. The user may also modify the source code, redistribute their modifications, even if they charge for it. When incorporating a new piece of software into a development project, having source code access along with explicit freedom to modify it, this software could quickly be integrated and molded into something fit for the problem at hand.

3.3 Scenarios

As a complement to the eleven criteria, we have also defined a set of three scenarios which will be implemented in each simulator. This serves both to test certain criteria as well as to gain a practical understanding for how to work with a given simulator.

Each scenario is based on a problem or study described in the published literature. The three scenarios are defined below.

3.3.1 Traffic lights

In the traffic lights scenario, a simple road network containing at least one set of traffic lights will be constructed.

This scenario will evaluate if the simulator allows the network in which the vehicles operate to dynamically influence the operation of the vehicles. It can also be thought of as a test of the communication criterion, as traffic lights constitute a rudimentary form of V2I communication.

This scenario is inspired by Vaudrin, Erdmann, and Capus's study [12], in which they studied throughput at traffic lights when AVs were added into the traffic flow.

3.3.2 Mining operations

In the mining operations scenario a simple graph network will be constructed with at least two producer nodes and at least two consumer nodes. Within the graph, vehicles will be moving to retrieve goods from the producers and moving it to the consumers. This makes the system a form of PDVRP.

This scenario will evaluate the goods criterion, as goods must "exist", in some sense, at the producer prior to being picked up. Goods could also be interactively generated, such as at the press of a button, which will evaluate the on-line interaction criterion.

This scenario is inspired by the work of Karlsson and Steffenburg [25] who explored intelligent agents needing to navigate an underground mine and deliver ore from "producers" to consumers.

3.3.3 Generative traffic

In the generative traffic scenario a small town road network will be constructed to test if traffic density can vary as a function of time. In particular, the scenario will seek to mimic the "rush hour" effect, when traffic volume drastically increases at certain times of the day.

This scenario will evaluate if a simulator allows conditional generation of traffic and if it supports the concept of a "day", such that certain events occur at given times each day.

This scenario is inspired by Guggisberg Bicudo and Berkenbrock [26] who simulated the traffic flows caused by the complex labor shifts in the town of Joinville, Brazil.

4

Results

In this chapter, we will present the results of our analysis as described in chapter 3 above. The chapter begins with a summary and overview of the results in section 4.1, then proceeding to an in-depth presentation of results from each simulator in turn, starting with MATSim in section 4.2, RinSim in section 4.3, and finally SUMO in section 4.4.

4.1 Overview

Below, the simulators' criteria fulfillment is summarized in table 4.1. Of the three simulators, SUMO fulfilled the most criteria — nine of eleven — while both MATSim and RinSim fulfilled four of eleven.

Considering the evaluation scenarios, none of the simulators allowed us to implement all three scenarios, lacking some crucial feature. Only SUMO allowed us to implement the traffic lights scenario, while only RinSim allowed us to implement the mining operations scenario. All three simulators allowed us to implement the generative traffic scenario in some form.

4.2 Multi-Agent Transport Simulation

In this section, we present the results for Multi-Agent Transport Simulation (MATSim), starting with its construction and design as well as some notable works in the published literature, then presenting criteria fulfillment and the evaluation scenarios. The work in this thesis was done using MATSim version 0.10.1.

4.2.1 Construction and design

MATSim is written in Java and is distributed either as source code or as a precompiled executable jar-file. It is also available for import into an existing project as a Maven dependency.

Table 4.1: Summary of the criteria fulfillment for the three simulators

	Simulator	**MATSim**	**RinSim**	**SUMO**
	Type	Traffic	Logistics	Traffic
	Criterion			
1.	Ext. control	-	-	✓
2.	On-line inter.	-	-	✓
3.	GUI	-	✓	✓
4.	Sep. conf.	✓	-	✓
5.	Goods	-	✓	-
6.	Comm.	-	✓	-
7.	3D	-	-	✓
8.	Fuel	-	-	✓
9.	Transparency	✓	-	✓
10.	Docs. & Community	✓	-	✓
11.	License	✓	✓	✓

MATSim's basic model is demand-focused and agent-based, where agents — representing individual humans — generate demand by needing to go places, such as to work, to school, or to the store. This demand, along with a definition of where agents "live", "work", and "study" cause the agents to move, in turn causing traffic. This traffic can be modeled in many ways, from each agent having their own car to multi-modal public transit systems. While in transit, agents evaluate their plan for the day and re-plan it, if necessary, ensuring they can meet their goals even if the traffic situation is working against them.

This approach to traffic generation allows MATSim to generate rich data about city-wide demand and traffic patterns, predicting rush hours and lulls in traffic flow, as well as revealing bottle necks in the network. While MATSim is a microscopic simulator, it does this at the cost of a detailed vehicle simulation — indeed, MATSim is far more focused on the agents themselves than their individual vehicles. We nevertheless categorize MATSim as a traffic simulator, as its primary purpose is to simulate the flow of traffic, though this flow is driven by agents going about their daily business.

MATSim is designed to be highly modular [27], allowing a user to use no more or less of the simulator than they need. Similarly, this allows a skilled user to develop their own modules for the simulator, extending its function as they see fit.

4.2.2 Notable work

MATSim has seen some use in academic literature. Below, we present a sample of published works which use or build upon MATSim.

Maciejewski and Nagel [28], [29] integrated MATSim with an optimizer for VRPs. Their first paper looks at enabling on-line interaction between the two programs,

while the second looks at using this system to study different taxi routing and assignment strategies in an urban environment — a Dial-a-Ride Problem (DARP). They found that none of their tested strategies was universally better — instead, the choice of optimal strategy is dependent on the supply-to-demand ratio

Novosel et al. [30] used MATSim as a supplemental simulator to EnergyPLAN [31] in a case study of the impact of the introduction of electric vehicles on Croatia's national power grid. In this study, MATSim was used to generate data on vehicle use which was used to calculate power consumption, in turn informing the primary EnergyPLAN simulation.

Schröder et al. [32] added freight activities to MATSim through a set of new agent types which generate and coordinate shipping activities. They find that this method can drive a rich transport simulation as well as generate valuable data on the economy of the freight system.

4.2.3 Criteria

As mentioned in the overview above, MATSim fulfilled four of the eleven criteria. It notably passed no primary criteria, though there are caveats for all of them.

External control MATSim does not, in itself, support external control. Instead, if a user wishes to utilize MATSim with some external process, it would be necessary to write a tie-in module to MATSim, interfacing the two processes. This method seems relatively commonplace — for example, as mentioned in section 4.2.2 Maciejewski and Nagel [28] integrated MATSim with an external optimizer for VRPs.

Lacking a standard method for interfacing MATSim to another process, without modifying its source code, MATSim fails the first criterion.

On-line interaction Tying into both the first and third criteria, it is difficult to have on-line interaction without either external control or a GUI. Maciejewski and Nagel [28], [29] showed that it is possible to have an external process interact on-line with MATSim but, as mentioned above, this requires a custom built interface with the second process.

Again similar to the first criteria, since there is no standard method for on-line interaction which does not require source code modification, MATSim fails the second criterion.

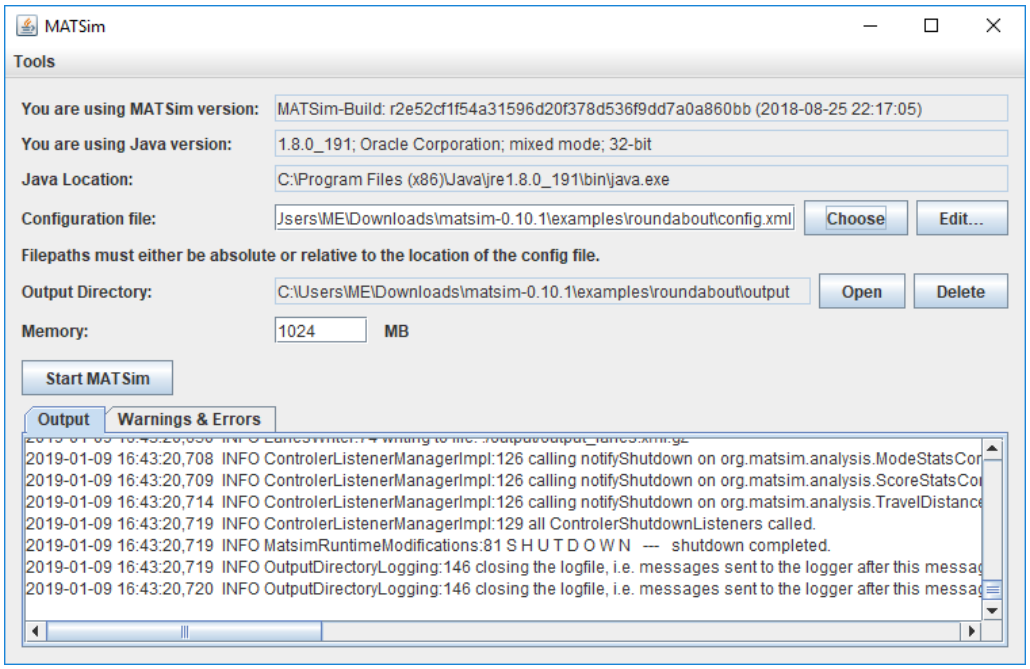

Figure 4.1: Screenshot of MATSim's scenario selection GUI.

GUI As hinted to above, MATSim does not have a visualization GUI of its own. It does come with a rudimentary GUI which can be used to select scenario file and start the simulator, but little else. This GUI is shown in figure 4.1.

To visualize a simulation one needs a third party visualizer. The most popular one is Simunto Via [33], a proprietary visualizer which is only capable of visualizing a scenario after MATSim has finished simulating it. An example of such a visualization is shown in figure 4.2. While this allows the user to rewind and fast-forward the playback, it does mean that a lengthy simulation can't be reviewed and debugged until after its completion.

There is a MATSim-native visualizer called OTFVis (On The Fly Visualizer) [34] which reportedly is capable of on-line visualization. However, this visualizer is not available without source code modification.

For the above reasons, MATSim fails the third criterion.

Separate configuration MATSim's configuration is done through XML files, describing its road network and agent plans. The internal syntax of these files is simple, which should allow a novice user to get started quickly. Thus, MATSim passes the fourth criterion.

Goods Looking again to Schröder et al. [32] shows that freight activities can be simulated in MATSim. However, there is no mention of the goods being physical entities moving through the network, or if they are properties of the vehicles

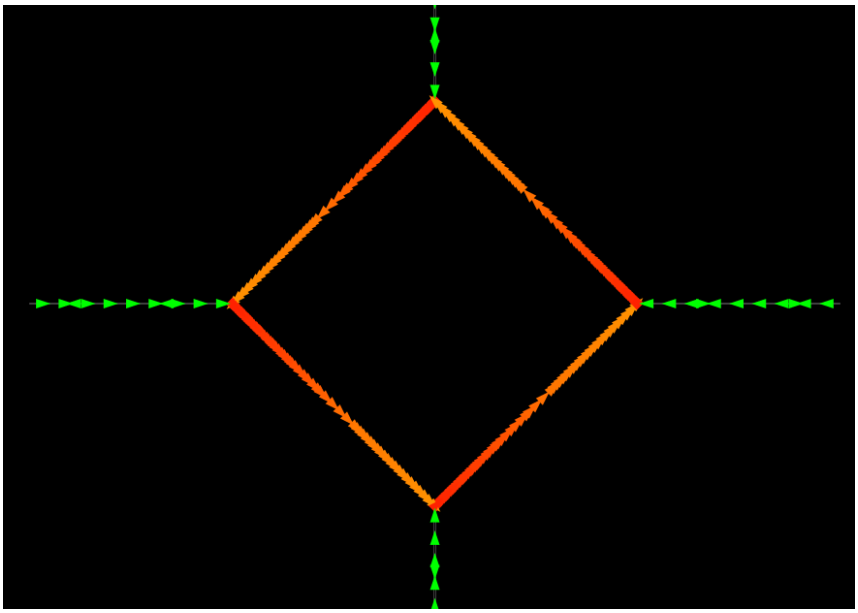

Figure 4.2: MATSim's roundabout scenario being visualized in Simunto's Via. The agents are color coded depending on their speed, where green is at target speed and red is far below target.

which transport them. As we are concerned with the movement of discrete, physical packages through a network, it follows that MATSim fails the fifth criterion.

Communication MATSim does not support communication in any way, nor have we found any reference using MATSim for simulation of communicating agents. Thus, it fails the sixth criterion.

Third dimension MATSim does not support height data in its networks. Consequently, it fails the seventh criterion.

Fuel MATSim does not support simulating fuel consumption in any meaningful way. Surprisingly, it does support vehicle refueling actions, at least for electric vehicles, as noted by Waraich and Bischoff [35]. Tangentially, Kickhöfer shows that it is possible to derive emissions data by studying simulation output [36]. This suggests that it should also be possible to derive fuel consumption data from the same output. However, Kickhöfer's models only consider vehicle average road speed, since acceleration, deceleration, and slope are not known — fuel consumption models without these input data will be less accurate.

That said, as MATSim does not support simulating fuel consumption, it follows that it fails the eighth criterion.

Transparency The internal models of MATSim, in particular its action-based agent engine are well explained, such as by Nagel and Flötteröd in [37]. MATSim itself also generates a wealth of output files by default, outlining agent movements and actions, which can assist a new user in understanding what the simulator is doing.

Thus, MATSim passes the ninth criterion.

Documentation & Community MATSim's primary reference is "The Book", as it is often known [38]. This book serves as a tutorial for new users, a guide to extending the MATSim code, and providing the reader with a sample of scenarios which have been simulated in MATSim.

The project is community-driven with an active group of developers. Throughout the project's eleven-year history, 45 developers have made contributions to MATSim's GitHub repository [39]. Of these, 21 have made contributions since the start of 2018, suggesting the project is being actively and continuously developed. The community has also held annual user meetings since 2009, suggesting that there is an active community of users in addition to an active developer group.

Based on the above, it is reasonable to conclude that MATSim passes the tenth criterion.

License MATSim is licensed under the GNU General Public License (GPL) version 2, with an explicit permission to license derivative works under the GPL version 3. The GPL is the FSF's gold standard license — written and maintained by the FSF itself — so is naturally recognized as a free license. Similarly, the OSI has approved the GPL, both version 2 and 3.

It should be noted that the Via visualizer software is neither free nor open source. While available for free from the Simunto website, using it requires a license. For private individuals, a free six months license is available after registration, but this also limits the number of visualized vehicles to 500. Full licenses for for researchers cost 1000 € while commercial licenses start at 4000 € per license per year. However, as the visualizer is not an integral part of MATSim, this does not inhibit MATSim itself from passing the eleventh criterion.

4.2.4 Scenarios

MATSim supported implementing one of the three evaluation scenarios. There is good reason to believe that there is support in MATSim to implement the remaining two scenarios as well, but only after modifying its source code. As such extensions are beyond the scope of this thesis, we did not implement those scenarios.

Traffic lights Using conventional configuration means, there is no readily available way for MATSim to simulate traffic lights or similar environmental traffic impediments. There is support for traffic lights and other signaling methods in the source code, but without modifying the source code, this functionality may be impossible to use. There is a slight possibility that it could be done, but there was no user documentation for the signaling module and no example scenarios which implemented traffic lights.

Mining operations Similarly to the traffic lights scenario above, there is no readily available module in MATSim which supports the simulation of goods in any meaningful way. Again, akin to the signaling module, there is a freight module available, but its user documentation is missing. Its Java developer documentation, meanwhile, gives no hints as to how one could make use of the module without modifying source code.

As MATSim's model is generally unconcerned with simulating real-world environments, it is improbable that non-agent-driven parcels could exist within the simulated world, especially independent of vehicles. Zilske et al. [40] talks of a transportation simulation toolkit for MATSim, which may add capabilities for goods simulations, but makes no mention of goods existing independently of their transport vehicles.

Generative traffic MATSim has an explicit daily cycle as part of its simulation — all agent actions are planned for and carried out during a 24 h day. Timekeeping is done with a simulated time-of-day clock, allowing the user to designate agents taking some action at a time of day, rather than at a simulator timecode. That is to say that events can be defined as taking place at 08:30, simulated time, rather than a more arbitrary "40 s from simulator start."

All MATSim traffic is explicitly defined, allowing a user to generate sudden traffic flows whenever they wish. Traffic is a side-effect of agent mobility, however, since a user specifies the routes, plans, and activities of individual agents, not individual vehicles — an agent may use many different modes of transport to carry out its plan.

This was tested by modifying the basic example scenario "equil", shown in figure 4.3, to delay the departure of some agents. In the scenario, 100 agents leave a specific point at 08:00, while we modified the scenario to delay a number of agents' departure to 10:00. This was done through modifying the trips as defined in `trips100.xml`, editing the start-of-day activity's end time.

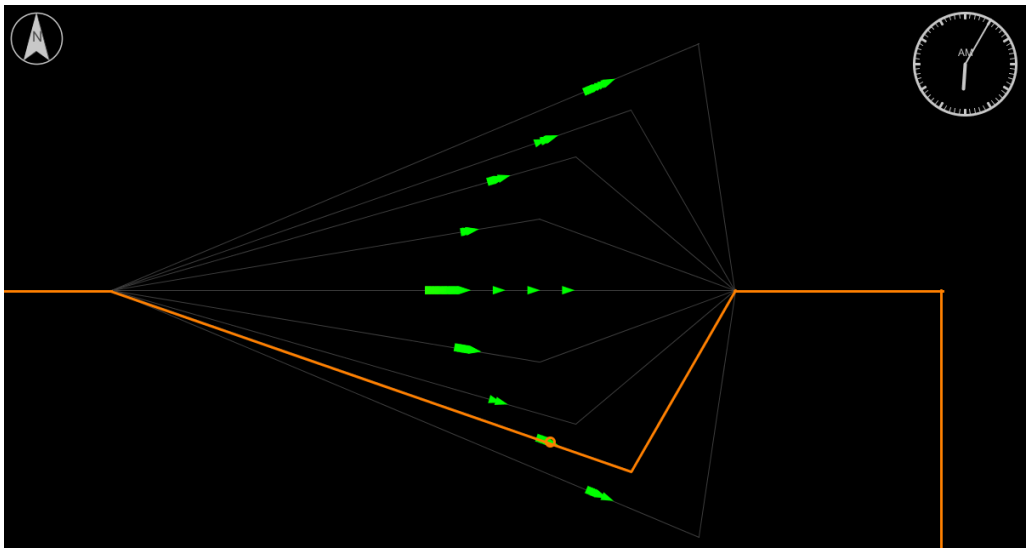

Figure 4.3: Via visualization of MATSim's example scenario with one agent selected, highlighting it and its path.

4.3 RinSim

Similarly to section 4.2, we will now present the results for RinSim. The work in this thesis was done using RinSim version 4.4.6 [41].

4.3.1 Construction and design

RinSim is developed in Java and is available for use either as source code or as a Maven dependency. This all but requires that RinSim is used as a library rather than as a standalone simulator. It also stands out from the other two in the sense that it is designed as a logistics simulator, rather than a traffic simulator. This means that it is focused on the efficient movement of goods through some network, rather than simulating accurate vehicle behavior and traffic flows.

RinSim is designed as a multi-agent system, where intelligent agents make and execute their own plans as well as interacting with the world and other agents. There are no preset agent types, only general agent abstract classes, so each simulation scenario is by necessity custom built to its purpose. This affords the developer a great deal of power, allowing RinSim to be used to simulate a vast number of scenarios, but does require a significant amount of work to create the scenario itself.

In terms of its internal model, RinSim is separated into strict modules. This means, for one thing, that the entire simulation shares the same model of time. It also means that the simulation model is kept separate from the multi-agent system and its solver, as well as the visualization. This segmentation is intended to make RinSim easily extensible.

4.3.2 Notable work

RinSim has not yet been used to any great extent in academic literature. All references we found, save one, were written by RinSim's creator and namesake Rinde van Lon — the exception being the recent Master's thesis by Karlsson and Steffenburg [25].

The seminal work on RinSim was written by van Lon and Holvoet in 2012 which introduces RinSim, the motivation behind its creation, and its internal structure [42]. Since then, van Lon and Holvoet have written on the topic of solving logistics problems using evolving agents in RinSim [43]. Later, van Lon, Branke, and Holvoet continued working in RinSim to study using genetic algorithms for planning missions for freight agents [44].

The last paper appears to have inspired Karlsson and Steffenburg's thesis, wherein they modeled a set of agents having to navigate a constrained, shared environment. The agents had to avoid colliding with each other with only limited capability to detect other agents' positions. All the while, the agents had to fulfill transport contracts, moving ore through a simulated mine. As noted previously, this work was the inspiration for the mining operations scenario.

4.3.3 Criteria

RinSim, just like MATSim, fulfilled four of the eleven criteria. However, RinSim fulfilled one primary criteria, which MATSim did not.

There is a caveat to all of RinSim's criteria — by virtue of RinSim's construction as a library rather than as a stand-alone simulator, it would be feasible to implement support for nearly all of the criteria below. However, as we have explicitly disallowed source code edits for the criteria, the fulfillment is based on what is supported in RinSim's core library.

External control RinSim does not itself support any interface to another process. Thus, it fails the first criterion.

On-line interaction RinSim does not support on-line interaction. RinSim's GUI does not provide a way to interact with the simulation while it is running, beyond controlling the rate at which it runs. Further, there is no way to interact with the simulator through a terminal or other keyboard input. Therefore, RinSim fails the second criterion.

GUI RinSim comes with a visualization GUI built in. While it is rather rudimentary, it is also flexible, having rendering defined per-scenario. It only supports

on-line visualization — viewing the simulation as it is carried out.

Thus, RinSim passes the third criterion.

Separate configuration RinSim scenarios are written as Java programs, using RinSim as a Java library. This affords RinSim a very high level of flexibility, but this comes at the cost of a significant learning curve. Since the separate configuration criterion requires that scenarios can be defined without modifying source code, RinSim fails the fourth criterion.

Goods RinSim is built around the concept of transporting goods, with special `parcel` abstract class provided in RinSim's core. This general `parcel` has a source, a destination, a size, and independent delays for loading/unloading it. However, this definition reveals limitations for certain problems. For example, it will cause difficulties when modeling a single-origin, multiple-destination PDVRP problem, as the `parcel`'s destination is singular and fixed from the moment of creation. Similarly, the `parcel`'s size is implemented as an integer, saying nothing of size or weight. This works with trucks and depots, which have an integer capacity, but does not allow for weight- or size limits for goods.

That being said, RinSim has support for inanimate goods which exist independently of their containers. Thus, it passes the fifth criterion.

Communication RinSim supports communication, allowing agents to send messages — with any content — to each other within the simulation. RinSim's communication model also allows for an agent to have its own communication range and reliability. The range impacts how far an agent can transmit a message and the reliability defines the chance of a transmission succeeding. Karlsson and Steffenburg made use of this communication model for agents to locate each other and perform collision avoidance, when necessary.

An example simulation — the `CommExample` included in RinSim — which uses communicating agents is shown in figure 4.4. In this scenario the agents move randomly, attempting to greet all other agents. As agents have randomly generated range and reliability, it may take a long time before everyone has met everyone.

Given the above, it follows that RinSim passes the sixth criterion.

Third dimension RinSim uses a simple two-dimensional positioning system, built around its `Point` class. This could potentially be extended to include a third dimension, but due to the ubiquitous use of the `Point` class, this would be a significant undertaking.

It follows that RinSim fails the seventh criterion.

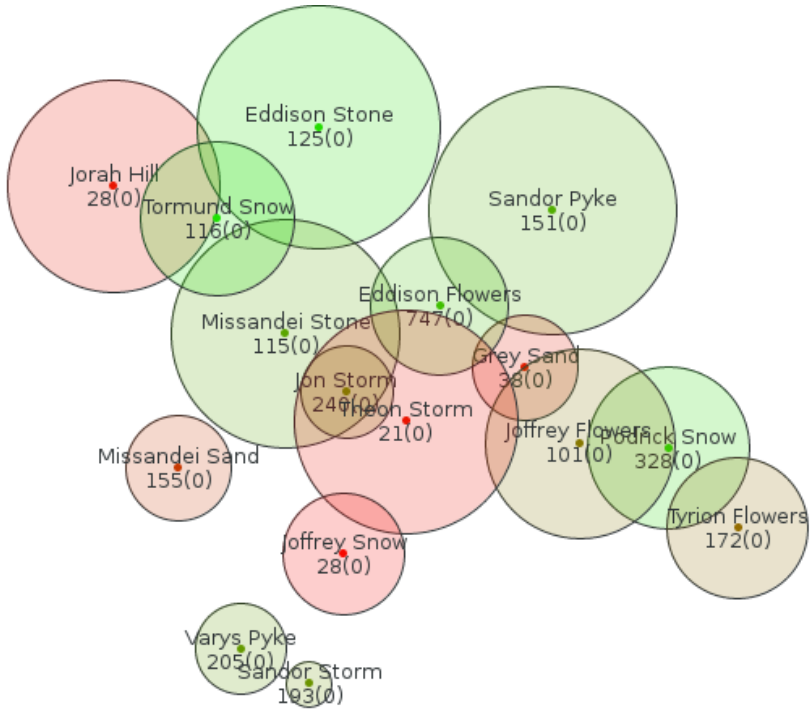

Figure 4.4: RinSim's communication demo, simulating 16 agents randomly moving around a plane, greeting each other as they go.

Fuel Vehicles in RinSim are simplified models of real-world vehicles, having no weight, size, or driving properties. Every vehicle — unless the basic model is extended — has infinite acceleration and deceleration and has a turning radius of zero. The concept of fuel does not exist in RinSim and, given the above, would require a significant amount of work to implement, as a kinematic model for a vehicle is also missing.

Thus, RinSim fails the eighth criterion.

Transparency It is difficult to judge the transparency criterion for RinSim. On the one hand, the example scenarios provided are difficult to understand and code quality is debatable. On the other hand, since a user by necessity needs to be familiar with the RinSim source code to construct scenarios, it could be argued that the internal models are transparent.

The intention of the transparency criterion was for transparency to come from the user being presented with or being able to seek information in the application, not the source code. Furthermore, understanding the source code requires a high level of technical knowledge, which is not something that could be expected of most users.

For the reasons above, it is reasonable to conclude that RinSim fails the ninth criterion.

Documentation & Community RinSim does not have any form of user documentation beyond how to download the source or import it as a Maven dependency. There is source code documentation, in the form of Javadocs, but it is incomplete.

As for community, RinSim has none — it is primarily and near-solely developed by its creator and maintainer Rinde van Lon. van Lon uses RinSim as a tool for researching genetic algorithms and for teaching multi-agent systems [42], which appears to drive development. However, only nine persons have ever contributed to the RinSim GitHub repository [45]. Of these nine, van Lon alone accounts for 84 % of contributions[1]. The number two contributor, Bartosz Michalik, accounts for 13 % of contributions and has not contributed since 2013.

Based on the above, it is reasonable to conclude that RinSim fails the tenth criterion.

License RinSim is licensed under the Apache 2.0 license, which is fully recognized and approved by both the OSI and the FSF. Thus, RinSim passes the eleventh criterion.

4.3.4 Scenarios

RinSim supported implementing two of the three scenarios, only failing to support the traffic lights scenario. The same caveat that applied to RinSim's criteria applies here, however — since RinSim behaves like a library, it is fully possible to build support for all the scenarios. Doing so was beyond the scope of this thesis, however.

Traffic lights The only option to build a traffic light or other environmental effect on the network is to build the functionality into the scenario code. In the case of traffic lights or some other time-varying effect, this requires constructing listeners to timing events from the simulator core. Interfaces for such listeners do exist, but developing a listener to interface with the simulator was beyond the scope of this evaluation scenario.

Mining operations This scenario is based on the `FactoryDemo` which is supplied as one of RinSim's example scenarios. A screenshot of this scenario is shown in figure 4.5. We were able to change the destination of parcels in order to spell out a different string as well as modify the network in which the Automatic Guided Vehicles (AGVs) are operating by randomly adding edges between nodes.

[1] Calculated as average of share of total number of commits and share of total number of lines of code changed.

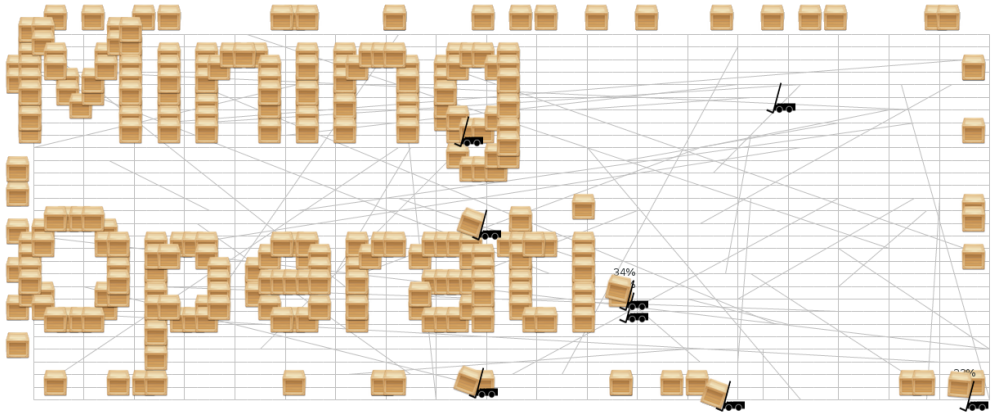

Figure 4.5: RinSim simulating AGVs as they move parcels in order to spell out "Mining Operations."

As noted above in the results for the goods criterion, RinSim cannot handle a scenario where a parcel has more than one possible recipient, so the original definition of the scenario could not be implemented. The parcels could, however, exist in the world in "depots", independently of vehicles.

Studying a basic form of this scenario was not difficult, since it is largely based on the existing demo. However, to conform with the definition of this scenario, a special `Parcel` class would need to be created in order to support open-ended deliveries. This was out of scope for this scenario.

Generative traffic This scenario is based on the `GradientFieldDemo`, another provided example. In it, buses and passengers are generated over time and the buses had to pick up the passengers, before they got tired of waiting, and deliver them to their destination. When all passengers had been delivered, the day was over. A screenshot of this scenario is shown 4.6.

RinSim's core supports both `VehicleAddEvents` and `ParcelAddEvents` — the passengers in this scenario are handled as parcels — which shows that RinSim is capable of generative traffic. While one could also think that this shows that RinSim supports the concept of a "day", that is a concept specifically designed in this scenario and is not a feature of the RinSim core.

4.4 Simulation of Urban MObility

Akin to the two previous sections, we will now present the results for Simulation of Urban MObility (SUMO). The work in this thesis was done using SUMO version 1.1.0.

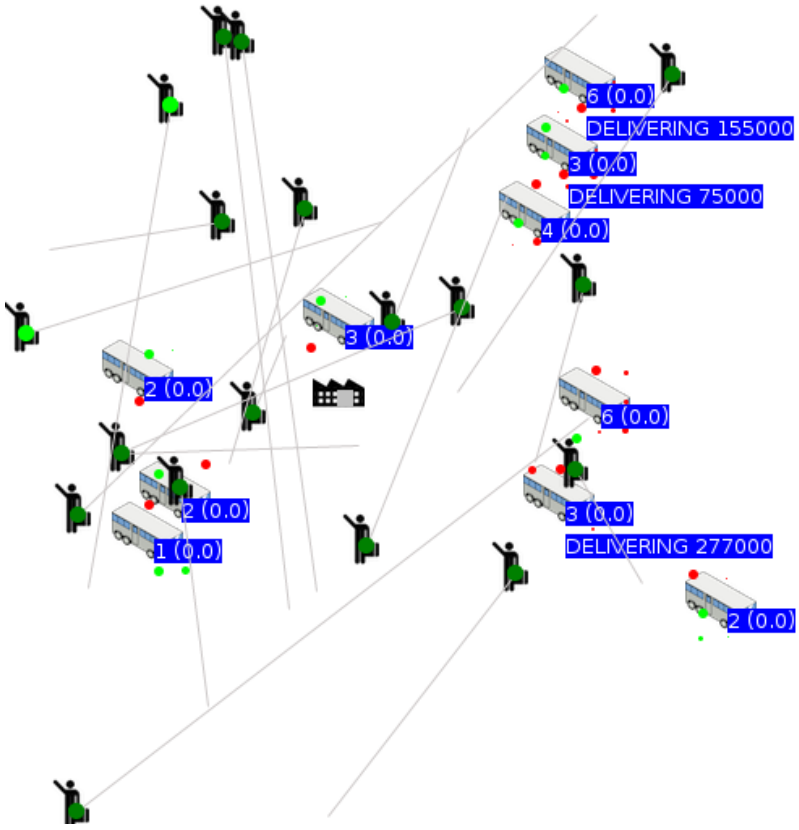

Figure 4.6: RinSim simulating buses in the process of picking up and delivering passengers.

4.4.1 Construction and design

SUMO is written in C++ and is available as source code or with an installer for most major operating systems.

SUMO is also the oldest of the three, having its roots in 2001, when the project was started by the German Aerospace Center [46]. It is a microscopic traffic simulator by design, though its capabilities have grown over the years, including both meso- and macroscopic models.

SUMO should be thought of as a suite of programs, rather than as one monolithic simulator. The standard installation comes with a host of utility programs and scripts, including generators for random traffic and road networks, converters from map data to road networks, or optimizers. There are also a number of interchangeable components, for example allowing the user to decide which driver model or router they want to use on a scenario-by-scenario basis.

One such driver model is Krauß's [9] — SUMO's default driver model. This model is based on one simple assumption: vehicles, in general, do not collide. It also requires per-vehicle definition of acceleration and deceleration. From this, the model can simulate vehicle interaction and flow at large scales. Krauß notes that traffic flow

comes in three states — free flow, synchronized flow, or jammed — where the model is capable of simulating all three and the transitions in between.

SUMO has been extended a number of times through the years, where the modifications either have been folded into SUMO itself, or have taken on a life of their own as a separate project. Such an example is iTETRIS (the Open Simulation Platform for Intelligent Transport System Services) [47], which has extended SUMO with complex emission and noise models, routers, and communication protocols.

4.4.2 Notable work

Of our three simulators, SUMO is the most used in the published literature. The following is a sample of notable work regarding SUMO and AVs.

Basso et al. [14] used SUMO as a part in their work for optimizing the routing of electric buses. SUMO was used to simulate the movement of the buses, generating realistic acceleration/deceleration data, which was later post-processed using Volvo's Global Simulation Platform (GSP) to calculate energy consumption.

Bjärkvik et al. [48] attempted to extend SUMO with a more advanced driver model for AVs called the Intelligent Driver Model (IDM). They showed that this was possible, though difficult, due to the interaction between "normal" drivers, using the Kraußmodel, and AVs using the IDM. In particular, they noted that lane changing became problematic, where IDM vehicles could deadlock against other vehicles on large roads.

Vaudrin, Erdmann, and Capus used SUMO to simulate and study how the introduction of AVs in traffic flow affected throughput at traffic lights [12]. Their model of an AV was not as advanced as Bjärkvik et al.'s, instead merely shortening the response delay between a light turning green and the vehicle starting to accelerate. Their results are inconclusive, however, as they cannot show a significant decrease in waiting time when AVs are introduced. As mentioned above, this work served as inspiration for the traffic lights scenario.

4.4.3 Criteria

SUMO fulfilled nine of the eleven criteria, including all primary criteria.

External control SUMO supports external control, though not entirely on its own. SUMO gets its external control capabilities through Traffic Control Interface (TraCI), originally described by Wegener et al. in 2008 [49]. TraCI itself is a simulator-agnostic protocol for traffic simulator control, but the developers of SUMO integrated it into SUMO's core shortly after its introduction. Nowadays, SUMO comes packaged with a Python library which fully implements the TraCI protocol,

allowing a developer to make use of external control out of the box. There are other language libraries, but the Python version is complete and is maintained as part of the SUMO suite.

Thus, the power of TraCI and the fact that SUMO supports it without any source code changes motivates SUMO passing the first criterion.

On-line interaction SUMO has full support for on-line interaction with a running simulation. The primary and most powerful method to facilitate on-line interaction is through TraCI, which allows an external process complete read/write access to the simulation. While TraCI is in use, SUMO delegates time-stepping to the TraCI client, ensuring that it will never miss out on information.

A user can also interact with a running simulation through the simulator GUI, though this does not afford the user as much power as the TraCI interface. The user can, however, read road or vehicle data, as well as plot time-varying values such as vehicle speed profiles. The GUI also allows the user to, for example, close a road segment or lane while the simulation is running. If the simulated vehicles do not use a dynamic router, this tends to eventually result in traffic grinding to a complete halt.

Given the two methods to influence a running simulation, it follows that SUMO passes the second criterion.

GUI SUMO includes two prominent GUI applications in its suite. The first is the simulator GUI itself, where a user can observe and interact with simulations as they play out. This GUI is independent of the simulator itself, which can be run in a non-graphical mode, though this severely restricts a user's ability to interact with a running simulation. The second GUI application is SUMO's graphical network editor, allowing a user to draw road networks for use in the main simulator.

Thus, SUMO passes the third criterion.

Separate configuration SUMO networks, traffic flows, vehicle types, and other configuration is defined in a series of XML-files. While cumbersome to type by hand — especially the often lengthy network definition files, which define all roads and their properties — these files are independent from the simulator source code. Thus, SUMO passes the fourth criterion.

Goods SUMO has extensive support for multi-modal person transport simulation, including pedestrians moving without a vehicle, which suggests that it may also support goods transport simulation. Indeed, there is a container model in place which is intended to simulate the movement of inanimate cargo through a network. However, this container model leaves a great deal to be desired.

A major shortcoming is that container routing is handled backwards — the packages route and handle themselves. In a real freight scenario, a freight planner would assign containers to a vehicle and then dispatch the vehicle to load and transport them. In SUMO's model, each container is "aware" of its full transport plan and will load itself onto an available vehicle if it is near enough and is headed to the container's goal. This is reminiscent of how a human would traverse public transport, but it does not translate well to goods transport. It should still be possible to implement a goods transport simulation in this model, however.

The goods criterion requires that we can simulate inanimate objects independent of the vehicles that transport them. However, as SUMO's containers can load and unload themselves they cannot be considered fully inanimate. Thus, SUMO fails the fifth criterion.

Communication SUMO does not support any model for V2V or V2I communication — collectively known as Vehicle-to-Somewhere (V2X) communication. Instead, if a user wishes to simulate vehicle communication, for whatever reason, this needs to be facilitated through attaching a network simulator to SUMO via TraCI. This was, in fact, exactly what Wegener et al. did in their paper which introduced TraCI [49].

While there is at least one tried-and-tested method to realize V2X communication in SUMO, it is not a feature of SUMO itself. Thus, it fails the sixth criterion.

Third dimension SUMO has support for three-dimensional positioning of all elements, including vehicles. SUMO specifically calculates vehicle slope dependent on road slope and travel direction, which could be a useful for fuel consumption calculations. Thus, SUMO passes the seventh criterion.

Fuel SUMO supports calculating fuel consumption for individual vehicles. However, these metrics are based on emission models which are included in the SUMO suite and may not accurately represent an experimental vehicle or modern electric vehicles. A user may define their own emission and consumption models to get around this problem, which can be done without editing source code.

Further, SUMO does not simulate an individual vehicle's fuel type or volume or an electric vehicle's battery capacity or charge. Thus, a vehicle in SUMO can never run out of fuel, nor does SUMO support refueling or recharging operations.

However, as the criterion only required support for fuel consumption simulation for conventional vehicles, SUMO passes the eighth criterion.

Transparency SUMO's inner workings are heavily based on peer-reviewed publications, meaning they are open to scrutiny. One example is the standard car following model, as mentioned above, which is based on Krauß's PhD dissertation [9].

Similarly, articles such as Behrisch et al. [10] describe the development of SUMO and how it has evolved since its inception.

During runtime a user can inspect a simulation as it is in progress through the GUI, viewing vehicle states and road properties as desired.

Thus, it is reasonable to conclude that SUMO passes the ninth criterion.

Documentation & Community SUMO is the most widely used simulator of the four with a significant presence in academic research — Krajzewicz noted that SUMO had been featured in hundreds of published articles by 2012 [50]. It is also well documented, sporting a publicly editable and well cited wiki, which also functions as its manual.

Regarding community, SUMO has an annual user conference — which in 2017 focused especially on autonomous mobility [51] — suggesting that there is a significant community of active users. SUMO's GitHub repository also suggests an active, albeit small, community of developers [52]. 16 developers in total have made contributions and all but three of these have made contributions since July 2018. That said, the project appears to only recently have migrated to GitHub, as there are no commits older than April 2018.

Based on the above, it is reasonable to conclude that SUMO passes the tenth criterion.

License SUMO is licensed under the Eclipse Public License 2.0, which is approved by the OSI. The FSF recognizes this as a free license, while also noting that it is incompatible with its preferred license, the GPL version 3, due to weaker redistribution terms.

Nevertheless, SUMO meets the criteria for FOSS, meaning it passes the eleventh criterion.

4.4.4 Scenarios

SUMO supported implementing two of the three scenarios, having extensive support for traffic lights and sufficient support for generative traffic, but lacking support for goods simulations.

The scenarios below were all based on the open scenarios provided by Bieker et al. [53], simulating roads and traffic in the city of Bologna, Italy.

Traffic lights The traffic lights scenario was based on the `Acosta` scenario, which already includes a number of TLSs, such as the one shown in figure 4.7. These

TLSs are presumably programmed to match the real-world timings of the modeled intersections in Bologna.

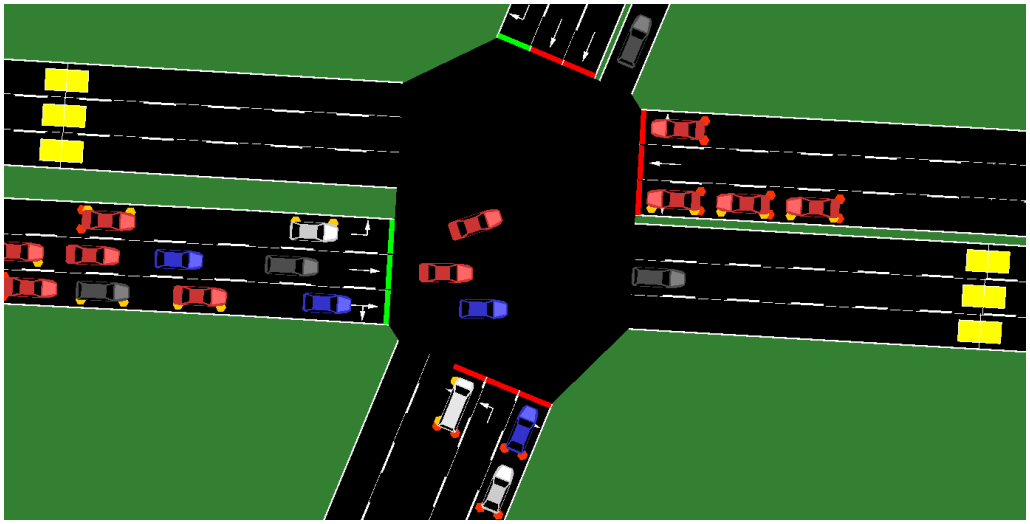

Figure 4.7: Screenshot of SUMO simulating a large intersection in the Acosta scenario, showing drivers obeying the traffic lights.

We were able to modify the light timing programs easily through modifying the `acosta_tls.add.xml` file, which describes each TLS's program. Through editing this file it was possible to make individual lights in an intersection permanently green, disable them to force drivers to yield, or simply write erratic timing. The resulting timings could be observed both in the main simulation view, as seen in figure 4.7 as colored bars by the intersection, or as a phase graph, as seen in figure 4.8. This phase graph could be viewed based either on the static program, showing what the phase graph *should* look like, or as a tracking graph, showing the *actual* phases over time in the intersection — figure 4.8 shows the latter.

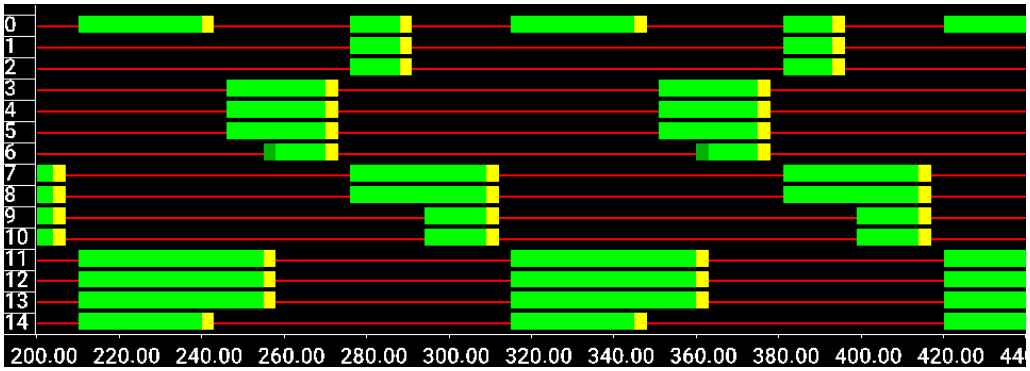

Figure 4.8: A SUMO traffic light tracking graph, showing the different phases of lights in a large intersection.

Though none of the TLSs in the `Acosta` scenario implemented it, SUMO supports dynamic light timing in addition to the static light programming tested above. Such dynamic programs can be based on either demand or waiting time, informed by induction loops embedded in the road, sensing cars passing or idling over them.

Such induction loops can be seen in figure 4.7 as yellow rectangles at the left and right edges. These loops only measure intersection throughput and do not inform the TLS. When used, however, they allow SUMO to simulate a complex feedback loop, where a TLS timing program impacts traffic conditions which in turn impact the TLS timings.

Mining operations The mining operations scenario was based on the `Acosta_persontrips` scenario, which extends the basic `Acosta` scenario with pedestrians and persons utilizing public transport. This suggested that it could be modified to simulate the movement of goods, rather than people.

There are two possibilities for implementing a goods transport scenario in SUMO. One can either re-interpret persons to "pretend" to be packages, or one can use SUMO's rudimentary container model. In a sense, the container model already is a re-interpretation of the passenger model, as it is the container itself, not some freight planner, which controls its route and how it is loaded/unloaded.

A problem with either option is that neither allows an open-ended delivery — a person or a container has a singular destination, so a VRPPD with multiple possible destinations for a single person or container cannot be easily simulated. Nor is it possible for containers to be of varying size — a SUMO transport vehicle can take a fixed number of containers, but a container cannot consume more than one "slot" in the vehicle.

The way SUMO implements loading and unloading of containers could cause problems in larger scale freight networks. As mentioned above, the container itself contains information about its source, destination, and route, including which vehicle will transport it. If that vehicle passes nearby the container's position, the container will load itself and, similarly, unload itself at the destination. While it should be possible to implement complex networks in this model, it does not match reality wherein a container is a dumb, inanimate object which is loaded and unloaded by separate operators. The container itself may have both a sender and an intended recipient, but it is not responsible for its routing or vehicle assignment.

Generative traffic All traffic in SUMO is inherently generative. A scenario's routes-file defines each vehicle precisely: where it will enter the simulation, at what time, and where it will exit. In most cases this definition also includes a predetermined route, though SUMO does support routing vehicles dynamically.

To test this, the `Acosta` scenario was again used as a base, this time to test injecting a stream of traffic at a certain point in time. This is possible through SUMO's `flow` definition. Instead of defining individual vehicles, this allows the user to define a flow of identical vehicles from some point in the network — the flow needs not have an endpoint — such that some number of vehicles are generated in a given time interval. In our evaluation, a flow of 42 vehicles was generated at one vehicle per ten seconds, starting at the green dot and moving to the blue dot shown in figure 4.9.

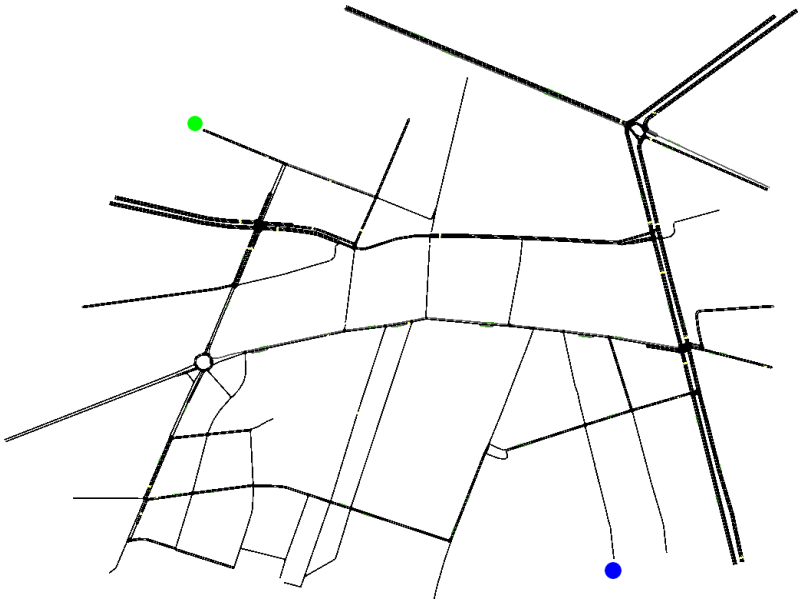

Figure 4.9: View of the Acosta scenario network in SUMO with an entry and exit to the network marked.

This could feasibly be used to simulate rush hour traffic, as describes in the scenario definition.

SUMO does not, however, have a general concept of a "day," instead having a clock counting seconds from the simulation start — the time of day in the simulation is not known or defined. If one wished to simulate identical traffic two days in a row, one would need to duplicate all route information from the first day and increment the departure times by 86 400 s. There *is* support in TLS programming for time-dependent program switching, allowing different timing programs to be used during the day, at night, or during the weekend. However, this is still dependent on the user interpreting SUMO's absolute time to some time of day in the simulation. Further, it is up to the user to ensure all definitions — routing and TLS — use the same reference point, as SUMO is only aware of its own absolute time. As an example, if the route definitions are written assuming 0 s to be Monday 00:00 while the TLS programs are written assuming 0 s to be Monday 08:00, SUMO will not be aware of this discrepancy, but the simulation will clearly be faulty.

5

Discussion

In this chapter, we will discuss the results presented above in order to compare and contrast the three simulators we have studied. In section 5.1, we will discuss our simulators from the context of our evaluation scenarios, looking at how well suited they are to handle such problems. In section 5.2 we will discuss SimMobility and why it, ultimately, did not get to be a part of this study. Finally, we close out the chapter with a brief discussion on the aspects of ethics and sustainability for both AVs and traffic simulation in section 5.3.

5.1 Comparison

The results presented in chapter 4 have given a general idea of what each simulator is capable of. In this section we will compare the three simulators, scenario by scenario, and discuss their relative strengths and shortcomings. In section 5.1.4, we will look at it from the other direction and discuss the sorts of problem each simulator is best suited for. Finally, in section 5.1.5 we will discuss the advantages and disadvantages of extending one simulator to fit a set of tasks or using different simulators for problems in the set.

5.1.1 Traffic lights

The traffic lights scenario studied whether or not each simulator would support a scenario where something in the environment — such as traffic lights — could affect the movement of vehicles. We found that only SUMO supported this out-of-the-box. RinSim, as mentioned previously, has the potential to support *any* scenario, including this one, since scenario configuration is practically the same thing as extending the simulator. Lastly, MATSim appears to support traffic lights though we were unable to test this, lacking adequate documentation.

The two simulators which ostensibly support traffic lights — SUMO and MATSim — are the best choices if one wishes to examine a traffic network with such environmental impediments. RinSim, while capable of simulating something like it, is designed around the idea of moving goods, not the movement of traffic. Similarly,

MATSim is generally more concerned with the movement of people rather than vehicles — the cars which are simulated in MATSim are caused by the movement of agents, not generated for the purpose of traffic. For much the same reason, MATSim does not have a good kinematic model of its vehicles — they accelerate and decelerate instantly. Thus, traffic lights in MATSim can influence the flow of traffic, but will not do so in a very realistic way.

SUMO is designed for realistic, microscopic simulation of vehicles in traffic, and so handles this scenario well. As mentioned in the results, SUMO's TLS programming allows for many modes beyond simply red or green and the simulated vehicles respond accordingly. Using the Krauß driver model, each vehicle has its own acceleration, deceleration, adherence to the maximum road speed, and reaction time.

An example of realistic traffic behavior which stems from this driver model is the "traffic wave" phenomenon, which occurs both in SUMO and in the real world. This phenomenon happens, for example, when multiple vehicles are waiting at a red light which turns green. The first vehicle will accelerate, then a short moment after that the second vehicle will have had time to react and starts accelerating. Another moment later and the third vehicle will accelerate, and so on, creating a "wave" moving backward through the waiting vehicles. This phenomenon can be seen in figure 4.7 (see page 37) in the vehicles entering the intersection from the left. The first row of vehicles are already in the intersection, while the second row has started accelerating. The third row and beyond still appear stationary.

The Vaudrin, Erdmann, and Capus study [12] which inspired this scenario was simulated in SUMO, so it is perhaps not surprising that it fared the best. Indeed, the study looked at adjusting the above mentioned reaction time of simulated AVs, along with changing TLS timing, to reduce the traffic wave phenomenon which, in turn, should increase throughput at an intersection.

5.1.2 Mining operations

The mining operations scenario studied if each simulator would support a scenario where goods were moved from some producers to some consumers — a Pickup and Delivery Vehicle Routing Problem (PDVRP). We found that only RinSim solved this in any meaningful way, though no simulator could simulate the exact scenario specified in section 3.3.2. Beyond merely simulating the movement of goods — which was handled by the goods criterion — the scenario required that goods from any producer could satisfy any consumer. In particular, a unit of goods could not have a singular intended recipient just as, in a mine, a "unit" of ore may have more than one processing station or cart that could receive it. Unfortunately, no simulator allowed such a transport network.

MATSim and SUMO both failed to even begin to implement this scenario, having both failed the goods criterion. SUMO gets a little further than MATSim by having its container model, though this model is even less compatible with the single-sender,

multiple-recipient PDVRP model than RinSim's parcel model. Both SUMO's container and RinSim's parcel have the singular sender and recipient as properties of the object, but SUMO's container also has the plan for traversing the transport network as a property. As noted in section 4.4.4, this is akin to how a person might reason about moving through public transportation, which does not translate over to the movement of inanimate objects through a transport network.

RinSim stands out in this scenario, not only for being the only simulator to support it, but also for being built around the idea of moving goods. RinSim, as noted in 4.3.1 and table 4.1 is a logistics simulator whereas the other two are traffic simulators. It is designed to study the interplay of agents carrying out transportation tasks.

As also noted above, RinSim is more of a simulation development library and less of a simulator in itself. While configuring RinSim involves Java development — a task which may be too complex for many users — it affords a skilled user an *immense* amount of power to simulate any scenario they might care to. The current, basic implementation of the `parcel` class in RinSim's core library does not allow a parcel to have more than one recipient. However, it is entirely possible, as shown by Karlsson and Steffenburg [25], to construct a scenario where the agents themselves have to choose where to deliver a parcel — the parcel itself has no preference.

Karlsson and Steffenburg's thesis was the inspiration for this scenario. They used RinSim in their work so it is, again, unsurprising that it did as well as it did. What is surprising, however, is how poorly the other two simulators did. SUMO could conceivably model the paired variants of the VRPPD, in particular the Dial-a-Ride Problem (DARP). MATSim notably has modeled the Dial-a-Ride Problem (DARP), as described in Maciejewski and Nagel [29], though this required extending MATSim and so is not a native capability of the simulator. However, when considering a scenario which requires the movement of inanimate goods rather than people, RinSim is most likely the best choice.

5.1.3 Generative traffic

The generative traffic scenario studied if each simulator would support traffic flows varying as a function of time, so as to simulate rush hours, for example. The scenario also investigated whether or not each simulator had a concept of a "day" and the cycles in operation that would entail. All three simulators had explicitly defined traffic, which thereby supports the first part of the scenario. The way they implemented this varied a great deal, however.

MATSim, as we have mentioned previously, does not explicitly define *traffic*. Rather, traffic is what happens when the agents, whose movements *are* explicitly defined, need to move. In a scenario where each agent has their own car, the definition of agent movement and traffic flow is synonymous. If a scenario incorporates public transport or multi-passenger vehicles, however, this is no longer the case. MATSim's model provides an organic simulation of the movement of people, but it may be

difficult to define a certain vehicle density, at a certain location, at a certain time. On the other hand, rush hour traffic may be emergent behavior from the agents' movement, removing the need for explicitly defining it.

SUMO, in contrast, does define traffic explicitly. Each vehicle has a given time and location when it will enter the simulation and most vehicles also have a predetermined path it will follow. As we showed in section 4.4.4, setting a specific traffic density along a specific path is easy.

RinSim does have core support for generative traffic but, as expected, requires the scenario to be specially developed in order to make use of this. Since RinSim does not have separate configuration, there is no ready-made parser for XML routing files, as in the other two simulators. Instead, the example scenario used in section 4.3.4 used a purpose-built file reader for an unfamiliar format.

The second part of the scenario — support for a "day" — was not well supported. Both RinSim and SUMO failed this part, neither having a concept of time beyond internal simulator time. SUMO counts internal time in seconds even if the simulation was run at time steps longer or shorter than a second, which offers some internal consistency. The same cannot be said for RinSim which only tracks simulator "ticks," without any relation to real or simulated time.

MATSim stands out in this scenario since it is explicitly built around the idea of a daily cycle. Given its focus on agents representing people, this is a sensible model. In the basic MATSim model, agents go through their day, following their plan, then score and evaluate the result at the end of the day. With this evaluation, they construct a new plan for the next day, and the cycle repeats.

Like the above two scenarios, it is not surprising that the best simulator in our findings is also the simulator that was used in the study which inspired it. Guggisberg Bicudo and Berkenbrock [26] used MATSim in a small example study to see if they could replicate the complex traffic patterns of Joinville, Brazil.

5.1.4 Use cases

Let us imagine a new scenario. You, the reader, have been tasked to lead the development of a new, electric AV-HT, from the first line on the drawing board to its maiden voyage. You are — hopefully — convinced that part of this project will require simulation. But which simulator will you use? Is there a *best* simulator?

The short answer is no. As we have discussed above, our three simulators are all good at different things, and none of them is generally better than any other. The skilled engineer will choose the right tool for each task and these simulators are just that: tools.

Above, we studied how each simulator fared in each of our evaluation scenarios. Let us now take each simulator and construct a hypothetical scenario where it would

excel.

MATSim Simulating the microscopic movement of individuals is where MATSim excels. If we are planning new bus routes, laying new tracks for light rail, or exploring the impact of introducing tolls on certain roads, MATSim is the recommended tool to use. The simulated agents have "homes" and somewhere to be during the day — be it work, school, or the supermarket. If we change the available methods of getting there, the agents will adapt and we can observe the emergent flow of people and traffic.

In context of AV-HT, moving goods instead of people, MATSim may not seem like a good choice. However, MATSim's agent-based computation could be used to model highly detailed freight networks, as discussed in Schröder et al. [32]. The goods themselves may not be simulated, but this approach could be used to test large-scale planning and scheduling strategies.

RinSim Imagine a scenario where you need to optimize a system in which electric AVs pick up goods at a single producer and deliver them to a single consumer. At either end there are limited numbers of loading and unloading bays and even fewer charging stations. We are not concerned with the *actual* power drain of these vehicles, but we know they need to charge every so often. The problem is: how do we schedule vehicles for pickups, deliveries, and charging to optimize throughput. RinSim would be well suited to simulate this scenario and may even optimize it as well.

If there are goods involved and vehicle kinematics are of lesser importance, RinSim will be the best choice. The same goes for multi-agent reasoning problems, where AVs have to negotiate and agree on something. RinSim has strong support for modeling complex goods and communications systems, and lends itself well to GPDP-type problems.

There are few real limitations to what one can do in RinSim. In our study, we have investigated what can be done with the core simulator, but in a real use case, the user has the power of the whole Java language at their disposal. With RinSim, there is no difference between a user and a developer.

SUMO The control systems for our imaginary AV-HT project are coming along and it is time to start testing them. Unit tests have gotten them so far, but they need to be tried in action. Unfortunately, our AV-HT is not allowed to be on the road yet, let alone near human drivers, so a live test is out of the question. Still, we need to look at the interaction between the AV-HT and its control center as it encounters different traffic conditions. In this scenario, SUMO could be used together with TraCI to connect a simulated AV-HT to the real world control systems. Then we would be free to throw whatever we wanted at the simulated vehicle to see how it would fare.

If realism is important, then SUMO is the best place to start. It is the only one of our three simulators where vehicles know to obey traffic laws, yielding, and staying more-or-less within the speed limit. We can also use it to model three-dimensional terrain and calculate fuel consumption metrics for the vehicles driving over it. We could model pollution — both noise and air — or connect it to any other application via TraCI, allowing the other application to make up for where SUMO falls short. There is, in theory, nothing stopping you from connecting SUMO to RinSim, creating a simulator system which fulfills all eleven criteria and supports all three scenarios we have studied.

5.1.5 Combining vs. extending

Given all of the above, we should still ask ourselves if it is easiest to extend one simulator or use a combination of simulators. There are instances where it may be relevant to use two simulators at the same time, with one informing the other. In another case they could be used at different stages in the process, depending on what each simulator does best. As an example, one could use SUMO to design an AV-HT and its control logic, accurately simulating its behavior on the road. Once that is done, one could use RinSim to lay out a model of the transport network, simulating different placements of loading bays and varying numbers of vehicles to find the optimal configuration.

The obvious downside to combining simulators is that the user must be familiar with more than one simulator. Alternatively, more than one person must be involved in the simulation and the simulation must move between them. However, this is not likely to be a problem in large corporations. Barring all extensions, the simulation is also dependent on the already implemented models in each simulator which may be flawed or insufficient for a given problem. On the other hand, combining different simulators allows a simulation work flow to be put in place quickly.

On the other hand, extending a simulator requires a great deal of time and effort. In exchange, it affords the developer absolute control of the simulation and eliminates any errors caused by translating a single scenario between multiple simulators. Bespoke extensions will need to be extensively verified, however, since they will not have been tested and validated by the simulator's community of users.

RinSim stands out as the best candidate for extension rather than combination, simply because it *must* be extended to be used. However, it has its niche, and for major modifications it may be easier to try to use it alongside another simulator. MATSim is intended to be modular, so will probably lend itself to both approaches rather well. Finally, SUMO is likely best suited to working alongside some other process — simulator or otherwise — via its external control capabilities. The easiest way to add capabilities to SUMO is by extending its external controller, rather than SUMO itself.

5.2 SimMobility — the fourth simulator

SimMobility [54] was originally intended to be the fourth simulator in this study. Unfortunately, a series of problems prevented it from being included. Below, we will briefly introduce it and discuss what made it fall short.

SimMobility is the spiritual successor to MITSimLab [55] and DynaMIT [56], all three developments of MIT's Intelligent Transportation Systems Lab. It is intended to simulate traffic and the movement of people at a massive scope — from fraction-of-a-second microscopic traffic simulations to time steps of months or years, simulating the change in mobility trends and land use as people change homes or jobs [11]. In the scope of our project, the microscopic model was deemed relevant to study.

At first glance, acquiring SimMobility would be easy, as the source code is hosted on GitHub. Unfortunately, acquiring the code was the *only* easy step. SimMobility is only available as C++ source code, requiring a user to compile it themselves. The documentation describing how to do so was lacking, leaving a number of details to guesswork or trial and error.

Once compiled came the problem of running SimMobility. In difference from all the simulators we did include in our study, SimMobility did not come with any sample scenarios, nor information about how to create a scenario from scratch. The documentation referred to a "prototypical city" which could be used for testing SimMobility, but this data was missing. It later became clear that one had to request access to this data from the developers, a process which may require signing a non-disclosure agreement.

As mentioned in chapter 1, a criteria for selection of a simulator is that it is freely available. While SimMobility's source code *is* available, there is no method — short of reverse-engineering the source — to use it, lacking both example data and documentation. The source code itself is available for anyone to download, but it is by no means FOSS. Yoni Rabkin, a licensing volunteer for the Free Software Foundation, assisted in reviewing SimMobility's license and found that it places restrictions on redistribution which are incompatible with free software [57]. Specifically, it prohibits a person for charging a fee for redistribution. We also found that the license infringes on the Derived Works clause of the OSI Open Source Definition [24, no. 3], as redistribution *must* be done through SimMobility's public GitHub repository. A derivative work may be used only within an organization, but such use still requires the source code, with documentation of changes made, to be submitted to the GitHub repository first. Thus, it is not suitable for commercial use, where edits may be confidential.

In summary, SimMobility was excluded from our study due to it lacking crucial information in order to run it and due to being neither free nor open source software.

5.3 Ethics and sustainability

At a cursory glance, it could appear that everything is well and good, ethically and in terms of sustainability, with the emergence of autonomous vehicles. Studies such as Fagnant and Kockelman [18] and Litman [19] show that AVs can allow traffic to flow more smoothly, with lower emissions, while also granting greater mobility to persons who may otherwise have trouble getting around. AVs also drive better than humans which will lower the rate of traffic accidents. All of this is likely true, but it is not the whole picture.

There is a famous ethics problem, known as the Trolley Problem, which is usually brought up when discussing autonomous vehicles. In brief, the problem states that you see a runaway trolley speeding towards a track where five people are in the way. Next to you is a switch lever, allowing you to divert the trolley to a side track where there is only one person in the way. What do you do?

Referring back to the AVs, what would it do? What *should* it do? In this scenario, it is the trolley, and must make a decision which may very well kill a human. Should it prioritize saving its passengers, even if that means injuring bystanders? Bonnefon, Shariff, and Rahwan found that their study participants would prefer utilitarian AVs — that is, AVs that would sacrifice its passengers for the greater good — so long as the respondent was not a passenger [58].

Answering these questions is a topic for another thesis, but it is nevertheless important to mention them as we discuss allowing computers to make ethical, potentially lethal, decisions. We should also ask who we blame when — not if, when — an AV injures someone. Is it the fault of the AV's owner or of its manufacturer? Do we treat the AV as infallible and blame the injured party? At this time, there is no way to say, although the last version seems unlikely.

Leaving the tangled ethics of the trolley problem and personal injury, let us consider electric AVs. As mentioned by Fagnant and Kockelman [18], AVs drive more efficiently than human drivers, causing lower emissions than a human would, even if the AV drives further. With that, an electrical AV should have even less of an impact on the environment. Again, this is true, but it is not without drawbacks. Electric vehicles themselves have no emissions, but the electricity is generated somewhere. If the power source is not clean, it becomes harder to justify the increased load from electric vehicles.

Automation of heavy transport is a promising avenue for this emerging technology, but there are problems to account for here as well. As noted in chapter 1, the transportation sector employs a large number of people. Increasing automation in this sector could oust these people from their jobs, increasing unemployment. A Pew Research study [59] shows that 81 % of respondents believe that *"many people who drive for a living would lose their jobs"* as a result of AVs becoming more widespread. In the same study, 65 % of respondents said they would feel unsafe sharing the road with an autonomous freight vehicle.

Finally, let us turn to simulation. There are many advantages of traffic simulation as compared to live road tests. Simulation is cheaper, faster, and safer, just to name a few. Increasing traffic simulation in vehicle development could help the vehicles reach the road faster and be safer on the road than if development had relied solely on live road tests. However, over-reliance on simulation brings its own set of problems. Careless engineering could build vehicles or control systems that work fine in the simplified, simulated world but are incapable of handling the complex and random nature of the real world. Simulation as a whole, much like the individual simulators discussed above, is not a silver bullet — it cannot solve all problems. As the saying goes: take everything — including simulation — in moderation.

6

Conclusion

In this thesis, we have studied three different simulators for traffic and logistics networks — Multi-Agent Transport Simulation (MATSim), RinSim, and Simulation of Urban MObility (SUMO). This study has entailed reviewing literature related to each simulator, determining its fulfillment of eleven criteria, and testing its support for three scenarios. Through this analysis, we have found that no simulator stands out as clearly "better" than the rest — rather, they are all better than the others for certain classes of problems. MATSim outperformed the others in problems relating to population simulation and intelligent route planning. RinSim excelled in problems related to the movement of goods and simulation of inter-agent communication. Finally, SUMO stood out in scenarios requiring realistic simulation of traffic flows and vehicle behavior.

6.1 Further study

Further study could take one of two routes. A study may choose a single simulator from our three to perform an in-depth study of its capabilities or extend it to handle some class of problems it currently cannot. The other route is to take a broader scope, looking at a greater number of simulators than have been covered in this work. This route would be particularly interesting if such a study compared the three FOSS simulators in this work to commercial simulators.

Bibliography

[1] Transport sector economic analysis - European Commission. Accessed: 2018-11-20. [Online]. Available: https://ec.europa.eu/jrc/en/research-topic/transport-sector-economic-analysis

[2] IPCC, "Summary for Policymakers," in *Climate Change 2014: Mitigation of Climate Change. Contribution of Working Group III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, O. Edenhofer *et al.*, Eds. Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press, 2014.

[3] R. Sims *et al.*, "Transport," in *Climate Change 2014: Mitigation of Climate Change. Contribution of Working Group III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, O. Edenhofer *et al.*, Eds. Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press, 2014.

[4] (2016, Sep.) Volvo first in the world with self-driving truck in underground mine. AB Volvo. Accessed: 2018-11-18. [Online]. Available: https://www.volvogroup.com/en-en/news/2016/sep/news-2297091.html

[5] (2018, Sep.) Ground-breaking innovations for future autonomous and electric transport solutions. AB Volvo. Accessed: 2018-09-17. [Online]. Available: https://www.volvogroup.com/en-en/news/2018/sep/news-3048902.html

[6] MATSim.org. Accessed: 2018-08-24. [Online]. Available: https://matsim.org/

[7] RinSim. Accessed: 2018-10-23. [Online]. Available: https://rinsim.rinde.nl/

[8] SUMO - Simulation of Urban Mobility. Accessed: 2018-08-18. [Online]. Available: http://sumo.dlr.de/index.html

[9] S. Krauß, "Microscopic modelling of traffic flow: Investigation of collision free vehicle dynamics," Ph.D. dissertation, University of Köln, Germany, 1998.

[10] M. Behrisch, L. Bieker, J. Erdmann, M. Knocke, D. Krajzewicz, and P. Wagner, "Evolution of SUMO's Simulation Model," in *Traffic and Transportation Simulation – Looking Back and Looking Ahead: Celebrating 50 Years of Traffic Flow Theory, A Workshop*, ser. Transportation Research Circular, C. Antoniou, G. List, and R. L. Bertini, Eds., no. E-C195. Washington D.C.: Transportation Research Board of the National Academies, Apr. 2015, pp. 64–83.

[11] M. Adnan, F. C. Pereira, C. M. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. Zegras, and M. Ben-Akiva, "SimMobility: A multi-scale integrated agent-based simulation platform," in *95th Annual Meeting of the Transportation Research Board*, 2016.

[12] F. Vaudrin, J. Erdmann, and L. Capus, "Impact of autonomous vehicles in an urban environment controlled by static traffic lights system," in *Proceedings of the SUMO 2017: Towards Simulation for Autonomous Mobility*, ser. Reports of the DLR-Institute of Transportation Systems, K. Lemmer, Ed., vol. 31. Berlin, Adlershof: German Aerospace Center, May 2017, pp. 81–92.

[13] I. Y. Hernández-Paniagua, J. C. Zavala-Reyes, P. López-Ramírez, U. Diego-Ayala, I. Rosas, and A. Jazcilevich, "Use of SUMO to assess human exposure and intake to $PM_{2.5}$ near motorways: Preliminary Results," in *Proceedings of the SUMO 2017: Towards Simulation for Autonomous Mobility*, ser. Reports of the DLR-Institute of Transportation Systems, K. Lemmer, Ed., vol. 31. Berlin, Adlershof: German Aerospace Center, May 2017, pp. 63–70.

[14] R. Basso, B. Kulcsár, B. Egardt, P. Lindroth, and I. Sanchez-Diaz, "Reliable energy consumption estimation for the Electric Vehicle Routing Problem," 2018, manuscript submitted for publication.

[15] M. Gallet, T. Massier, and T. Hamacher, "Estimation of the energy demand of electric buses based on real-world data for large-scale public transport networks," *Applied Energy*, vol. 230, pp. 344–356, 2018.

[16] P. Gora, "Simulation-based traffic management system for connected and autonomous vehicles," in *Road Vehicle Automation 4*, G. Meyer and S. Beiker, Eds. Cham: Springer International Publishing, 2018, pp. 257–266.

[17] S. Gong, J. Shen, and L. Du, "Constrained optimization and distributed computation based car following control of a connected and autonomous vehicle platoon," *Transportation Research Part B: Methodological*, vol. 94, pp. 314–334, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261516303836

[18] D. J. Fagnant and K. M. Kockelman, "The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 1–13, 2014.

[19] T. Litman, *Autonomous vehicle implementation predictions.* Victoria, Canada: Victoria Transport Policy Institute, 2017.

[20] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[21] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems. Part I: Transportation between customers and depot," *Journal für Betriebswirtschaft*, vol. 58, no. 1, pp. 21–51, 2008.

[22] ——, "A survey on pickup and delivery problems. Part II: Transportation be-

tween pickup and delivery locations," *Journal für Betriebswirtschaft*, vol. 58, no. 2, pp. 81–117, 2008.

[23] What is free software? - GNU Project - Free Software Foundation. Accessed: 2018-12-21. [Online]. Available: https://www.gnu.org/philosophy/free-sw.html

[24] The Open Source Definition (Annotated) | Open Source Initiative. Accessed: 2018-12-21. [Online]. Available: https://opensource.org/osd-annotated

[25] S. Karlsson and J. Steffenburg, "Self-organizing multi-agent systems for shared space applications: Using genetic algorithms and contract net protocols to solve the pickup and delivery problem," Master's thesis, Department of Mechanics and Maritime Sciences, Division of Vehicle Engineering and Autonomous Systems, Applied Artificial Intelligence Research Group, Chalmers University of Technology, SE-412 96 Gothenburg, 2018.

[26] D. Guggisberg Bicudo and G. R. Berkenbrock, "Joinville," in *The Multi-Agent Transport Simulation MATSim*, A. Horni, K. Nagel, and K. W. Axhausen, Eds. London: Ubiquity Press, 2016, ch. 72, pp. 445–446, License: CC-BY 4.0. [Online]. Available: http://dx.doi.org/10.5334/baw

[27] K. W. Axhausen, M. Balmer, K. Meister, M. Rieser, and K. Nagel, "Agent-based simulation of travel demand: Structure and computational performance of MATSim-T," in *Proceedings of the 2nd TRB Conference on Innovations in Travel Modeling*, ser. Arbeitsbericht Verkehrs-und Raumplanung, vol. 504. IVT ETH Zurich, 2008.

[28] M. Maciejewski and K. Nagel, "Towards multi-agent simulation of the dynamic vehicle routing problem in MATSim," in *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, Eds. Berlin, Heidelberg: Springer, 2011, pp. 551–560.

[29] ——, "Simulation and dynamic optimization of taxi services in MATSim," *VSP Working Paper, TU Berlin, Transport Systems Planning and Transport Telematics*, no. 13-05, 2013.

[30] T. Novosel, L. Perković, M. Ban, H. Keko, T. Pukšec, G. Krajačić, and N. Duić, "Agent based modelling and energy planning – Utilization of MATSim for transport energy demand modelling," *Energy*, vol. 92, pp. 466–475, 2015.

[31] EnergyPLAN | Advanced energy systems analysis computer model. Accessed: 2018-12-12. [Online]. Available: https://www.energyplan.eu/

[32] S. Schröeder, M. Zilske, G. Liedtke, and K. Nagel, "A computational framework for a multi-agent simulation of freight transport activities," in *91st Annual Meeting of the Transportation Research Board*, no. 12-4152, 2012.

[33] Via Features. Accessed: 2018-12-13. [Online]. Available: https://www.simunto.com/via/

[34] D. Strippgen, "OTFVis: MATSim's Open-Source Visualizer," in *The*

Multi-Agent Transport Simulation MATSim, A. Horni, K. Nagel, and K. W. Axhausen, Eds. London: Ubiquity Press, 2016, ch. 34, pp. 225–234, License: CC-BY 4.0. [Online]. Available: http://dx.doi.org/10.5334/baw

[35] R. A. Waraich and J. Bischoff, "Electric vehicles," in *The Multi-Agent Transport Simulation MATSim*, A. Horni, K. Nagel, and K. W. Axhausen, Eds. London: Ubiquity Press, 2016, ch. 14, pp. 93–96, License: CC-BY 4.0. [Online]. Available: http://dx.doi.org/10.5334/baw

[36] B. Kickhöfer, "Emission modeling," in *The Multi-Agent Transport Simulation MATSim*, A. Horni, K. Nagel, and K. W. Axhausen, Eds. London: Ubiquity Press, 2016, ch. 36, pp. 247–252, License: CC-BY 4.0. [Online]. Available: http://dx.doi.org/10.5334/baw

[37] K. Nagel and G. Flötteröd, "Agent-based traffic assignment," in *The Multi-Agent Transport Simulation MATSim*, A. Horni, K. Nagel, and K. W. Axhausen, Eds. London: Ubiquity Press, 2016, ch. 47, pp. 315–326, License: CC-BY 4.0. [Online]. Available: http://dx.doi.org/10.5334/baw

[38] A. Horni, K. Nagel, and K. W. Axhausen, Eds., *The Multi-Agent Transport Simulation MATSim*. London: Ubiquity Press, 2016, License: CC-BY 4.0. [Online]. Available: http://dx.doi.org/10.5334/baw

[39] GitHub - matsim-org/matsim. Accessed: 2019-01-09. [Online]. Available: https://github.com/matsim-org/matsim

[40] M. Zilske, S. Schröder, K. Nagel, and G. Liedtke, "Adding freight traffic to MATSim," *VSP Working Paper, TU Berlin, Transport Systems Planning and Transport Telematics*, no. 12-02, 2012.

[41] RinSim v4.4.6 | Zenodo. Accessed: 2019-01-14. [Online]. Available: https://zenodo.org/record/1286654

[42] R. R. S. van Lon and T. Holvoet, "RinSim: A simulator for collective adaptive systems in transportation and logistics," in *Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth International Conference on*. IEEE, 2012, pp. 231–232.

[43] ——, "Evolved multi-agent systems and thorough evaluation are necessary for scalable logistics," in *Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS 2013)*. IEEE, 2013, pp. 16–19.

[44] R. R. S. van Lon, J. Branke, and T. Holvoet, "Optimizing agents with genetic programming: an evaluation of hyper-heuristics in dynamic real-time logistics," *Genetic programming and evolvable machines*, vol. 19, no. 1-2, pp. 93–120, 2018.

[45] GitHub - rinde/RinSim. Accessed: 2019-01-04. [Online]. Available: https://github.com/rinde/RinSim

[46] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO — Simulation

of Urban MObility: An overview," in *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, vol. 42, 2011.

[47] iTETRIS Platform. Accessed: 2019-01-14. [Online]. Available: http://www.ict-itetris.eu/

[48] E. Bjärkvik, F. Fürer, M. Pourabdollah, and B. Lindenberg, "Simulation and Characterisation of Traffic on Drive Me Route around Gothenburg using SUMO," in *Proceedings of the SUMO 2017: Towards Simulation for Autonomous Mobility*, ser. Reports of the DLR-Institute of Transportation Systems, K. Lemmer, Ed., vol. 31. Berlin, Adlershof: German Aerospace Center, May 2017, pp. 1–13.

[49] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*. ACM, 2008, pp. 155–163.

[50] D. Krajzewicz, "Summary on Publications citing SUMO, 2002–2012," in *Proceedings of the 1st SUMO User Conference 2013*, ser. Reports of the DLR-Institute of Transportation Systems, K. Lemmer, Ed., vol. 21. Berlin, Adlershof: German Aerospace Center, May 2013, pp. 11–24.

[51] K. Lemmer, Ed., *Proceedings of the SUMO 2017: Towards Simulation for Autonomous Mobility*, ser. Reports of the DLR-Institute of Transportation Systems, vol. 31. Berlin, Adlershof: German Aerospace Center, May 2017.

[52] GitHub - eclipse/SUMO. Accessed: 2019-01-06. [Online]. Available: https://github.com/eclipse/sumo

[53] L. Bieker, D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano, "Traffic Simulation for All: A Real World Traffic Scenario from the City of Bologna," in *Modeling Mobility with Open Data*, M. Behrisch and M. Weber, Eds. Cham: Springer International Publishing, 2015, pp. 47–60.

[54] SimMobility – Integrated Simulation Platform | INTELLIGENT TRANSPORTATION SYSTEMS LAB. Accessed: 2018-11-18. [Online]. Available: https://its.mit.edu/software/simmobility/

[55] MITSIMLab | INTELLIGENT TRANSPORTATION SYSTEMS LAB. Accessed: 2018-08-24. [Online]. Available: https://its.mit.edu/software/mitsimlab

[56] DynaMIT | INTELLIGENT TRANSPORTATION SYSTEMS LAB. Accessed: 2018-08-24. [Online]. Available: https://its.mit.edu/software/dynamit

[57] Y. Rabkin *et al.*, Private communication, 2018–2019.

[58] J.-F. Bonnefon, A. Shariff, and I. Rahwan, "The social dilemma of autonomous vehicles," *Science*, vol. 352, no. 6293, pp. 1573–1576, 2016.

[59] Pew Research Center, "Automation in Everyday Life," October 2017.