institute of telecommunications

MASTER THESIS

Simulating Vehicle-to-Vehicle Connectivity on Real-World Street Networks

Author:
Markus GASSER, BSc

Matriculation Number:
0925878

Supervisor:
Prof. Christoph MECKLENBRÄUKER

Co-Supervisor:
Dipl.-Ing. Thomas BLAZEK

A thesis submitted in fulfillment of the requirements for the degree of Diplomingenieur

at the

INSTITUTE OF TELECOMMUNICATIONS

at

TU WIEN

Vienna, September 21, 2017

Abstract

Vehicular ad hoc networks (VANETs) are a widely discussed topic in the scientific community of the recent past. Many publications focus on analysis and simulation of communication between vehicles on idealized street networks. These networks consist of streets forming a rectangular grid (Manhattan grid), where either the street placement occurs in a fixed and constant interval or is determined by a random process. Additionally, most scientific works use a maximum Euclidean distance to determine whether two vehicles are connected or not.

In this work we present a simulation framework that allows the simulation of connectivity in VANETs and the derivation of multiple static and dynamic network metrics from these simulations. In contrast to conventional idealized street networks the framework uses real-world street networks and building data. The street data is used to place and move vehicles around the streets and the building data is used to assign one of the propagation condition conditions line-of-sight (LOS), obstructed-line-of-sight (OLOS) or non-line-of-sight (NLOS) to each link connecting two vehicles. The status of a link can be either determined by a maximum Euclidean distance or a maximum pathloss, using models derived from vehicular measurements.

The framework allows the placement of vehicles statically and randomly with a uniform distribution, or by using external traffic simulators that routes and moves the vehicles in the street network. Results of the simulations include the biggest connected cluster size, path redundancies and link and connection durations.

Furthermore, we present and discuss simulation results for different scenarios and compare them to results of idealized street networks.

Kurzfassung

Fahrzeug-Ad-Hoc-Netzwerke basieren auf Technologien, die in frühester Vergangenheit verstärkt wissenschaftlich untersucht wurden. Oft beruhen diese Untersuchungen auf Simulationen mit idealisierten Straßennetzen. Diese Netze bestehen aus Straßen, die ein rechteckiges Raster (Manhattan Grid) bilden, wobei Straßenabstände entweder konstant sind oder von einem Zufallsprozess generiert werden. Des weiteren wird meist die Euklidische Distanz benutzt um den Status von Verbindungen zwischen Fahrzeugen zu bestimmen.

In dieser Arbeit präsentieren wir eine Software, welche die Simulation von Konnektivität in Fahrzeug-Ad-Hoc-Netzwerken ermöglicht. Im Gegensatz zum Verbreiteten Ansatz der Verwendung von idealisierten Straßennetzen, erlaubt dieser Simulator die Verwendung von realen Straßen- und Gebäudedaten. Die Straßendaten werden dazu benutzt um Fahrzeuge auf den Straßen zu platzieren, während die Gebäudegrundrisse dazu verwendet werden die Ausbreitungsbedingungen zwischen Fahrzeugen zu bestimmen. Diese Ausbreitungsbedingungen können unterteilt werden in Sichtverbindung (LOS), versperrte Sichtverbindung (OLOS) und keine Sichtverbindung (NLOS). Das Zustandekommen einer Verbindung kann aufgrund der Euklidischen Distanz oder eines Pfadverlustmodells bestimmt werden.

Fahrzeuge können entweder statisch und mit einer Gleichverteilung im Straßennetz platziert werden, oder durch die Benutzung einer externen Software auf den Straßen bewegt werden. Die Resultate aus diesen Simulationen umfassen die Bestimmung des größten verbundenen Gruppe an Fahrzeugen, die Pfadredundanzen, sowie die physikalischen und logischen Verbindungsdauern.

Wir präsentieren weiteres Simulationsergebnisse für verschiedene Szenarios und vergleichen diese mit Ergebnissen von idealisierten Straßennetzen.

Acknowledgements

Thank you Christoph and Thomas for making this thesis possible and sharing your knowledge with me.

I express my deepest gratitude to Richard, Barbara, Oliver and Juliane for supporting and enduring me during all these years.

At long last, thank you Doris – for everything.

Contents

List of Figures

List of Tables

Acronyms

ADAS Advanced Driver-Assistance Systems

AV Autonomous Vehicle

BFS Breadth-First Search

BSS Basic Service Set

CA Cellular Automaton

CDF Cumulative Distribution Function

ECDF Empirical Cumulative Distribution Function

GNU GNU's Not Unix

ITS Intelligent Transport Systems

ITS-G5 Intelligent Transport Systems at 5 GHz

LOS Line-Of-Sight

MAC Medium Access Control

NLOS Non-Line-Of-Sight

OFDM Orthogonal Frequency-Division Multiplexing

OLOS Obstructed-Line-Of-Sight

OSM OpenStreetMap

PDP Power Delay Profile

pmf Probability Mass Function

SUMO Simulation of Urban MObility

TLS Traffic Light System

UTM Universal Transverse Mercator

V2I Vehicle-To-Infrastructure

V2V Vehicle-To-Vehicle

V2X Vehicle-To-Everything

VANET Vehicular Ad hoc NETwork

WAVE Wireless Access in Vehicular Environments

WSS Wide-Sense Stationary

XML eXtensible Markup Language

Chapter 1

Introduction

This chapter serves as an introduction to the thesis. It gives a brief overview of technologies using vehicular communication from a technical and societal point of view. The relevance of the work is explained and its existence motivated. Furthermore, this chapter gives an overview on the state of the art in relevant research areas by presenting key publications we base our work on. It is concluded by explaining the structure of the rest of this text.

1.1 Overview and Motivation

The focus of development in the automotive industry on assisted and autonomous driving in recent years indicate that these systems will have a significant impact on the road traffic of the future [1]. They will consequently play a big role in the lives of people driving on and walking these roads.

Advanced driver-assistance systems (ADAS) are the first step in this innovation process. Using different technologies they have the goal to improve car and road safety and simplify the driving process. They do so by assisting the driver in its human-machine interaction. The applications of ADAS range from adaptive cruise control, where the speed of a car is automatically adjusted to keep a safe distance to the preceding vehicle, to traffic sign recognition, where traffic signs along the road are detected by the vehicle and displayed to the driver on the dashboard of the car. Technologies that are used to realize these applications include radar, image processing and computer vision. Upcoming implementations of ADAS will also leverage communication between cars themselves, termed vehicle-to-vehicle (V2V) communication and between cars and the infrastructure along the road, termed vehicle-to-infrastructure (V2I) communication. Both types can be summarized into vehicle-to-everything (V2X) communication.

Autonomous vehicles (AVs) go one step further than ADAS by completely eliminating the need for interaction between the driver and the vehicle during the driving process. As in the assisted case this goal will be achieved by using on-vehicle sensors and processing as well as V2X communication.

Studies predict that the trends of ADAS and AVs represent one way how road safety will be increased. A report comes to the conclusion that widespread deployment of AVs could eliminate as much as 90 percent of all car accidents in the USA [2]. Another study concludes that ADAS could help to avoid up to 79 percent of all traffic accidents [3]. Statistics show that road traffic related deaths accounted for an estimate of 2.52 percent of the total deaths worldwide in the year 2010[1]. This has

[1]An estimated 1 328 500 road traffic related deaths out of 52 769 700 total deaths worldwide occurred during 2010 [4].

been an increase from an estimated 1.95 percent in 1990[2]. While the absolute number of road accident fatalities decreased constantly between 2005 and 2014 in the European Union, they are still a prevalent cause of death [5]. These numbers and trends show that much needs to be done to increase safety on the road and in turn decrease traffic related deaths.

Increased road safety might be the most important potential consequence of widespread deployment of AVs, but definitely not the only one. By automating the driving process, acceleration and breaking can be optimized and therefore become more fuel efficient. By additionally removing human errors, congestions will be reduced, which leads to less fuel usage in addition to time savings for people traveling on the road. The increase in fuel efficiency will decrease the travel expenses and reduce emissions, which otherwise burden the environment. Combined with the advancing deployment of electric cars, the result could be a reduction of up to 90 percent of greenhouse-gas emissions per distance traveled by 2030 [6].

While people could spend less time in their vehicles due to less congestion, they would be able to use the remaining time in their vehicles on other activities, such as work, leisure or even to sleep. It has been predicted that AVs will free one billion hours from the driving process every day globally [2].

Being a core part of ADAS and AVs, V2X communication technologies experienced a rapid development in the recent past. The communication can be realized by vehicular ad hoc networks (VANETs), the decentralized creation of a wireless networks for data exchange by moving vehicles. The foundation of all major VANET implementations is IEEE 802.11p [7], an amendment to the popular IEEE 802.11 family of standards, commonly referred to as WiFi. For the higher layers there are competing standards, the two most important being defined by IEEE and ETSI. IEEE proposed the 1609.x standards, with the full stack including 802.11p being named Wireless Access in Vehicular Environments (WAVE). ETSI developed their own standard named Intelligent Transport Systems at 5 GHz (ITS-G5).

IEEE 802.11p, the fundamental standard of vehicular communication, defines the physical and part of the medium access control (MAC) layer. The physical layer still employs orthogonal frequency-division multiplexing (OFDM) as in 802.11a [8], but instead of using channels of 20 MHz bandwidth, the bandwidth has been halved to 10 MHz. This change can be implemented in a straightforward manner, by doubling the OFDM timing parameters and it results in half the data rate. The second major change on the physical layer is the requirement of improved receiver performance in adjacent channel rejection. These adaptations have the goal to allow communication within a longer range and support the vehicular channel, which consist of high speeds and large multipath components. On the MAC layer standard 802.11 defines a basic service set (BSS) as a group of devices forming a network. 802.11p allows the usage of special vehicular BSSs and a wildcard BSS where the process of joining the network is eliminated. This allows the instantaneous data exchange that is needed for vehicular safety applications.

Before V2X communication technologies will see widespread deployment in mass products they need to be examined and evaluated first. While hardware and testbed developments are time-consuming and expensive, the technologies are constantly evolving and legal frameworks are still being developed. This makes software simulations attractive as a first iteration in investigating and developing V2X technologies.

[2]An estimated 907 900 road traffic related deaths out of 46 511 200 total deaths worldwide occurred during 1990 [4].

In this thesis we introduce a simulation framework for vehicular connectivity that could help to answer questions regarding V2V communication. The main characteristic of this framework is to simulate connectivity on real world street networks. It enables the gathering of insights into the differences concerning the connectivity on idealized street networks used in publications such as [9] and street networks from real world cities. Additionally, it allows us to compare connection metrics for different places around the world.

By developing and subsequently using this framework we try to answer the following crucial questions:

– What density of vehicles equipped with V2V technology is needed, so that the networks is fully connected, i.e. messages can propagate through the whole network?

– For how long does the average link in a network of moving vehicles last?

– Is there a connection between the distance of a vehicle from the nearest intersection and the number of other vehicles it is connected to?

1.2 State of the Art

Network connectivity in VANETs have been studied extensively in the scientific community in the recent past. In this section we will briefly present relevant publications of the subject, mention their insights, shortcomings and context in regard to this thesis.

1.2.1 Analyzing Connectivity in Vehicular Networks

In [9] the authors present a comprehensive study on the connectivity of VANETs in urban environments. The work is based on the assumption of a Cartesian grid of streets, that consists of evenly spaced horizontal and vertical two-lane, bi-directional streets. This street network type is also known as Manhattan grid. The vehicle movement is based on a cellular automaton (CA) approach [10] where during each time step each vehicle is in a state and updates its state according to a set of rules. The connection of any two vehicles on the street network is determined by their Euclidean distance, comparing it to a predefined maximum. This maximum depends on the propagation condition of the link and can be either line-of-sight (LOS) or non-line-of-sight (NLOS).

The authors present various simulation results, including static and dynamic connectivity metrics. The static characteristics of the networks are captured by the average network connectivity. For results regarding additional static metrics such as path redundancy, the authors refer to [10], [11] by the same research group. Dynamic characteristics include the average link and connection duration and the re-healing time.

As a further contribution the publication derives closed form expressions for the average link duration in different scenarios for vehicles moving on the Manhattan grid, e.g. the two cars under investigation are approaching, the two cars are traveling on perpendicular streets, etc. Additionally an expression for the conditional distribution of the re-healing time is presented. These analytical results are then validated by simulation results.

This publication is a valuable resource presenting analytical and simulation results that will be used in this thesis. However, the use of the Manhattan grid as street network and the use of Euclidean distance to determine connections are a highly simplified model. The use of real world street networks and other metrics to decide the state of connections would constitute a more realistic model therefore allow further insights.

Another study on network connectivity is presented in [12]. It classifies different traffic mobility models by trying to assess their realism and comparing the network connectivity metrics resulting from their usage. In their conclusion they emphasize the importance of realistic models for mobility due to the differences in the simulation results. However, they base their simulations and analysis on an idealized street network consisting of a 3 x 3 block grid. Additionally, no buildings are considered, resulting in a very simplistic modeling of the transmission range. Furthermore, only a very high density network is simulated (20 vehicles per km and lane) which does not allow the investigation of weakly connected or disconnected networks.

1.2.2 Real-World Street Networks

Many publications (e.g. [9]–[12]) use strictly regular street networks, such as the Manhattan grid. As an alternative street network data from real cities provided by various data sources can be used. This is one method to make models more realistic and to allow the comparison of results for different environments. One data source, that collects accurate and exhaustive data and makes them available publicly, is OpenStreetMap (OSM) [13]. Most publications that use real-world street networks that have been investigated rely on OSM as their data source [14]–[17].

The suitability of OSM street and building data to model V2X channels has been investigated in [14]. The authors compare building data offered by OSM with official data from the municipality and come to the conclusion that OSM offers sufficient accuracy to model vehicular communication channels. It is mentioned that height information is mostly missing from building data, but this can be resolved by using the mean height of buildings in the area. Furthermore, the authors observe that the coverage and accuracy of data in OSM varies depending on the geographical region, but is expected to improve drastically with time.

In [15] a simulator for V2V channel modeling is presented. It uses street and building data from OSM to separate each link into LOS, obstructed-line-of-sight (OLOS) and NLOS types and calculate its pathloss and shadow fading based on it. Additionally, it models small-scale fading in a stochastic manner, based on surrounding objects. As results the simulator delivers the received power and small scale fading characteristics of each link. Data-sets of buildings, street networks and vehicle traces for four European cities are already included in the simulator. However, it does not provide a trivial method to import additional data from OSM. Building data has to be manually downloaded from the OSM website, while street data can not be imported from OSM at all. The project is therefore clearly limited in the variety of different environments it can simulate and analyze. Additionally, its result offers channel characteristics, which would need to be further processed to gather insights about the connectivity in VANETs.

Different other publications base their research on the simulator presented in [15]. In [17] the author study performance metrics, such as received signal strength and packet delivery ratio, while no network connectivity results are presented. Furthermore, only one geographical area, (Doha, Qatar) and one vehicle density have been investigated. In [16] a closed form expression for the probability of a connection being in LOS is derived, using the distance between the two vehicles as input and its parameters depending on the geographical area it has been fitted to. This probabilistic approach could be useful in reducing the computational complexity of larger simulations, e.g. when analyzing network connectivity on large areas with a high density of vehicles.

To efficiently extract street and building data from OSM, [18] introduced OSMnx, a software library, that can be used directly or by integrating it into other projects. OSMnx is written in the Python programming language and downloads street and building data from an area, by supplying just its name. It then converts the street network data into a graph object, retaining all the geometrical information. The building data is processed and saved as a list of polygons. This allows a flexible and efficient subsequent use. These properties make OSMnx a viable tool to retrieve real-world street networks and building data in VANET simulations.

1.2.3 Pathloss Models for Vehicular Channels

To analyze network connectivity in VANETs one has to determine if any two vehicles can connect to each other, as a first step. This is typically achieved by checking if the link between vehicles is in LOS or NLOS. Subsequently, the Euclidean distance between the vehicles is compared to a maximum distance of the corresponding propagation condition and the vehicles are determined to be connected if their distance is lower than the maximum. This is the approach applied in [9]–[12].

As an alternative to using the Euclidean distance between two vehicles to determine if they are connected, the pathloss could also be used. By employing appropriate models for each propagation condition, the pathloss between two vehicles can be calculated and the connection be set if the pathloss is below a maximum. This approach has the advantage of condensing the different maximum distances for each propagation condition in the Euclidean case into one maximum pathloss value. Additionally, using the pathloss, the effect of shadow fading can also be modeled. This alternative method raises the questions, which pathloss models are suitable for the environment that has to be simulated.

In [19] the authors present a pathloss and shadow fading model for V2V channels. It is applicable in the case that the line of sight connection is not obstructed (LOS) and in the case that it is obstructed by other vehicles (OLOS). The model is based on extensive measurements using a channel sounder between two vehicles, moving in urban and highway environments. The measured channel transfer functions were transformed to distance dependent pathlosses. The authors argue that the measured data can be best represented by a dual-slope model for the path loss and by a log-normal distribution for the shadow fading. By separating the highway and urban environment measurements separate pathloss exponents and shadow fading standard deviations can be provided for each scenario. This results in a pathloss model equation where the deterministic part of the result is only dependent on the distance between the two vehicles.

A pathloss and shadow fading model for V2V communication that covers the NLOS case is introduced in [20]. It is applicable if the line of sight is blocked by buildings and receiver and transmitter are moving on intersecting streets. The model is based on measurements in urban and suburban areas of Munich, Germany. These measurements have been conducted using off-the-shelf hardware, measuring the received power. By fitting the measured data to a modified virtual source model [21], an equation for the pathloss that is dependent on the distance between transmitter and intersection, distance between receiver and intersection, wavelength of the signal, the scenario (urban or suburban) and street widths is derived. As in [19], the shadow fading was best matched by a log-normal distribution.

1.2.4 Vehicle Movement

Various models are available to describe vehicular traffic flow. They can be roughly classified into three categories, namely microscopic, macroscopic and mesoscopic models. Microscopic models describe the dynamics of each vehicle individually. They update each vehicles speed by considering the speed and positions of other vehicles in the area. Macroscopic models on the other hand describe average quantities for a whole area, like density and average speed and disregard individual vehicles. The third class, mesoscopic models combine approaches from microscopic and macroscopic models.

Since we are interested in individual vehicle positions and movements to realistically simulate VANET connectivity, we will focus on microscopic traffic flow models and try to classify them further. The most trivial types are stochastic models. According to these models the street network is represented by a graph and vehicles move on the edges of this graph randomly. They choose random paths and travel with random speeds. A representative of this class is the Constant Speed Motion model [22], where each vehicle chooses a random speed, that is uniformly distributed between a minimum and a maximum speed. Further they select a random source and destination and route their path between them, using a shortest path algorithm. The Manhattan model [23] extends the Constant Speed Motion model by updating the vehicle speed according to a safety distance to the vehicle in front and therefore avoiding the overlap of vehicles.

The most prevalent type of microscopic traffic flow models are car-following models. The dynamics of each vehicle are described by them as a function of position and speed of the vehicle in front of it. They determine the change of speed of the respective vehicle. If the time domain and space domain are discretized a CA model is obtained, which can be solved in a computationally efficient manner. Additionally, as it is the most common model, there are many extensions to it. In [24] the model is extended to allow the modeling of multi-lane traffic, line changing and traffic jams.

[12] compares different models by investigating the connection between vehicle speed and density in an area, simulating perturbation, congestion and traffic at an intersection. Using the simulation results they try to assess the realism of the models. They conclude that the car-following model provides a faithful representation of real world dynamics, while the stochastic models fail all the tests.

To simulate vehicle movement using traffic flow models, different approaches are available. There are publications, such as the already mentioned [9], that develop and implement their own vehicle movement models, in addition to investigating the communication between them. Alternatively there are publications and software projects that focus mainly on vehicle movement and their results can be used as basis to simulate VANETs.

Simulation of Urban Mobility (SUMO) [25] is a open source traffic simulation suite that is widely used by academics. Road networks can be manually generated with it, but also imported from different sources such as OSM [13]. It offers microscopic traffic simulations by representing each vehicle explicitly, giving it an identifier, a source and a destination and a departure time. A path between source and destination for each vehicle is calculated using a routing algorithm, such as shortest path calculation under different cost functions. The movement of the cars is modeled by an extended car-following model [24]. SUMO offers the possibility to save all vehicle positions during each time step of the simulation, therefore enabling the user to subsequently analyze these vehicle traces or simulate other aspects on their basis. Using these traces in VANET simulations could yield more realistic results than a simple random process to distribute the vehicles or implementing a rudimentary car movement model.

Another microscopic traffic simulator has been introduced in [26]. It offers the same basic features of SUMO, including the possibility of importing real world street networks from OSM. Additionally, its focus lies on cooperate routing and intelligent transport systems (ITS). This would make it a great candidate to further investigate VANET connectivity, but the software has not been publicly released.

1.3 Structure of the Thesis

This document is divided into 4 chapters, with their content briefly outlined in this section.

Chapter 1 gives an overview of the whole thesis. It introduces the topic of V2V communication and explains why realistic simulations are needed. Additionally, it presents the current state of research in this area.

Chapter 2 is a brief introduction to the graph theoretic concepts that are used in the simulation process and when deriving network connectivity metrics from the simulation results.

Chapter 3 introduces the simulation framework that has been developed within the scope of this thesis. It gives an overview of the whole simulation process and explains the different process steps in detail. Furthermore, it offers background information about the models, algorithms and theoretic results used using the framework.

Chapter 4 gives a detailed explanation of the network connectivity metrics, that can be derived from the simulation results. Based on individual connections between vehicles, these metrics capture global properties of the VANET.

Chapter 5 presents simulations that have been executed by the framework. It lists the simulation parameters and shows simulation results. Finally, it draws conclusions from these simulation results and compares them to results from simulations on idealized street networks presented in scientific publications.

Chapter 6 is the final chapter of the document, it draws conclusions from the work presented herein and mentions further tasks that could be done based on the results of this thesis.

Chapter 2

Graph Theory Introduction

The simulation process and the subsequent connection analysis heavily relies on concepts from graph theory. Among other things, graphs are needed to route vehicles in a street network and represent connections between vehicles in the vehicular ad hoc network (VANET). We will therefore introduce the basic concepts, definitions and algorithms from graph theory, that are used in the simulation framework. This short introduction does by no mean claim completeness.

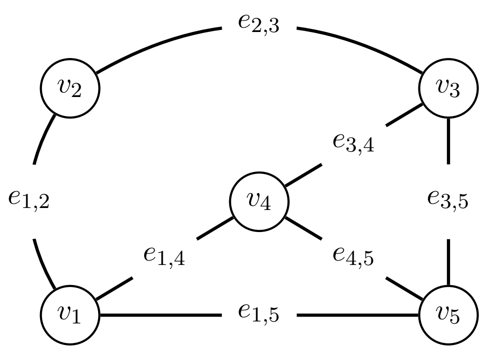
2.1 Types of Graphs

In the most common an basic definition a *graph* is a length-2 sequence $G = (V, E)$. The first element of this sequence is the set of *vertices* or *nodes* $V = \{v_1, \ldots, v_{|V|}\}$. The second element is the set of *edges* $E \subseteq \{e_{i,j} | i, j \in \{1, \ldots, |V|\}, i < j\}$. The elements of the edge set are 2-element subsets of the node set, i.e. $e_{i,j} = (v_i, v_j)$ with $v_i, v_j \in V$, as they reflect the association between nodes. The cardinality of the node set $|V|$ is termed the *order* of the graph, whereas the cardinality of the edge set $|E|$ is referred to as the *size* of the graph. To avoid ambiguity this type of graph can be characterized more specifically as *undirected*, *unweighted* and *simple* graph. Such a graph could describe a communication network, where the participants are represented by nodes and the bidirectional communication channels by edges.

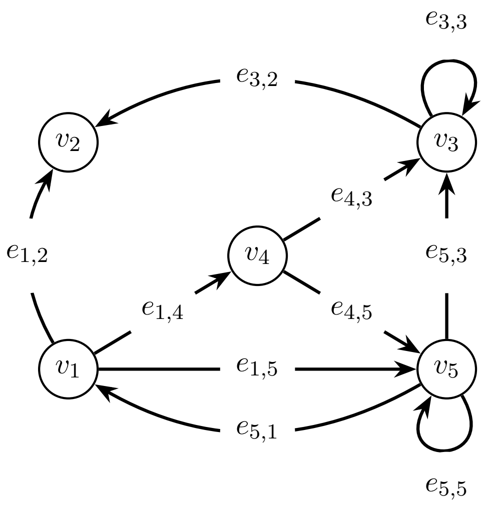
To introduce a direction to the association between nodes, we define a *directed graph* as a length-2 sequence $G = (V, E)$, where the edges are not sets but sequences $e_{i,j} = (v_i, v_j)$ and are therefore ordered. The set is described as being directed from v_i to v_j, v_i is termed the *tail* of the edge and v_j is termed the *head* of the edge. v_j is said to be a *successor* of v_i, while v_i is said to be a *predecessor* of v_j. The set of all edges E is a subset of the Cartesian square of the node set, i.e. $E \subseteq V \times V = \{(v_1, v_2) | v_i \in V \; \forall i = 1, 2\}$. Note that by this definition, directed graphs can include edges $v_{i,i}$ that connect nodes to themselves, termed *loops*. Directed graphs could describe communication networks where the channels between two participants are unidirectional, i.e. a participant in the network that can send to another one, can not necessarily receive from it.

A *multigraph*, in contrast to a simple graph, is a graph that contains *multiple edges*. Multiple edges are edges that connect the same two nodes. Therefore, E containing all edges, is a multiset.

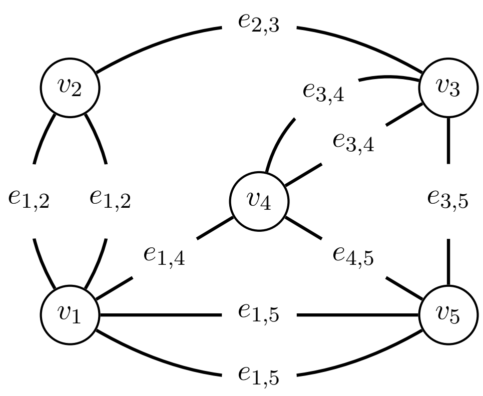
A graph that has a weight assigned to each edge, is termed a *weighted graph*. It can formally be defined as length-3 sequence $G = (V, E, w)$, with w being a function that maps a weight to each edge, as $w : E \to \mathbb{R}$. The concept of weighted graphs is especially important when path finding is involved. A weighted graph could describe a street network[1], where nodes represent intersections and edges represent streets. If the length of each street is chosen as the weight of the corresponding edge, a shortest path algorithm that considers weights, would find the path with the shortest length, between arbitrary nodes.



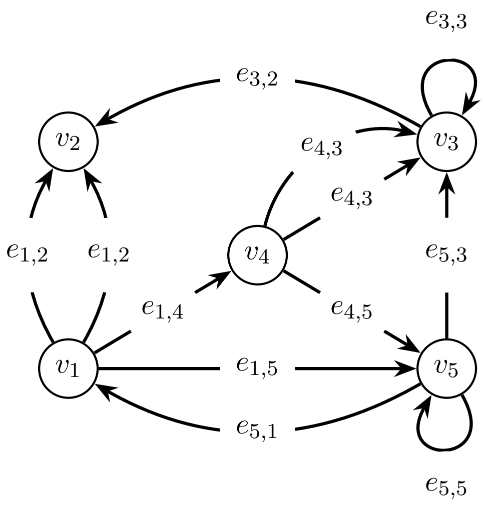
(A) Graph



(B) Directed graph



(C) Multigraph



(D) Directed multigraph

FIGURE 2.1: Examples of different graph types

Examples of the introduced graph types are depicted in Figure 2.1. Figure 2.1a shows an undirected, unweighted and simple graph. The nodes are represented by circles and the edges by lines connecting the circles. For the directed graphs Figures 2.1b and 2.1d the edges are represented by arrows, where the arrowhead points towards the head of the edge. In Figures 2.1c and 2.1d, multigraphs are depicted. Representative for all graphs we can state the graph in Figure 2.1a as $G_A = (V_A, E_A) = (\{v_1, \ldots, v_5\}, \{e_{1,2}, e_{1,4}, e_{1,5}, e_{2,3}, e_{3,4}, e_{3,5}, e_{4,5}\})$.

[1]Furthermore, this graph would ideally be directed, to consider one-way streets, and a multigraph, to consider multiple streets connecting the same intersections.

2.2 Graph Sequences

To capture the temporal evolution of graph edges we use a sequence of graphs $\underline{G} = (G[1], \ldots, G[|\underline{G}|])$. Each element of the sequence contains a graph with the same nodes and time index n dependent edges, $G[n] = (V, E[n])$. This sequence description is useful in the context of VANETs, where connections between vehicles remain established only for a limited amount of time.

2.3 Neighborhood

To define the neighborhood of an node, a method to derive a graph from another graph has to be introduced first. For a graph $G = (V, E)$ and a subset of its nodes $V' \subset V$, $G[V']$ is the graph whose node set is V' and whose edge set E' consists of all the edges in E that have both endpoints in V'. $G[V']$ is called the *subgraph* induced in G by V'. Formally we can describe it as

$$\begin{gathered} G[V'] = G(V', E'), \\ \text{with} \\ V' \subset V\ , \quad E' = \left\{ e_{i,j} \in E \middle| v_i, v_j \in V' \right\}\ . \end{gathered} \tag{2.1}$$

Based on the definition of an induced subgraph the neighborhood can be defined. The *neighborhood* of a node v_i is the subgraph induced in G by the node v_i and all nodes that are connected to it with an edge. This can be expressed as

$$N(v_i) = G[v_i \cup \{v_j \in N_s(v_i)\}]\ , \tag{2.2}$$

with $N_S(v_i)$ being the *set of neighbors*, given by

$$N_S(v_i) = \{v_j \in V | e_{i,j} \in E\}\ . \tag{2.3}$$

By this definition, the subgraph is more specifically termed *closed* neighborhood, since it includes the node v_i itself. The *open* neighborhood of v_i would not include it.

In the case of directed graphs, where the terms predecessor and successor have been introduced, the set of neighbors can be formed, by first defining the *set of successors* $\Gamma^+(v_i)$ of a node v_i as

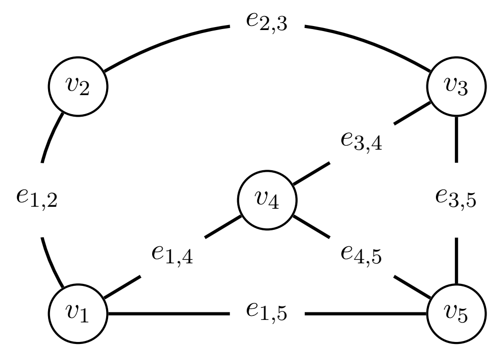
$$\Gamma^+(v_i) = \{v_j \in V | e_{i,j} \in E\}\ , \tag{2.4}$$

and the *set of predecessors* $\Gamma^-(v_i)$ of a node v_i as

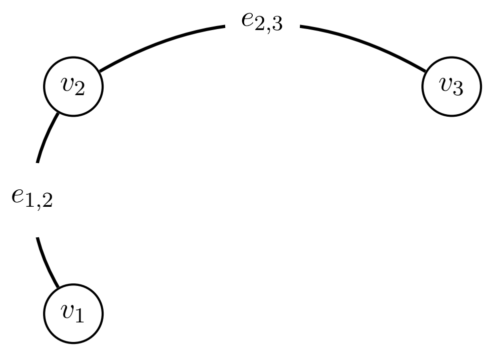
$$\Gamma^-(v_i) = \{v_j \in V | e_{j,i} \in E\}\ . \tag{2.5}$$

Their union then forms the set of neighbors $N_S(v_i) = \Gamma^+(v_i) \cup \Gamma^-(v_i)$. When working with undirected graphs the three sets are equal, $N_S(v_i) = \Gamma^+(v_i) = \Gamma^-(v_i)$.

To illustrate the concept of neighborhood we reuse the graph G_A from Figure 2.1a and show it side by side to the neighborhood $N(v_2)$ of node v_2 in Figure 2.2.



(A) Graph G_A



(B) Neighborhood $N(v_2)$

FIGURE 2.2: Examples of graph and neighborhood

2.4 Adjacency Matrix

Every finite graph[2] can be compactly represented by a matrix. The *adjacency matrix* of the graph $G = (V, E)$ is a square $|V| \times |V|$ matrix $\mathbf{A}$ with its elements $a_{i,j}$ determined by the edge set E. For a simple graph the elements $a_{i,j}$ are 1 if the graph G has an edge from node v_i to node v_j,

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,|V|} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,|V|} \\ \vdots & \vdots & \ddots & \vdots \\ a_{|V|,1} & a_{|V|,2} & \cdots & a_{|V|,|V|} \end{pmatrix}, \tag{2.6}$$

with

$$a_{i,j} = \begin{cases} 1 & \text{if } e_{i,j} \in E \\ 0 & \text{if } e_{i,j} \notin E \end{cases} \quad i, j \in \{1, \ldots, |V|\} . \tag{2.7}$$

For multigraphs the value of the matrix elements $a_{i,j}$ correspond to the number of edges between the nodes v_i and v_j,

$$a_{i,j} = \sum_{e \in E} \delta(e, e_{i,j}) , \tag{2.8}$$

where δ is the Kronecker delta function defined as

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases} . \tag{2.9}$$

For weighted graphs the elements $a_{i,j}$ are equal to the weight of the corresponding edge, as

$$a_{i,j} = \sum_{e \in E} \delta(e, e_{i,j}) w(e_{i,j}) . \tag{2.10}$$

Undirected graphs, having only bidirectional connection between the nodes, result in a symmetric adjacency matrix. Their diagonal elements are all zero, because loops are not possible for this graph type.

[2]A graph is *finite* if the node and edge sets are finite. All graphs that will be treated in this work are finite.

As an example, the unweighted undirected simple graph depicted in Figure 2.1a results in an adjacency matrix given by

$$\mathbf{A}_A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} . \tag{2.11}$$

whereas the adjacency matrix for the directed multigraph in Figure 2.1d is given by

$$\mathbf{A}_D = \begin{pmatrix} 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} . \tag{2.12}$$

2.5 Path Finding

A fundamental application of graph theory is the *shortest path problem*. It is defined as the problem of finding a path between two nodes in a graph, so that the summed weights or the number of the edges that are traversed, is minimized. Its importance becomes clear in its applications. It can be used to find the shortest route between two points in a street network or the shortest route in a communication network with relays between two communication nodes. It can also answer the more fundamental question, about the existence of a path between two nodes. To define an algorithm that solves this problem we first have to introduce some terms.

We define a *path* as an alternating sequence of nodes and edges which begins and ends with nodes. In the sequence the edges connect the preceding node with the succeeding node in the sequence. In a path all nodes, except the first and last in the sequence, and all edges are distinct. We can therefore denote a path with *length* k from node v_i to node v_j as

$$p_{i,j} = (v_i, e_{i,(1)}, v_{(1)}, \dots, v_{(k-1)}, e_{(k-1),j}, v_j) . \tag{2.13}$$

Alternatively that are two other ways to denote the path $p_{i,j}$, that are useful depending on the situation. The node representation $p^V_{i,j}$ is the sequence of the $k+1$ nodes that are traversed expressed as

$$p^V_{i,j} = (v_i, v_{(1)}, \dots, v_{(k-1)}, v_j) . \tag{2.14}$$

The edge representation $p^E_{i,j}$, on the other hand, is the sequence of the k edges that are traversed expressed as

$$p^E_{i,j} = (e_{i,(1)}, e_{(1),(2)}, \dots, e_{(k-2),(k-1)}, e_{(k-1),j}) . \tag{2.15}$$

A graph is called *connected* if there exists a path between any pair of nodes, otherwise it is called *disconnected*. Further we define the *connectivity indicator* $A(G, i, j)$ which takes the value 1 if there is a path between v_i and v_j, and 0 if there is no path.

$$A(G, i, j) = \begin{cases} 1 & \text{if } \exists p_{i,j} \text{ in G} \\ 0 & \text{if } \nexists p_{i,j} \text{ in G} \end{cases} . \tag{2.16}$$

With these terms defined, the shortest path problem can be investigated. To find the shortest paths in an unweighted undirected simple graph $G = (V, E)$, the breadth-first search (BFS) algorithm [27] can be used. It determines all shortest paths from a source node $v_s \in V$ to all other nodes in the graph. In the context of unweighted graphs, shortest refers to the length of the path, i.e. the number of nodes that are traversed. Furthermore, the algorithm can be used for either weighted or unweighted, directed or undirected, and simple or multigraphs to determine if the graph is connected or if a path between two nodes exists. This means we can use it to evaluate $A(G, i, j)$ in Equation (2.16). For other types of graphs it has first to be converted into a unweighted undirected simple graph.

The algorithm is described by the pseudo-code in Algorithm 1. The lengths of the paths are returned by $d[v]$ for all $v \in V$. The shortest path from v_s to any v_d can then be constructed in reverse by recursively evaluating $p[v]$, starting from $p[v_d]$. A graph is connected if $\forall v \in V : d[v] \neq \infty$. To only determine the path to one destination node $v_d \in V$, the algorithm has to be adapted by adding the declaration $v_d \in V$ to the parameter list on Line 1 and expanding Line 17 by $\vee d[v_d] \neq \infty$.

Algorithm 1 BFS algorithm

```
1: procedure SHORTEST-PATH($G = (V, E), v_s \in V$)
2:     for all $v_i \in V$ do
3:         $d[v_i] \leftarrow \infty$                    ▷ Set all distances to infinity
4:     end for
5:     $d[v_s] \leftarrow 0$                    ▷ Set the distance to the source node to 0
6:     $n \leftarrow 0$                         ▷ Set the current distance to 0
7:     repeat
8:         for all $v_c \in \{v \in V | d[v] = n\}$ do
9:             for all $v_n \in N_S(v_c)$ do
10:                if $d[v_n] = \infty$ then
11:                    $d[v_n] \leftarrow n + 1$                ▷ Set the distance
12:                    $p[v_n] \leftarrow v_c$                  ▷ Set the predecessor
13:                end if
14:            end for
15:        end for
16:        $n \leftarrow n + 1$
17:    until $n = |V|$
18:    return $\underline{p}, \underline{d}$              ▷ Return predecessors and distances
19: end procedure
```

An alternative algorithm to find the shortest paths has been defined in [28] and is referred to by its authors name as Dijkstra's algorithm. It applies to connected graphs that can either be undirected or directed, unweighted or weighted and simple or multigraphs. In the context of weighted graphs, shortest refers to the sum of edge weights that are traversed on the path. In case of a weighted graph it has the condition of all weights being positive,[3] while for unweighted graphs the weight can simply be set to 1 for all edges $e \in E$.

For a graph $G = (V, E)$ we define the *extended weight* of edge v_i to edge v_j from V as

[3]An algorithm that can be applied to weighted graphs with negative weights is the Bellman-Ford algorithm [29].

$$w_e(v_i, v_j) = \begin{cases} 0 & \text{if } v_i = v_j \\ w_e(e_{i,j}) & \text{if } e_{i,j} \in E \\ \infty & \text{otherwise} \end{cases}, \tag{2.17}$$

where $w(e_{i,j})$ is the weight of the edge $e_{i,j}$ in a weighted graph.

Dijkstra's algorithm to find the shortest path in G from the source node v_s to the other nodes v is then given by the pseudo-code described in Algorithm 2. It returns the shortest distances from node v_s to each node $v_d \in V$ as $d[v_d]$, that is the sum of the weights along the shortest path. Additionally, it returns the predecessors for all nodes on the shortest path $p[v]$. The shortest path from v_s to any v_d can be constructed the same way as with the BFS algorithm, in reverse by recursively evaluating $p[v]$, starting from $p[v_d]$.

Algorithm 2 Dijkstra's Shortest Path algorithm

```
 1: procedure SHORTEST-PATH($G = (V, E), v_s \in V$)
 2:     for all $v_i \in V$ do
 3:         $d[v_i] \leftarrow \infty$                    ▷ Set all distances from $v_s$ to infinity
 4:         $q[v_i] \leftarrow 0$                         ▷ Set all nodes to unvisited
 5:     end for
 6:     $d[v_s] \leftarrow 0$
 7:     $q[v_s] \leftarrow 1$
 8:     $n \leftarrow 1$                                  ▷ Set the number of visited nodes to 1
 9:     repeat
10:         $v_c \leftarrow \arg\min_{v \in V | q[v]=0} \{d[v]\}$
11:         for all $v_n \in \Gamma^+(v_c)$ do
12:             if $(q[v_n] = 1) \wedge (d[v_n] > d[v_c] + w_e(v_c, v_n))$ then
13:                 $d[v_n] \leftarrow d[v_c] + w_e(v_c, v_n)$    ▷ Set the distance of $v_s$ to $v_n$
14:                 $p[v_n] \leftarrow v_c$                      ▷ Set the predecessor of $v_n$
15:             end if
16:         end for
17:         $q[v_c] \leftarrow 1$
18:         $n \leftarrow n + 1$
19:     until $n = |V|$
20:     return $\underline{p}, \underline{d}$             ▷ Return predecessors and distances
21: end procedure
```

In case the goal is to find only the path to one specific node v_d, this can be achieved by minimal changes to the algorithm. The modifications consist of replacing Line 19 with $q[v_d] = 1$ and adding the destination declaration $v_d \in V$ to the parameters on Line 1.

Chapter 3

Simulating Connectivity in Vehicular Networks

This chapter presents the simulation framework developed as part of this thesis. It allows the simulation of connectivity in vehicular ad hoc networks (VANETs). First, a high level overview of the functionality, implementation and availability of the framework is given. Subsequently, the different simulation stages are presented in more detail. This includes theoretic background on the models that are being employed.

3.1 Simulation Framework Overview

The simulation framework is written in the Python[1] programming language and relies heavily on third party libraries and software. The most important ones are OSMnx [18] for real world map retrieval, NetworkX [30] to work with graphs and networks, Simulation of Urban Mobility (SUMO) [25] to model vehicle movement and SciPy [31] for general numerical calculations.

The framework is freely available on the Internet[2] under the permissive GNU's not Unix (GNU) General Public License. This gives anyone the opportunity to reproduce the results presented in Chapter 5, run own simulations and expand and redistribute the framework.

The simulation process consists of multiple stages, that are explained in detail in the following sections. The process starts by loading and processing street networks, including the street data itself, building data as well as data to determine the propagation condition between two points.

All the following steps will be repeated in a loop for the defined vehicle densities. First, vehicle snapshot positions are generated. The simulation framework supports two methods to generate these. Employing the first method the vehicles are placed randomly with a uniform distribution along the streets and independently between repetitions. As an alternative the framework also allows more realistic placement and movement of vehicles via SUMO [25].

The resulting vehicle snapshots are then iterated, whereby in each iteration the connected vehicles are determined. This is done by iterating all pairs of vehicles in the snapshot. For each pair, the propagation condition is determined: If there are buildings on the line connecting the pair, the connection is assumed to be non-line-of-sight (NLOS). If there are other vehicles in between, the connection is assumed to be obstructed-line-of-sight (OLOS) and if nothing is in between, the connection is assumed to be line-of-sight (LOS).

[1]Available at `https://www.python.org`
[2]Available at `https://v2v.sad.bz`

To then determine if a pair of vehicles is connected, two different metrics, pathloss and Euclidean distance, are supported by the framework. If using the former, a pathloss model corresponding to the respective propagation condition is used to determine the pathloss between the two vehicles. The connection is active if the pathloss is smaller than a set threshold. If the latter is used, the connection is active if the distance between the two vehicles is smaller than a set threshold for the respective propagation condition.

The resulting connections between vehicles in each time step are saved and further analyzed to derive metrics that are described in Chapter 4. The whole simulation process is depicted by a flowchart in Figure 3.1. More specifically, it shows the case where pathloss is used as a connection metric. However, it behaves analogous to the case where Euclidean distance is used.

3.2 Real-World Street Networks

The simulation framework uses the OSMnx [18] Python package to retrieve real world street networks, which in turn uses OpenStreetMap (OSM) [13] as main data source. When using OSMnx, a place name is passed to it which is translated to a polygon corresponding to the place's boundary geometry. A buffer is then added and the street data within the extended boundary is downloaded from OSM. From this data the street network is constructed by placing directed edges for one-way streets and reciprocal directed edges in both directions for bidirectional streets. Afterwards, OSMnx corrects the topology and determines degrees and node types. As a result nodes in the network represent dead-ends, the point from which an edge self-loops or the intersection of multiple streets.

The network is then truncated to the original boundary geometry. This buffering and subsequent truncation ensures that intersections with an incident edge connecting to a node outside of the boundary, are not represented as dead-ends. The result is a weighted directed multigraph (see Section 2.1), describing the street network. The lengths of the streets are chosen to be the weights of the graph. The direction of the edges corresponds to the driving direction of the street. Finally, the graph is a multigraph, so it can represent multiple streets between the same two intersections, as well as streets starting and ending in the same intersection.

To illustrate the context, Figure 3.2 shows an exemplary street network and its corresponding graph with the nodes/intersections numbered.

Additional to the street network, OSMnx also downloads the building geometries within the boundary of the place and returns them as a list of polygons. Both street network and building data are finally projected onto the Universal Transverse Mercator (UTM) coordinate system [32].

To reduce the computational complexity, the simulation framework supports the simplification of the building data but in turn sacrifices geometrical accuracy. More specifically, the determination of the propagation condition of each possible link between two vehicles needs to check if there are buildings on the straight line connecting the two vehicles. The complexity of this check grows linearly with the number of total building edges, because it needs to be verified if the connecting line intersects any of the building edges. So by reducing the number of edges, the complexity of the propagation condition identification is reduced.

The first step of the simplification algorithm consists of iterating all pairs of buildings and merging the ones that are not further apart geometrically than a certain threshold. If the two polygons are intersecting, then the merge consists of the

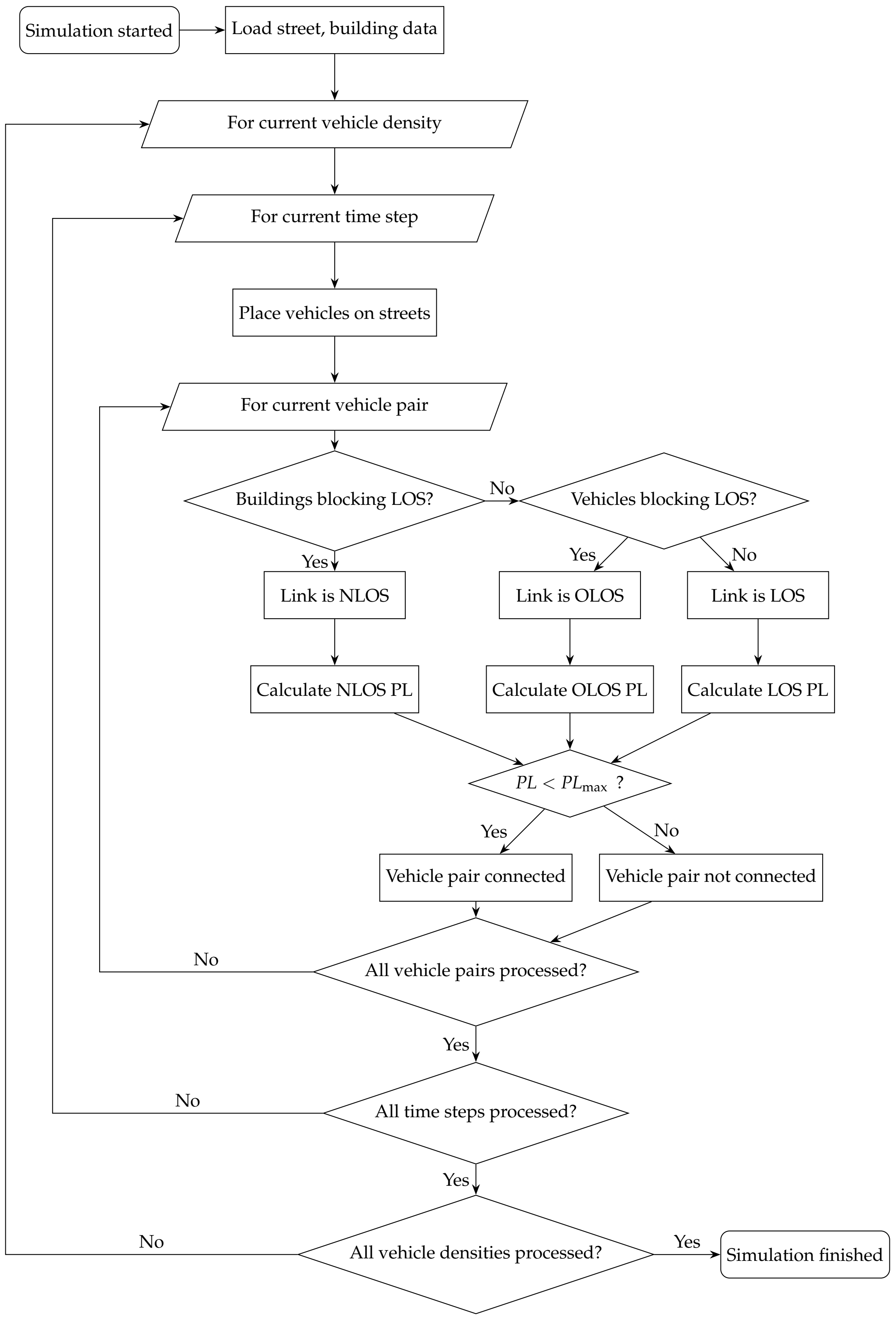
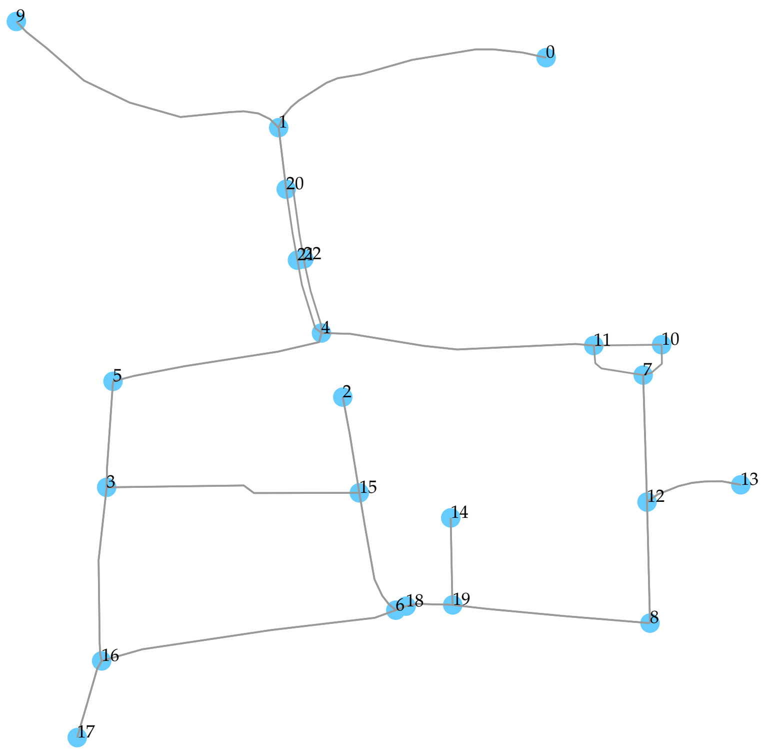
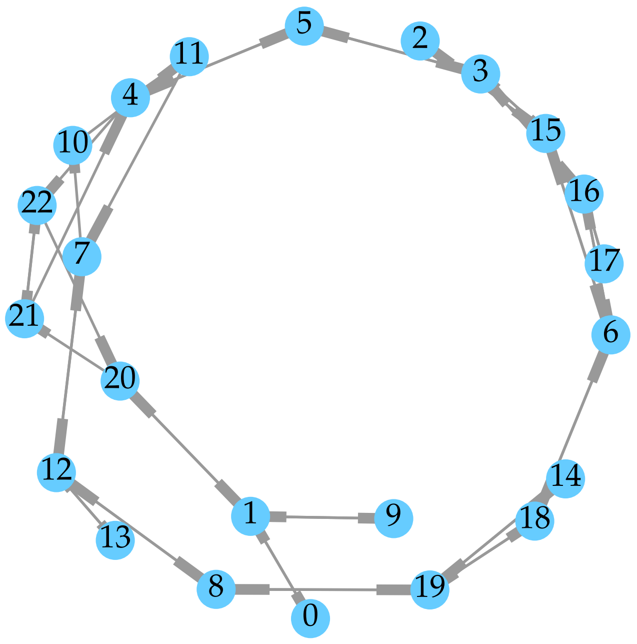


FIGURE 3.1: Flowchart of the simulation process



(A) Street network



(B) Graph representation

FIGURE 3.2: Street network and graph representation

unary union of the polygons. If they are not intersecting, the two pairs of the closest edges of the polygons are determined and from these four points a valid polygon, the filler polygon is constructed by connecting permutations of the points. The merged polygon then consists of the unary union of the two polygons and the filler polygon.

In the next step the interiors of all polygons are removed and the exteriors of the polygons are simplified using the *Douglas-Peucker algorithm* [33]. The algorithm simplifies a curve, represented by a connected series of line segments. It does so by reducing the number of its defining points while ensuring a maximum distance $\varepsilon > 0$ from any point on the simplified curve to the nearest point on the original curve.

The recursive algorithm starts by taking the original curve defined by the series of points of the connecting line segments $\mathbf{C}_{\text{orig}} = (\mathbf{p}_0, \ldots, \mathbf{p}_n)$, constructing a new line defined by the first and last point of the original curve $\overline{\mathbf{p}_0\mathbf{p}_n}$ and searching for the point with the largest distance to the new line as

$$\mathbf{p}_a = \underset{\mathbf{p}_i \in \{\mathbf{p}_1, \ldots, \mathbf{p}_{n-1}\}}{\arg\max} \; d(\mathbf{p}_i, \overline{\mathbf{p}_1\mathbf{p}_n}) \,. \tag{3.1}$$

If $d(\mathbf{p}_a, \overline{\mathbf{p}_1\mathbf{p}_n}) \leq \varepsilon$ the algorithm is aborted and the simplified curve is formed by $\mathbf{C}_{\text{simp}} = (\mathbf{p}_0, \mathbf{p}_n)$. Otherwise, the temporary curve is formed by $\mathbf{C}_{\text{simp}} = (\mathbf{p}_0, \mathbf{p}_a, \mathbf{p}_n)$ and the process repeated with the two section curves $\mathbf{C}_1 = (\mathbf{p}_0, \ldots, \mathbf{p}_i)$ and $\mathbf{C}_2 = (\mathbf{p}_i, \ldots, \mathbf{p}_n)$. This process is then repeated recursively until all recursions are aborted ensuring the maximum distance ε of the simplified curve to the original one.

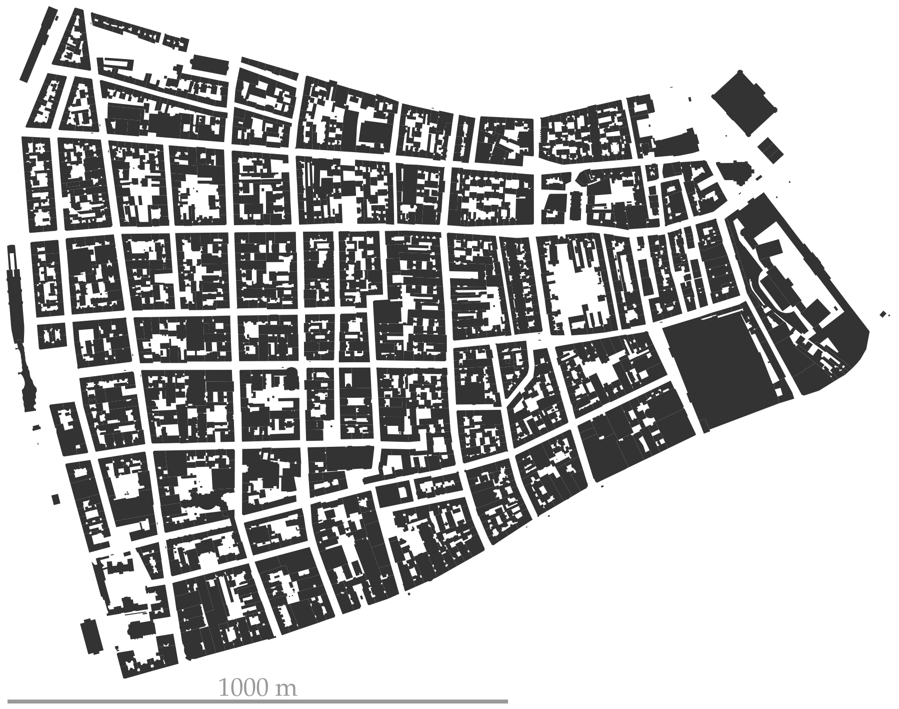
An example for original and simplified building data for Neubau, Vienna, Austria can be seen in Figure 3.3. A tolerance of one meter has been selected for the process. It can clearly be seen that the number buildings has been greatly reduced and the shapes of the resulting polygons have been simplified. This has been achieved while not changing the exteriors of the city blocks visibly. The result indicates that LOS conditions between points on streets should not change significantly due to the applied building simplification.

The effectiveness of the simplification algorithm for different places can be seen in Table 3.1. It shows the number of buildings and their edges with and without simplification for three different places. The tolerance has again been set to 1 meter. The number of buildings could be reduced by up to a factor of 5 and the sum of the number of all edges by up to a factor of 2.75. As intuition would suggest, the algorithm is more effective in areas where the building density is high for constant tolerances. Results regarding the error during simulations when using the simplified building data instead of the original one are presented in Section 5.4.

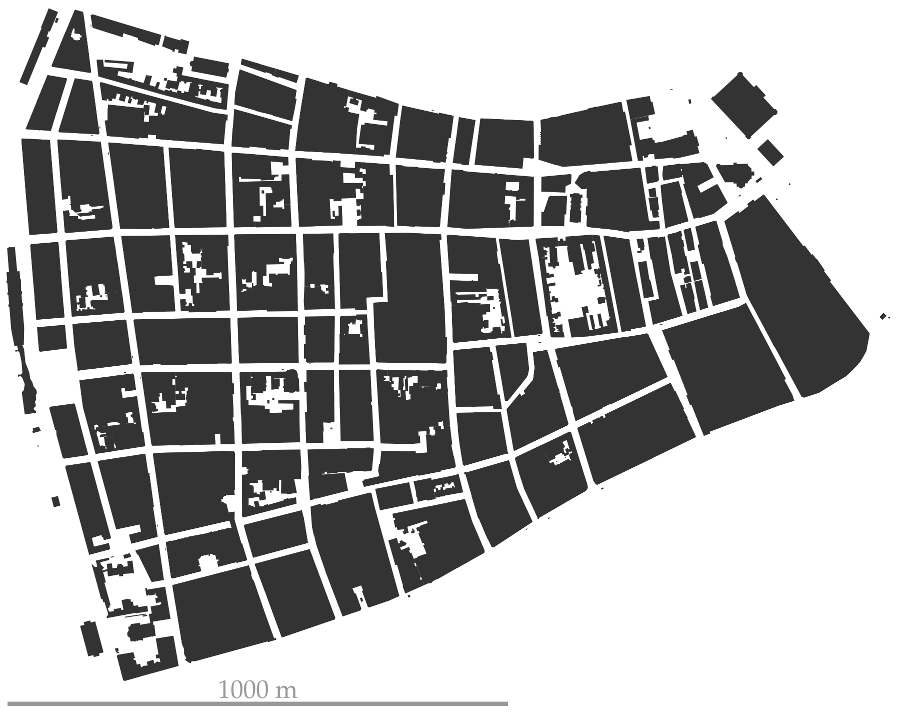
	Original		Simplified	
Place	Buildings	Edges	Buildings	Edges
Salmannsdorf, Vienna, Austria	388	3987	326	3097
Neubau, Vienna, Austria	1876	27 826	382	10 123
Upper West Side, New York, USA	4860	54 124	967	28 796

TABLE 3.1: Number of buildings and edges with and without simplification

The simulation framework generates an additional network based on the street network. This *propagation network* can be used during the simulation to determine



(A) Original buildings



(B) Simplified buildings

FIGURE 3.3: Original and simplified building footprints

the propagation condition (NLOS, OLOS, LOS) of connections. It takes the street network graph, converts it to an undirected graph and then iterates all pairs of nodes. If the nodes are not already connected by an edge, the Euclidean distance is smaller than a certain maximum and a straight line between the two nodes is not intersected by any building, an edge between the two nodes is added. This edge is represented geometrically by the straight line connecting the two nodes.

3.3 Vehicle Placement and Movement

The framework allows the placement of vehicles along roads via two different methods. The first method consists of placing vehicles randomly and uniformly distributed on the streets, therefore achieving a placement that is independent between time steps. A second, more sophisticated method uses the external software SUMO [25] and allows a more realistic simulation with vehicles moving in time between their randomly selected sources and destinations.

The number of vehicles that will be placed can be set by three different parameters independent of the placement method that has been chosen. It can be specified directly by supplying the total count of vehicles. Alternatively it can be specified by setting the area density (vehicles per area of the street network in m^{-2}) or the length density (vehicles per total road length of the street network in m^{-1}).

3.3.1 Static Placement with Uniform Distribution

The *static placement* of vehicles distributes every vehicle randomly by drawing from a uniform distribution and thus assigning each point along a street the same probability for being chosen as a vehicle position.

The vehicle placement process starts by calculating the length of each street in the network. These lengths are then used as probabilities when drawing streets that vehicles will be placed in. To finally place each vehicle on a point along the street, a number from the uniform distribution, with minimum 0 and the length of the street as maximum is drawn. This results in all points on a street being equally likely to be chosen as position for a vehicle. More specifically it results in points on one-way streets being half as likely as points on two-way streets. This algorithm is equivalent to a Poisson point process that distributes vehicles along a line. The probability $P(k, l)$ of finding k vehicles on a street segment of length l is then given by

$$P(k, l) = \frac{(\beta l)^k e^{-\beta l}}{k!} , \tag{3.2}$$

where β represents the vehicles density, in vehicles per street length.

3.3.2 Dynamic Placement with Vehicle Movement

To generate more realistic vehicle placements than the uniform distribution, the simulation framework uses a *dynamic placement* that interacts with the traffic simulator SUMO. It has been chosen to simulate vehicle movement because it is publicly available and uses the *car-following model*. This model is widely used and has been determined to model real-world vehicular dynamics, also in regard to vehicular communication [12]. SUMO allows the generation of vehicle positions by first randomly choosing start and destination positions for every vehicle, determining a route between them and then moving the vehicles along this route considering other vehicles and traffic light systems (TLSs).

SUMO employs an extended car-following model that is defined in [24]. The model development begins by distinguishing between two modes a vehicle can move in, *free motion* and *interaction with other vehicles*. Free motion is the case, when there is no other vehicle in front of a vehicle and its speed v is bound by a maximum $v \leq v_{\text{max}}$. This maximum can be interpreted as the desired speed the driver wants to reach. In the other case, when vehicles are in front, the speed v is bound by the maximum $v \leq v_{\text{safe}}$, that can be interpreted as the speed guaranteeing collision free movement.

The model further assumes that there is a maximum acceleration a and deceleration b, resulting in the conditions

$$\begin{aligned} -b \leq \frac{dv}{dt} \leq a, \\ a > 0, \quad b > 0. \end{aligned} \tag{3.3}$$

When considering only discrete time steps in increments of Δt and by combining all the previous conditions we arrive at the update scheme for the vehicles given by

$$v(t+\Delta t) \leq \min\left\{v_{\text{max}}, v(t) + a\Delta t, v_{\text{safe}}\right\} \, . \tag{3.4}$$

The information how vehicles interact has to be determined by an equation defining v_{safe} that must fulfill the condition

$$v(t+\Delta t) \geq v(t) - b\Delta t \, . \tag{3.5}$$

To arrive at an expression for the safe speed a condition on the gap between the vehicles is defined first. The *gap* g between the leading vehicle (denoted by the l) and the following vehicle (denoted by the f), is given by $g = x_l - x_f - l$, where x is the position of the respective vehicle and l is the length of a vehicle. To guarantee the absence of collisions the gap has to be larger than some non-negative desired gap g_{des} and fulfill the condition

$$\frac{dg}{dt} \geq \frac{g_{\text{des}} - g}{\tau_{\text{des}}} \, , \tag{3.6}$$

with the relaxation time τ_{des} and the desired gap g_{des}, both of which can be functions of g and the vehicles speeds. By again applying a discretization in the time domain and using $\dot{g}(t) = v_l(t) - v_f(t)$ this yields the update rule

$$v_f(t+\Delta t) \leq v_l(t) + \frac{g(t) - g_{\text{des}}(t)}{\tau_{\text{des}}(t)} \, . \tag{3.7}$$

The vehicle position is then updated according to

$$x(t+\Delta t) = x(t) + v(t+\Delta t)\Delta t \, . \tag{3.8}$$

This then results in the equations defining the car-following model from [24] as

$$\begin{aligned} v_{\text{safe}}(t) &= v_l(t) + \frac{g(t) - g_{\text{des}}(t)}{\tau_{\text{des}}} \\ v_{\text{des}}(t) &= \min\left\{v_{\text{max}}, v(t) + a(v)\Delta t, v_{\text{safe}}(t)\right\} \, , \\ v(t+\Delta t) &= \max\left\{0, v_{\text{des}}(t) - \eta\right\} \\ x(t+\Delta t) &= x(t) + v(t+\Delta t)\Delta t \end{aligned} \tag{3.9}$$

where non-ideal driving has been introduced by the random *perturbation* η. The model further defines multiple ways to set the desired gap g_{des} and starts by setting $g_{\text{des}} = \tau v_l$ with τ being the reaction time. Additionally, the model is extended to support multi-lane traffic with lane changing and the modeling of traffic congestion.

The frameworks prepares for the SUMO process by converting the street network from our simulation framework to an Extensible Markup Language (XML) format that can be read by SUMO. SUMO then generates random trips for each vehicle by selecting random points on the street network as start and destination and calculating routes between them. To prevent all TLSs from being synchronized it calculates TLS signal offsets based on a fraction of the vehicle routes. The SUMO simulation is then run and the vehicle positions for every time step are saved. These vehicle position snapshots are then loaded by the simulator and each vehicle position is moved to the nearest point on the street network to account for small differences between the coordinates of the two street networks.

Figure 3.4 shows two instances of the two different vehicle distributions. In both cases 250 vehicles, represented by blue circles, have been placed inside the street network of Neubau, Vienna, Austria. In Figure 3.4a they have been placed according to the uniform distribution whereas Figure 3.4b is a snapshot of a SUMO simulation, after a 10 minute warm-up period. It can clearly be seen that in the uniform case all parts of the street network are covered by vehicles. In the SUMO case on the other hand, there are regions where no or few vehicles ar residing and other parts show a high density of vehicles, where congestion formation can already be observed.

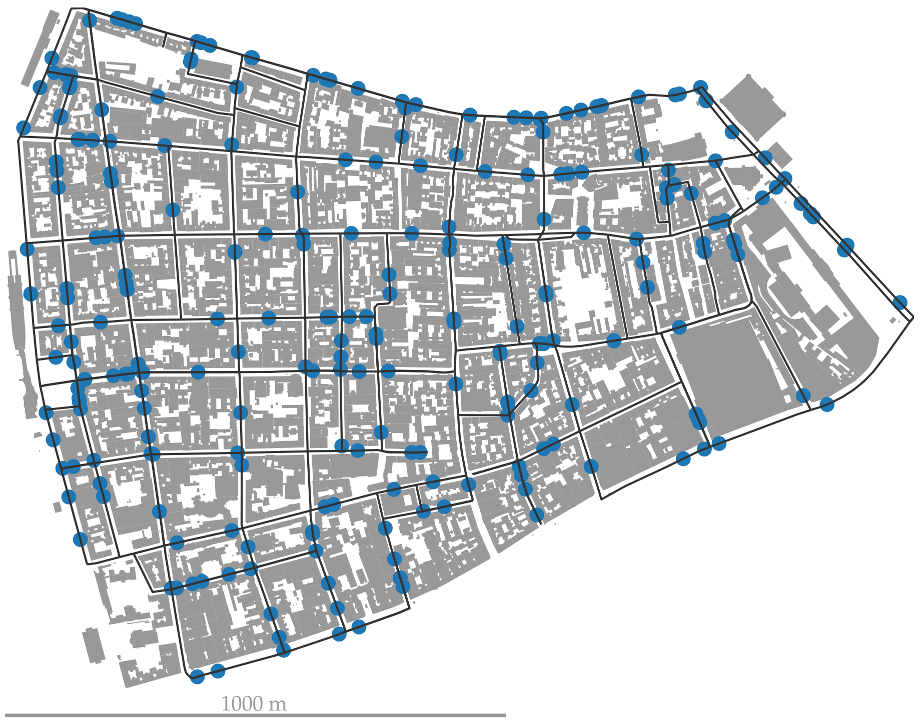
The described characteristics could be an early indicator that the uniform distribution of vehicles in the street network is not a realistic model to investigate the connectivity of a VANET. Since we expect high and low density areas in reality, similar to the SUMO case, there will be unconnected vehicles in the low density areas that can not be modeled by the uniform distribution.

3.4 Propagation Conditions

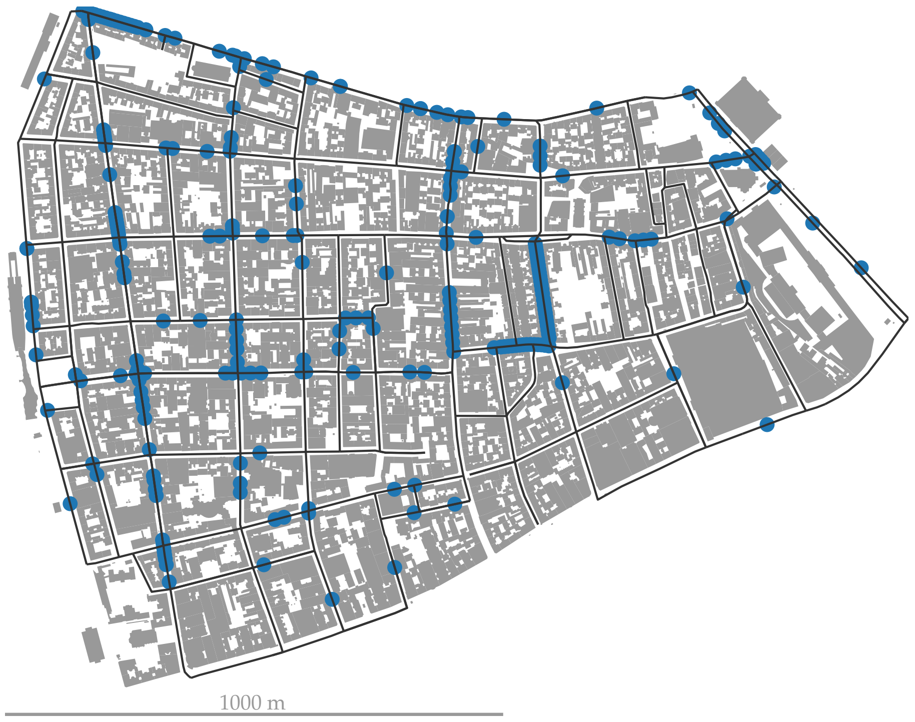
The simulation frameworks supports different *propagation conditions*, depending on the employed connection metric (see Section 3.5). If the Euclidean distance is used as the connection metric, only LOS and NLOS are used as propagation conditions. The connection is LOS if there is free space between the two vehicles and NLOS if there are buildings between the two vehicles. More specifically the simulation framework only investigates if the direct geometrical line connecting the two vehicles does not intersect any buildings. This is of course a simplification of the more accurate model, where the propagation is only assumed to be LOS if the first Fresnel zone is free of any obstructions [34, p. 58]. However, this simplification has been accepted as necessary, because of the much higher computational complexity of the alternative. Eventually, this approach makes the simulation result also comparable to published results in [9]–[11] using the same model.

If the pathloss model is used as a connection metric, additionally to LOS and NLOS there can be OLOS propagation between two vehicles if another vehicles blocks the line of sight. Furthermore, NLOS is split into NLOS between vehicles on orthogonal streets and NLOS between vehicles on parallel streets. This distinction becomes necessary because the used pathloss model distinguishes the same cases [19], [20].

To determine the propagation condition, the simulator iterates all pairs of vehicles. For each pair it constructs a direct line between the vehicles. It checks if this



(A) Static placement



(B) Dynamic placement

FIGURE 3.4: Vehicle placements

line intersects any of the buildings by iterating all buildings and for each building iterating all line segments forming the building polygon. It subsequently verifies if the polygon line segment intersects the connection line. If there is no intersection detected, the connection is either marked as LOS in case of Euclidean distance simulation or as OLOS or LOS with further investigation needed in case of pathloss simulation. In the case of an intersection, the connection is marked as NLOS when simulating with Euclidean distance and NLOS between vehicles on parallel or orthogonal streets with further investigation needed when simulating with pathloss.

As explained, in the case of pathloss simulation further distinctions are needed. To distinguish between OLOS and LOS, all other vehicles not part of the pair under investigation are iterated and checked if they intersect the line connecting the pair. In this simulation step the vehicles are modeled as circles with a set radius. If an intersection is found, the connection is marked as OLOS, otherwise it is marked as LOS.

The distinction between NLOS on parallel and orthogonal streets is done by calculating the shortest path between the two vehicles on the street network. This calculation is done using Dijkstra's algorithm, which is explained in Algorithm 2. The shortest path is then converted to a line segment and the angles along the line segments are summed up. If this sum angle is greater than a certain threshold, the two vehicles under investigation are assumed to be on parallel streets, otherwise they are assumed to be on orthogonal streets. Additionally, the position of the largest angle is saved, to later determine the location of the intersection for the pathloss model.

During the development of this algorithm we observed that the intuitive threshold sum angle of $\pi/2$ is too small and an angle $\pi/2 < \alpha < \pi$ should be chosen, to account for street irregularities. Furthermore, it should be noted that the shortest path algorithm of the framework uses the street lengths as weight which is sub-optimal to the more appropriate weight of sum angles along the street. Nevertheless, this simplification does not impair the accuracy of the algorithm considerably.

Figure 3.5 and Figure 3.6 show two exemplary results of determining the propagation conditions. 250 vehicles were placed randomly with uniform distribution on the streets of two areas. The center vehicle is marked by 'X' and all others are marked by circles, their colors corresponding to the propagation condition in regard to the center vehicle.

We first investigate Figure 3.5, corresponding to Upper West Side, New York, USA. The vehicle closest to the center vehicle in each of the two directions is determined to be in LOS, as would seem intuitive. Since no other vehicles were placed on the same street, no OLOS links exist. Vehicles on parallel streets are detected as such, whereas some vehicles that seem to be on intersecting streets have been detected as being on parallel streets. This can be explained by the simplifications made during the routing process.

This behavior can not be observed in Figure 3.6, corresponding to Neubau, Vienna, Austria. Here, all vehicles on intersecting streets are correctly detected as such. Additionally, the vehicles marked as being on parallel streets also correspond to intuition. The vehicles on the same street as the center vehicle but with the link being blocked by other vehicles are correctly determined to be in OLOS.

In general, it can be concluded from these examples, that the distinction between LOS, OLOS and NLOS works well and that further improvements in detecting orthogonal and parallel streets should be examined. The performance mainly depends on the topography of the selected area and the selected maximum sum angle.

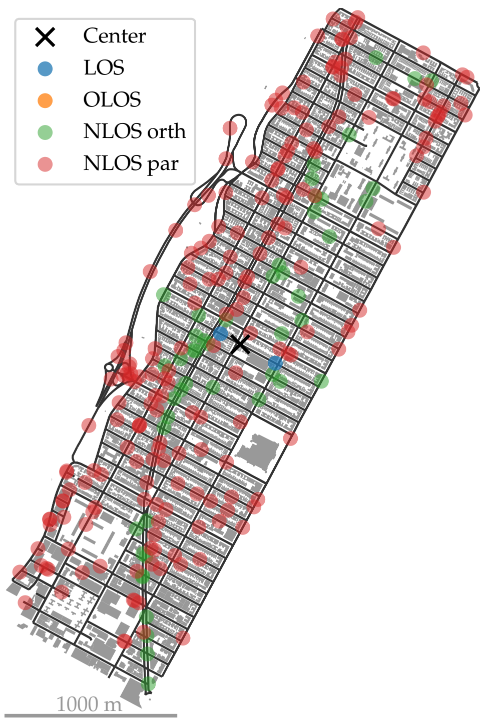


FIGURE 3.5: Propagation conditions - Upper West Side, New York, USA

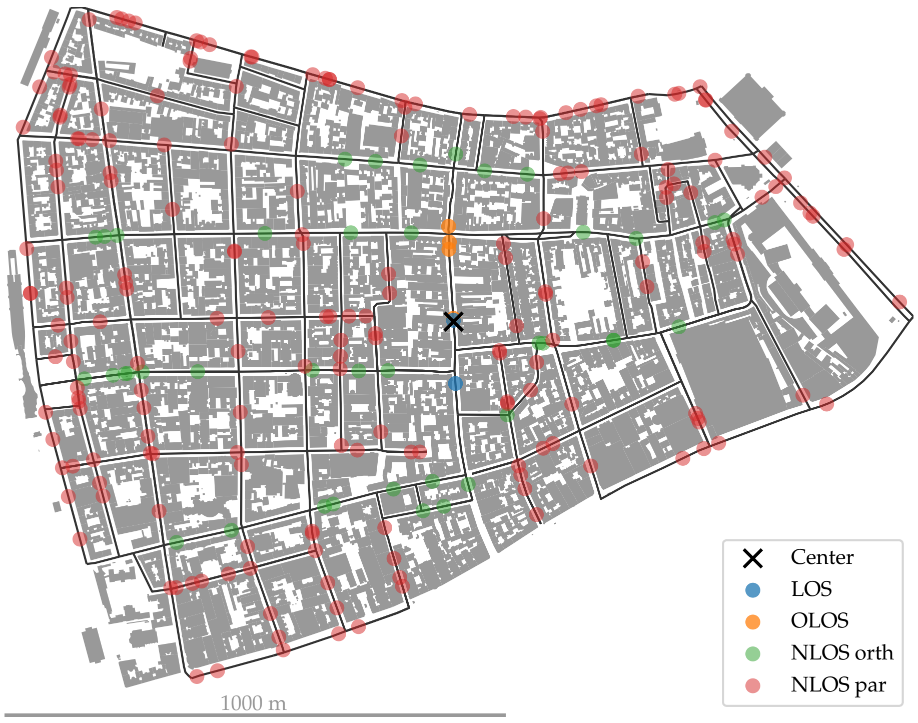


FIGURE 3.6: Propagation conditions - Neubau, Vienna, Austria

3.5 Connection Metrics

To determine if two vehicles, with a previously determined propagation condition, are connected or not the simulation framework allows the use of two different *connection metrics*, the Euclidean distance and the pathloss.

3.5.1 Euclidean Distance

Using the *Euclidean distance* is a trivial method to decide if two vehicles on a street network are connected or not. It has been used by many publications such as [9]–[12]. Using this method the simulator calculates the distances between all pairs of vehicles. It then compares these distances to a specified maximum for the corresponding propagation condition. If the distance is smaller the vehicles are set to be connected otherwise they are not connected.

3.5.2 Pathloss

As an alternative to specifying a maximum Euclidean distance, the framework allows to model the *pathloss* and *shadow fading loss* between vehicles and compare them to an adjustable maximum value in order to decide if a vehicle pair is connected or not. Together with small scale fading, pathloss and shadow fading constitute the three phenomena, that are generally used to model signal attenuation over a wireless channel.

Using the pathloss and shadow fading for connection decisions has the disadvantage that it is computationally more complex because it involves the application of a pathloss formula in addition to all the steps also required for the Euclidean distance method. However, it offers the advantages of modeling the random nature of shadow fading and the need of only one parameter, namely the maximum pathloss, instead of different maxima for each propagation condition. The simulator uses the pathloss model specified in [19] for OLOS and LOS propagation and the one given in [20] for NLOS links.

In [19] the authors use measurements between two moving cars using rooftop mounted omnidirectional antennas to derive a pathloss model for OLOS and LOS propagation. The measurements have been conducted in an urban and highway environments. These properties make the model applicable to our goal of modeling VANET connectivity.

The measurements have been conducted using a channel sounder with a bandwidth of 200 MHz and a center frequency of 5.6 GHz. To derive a pathloss formula from the measured channel transfer functions $H(f,t)$ they are transformed to the equivalent complex time varying channel impulse responses $h(t,\tau)$ via an inverse Fourier transform. From the impulse response the power delay profile (PDP) can be derived, which is additionally averaged to eliminate the effect of small scale fading, yielding the average PDP as

$$P_h(t_k,\tau) = \frac{1}{N_{\text{avg}}} \sum_{n=0}^{N_{\text{avg}}-1} |h(t_k + n\Delta t, \tau)|^2 \ , \tag{3.10}$$

for $t_k = 0, N_{\text{avg}}\Delta t, \dots, \lfloor N_t/N_{\text{avg}} - 1 \rfloor N_{\text{avg}}\Delta t$. N_{avg} is the averaging period and its chosen value corresponds to a relative movement of the two cars by 15λ satisfying the wide-sense stationary (WSS) assumption. N_t is the total number of measurements.

After suppressing the noise the averaged channel gain is obtained as

$$G_h(t_k) = \sum_{\tau} P_h(t_k, \tau) \, , \tag{3.11}$$

with τ being the propagation delay. By finally compensating for the antenna gains G_a and the implementation losses P_{IL} we retain the distance dependent pathloss as

$$PL(d) = 2G_a - P_{\mathrm{IL}} - 10\log_{10} G_h(d) \, . \tag{3.12}$$

To determine the statistical properties of the shadow fading the data sets for LOS and OLOS are investigated separately and the distance dependent channel gain $G_h(d)$ is divided into log-spaced distance bins. By investigating these datasets individually the authors come to the conclusion that the large-scale variations can be modeled by a random variable with log-normal distribution. This coincides with the most widely accepted approach to model shadow fading.

To model the pathloss the authors use a *dual-slope model*, given by

$$PL(d) = \begin{cases} PL_0 + 10n_1 \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma & \text{if } d_0 \le d \le d_b \\ PL_0 + 10n_1 \log_{10}\left(\frac{d_b}{d_0}\right) + 10n_2 \log_{10}\left(\frac{d}{d_b}\right) + X_\sigma & \text{if } d > d_b \end{cases} \, . \tag{3.13}$$

This model is a piecewise linear approximation where the mean pathloss increases with exponent n_1, termed the *pathloss exponent*, until the *breakpoint distance* d_b and with exponent n_2 afterwards. The *reference distance* d_0 is the distance which results in the *reference pathloss* PL_0 and also the minimal distance the model can be used for. The shadow fading is modeled by the zero mean Gaussian random variable X_σ with standard deviation σ_1 in the first region and σ_2 in the second.

The breakpoint distance is generally assumed to be the distance where the first Fresnel zone touches the ground i.e. the distance where the first signal bounces of the ground that travels $d_b + \lambda/4$ being reflected by the ground, while traveling from the transmitter to the receiver. This would result in a breakpoint distance of $d_b = 4h_{\mathrm{tx}}h_{\mathrm{rx}}/\lambda - \lambda/4 = 161$ m with h_{tx} and h_{rx} being the height transmitter and receiver respectively. However, a breakpoint distance of 104 m has been chosen to better fit the measurement data.

After selecting a reference distance of $d_0 = 10$ m, the pathloss exponents are selected to fit the median value of the data sets in the least square sense. This leads to the parameters listed in Table 3.2 that are used in the simulation framework together with the dual-slope equation.

Propagation	Scenario	n_1	n_2	PL_0	σ
LOS	Highway	-1.66	-2.88	-66.1	3.95
	Urban	-1.81	-2.85	-63.9	4.15
OLOS	Highway	—	-3.18	-76.1	6.12
	Urban	-1.93	-2.74	-72.3	6.67

TABLE 3.2: Parameters for the OLOS and LOS pathloss model

To determine the NLOS pathloss between vehicles that have buildings blocking their line of sight, the model proposed in [20] has been chosen. It has been derived from an extensive measurement campaign between vehicles in urban and suburban scenarios and is recommended by [19] as an appropriate method.

The measurements have been conducted in the city of Munich, Germany using off-the-shelf radios that implement IEEE 802.11p [7], that report per packet reception power values. Two cars have been equipped with these radios and placed on intersecting streets. The transmitting car has been placed on different fixed positions, while receiving car drove on the crossing street passing the intersection. The intersections have been chosen to represent a wide variety of widths of the two intersecting streets, but all crossing at an angle of 90 degrees.

To model the pathloss as a function of distances, street widths and suburban/urban differences the *Virtual Source model* proposed by [21] as

$$PL = \begin{cases} 10\log_{10}\left(\frac{1}{\alpha}\left(\sqrt{\frac{2\pi}{x_t w_r}}\frac{4\pi d_t d_r}{\lambda}\right)\right) & \text{if } d_r \leq d_b \\ 10\log_{10}\left(\frac{1}{\alpha}\left(\sqrt{\frac{2\pi}{x_t w_r}}\frac{4\pi d_t d_r^2}{\lambda d_b}\right)\right) & \text{if } d_r \leq d_b \end{cases}, \tag{3.14}$$

has been selected. It takes the distances between the intersection and transmitter d_t, the distances between intersection and receiver d_r, the distance between transmitter and the wall parallel to the street x_t and the receiver street width w_r into account. Additional parameters are the wavelength λ and the breakpoint distance d_b. The breakpoint distance is determined by $d_b = (4h_t h_r)/\lambda$ where h_t and h_r are the heights of the transmitter and receiver, respectively. The higher losses at distances greater than the breakpoint distance reflect the dominance of diffraction over reflection in the region.

This model equation was adapted to better fit the influence of the street width and the distance from the wall. Additionally, a suburban loss factor was introduced to distinguish between urban and suburban scenarios. The measured receive power values where then binned into distance regions and the median of each region determined. To resolve the variables of the model equation, it was fitted to the median values. This results in a pathloss equation given by

$$PL(d_r, d_t, w_r, x_t, i_s) = 3.75 + i_s 2.94 + X_\sigma + \begin{cases} 10\log_{10}\left(\left(\frac{d_t^{0.957}}{(x_t w_r)^{0.81}}\frac{4\pi d_r}{\lambda}\right)^{2.69}\right) & \text{if } d_r \leq d_b \\ 10\log_{10}\left(\left(\frac{d_t^{0.957}}{(x_t w_r)^{0.81}}\frac{4\pi d_r^2}{\lambda d_b}\right)^{2.69}\right) & \text{if } d_r > d_b \end{cases}. \tag{3.15}$$

The selector i_s has to be set to 1 to model a suburban scenario and to 0 to model an urban one. The shadow fading X_σ is a Gaussian random variable with zero mean and standard deviation $\sigma = 4.1\text{dB}$. This has been derived by centering the power probability distributions to their average for each intersection and bin.

Since the model depends on the distances of the vehicles to the intersection, the simulation framework needs to determine the location of it. Finding the intersection is not as trivial on real-world street networks, as it is on idealized networks, such as the Manhattan grid. The simulation framework chooses the position of the maximum angle along the path that has been determined in Section 3.4 as the intersection location.

For the pathloss formula in the NLOS case, reciprocity does not hold, meaning the pathloss does not remain constant when the transmitting vehicle changes to receiving vehicle and the receiving vehicle to transmitting. Therefore, we determine both pathlosses and take the maximum of the two. This guarantees, that the two vehicle will be simulated only as connected, if both pathlosses lie below the maximum pathloss.

Since the model defined in [20] only applies to vehicles on intersecting streets, connections between vehicles on parallel streets are still not modeled. However, [19] argues, that the pathloss between vehicles on parallel streets is very high ($> 120\text{dB}$). We therefore ignore this case in the simulator and set the pathloss to infinity.

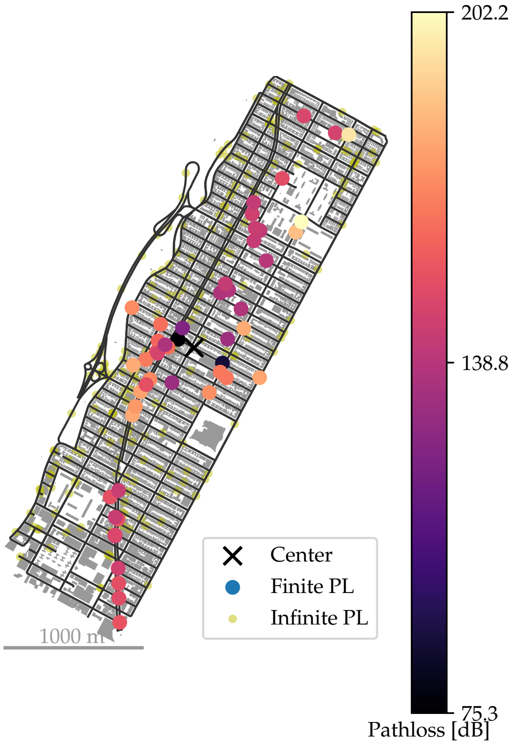


FIGURE 3.7: Pathloss - Upper West Side, New York, USA

Figure 3.8 and Figure 3.7 show the street networks of Vienna, Neubau, Austria and Upper West Side, New York USA respectively. In each network 250 vehicles have been placed randomly with a uniform distribution. The 'X' marker indicates the position of the center-most vehicle, the bigger round markers correspond to vehicles with finite pathloss. Their colors map to their respective pathloss in regard to the center vehicle. The vehicles that have infinite pathloss are marked by smaller circles. It can clearly be seen that vehicles on the same street as the center vehicle have the lowest pathloss, being either in LOS or OLOS. Vehicles on orthogonal streets are in NLOS to the center vehicle and therefore have higher pathlosses, that again rise with increasing distance. Vehicles on parallel streets have infinite pathloss.

3.6 Vehicle Connections

After each links connection metric has been calculated, they are compared to a predefined maximum, corresponding to the propagation condition. If the metric is smaller than this maximum the two vehicles are determined to be connected, otherwise they are not connected.

From this data the *vehicle connection graph sequence* $\underline{G}$ is created. The elements of the sequence are given by $G[n] = (V, E[n])$, as defined in Section 2.2. The sequence

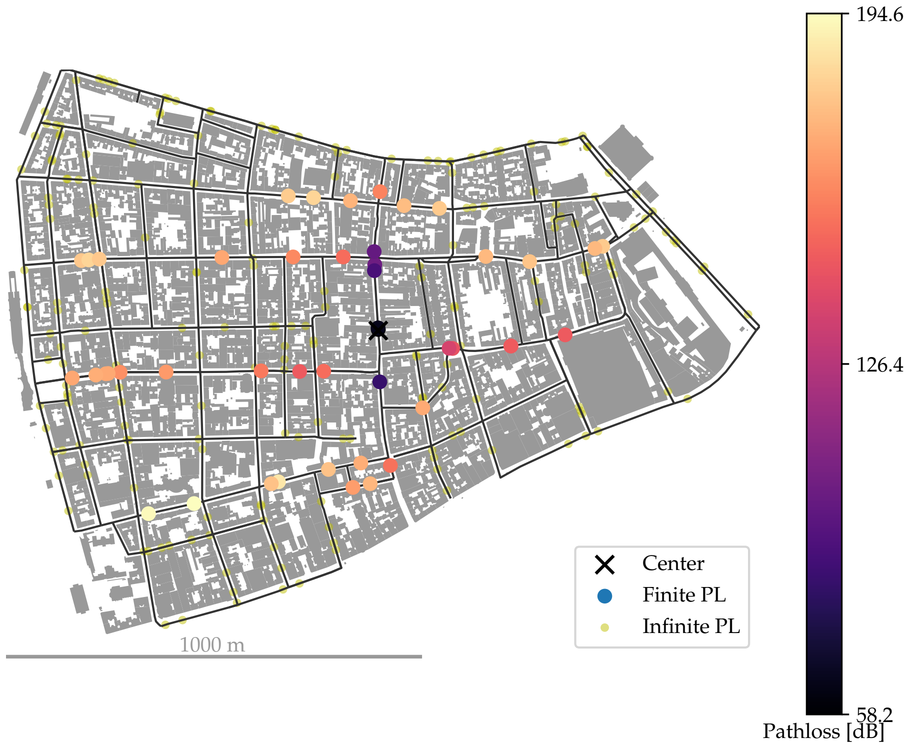


FIGURE 3.8: Pathloss - Neubau, Vienna, Austria

has length $N = |G|$, where N is the number of iterations for static vehicle placement and the time duration for dynamic vehicle placement. In the graph $G[n]$ vehicles are represented by the nodes $v \in V$. Connections between the vehicles are represented by time/iteration dependent edges $E[n]$. An edge $e_{i,j}$ is in the set $E[n]$ if there is a connection between vehicle v_i and v_j, with $i < j$, in the instance with index n. This graph is generated for each iteration in the static case respectively time step in the dynamic case. To unify these cases we will use the index n for both.

The graphs $G[n]$ have the properties of being undirected, unweighted and simple, as defined in Section 2.1. They are undirected, since our definition of a network connection assumes reciprocity, meaning all communication channels are bidirectional. This fact is also reflected in the definitions of the network metrics. Furthermore, the graph is unweighted, since the network connections reflect a binary status, either a communication channel between two nodes exists or it does not. There is no notion of quality, data-rate or distance in our model of a communication network. Finally, it is a simple graph, since no more than one communication channel can exist between a given pair of nodes and a node can not be connected to itself.

Since all the information about these graphs needs to be saved, the corresponding adjacency matrices are calculated as in Equation (2.6). It is saved instead of directly saving the graph object, to save disk space while not losing any information. The graph objects can easily be generated from the adjacency matrices. The obtained list of graphs can then be analyzed further, to obtain connectivity metrics that characterize the networks.

Exemplary vehicle snapshot are depicted in Figure 3.9 and Figure 3.10 for Upper West Side, New York, USA and Neubau, Vienna, respectively. In each network 250 vehicles were placed and propagation conditions and pathlosses in regard to the center vehicle, marked by an ’X’, calculated. The vehicles are connected to the center

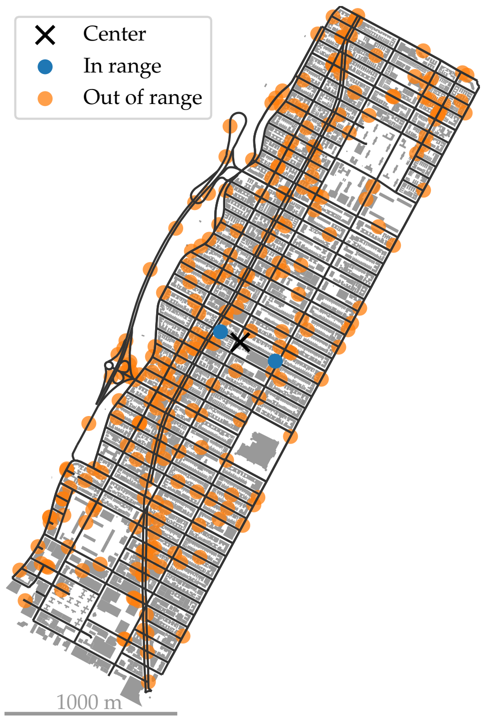


FIGURE 3.9: Connection status - Upper West Side, New York, USA

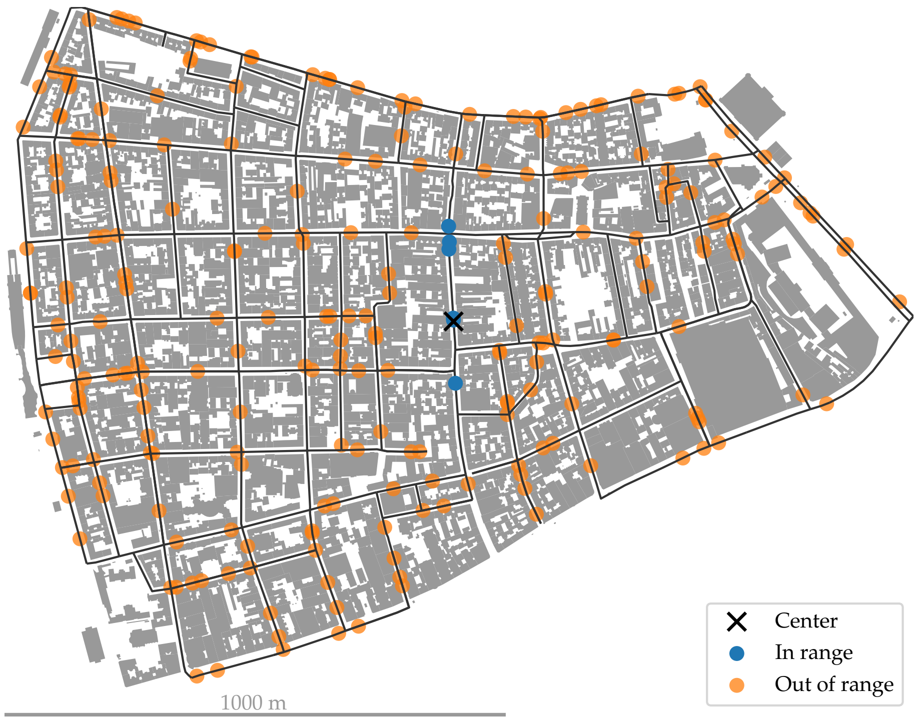


FIGURE 3.10: Connection status - Neubau, Vienna, Austria

vehicle if the pathloss was smaller than 120 dB. It can be observed that in Figure 3.9 no vehicles outside of the same street as the center vehicle are connected to it. This is due to the higher pathloss between NLOS links. In Figure 3.10 vehicles on the same street are connected to the center vehicle.

Chapter 4

Connectivity Metrics for Vehicular Networks

To characterize the connectivity of the simulated networks various metrics can be formulated. They capture different aspects of the network and can be derived from the sequence of vehicle connection graphs $\underline{G}$, that result from the process described in Chapter 3. There are two types of metrics, static and dynamic ones. *Static metrics* investigate each time instance of the graph individually and therefore neglect the temporal dimension of the network. Therefore, to simplify the notation the sequence element index n is omitted and the graph in the sequence denoted by G when considering these metrics. *Dynamic metrics*, on the other hand, consider the temporal evolution of the network, and therefore the whole sequence of graphs $\underline{G}$ will be investigated. The computation of the metrics described in this chapter is supported by the simulation framework introduced in Chapter 3 and can be done, after the connections have been determined.

4.1 Network Connectivity

The *network connectivity* $NC(G)$ of a graph $G = (V, E)$ is a static metric and is defined as

$$NC(G) \triangleq \max_{i \in \{1,\ldots,|V|\}} \left\{ \frac{1}{|V|} \sum_{j=1}^{|V|} A(G, i, j) \right\} , \tag{4.1}$$

where $A(G, i, j)$ is the connectivity indicator defined in Equation (2.16). If the network connectivity of a graph is 1, there is a path between any pair of nodes v_i and v_j and the graph is connected. For a disconnected graph the network connectivity is smaller than 1 and therefore at least one pair v_i and v_j exists, between which there is no path.

The network connectivity can also be interpreted through a cluster level analysis, if a cluster is defined as a group of nodes with any two nodes having paths between them. As a consequence, nodes in different clusters have no path between them. It follows that the network connectivity is the relative size of the largest cluster. If the largest cluster contains the nodes $V_l \subseteq V$, then the network connectivity is $NC = |V_l|/|V|$.

Regarding the vehicular ad hoc network (VANET) that is represented by the graph G, it is obvious that a high network connectivity is desirable, with $NC = 1$ being the optimal case. This would guarantee that messages sent in the network could potentially be received by every node. It is also intuitive that the network connectivity strongly depends on the vehicle density in an area and it is therefore an issue that will be investigated later on.

The simulation framework can efficiently determine the network connectivity of a graph $G = (V, E)$ by applying the breadth-first search (BFS) algorithm described in Algorithm 1, $|V|$ times, using each $v \in V$ as departure node v_d and counting the resulting distances $d[v_d] \neq \infty \; \forall v_d \in V$. The maximum of these values is then divided by $|V|$, yielding the network connectivity. The framework determines and saves the network connectivities $NC[n]$ for the whole sequence $\underline{G}$. Additionally, it determines the average network connectivity as

$$\overline{NC} = \frac{1}{|N|} \sum_{n=1}^{N} NC(G[n]), \tag{4.2}$$

which is a more meaningful metric for the network, since it is not subject to statistical fluctuations, if N is chosen large enough.

4.2 Path Redundancy

While the network connectivity is a suitable metric to determine whether a network is connected, it does not capture how well-connected it is. This richness of connections is reflected by the *path redundancy*, another static network metric. On one side it can be interpreted as a robustness of the network, since a message that can travel two redundant paths can still reach its recipient if a relay along one path becomes unavailable. However, the redundancy also indicates how many duplicate messages the recipient will receive, implying the severity of possible broadcast storms [11].

The path redundancy is the maximum number of disjoint paths that are available between two nodes and can be defined in terms of nodes or edges. Two paths are *redundant* if they start and end at the same nodes, they are *edge-disjoint* if the two paths don't traverse any common edges and they are *node-disjoint* if they don't traverse any common nodes, except the first and the last one. So the edge-disjoint path redundancy $K_{i,j}^{E-\text{dis}}$ is defined as the maximum number of edge-disjoint paths, and the node-disjoint path redundancy $K_{i,j}^{V-\text{dis}}$ as the maximum number of node-disjoint paths.

Additionally, we define the *local edge connectivity* $\kappa_{i,j}^{E}$ as the minimum number of edges that must be removed from the graph to disconnect v_i and v_j, and the *local node connectivity* $\kappa_{i,j}^{V}$ as the minimum number of nodes that must be removed to disconnect them. They are also referred to as the minimum edge and minimum node cut, respectively. It is intuitive, however not trivial to prove, that the edge connectivities are equal to the path redundancies, $K_{i,j}^{E-\text{dis}} = \kappa_{i,j}^{E}$ and $K_{i,j}^{V-\text{dis}} = \kappa_{i,j}^{V}$ [35]. This justifies the denotation of redundancy, since it is a measure of resilience of the graph to node or edge elimination.

The m-th path between two nodes v_i and v_j in the graph $G = (V, E)$ denoted by $p_{i,j,m}$ can be expressed using the edge representation $p_{i,j,m}^{E}$ from Equation (2.15). Alternatively the path can be denoted by the node representation $p_{i,j,m}^{V}$ from Equation (2.14).

We can therefore define the set of all edge-disjoint paths between v_i and v_j in G as

$$P_{i,j}^{E\text{-dis}} = \left\{ p_{i,j,m} : p_{i,j,m}^{E} \cap p_{i,j,m'}^{E} = \emptyset \; \forall k \neq k' \right\} \; . \tag{4.3}$$

Furthermore, we define the set of all node-disjoint paths between v_i and v_j in G as

$$P_{i,j}^{V\text{-dis}} = \left\{p_{i,j,m} : \left(p_{i,j,m}^{V} \cap p_{i,j,m'}^{V}\right) \setminus \{v_i, v_j\} = \emptyset \ \forall k \neq k'\right\} . \tag{4.4}$$

The edge-disjoint and node-disjoint path redundancy are therefore the maximum cardinalities of these sets,

$$K_{i,j}^{E-\text{dis}} = \max\left\{\left|P_{i,j}^{E\text{-dis}}\right|\right\} , \tag{4.5}$$

and

$$K_{i,j}^{V-\text{dis}} = \max\left\{\left|P_{i,j}^{V\text{-dis}}\right|\right\} , \tag{4.6}$$

respectively.

To extend this notion of path redundancy to the whole network we can determine the set of edge-disjoint and node-disjoint path redundancies for all unique pairs of nodes as

$$D^{E\text{-dis}} = \left\{\left|P_{i,j}^{E\text{-dis}}\right| : i, j \in \{1, \ldots, |V|\}, i < j\right\} , \tag{4.7}$$

and

$$D^{V\text{-dis}} = \left\{\left|P_{i,j}^{V\text{-dis}}\right| : i, j \in \{1, \ldots, |V|\}, i < j\right\} , \tag{4.8}$$

respectively.

To find the sets with maximum cardinality, the Max-flow Min-cut theorem [36] can be used by calculating the maximum flow in the network. Alternatively, [37] proposes to iteratively find the shortest path between the two nodes using the BFS algorithm from Algorithm 1. After each iteration the nodes along the path are removed from the graph and the path count is incremented. This however yields not the exact node-disjoint path redundancy, but a lower bound, since a shorter path could use nodes that may belong to two different node independent paths, if the paths were longer. It is however intuitive to always use the shortest path, so that a minimum number of nodes is removed.

The simulator determines the two redundancies for all unique pairs of nodes and generates its probability mass functions (pmfs). From them the average, minimum and maximum path redundancies can be extracted, which are representative metrics for the robustness of the network.

4.3 Nodal Degree

The *nodal degree* $n(v_i)$ of a node v_i in a graph $G = (V, E)$ is defined as the number of neighbors the node has in the graph. It is equal to the cardinality of the set of neighbors (Equation (2.3)) or equivalently the order of the neighborhood of v_i (Equation (2.2)) reduced by one:

$$n(v_i) = |N_S(v_i)| . \tag{4.9}$$

Applied to the VANET it is the number of other vehicles that are in transmission range of a vehicle and can be interpreted as the local density of the network from a physical connectivity point of view.

The simulation framework determines and saves the nodal degree of all nodes.

4.4 Link Duration

To investigate the temporal evolution of a network we define a *link* between two nodes v_i and v_j as the existence of the edge $e_{i,j} \in E$ in the graph $G = (V, E)$. Equivalently it can be expressed as v_i and v_j being neighbors. The link duration can subsequently be interpreted as the duration the link between two nodes exists uninterrupted in time and it is therefore a dynamic network metric.

First we define the *total link duration*, which is given for the sequence of graphs $G[n] = (V, E[n])$ and any pair of nodes with indices i and j at time instant n as

$$\begin{aligned} T_{l,t}[n] &\triangleq \begin{cases} n_f - n_0 + 1 & \text{if } e_{i,j} \in E[n] \\ 0 & \text{if } e_{i,j} \notin E[n] \end{cases} \\ &\quad \text{with} \\ n_0, n_f &: e_{i,j} \notin E[n_0 - 1], \\ &\quad e_{i,j} \notin E[n_f + 1], \\ &\quad e_{i,j} \in E[k] \; \forall k \in \{n_0, \dots, n_f\} \,. \end{aligned} \tag{4.10}$$

It is therefore the duration for which an edge between nodes v_i and v_j has existed and will still exist at time n. To only determine the duration an edge will still exist in the future, we define the *residual link duration* as

$$\begin{aligned} T_{l,r}[n] &\triangleq \begin{cases} n_f - n + 1 & \text{if } e_{i,j} \in E[n] \\ 0 & \text{if } e_{i,j} \notin E[n] \end{cases} \\ &\quad \text{with} \\ n_f &: e_{i,j} \notin E[n_f + 1], \\ &\quad e_{i,j} \in E[k] \; \forall k \in \{n, \dots, n_f\} \,. \end{aligned} \tag{4.11}$$

Since the simulation framework determines and saves all link durations, it is also of interest to determine *unique link duration*, that is the duration of a link period $n_0, \dots, n_f$ should only be counted once. We therefore define the unique link duration as

$$\begin{aligned} T_{l,u}[n] &\triangleq \begin{cases} n_f - n + 1 & \text{if } e_{i,j} \in E[n] \,,\; e_{i,j} \notin E[n-1] \\ 0 & \text{otherwise} \end{cases} \\ &\quad \text{with} \\ n_f &: e_{i,j} \notin E[n_f + 1], \\ &\quad e_{i,j} \in E[k] \; \forall k \in \{n, \dots, n_f\} \,. \end{aligned} \tag{4.12}$$

The index where the respective link duration is returned has been arbitrarily chosen to be the beginning of the link duration, which corresponds to n_0 in the definition of the total link duration.

The simulation framework determines the three different link durations for all unique node sets $\{v_i, v_j\}$ and all time instances n. It saves the ones different from zero in a sequence. The pseudo code in Algorithm 3 illustrates the process. Additionally, the framework calculates their empirical pmf and the averages for all three cases.

Algorithm 3 Link Durations

```
1: procedure LINK-DURATIONS($G[n] = (V, E[n])$)
2:     $\underline{\mathcal{T}}_{l,t} = ()$                                  ▷ Initialize empty sequence
3:     $\underline{\mathcal{T}}_{l,r} = ()$
4:     $\underline{\mathcal{T}}_{l,u} = ()$
5:     for all $i \in \{1, \dots, |V|\}$ do
6:         for all $j \in \{i+1, \dots, |V|\}$ do                ▷ Iterate all unique node pairs
7:             for all $k \in \{1, \dots, |G[n]|\}$ do             ▷ Iterate all time instances
8:                 if $T_{l,t}[k] \neq 0$ then
9:                     $\underline{\mathcal{T}}_{l,t}\left[|\underline{\mathcal{T}}_{l,t}| + 1\right] \leftarrow T_{l,t}[k]$      ▷ Add total duration to sequence
10:                end if
11:                if $T_{l,r}[k] \neq 0$ then
12:                    $\underline{\mathcal{T}}_{l,r}\left[|\underline{\mathcal{T}}_{l,r}| + 1\right] \leftarrow T_{l,r}[k]$      ▷ Add unique duration to sequence
13:                end if
14:                if $T_{l,u}[k] \neq 0$ then
15:                    $\underline{\mathcal{T}}_{l,u}\left[|\underline{\mathcal{T}}_{l,u}| + 1\right] \leftarrow T_{l,u}[k]$      ▷ Add residual duration to sequence
16:                end if
17:            end for
18:        end for
19:    end for
20:    return $\underline{\mathcal{T}}_{l,t}$, $\underline{\mathcal{T}}_{l,r}$, $\underline{\mathcal{T}}_{l,u}$              ▷ Return all sequences of link durations
21: end procedure
```

4.5 Connection Duration and Period

To not only capture the immediate physical attachment between two nodes in terms of links, the notion of connections is introduced. A *connection* between a pair of nodes v_i and v_i exists the connectivity indicator in Equation (2.16) $A(G, i, j) = 1$. Connections reflect the attachment of two vehicles in terms of a network, instead of a direct physical attachment. If the higher layers of a communication protocol support routing and relaying, messages can not only propagate via links but also via connections and therefore potentially reach much further. To investigate the time evolution of connections we define the connection duration as the time interval a connection between two nodes exists uninterrupted.

As in the link case, we can define three different connection durations and due to the analogy they will be defined briefly. The *total link duration* for any pair of nodes with indices i and j in the graph $G[n]$ at time instant n is defined as

$$\begin{aligned} T_{c,t}[n] &\triangleq \begin{cases} n_f - n_0 + 1 & \text{if } A(G[n], i, j) = 1 \\ 0 & \text{if } A(G[n], i, j) = 0 \end{cases} \\ &\text{with} \\ n_0, n_f :\ & A(G[n_0 - 1], i, j) = 0, \\ & A(G[n_f + 1], i, j) = 0, \\ & A(G[k], i, j) = 1\ \forall k \in \{n_0, \dots, n_f\}\,. \end{aligned} \tag{4.13}$$

The *residual link duration* for any pair of nodes with indices i and j in the graph $G[n]$ at time instant n is defined as

$$
\begin{gathered}
T_{c,r}[n] \triangleq \begin{cases} n_f - n + 1 & \text{if } A(G[n], i, j) = 1 \\ 0 & \text{if } A(G[n], i, j) = 0 \end{cases} \\
\text{with} \\
\begin{aligned} n_f :\, & A(G[n_f + 1], i, j) = 0, \\ & A(G[k], i, j) = 1 \; \forall k \in \{n, \dots, n_f\} \,. \end{aligned}
\end{gathered}
\tag{4.14}
$$

The *unique link duration* for any pair of nodes with indices i and j in the graph $G[n]$ at time instant n is defined as

$$
\begin{gathered}
T_{c,u}[n] \triangleq \begin{cases} n_f - n + 1 & \text{if } A(G[n], i, j) = 1 \,,\; A(G[n-1], i, j) = 0 \\ 0 & \text{otherwise} \end{cases} \\
\text{with} \\
\begin{aligned} n_f :\, & A(G[n_f + 1], i, j) = 0, \\ & A(G[k], i, j) = 1 \; \forall k \in \{n, \dots, n_f\} \,. \end{aligned}
\end{gathered}
\tag{4.15}
$$

The simulation framework determines the total, residual and unique connection durations for all unique pairs of nodes and all time instants. The process is analogue to the one for link durations described in Algorithm 3. Additionally, their averages and pmfs are calculated and stored. To gather more insight into the dynamics of the network the number of connected periods

$$
N_{CP} = |\{T_{c,u}[n] | n \in \{1, \dots N\} : T_{c,u}[n] \neq 0\}| \;, \tag{4.16}
$$

for all unique pairs of nodes are calculated and saved.

4.6 Re-Healing Time

To investigate disconnected nodes, i.e. nodes that have no path between them, we introduce the *re-healing time*. It is the time duration two nodes have to wait until a connection is again established. The re-healing time is an extremely important metric for higher layer communication protocols that implement a store-and-carry-forward mechanism, where nodes store sent messages that they were tasked to relay until a path to the recipient becomes available. It also impacts the optimal frequency of periodic beacon messages, where nodes announce their existence to the network. A high frequency tends to overflow the network while a low frequency can result in nodes missing the joining and leaving of a node completely.

We therefore define the *unique re-healing time* for any pair of nodes with indices i and i in the graph $G[n]$ at time instant n as

$$
\begin{gathered}
T_{r,u}[n] \triangleq \begin{cases} n_f - n + 1 & \text{if } A(G[n], i, j) = 0 \,,\; A(G[n-1], i, j) = 1 \\ 0 & \text{otherwise} \end{cases} \\
\text{with} \\
\begin{aligned} n_f :\, & A(G[n_f + 1], i, j) = 1, \\ & A(G[k], i, j) = 0 \; \forall k \in \{n, \dots, n_f\} \,. \end{aligned}
\end{gathered}
\tag{4.17}
$$

In the set of re-healing times we collect all unique re-healing times different from zero as

$$S_{rt} = \{T_{c,u}[n] | n \in \{1, \dots N\} : T_{r,u}[n] \neq 0\} \ , \tag{4.18}$$

to determine the average re-healing time as

$$\overline{T}_r = \frac{1}{|S_{rt}|} \sum_{T_r \in S_{rt}} T_r \tag{4.19}$$

and the empirical pmf from all elements in S_{rt}.

4.7 Information Bottleneck Method

Depending on the analyzed properties of the VANET we might get empirical distributions in two variables X and Y, in the form of their joint pmf $p(x, y)$ as a result. A natural objective would then be to simplify the distribution by clustering X while retaining most information. This can generally be stated as the question, what features of X are relevant to predict Y?

To rephrase the question in an information theoretic context we first introduce the *mutual information* between two random variables X and Y as

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) = \sum_{x \in X, y \in Y} p(x)p(y|x) \log \left(\frac{p(y|x)}{p(y)} \right) \ . \tag{4.20}$$

The question is then, what is the compressed representation of X, $\tilde{X}$, that retains the most mutual information $I(\tilde{X}; Y)$, while being constrained on the mutual information $I(X; \tilde{X})$. This representation is found by the *Information Bottleneck method* [38] which yields a set of self-consistent equations for the representation:

$$\begin{cases} p(\tilde{x}|x) = \frac{p(\tilde{x})}{Z(\beta,x)} \exp(-\beta D_{KL}[p(y|x)||p(y|\tilde{x})]) \\ p(y|\tilde{x}) = \sum_x p(y|x)p(\tilde{x}|x)\frac{p(x)}{p(\tilde{x})} \\ p(\tilde{x}) = \sum_x p(\tilde{x}|x)p(x) \end{cases} , \tag{4.21}$$

where

$$D_{KL}[p||q] = \sum_y p(y) \log \left(\frac{p(y)}{q(y)} \right) \ , \tag{4.22}$$

is the *Kullback-Leibler divergence* [39]. β is a Lagrange multiplier and $Z(\beta, x)$ a normalization function. For the limit $\beta \to \infty$ the mapping $p(\tilde{x}|x)$ becomes deterministic and leads to the *Agglomerative Information Bottleneck algorithm* [40].

To show how the algorithm works we look at a m-partition of X, Z_m, in which several components will be merged into a $\overline{m}$-partition $\overline{Z}_m$, by merging the components $\{z_1, \dots, z_k\} \subseteq Z_m$ into $\overline{z}_k \in Z_{\overline{m}}$ with $\overline{m} = m - k + 1$. For the new component $\overline{z}_k$ we can define its probability distributions as

$$\begin{cases} p(\overline{z}_k|x) = \begin{cases} 1 & \text{if } x \in z_i \text{ for some } 1 \leq i \leq k \\ 0 & \text{otherwise} \end{cases} \quad \forall x \in X \\ p(y|\overline{z}_k) = \frac{1}{p(z)_k} \sum_{i=1}^{k} p(z_i, y) \ \forall y \in Y \\ p(\overline{z}_k) = \sum_{i=1}^{k} p(z_i) \end{cases} . \tag{4.23}$$

For other $z \in Z_{\overline{m}}, z \neq \overline{z}_k$ the probability distributions stay the same as in Z_m.

We can define the merge prior distribution $\Pi_k = (\pi_1, \dots, \pi_k)$, where π_i is the prior probability of z_i in the merged subset, $\pi_i = p(z_i)/p(\overline{z}_k)$. We observe that the merge $\{z_1, \dots, z_k\} \Rightarrow \overline{z}_k$ results in the decrease of mutual information $I(\tilde{X}; Y)$ given by

$$\begin{aligned} \delta I_y(z1, \dots, z_k) &\triangleq I(Z_m; Y) - I(Z_{\overline{m}}; Y) \\ &= p(\overline{z}_k)\, JS_{\Pi_k}[p(Y|z_1), \dots, p(Y|z_k)] \geq 0 , \end{aligned} \tag{4.24}$$

and termed Y-information decrease. JS_Π is the *Jensen-Shannon divergence* with the priors Π, defined as

$$JS_\pi[p_1, p_2, \dots, p_M] \triangleq H\left[\sum_{i=1}^{M} \pi_i p_i(x)\right] - \sum_{i=1}^{M} \pi_i H[p_i(x)] , \tag{4.25}$$

where $H[p(x)]$ is the *Shannon entropy*, given by

$$H[p(x)] = -\sum_x p(x) \log(p(x)) . \tag{4.26}$$

In a decision theoretic problem, the Jensen-Shannon divergence of the conditional pmfs is identical to the mutual information between the sample spaces of the classes, expressed as

$$JS_{p(y_1),\dots,p(y_M)}[p(x|y_1), \dots, p(x|y_M)] = H(X) - H(X|Y) = I(X;Y) \tag{4.27}$$

The decrease of mutual information $I(\tilde{X}; X)$ can be determined as

$$\begin{aligned} \delta I_x(z1, \dots, z_k) &\triangleq I(Z_m; X) - I(Z_{\overline{m}}; X) \\ &= p(\overline{z}_k)\, H[\Pi_k] \geq 0 , \end{aligned} \tag{4.28}$$

and is termed X-information decrease. It can now be proven that any merge of k components can be equivalently realized by $(k-1)$ consecutive merges of pair of components. Furthermore, for every $k \geq 2$, $\delta I_y(z_1, \text{¸}\dots, y_k) \leq \delta I_y(z_1, \dots, z_k, z_{k+1})$ and $I_x(z_1, \text{¸}\dots, y_k) \leq \delta I_x(z_1, \dots, z_k, z_{k+1})$. It follows further, that for every $1 \leq m \leq N$ an optimal m-partition can be realized by $(N-m)$ consecutive merges of pairs.

This leads to the Agglomerative Information Bottleneck algorithm. Because of the aforementioned properties, the merges can be performed iteratively by searching for the pair of clusters, that minimize the reduction of information δI_y.

Since in our applications, it makes sense to only merge adjacent clusters, we add this modification to the algorithm. The modified version is described by pseudocode in Algorithm 4, whereas if Lines 17 and 19 to 21 are omitted, we obtain the original algorithm described in [40].

The need for this modification is best understood by applying the algorithm to an example. From simulation and analysis of their results we can obtain distances

Algorithm 4 Modified Agglomerative Information Bottleneck method

```
1: procedure MOD-AGGLOM-INFO-BOTTLENECK($p(x,y), N = |X|, M = |Y|$)
                                                                    ▷ Initialization
2:     for $i = 1, \dots, N$ do
3:         $z_i \leftarrow \{x_i\}$
4:         $p(z_i) \leftarrow p(x_i)$
5:         $p(y|z_i) \leftarrow p(y|x_i)\ \forall y \in Y$
6:         for $j = 1, \dots, N$ do
7:             if $j = i$ then
8:                 $p(z|x_j) \leftarrow 1$
9:             else
10:                $p(z|x_j) \leftarrow 0$
11:            end if
12:        end for
13:    end for
14:    $Z \leftarrow \{z_1, \dots, z_N\}$
15:    for $i = 1, \dots, N$ do
16:        for $j = i + 1, \dots, N$ do
17:            if $j - i = 1$ then
18:                $d_{i,j} \leftarrow (p(z_i) + p(z_j))\ JS_{\Pi_2}[p(y|z_i), p(y|z_j)]$
19:            else
20:                $d_{i,j} \leftarrow \infty$ ▷ Non-adjacent clusters get assigned infinite divergence
21:            end if
22:                                  ▷ Every $d_{i,j}$ points to the corresponding couple in $Z$
23:        end for
24:    end for
                                                                    ▷ Loop
25:    for $t = 1, \dots, N - 1$ do
26:        Find $\{\alpha, \beta\} \leftarrow \arg\min_{i,j} d_{i,j}$
27:        merge $\{z_\alpha, z_\beta\} \Rightarrow \overline{z}$
28:            $p(\overline{z}) \leftarrow p(z_\alpha) + p(z_\beta)$
29:            $p(y|\overline{z}) \leftarrow (p(z_\alpha, y) + p(z_\beta, y)) / p(\overline{z})$
30:            for all $x \in X$ do
31:                if $x \in \{z_\alpha, z_\beta\}$ then
32:                    $p(\overline{z}|x) = 1$
33:                else
34:                    $p(\overline{z}|x) = 0$
35:                end if
36:            end for
37:        end merge
38:        Update $Z \leftarrow \{Z \setminus \{z_\alpha, z_\beta\}\} \cup \{\overline{z}\}$    ▷ $Z$ is the $(N - t)$-partition of $X$
39:        Update $d_{i,j}$ costs and pointers w.r.t. $\overline{z}$
40:    end for
41: end procedure
```

between the vehicles X and their path redundancy Y. Their distribution can be expressed using the empirical joint pmf $p(X, Y)$. We can apply the Agglomerative Information Bottleneck algorithm to $p(X, Y)$ to partition the distances X into clusters. However, it is intuitive to only cluster adjacent distances, resulting in continuous distance regions.

Chapter 5

Simulations

This chapter presents simulation setups and results, as well as conclusions drawn from them. These simulations have been conducted using the simulation framework presented in Chapter 3 and the results analyzed by metrics defined in Chapter 4.

5.1 Manhattan Grid vs. Real-World Maps

The main characteristics of the simulation framework is the usage of real-world street network data. We therefore want to investigate the impact of using these maps in comparison to the idealized Manhattan grid. The results for Manhattan grid simulations originate from [9]. It has been chosen as source because it is a comprehensive work on vehicular ad hoc network (VANET) connectivity and offers a wide range of connectivity metrics resulting from simulations.

5.1.1 Simulation Setup

The simulations in [9] were conducted using a Manhattan grid, consisting of a 2 x 2 km road network with each road block being 125 m long. The resulting street network is a two-dimensional regular square grid. Two different places to compare this scenario with, were chosen. The first street network consists of the area Upper West Side, New York, USA, a part of Manhattan. This area has been chosen because it has similar properties as the Manhattan grid, namely only perpendicular and parallel streets, and comparable intersection density and average road length. Additionally, we simulated Neubau, Vienna, Austria, to have further simulation results, originating from a street network with different properties.

In the reference work [9], vehicle movement is modeled via a simple cellular automaton (CA) approach described in [10], whereas we use the more sophisticated car-following model of Simulation of Urban Mobility (SUMO) [24], as described in Section 3.3.2. The maximum speed of SUMO was set to 10 m/s, the same value used for the cellular automaton model.[1] For an additional comparison, we repeated the simulation with static vehicle placement, where vehicles are distributed randomly following a uniform distribution.

To model the traffic light system (TLS), the same timing parameters as in [9] have been chosen. However, they apply a signal offset of 10 seconds to neighboring traffic lights in each direction. This is not possible in a non-ideal street network and therefore the TLSs offsets were determined by the routes of the simulated vehicles.

The distinction between line-of-sight (LOS) and non-line-of-sight (NLOS) links in the reference paper is done by checking if the two vehicles are on the same street,

[1]This can not be stated with certainty, because of a mistake in [9, Table 1], where both 15 m/s and 36 km/h are stated as maximum speed.

or at an intersection where they can see each other. Our simulation, on the other hand, considers building data and employs the process described in Section 3.4. The Euclidean distance has been used in both cases to determine connections. Additionally, the maximum distances for both propagation conditions have been chosen consistently.

Both simulation scenarios were repeated with multiple vehicle densities, ranging from 10 to 160 veh/km^2. Vehicle snapshots of a 2000 s period have been used to derive the resulting metrics, so that statistical fluctuations were eliminated. A warm-up period of 1000 s has been employed, where vehicle snapshots have been discarded. This results in only the steady-state in the network being considered. The simulation parameters, that are consistent in our simulation and the ones conducted by [9], are summarized in Table 5.1, while the differing parameters are listed in Table 5.2.

Parameter	Value
Vehicle densities	$\{10, 20, \ldots, 80, 120, 160\}$ $1/\text{km}^2$
Maximum vehicle speed	10 m/s
TLS cycle time	45 s
TLS yellow light duration	2 s
TLS green / red light ratio	0.5
Connection metric	Euclidean distance
Maximum distance LOS	250 m
Maximum distance NLOS	140 m
Simulation time	3000 s
Warm-up time	1000 s
Time resolution	1 s

TABLE 5.1: Consistent simulation parameters

Parameter	Manhattan grid	Real world network
Place	Idealized Manhattan grid	Upper West Side, New York, USA
Network area	4 km^2	4.82 km^2
Intersection density	6.4×10^{-5} $1/\text{m}^2$	7.90×10^{-5} $1/\text{m}^2$
Average road length	125 m	120.86 m
TLS signal offset	10 s	Determined by vehicle routes
Vehicle placement	CA	{SUMO, Uniform distribution}

TABLE 5.2: Differing simulation parameters

5.1.2 Results and Conclusion

From the simulations described in the previous section and subsequent analysis multiple connectivity metrics were derived. We first look at the network connectivity. Figure 5.1 shows the connectivities resulting from our simulation, as well as the ones for the Manhattan grid listed in [9]. We can observe that in the Manhattan grid case, vehicle densities of 80 veh/km^2 result in fully connected networks on average. The relation between vehicle density and average network connectivity is similar in the case where the real street network of Upper West Side has been simulated and vehicles were placed statically. The connectivity is slightly worse at densities

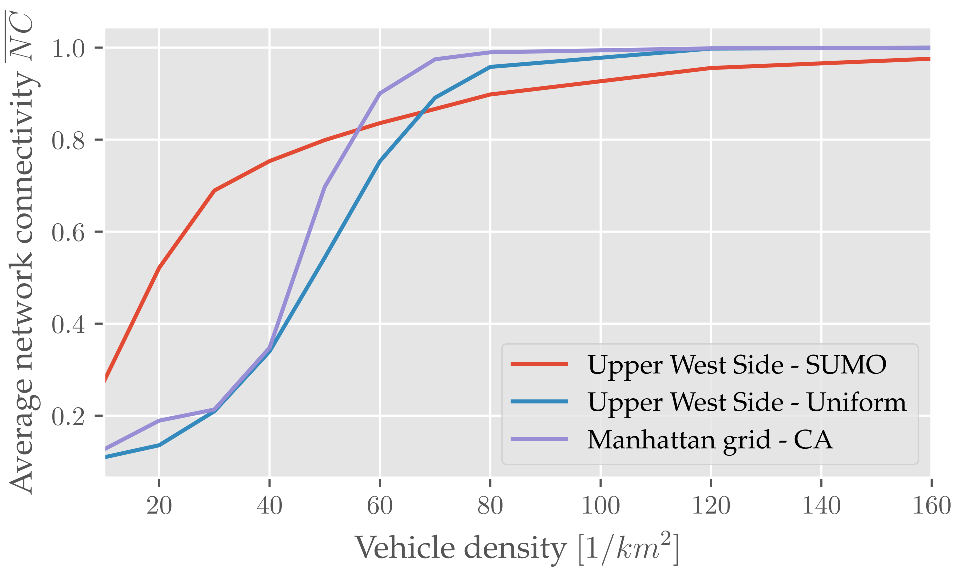


FIGURE 5.1: Network connectivity

higher than $40\ \mathrm{veh/km^2}$ and reaches 96 percent at $80\ \mathrm{veh/km^2}$ while they are fully connected on average only at $120\ \mathrm{veh/km^2}$. The situation is fundamentally different when placing the vehicles via SUMO. While the average network connectivity is higher than in the Manhattan grid at densities lower than $60\ \mathrm{veh/km^2}$, it only grows slowly afterwards and does not reach 100 percent in the simulated density region.

These observations can be explained by essential differences in the two simulation settings. The CA vehicle movement in combination with the Manhattan grid in [9] leads to a highly uniform distribution of cars in the network, similar to the random uniform distribution in the real street network. However, the real street network does not offer the same regular geometrical patterns, resulting in clustering effects of vehicles. There are regions, mainly at intersections on main streets, where there is a high density of vehicles. In other areas, mainly side roads, blind alleys and remote streets, there are very few vehicles residing at any moment. The low density leads to a disconnection from the main clusters and a separation between the main clusters.

The Manhattan grid has an intersection every $125\ \mathrm{m}$, while the LOS range of the vehicles is $250\ \mathrm{m}$. This means a vehicle moving on a street always sees two intersections, while cars currently at an intersection see four. If at every intersection there is one car, the network would already be fully connected. This is the case from a certain threshold density onwards, while for lower densities there will be no large clusters. This explains the very low connectivity at low vehicle densities, the steep increase and full connectivity at higher densities. For the real network there is already a higher average connectivity at low vehicle densities due to large clusters on main streets, however the increase at high vehicle densities is small, due to the vehicles in remote side roads.

In the following we will investigate the average link duration. In the Manhattan grid simulation result [9, Figure 4] we see a nearly constant value of $20\ \mathrm{s}$ over all densities, with only a slight increase with increasing density. Figure 5.2 shows the average unique link duration for different vehicle densities in Upper West Side and Neubau. It exhibits the same behavior of constant averages at low densities. However, with high densities, the average link duration increases substantially in both areas, in Neubau much steeper than in Upper West Side.

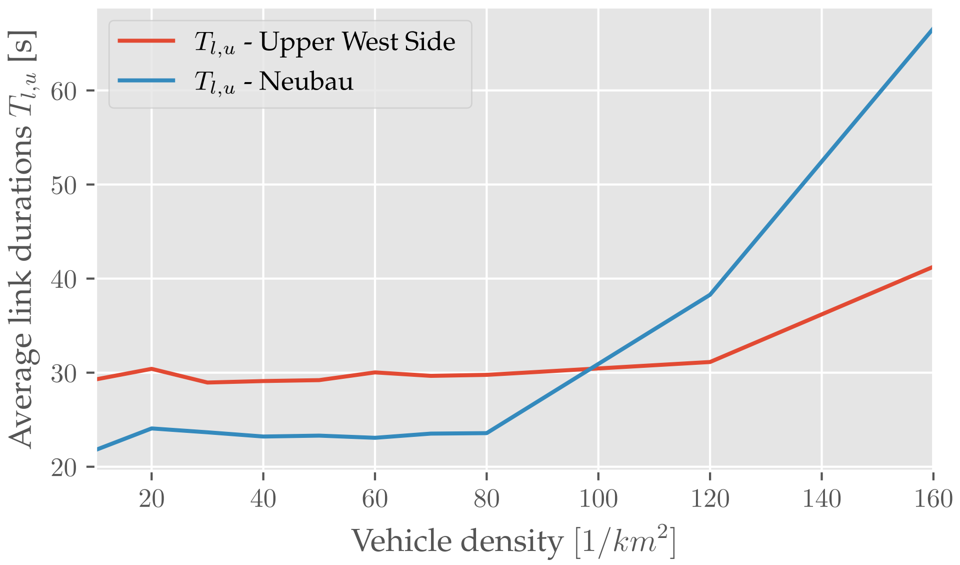


FIGURE 5.2: Unique link duration

The longer link durations with increasing vehicle densities can be attributed to traffic congestion. With higher densities the roads become overcrowded in bottleneck areas, which results in reduced speeds of the vehicles. Therefore, vehicles stay near each other for longer periods which increases their link durations. Due to the regularity of the Manhattan grid, traffic does not concentrate on areas and therefore congestions are not likely to occur with the chosen vehicle densities. This results in an average link duration that is independent of the vehicle density within some bounds.

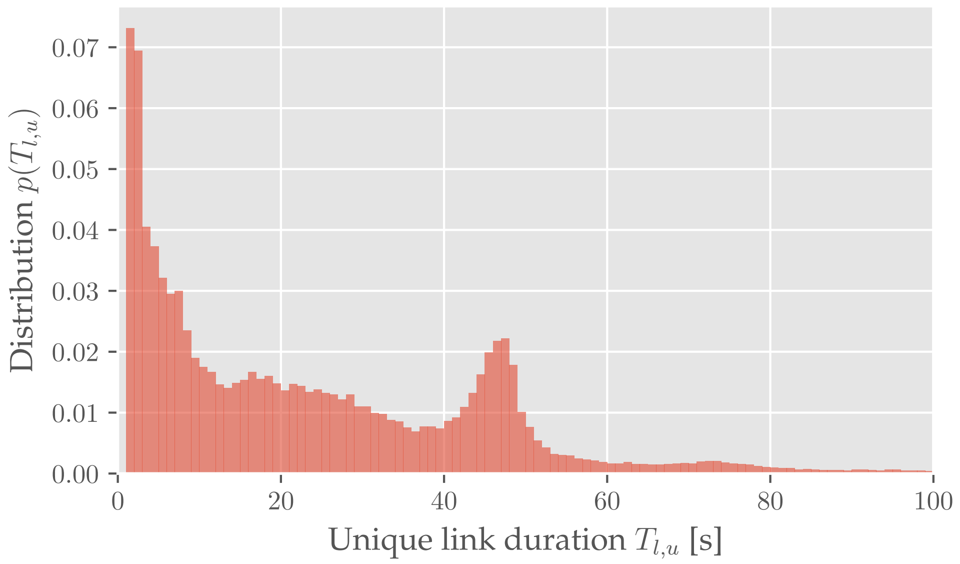


FIGURE 5.3: Unique link duration distribution

We moreover investigate distribution of the link duration. Figure 5.3 shows the unique link duration distribution for Upper West Side and a vehicle density of 70 veh/km^2. It exhibits qualitative similarities to the distribution for the Manhattan grid in [9, Figure 5], both showing a multimodal distribution.

Both distributions have modes at around 20 and 45 s. These values coincide approximately with half of and a full duration of the TLS cycle duration. A potential correlation between the cycle duration and the peaks in the distribution will therefore be investigated further in Section 5.2.

While the average link duration in the real street network is 30 s, nearly 20 percent of the links last for only 3 seconds or less. Due to many short-lived links, the robustness of upper-layer protocols becomes important. These protocols should offer a fast connection establishment and quick routing mechanisms.

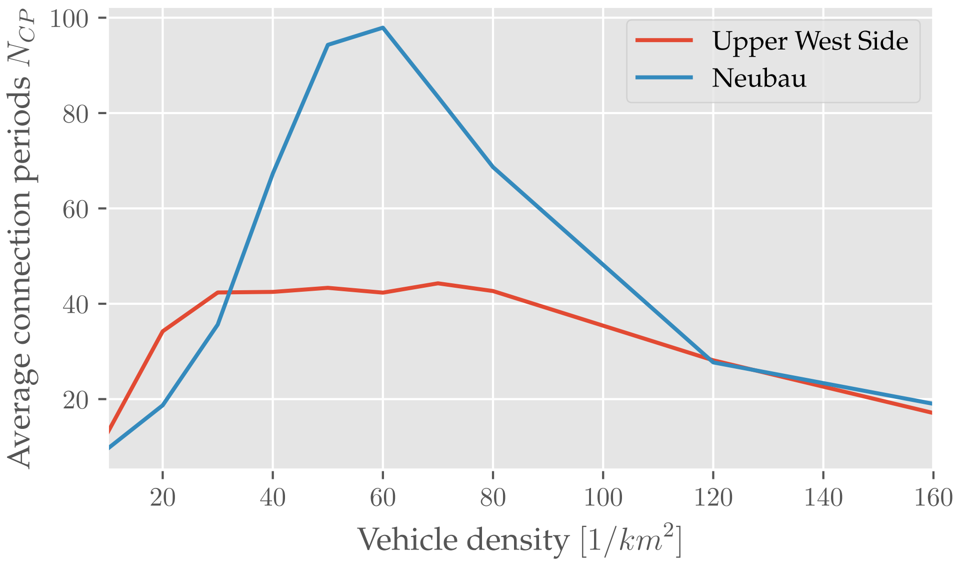


FIGURE 5.4: Number of connection periods

We will now turn our focus from links to connections. Figure 5.4 shows the average number of connection periods. The values have been normalized to the number of pairs of vehicles, to make the results from different networks comparable. As before, a vehicle density of 70 km^2 has been simulated for two areas. Both curves have a peak at approximately 60 $\mathrm{veh/km}^2$, being much more distinctive in the case of Neubau. Qualitatively this coincides with result in the Manhattan grid [9, Figure 6a], where the peak is at 40 $\mathrm{veh/km}^2$.

For low densities, the vehicles are mostly disconnected and seeing only few connections during the whole simulation duration. With increasing density, the number of connection periods also increases. From a certain density onwards, the network becomes well-connected and the connections are therefore long-lived. This results in the number of connections to decrease with rising density. The result is a peak of number of connections at some medium density.

In Figure 5.5 the average connection duration over vehicle densities for the two areas is depicted. These results are compared to the corresponding ones for the Manhattan grid [9, Figure 6b]. Generally the Manhattan grid offers much longer connections on average then the two real street networks. At 80 $\mathrm{veh/km}^2$, the average network connection lasts for 400 s in the Manhattan grid, while in Upper West Side it lasts for 38 s and even less, 23 s in Neubau. However, the dependence of the average duration on the vehicle density in the Manhattan grid and Neubau show striking similarities. In the low density regions (< 40 $\mathrm{veh/km}^2$), the average connection duration is slightly decreasing with increasing density while it shows an exponential increase from 40 to 100 $\mathrm{veh/km}^2$. This differs strongly from results in Upper

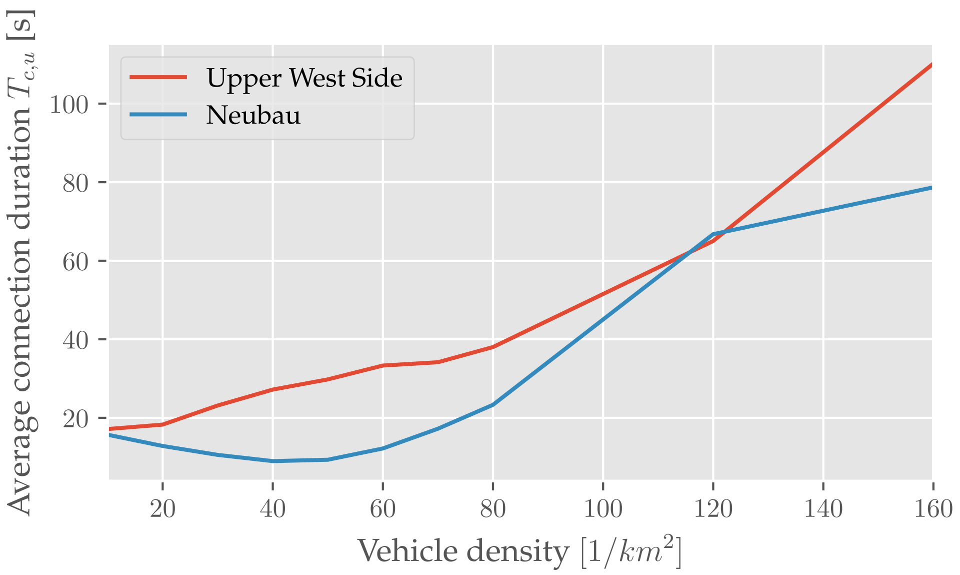


FIGURE 5.5: Unique connection duration

West Side, where the average connection duration grows mostly linearly with the vehicle density.

It is remarkable and seems counterintuitive at first, that at low vehicle densities, the average connection lasts for a shorter time period, than the average link (cf. Figure 5.2). However, this can be explained by the fact, that in a disconnected network a single vehicle that establishes a path to the largest cluster and later disconnects, established only one link, while it established a connection to each vehicle in the cluster. It has therefore a greater effect in the statistic of connections, while having only a small one in the statistics of links.

Regarding the requirements for VANETs it can be deduced from these results, that a very high density of vehicles with vehicle-to-vehicle (V2V) communication equipment is needed in real street networks, to support reliable communication with long average connection durations and few connection periods.

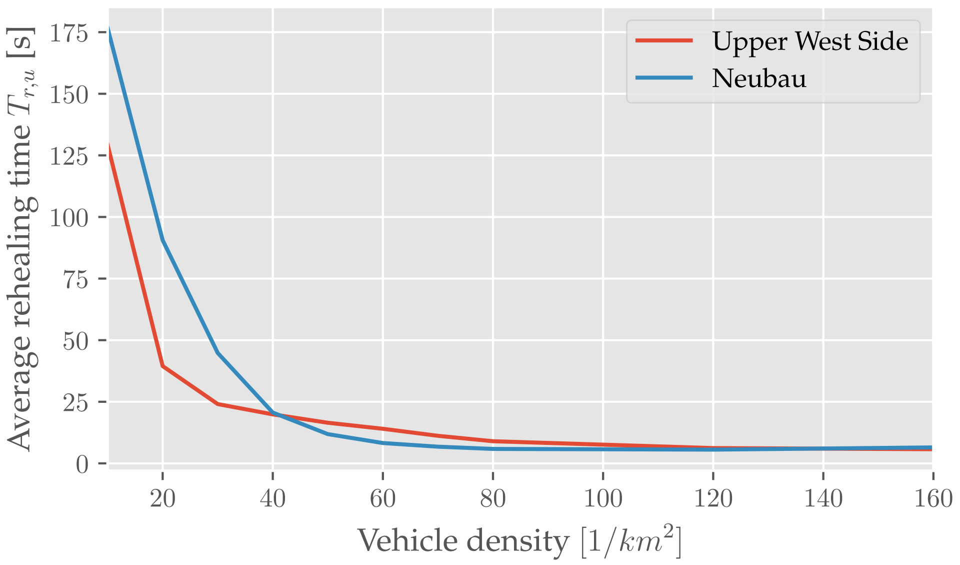


FIGURE 5.6: Unique re-healing time

Finally, we investigate disconnected networks, using the data depicted in Figure 5.6. It shows the average unique re-healing time over vehicle densities. We can again compare these results to the ones for the Manhattan grid in [9, Figure 7].

The curves show a good qualitative agreement, both being exponentially decreasing with rising vehicle density. The values vary drastically and are much higher in real street networks. In the Manhattan grid the mean re-healing time is already below $5\ \mathrm{s}$ at $50\ \mathrm{veh/km^2}$ while in both Neubau and Upper West Side it does not reach this value even in a very dense network of $160\ \mathrm{veh/km^2}$. Additionally, no improvement can be observed in the region with densities higher than $80\ \mathrm{veh/km^2}$, suggesting a minimum re-healing time of approximately $5\ \mathrm{s}$. This can be attributed to the network connectivity never reaching 100 percent in all cases and therefore disconnected vehicles still existing. The higher average re-healing times in real-world street networks increases the importance of effective store-and-carry-forward mechanisms as well as delay tolerance in upper layer protocols.

To summarize the insights from the comparison with the Manhattan grid, it can be stated that using the real-world street networks can reproduce effects that also occur in reality. However, these effects do not manifest themselves during simulations using a Manhattan grid. Therefore, simulations using real street data is always advisable. The Manhattan grid however offers a first estimate in terms achievable performance for the investigated network metrics.

The relevant connectivity metrics were worse in real-world street networks than the results presented in [9] for the Manhattan grid for the same vehicle densities. This means, to achieve a certain average connection duration, link duration or re-healing time, a much higher vehicle density on real world street networks is needed. The results from the Manhattan grid could be used as an upper bound on the achievable performance metrics.

Since for the most metrics, the performance of the real world network is worse than in the Manhattan grid, the requirements to deployments, especially regarding the minimum vehicle density, enhanced. Additionally, conditions to timing parameters of higher layer protocols need to be tightened. The duration of joining a network needs to be low, resulting from short link durations. The routing and discovery mechanism needs to be quick because of short connection durations and many connection periods. Finally, the store-and-carry-forward mechanism needs to support long storage periods, because of increased re-healing times.

5.2 Vehicle Speed and Traffic Light Systems

The vehicle movement model of SUMO allows the setting of a multitude of parameters that either influence the behavior of vehicles directly or indirectly through the underlying street network. It is of interest how these parameters affect the connectivity metrics of a VANET. We will assess the significance of the maximum speed a vehicle can move with and the effect of the cycle time of the TLS on the street network.

5.2.1 Simulation Setup

To determine the influence of the maximum vehicle speed and the TLS cycle time we simulate the area of Upper West Side, New York, USA using three different vehicle densities. The Euclidean distance was used as connection metric, where links have a maximum range of 250 and $140\ \mathrm{m}$ for LOS and NLOS, respectively. The simulations were repeated with two maximum vehicle speeds, 10 and 15 m/s, and two TLS cycle times, 45 and 90 s. The complete set of simulation parameters is listed in Table 5.3.

Parameter	Value
Place	Upper West Side, New York, USA
Vehicle densities	$\{25, 50, 75\}\ 1/\mathrm{km}^2$
Vehicle placement	SUMO
Maximum vehicle speed	$\{10, 15\}\ \mathrm{m/s}$
TLS cycle time	$\{45, 90\}\ \mathrm{s}$
TLS yellow light duration	2 s
TLS green / red light ratio	0.5
Connection metric	Euclidean distance
Maximum distance LOS	250 m
Maximum distance NLOS	140 m
Simulation time	3000 s
Warm-up time	1000 s
Time resolution	1 s

TABLE 5.3: Simulation parameters

5.2.2 Results and Conclusion

We will first investigate the influence of the two parameters on the link duration. We observed that the link duration distribution did not differ significantly for the different vehicle densities. We therefore only show results for 75 $\mathrm{veh/km}^2$, since it has the most possible connections and is therefore subject to statistical fluctuations the least.

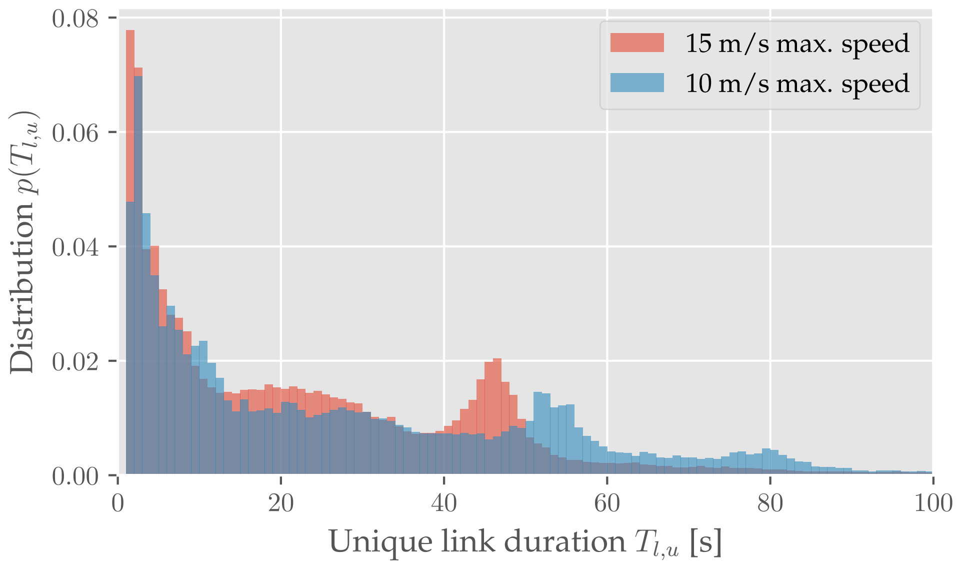


FIGURE 5.7: Unique link duration distribution - maximum speed

The unique link duration distributions for the two different maximum vehicle speeds are depicted in Figure 5.7. Both distributions have a multimodal character and a strong similarity, but the distribution for 10 m/s is shifted to longer durations in regard to the distribution for 15 m/s. This corresponds to the intuitive notion that higher speeds lead to shorter link durations. In the simple scenario representing the case of the largest reduction in link durations, two vehicles are approaching each other. Both travel with the same speed v_l instead of $v_l < v_h$ and have a maximum communication range of d. The link duration then decreases from $d/(2v_l)$ to $d/(2v_h)$ resulting in $\Delta T_l = d(v_h - v_l)/(2v_h v_l)$. Using the same parameters as in the simulation this would yield a reduction of 4.17 s of the link duration. In the best case scenarios, where one vehicle is directly following another or both vehicles are not moving and in range, the maximum speed has no impact on their link duration. The median link duration in the simulations were 22 and 18 s for maximum speeds of 10 and 15 m/s maximum speeds, respectively. The decrease by 4 s corresponds to the largest possible reduction for a single link.

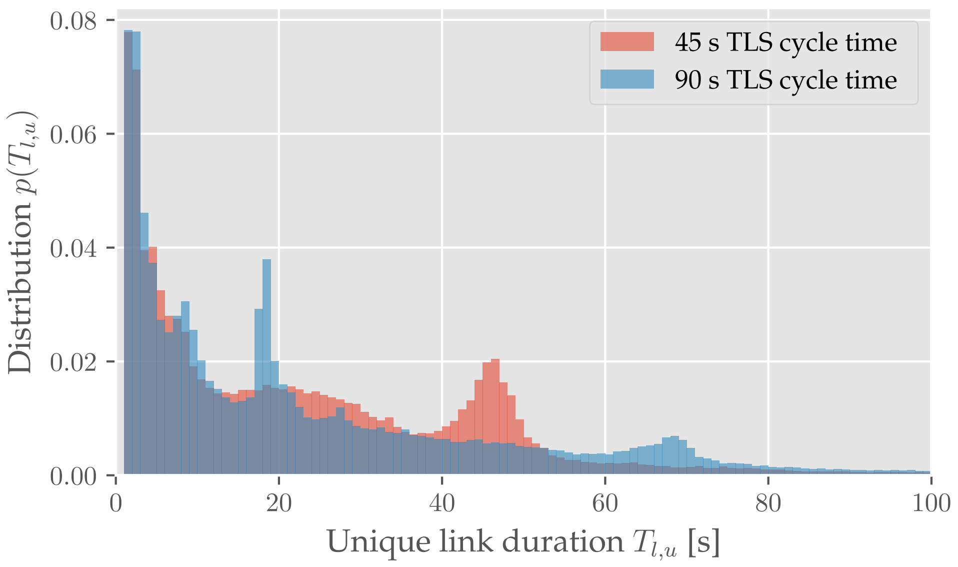


FIGURE 5.8: Unique link duration distribution - TLS cycle time

Figure 5.8 shows the unique link duration distributions for the two different TLS cycle times. Both distributions have a multimodal character but with significant differences. For link durations between 1 and 15 s seconds the distributions are almost identical. The mode at 18 s is much stronger and thinner for 90 s cycles and the mode at 47 s in the 45 s case is shifted to 69 s. We suspect that the second mode results from vehicles approaching each other, while the third mode results from vehicles driving in the same direction. This would explain the shift of the third mode, since the vehicles driving in the same direction have to stop at a red light longer if the cycle time is longer. The simulation result shows that the cycle time strongly influences the link duration characteristics.

In conclusion, it can be stated, that the model parameter choices strongly influence the connectivity results. While increasing the maximum vehicle speed leads to a shifted link duration distribution with shorter links, adapting the TLS cycle time has more complex consequences changing and shifting the modes of the distribution.

5.3 Euclidean Distance vs. Pathloss

The developed simulation framework supports the Euclidean distance and pathloss models to determine if vehicles are connected or not. We want to compare these

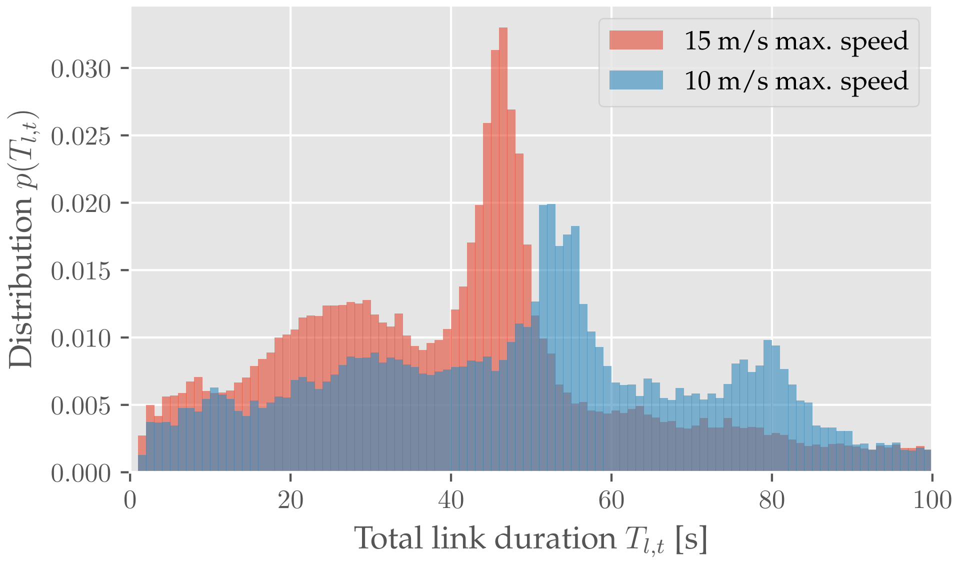


FIGURE 5.9: Total link duration distribution - maximum speed

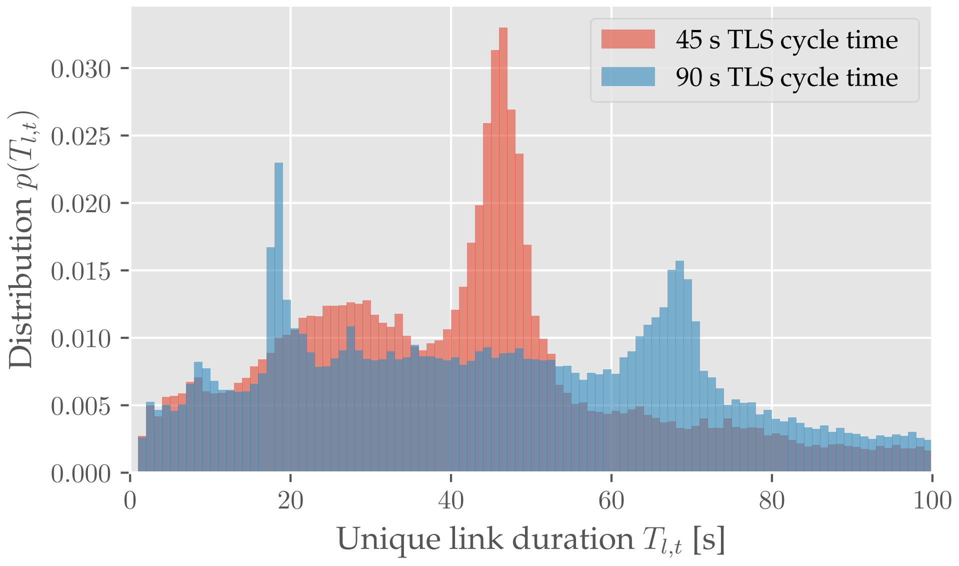


FIGURE 5.10: Total link duration distribution - TLS cycle time

two methods in regard to differences in simulation results and their runtime. If both methods deliver nearly identical results for adequately chosen parameters and the Euclidean distance method is much less computationally expensive, the pathloss models could be used to determine a maximum Euclidean distance, and this distance be used instead in the subsequent simulation.

5.3.1 Simulation Setup

To compare both methods a simulation in the area of Upper West Side, New York, USA has been set up. We arbitrarily chose a maximum pathloss of 100 dB. From the pathloss formulas for LOS and NLOS in Equation (3.13) and Equation (3.15) respectively, we eliminate shadow fading by setting $X_\sigma = 0$ and therefore receiving the average pathloss. We then extract the distances for a pathloss of 100 dB. For the NLOS pathloss we assume that half of the distance is between transmitter and intersection and the other half between intersection and receiver. This method results in a maximum distance of 434.33 and 40.72 meters for LOS and NLOS, respectively.

The simulation has been repeated using the Euclidean distance and pathloss using 100 iterations each time. The whole parameter set is summarized in Table 5.4.

Parameter	Value
Place	Upper West Side, New York, USA
Vehicle densities	$\{10, 20, \ldots, 80, 120, 160\}$ $1/\text{km}^2$
Vehicle placement	Uniform distribution
Iterations	100
Connection metric	{Euclidean distance, Pathloss}
Maximum Euclidean distance LOS	414.33 m
Maximum Euclidean distance NLOS	40.72 m
Maximum pathloss	100 dB

TABLE 5.4: Simulation parameters

5.3.2 Results and Conclusion

To compare the two connection metrics to determine vehicle links, we analyze the network connectivity and the runtime of the simulation. Figure 5.11 shows the average network connectivity for both methods over all the vehicle densities. The two curves show a good agreement, where smaller densities lead to higher differences in average network connectivity, the maximum difference being 7 percent at 30 veh/km^2. For very low and high densities the curves overlap almost perfectly. Generally, the usage of the pathloss model leads to higher average network connectivities than the usage of the Euclidean distance.

Looking at the runtime of the two simulations we get the result that the one using the pathloss takes 48 times longer than the one using the Euclidean distance. This grave difference can be attributed to the much more complex simulation process in the pathloss case. Not only does the simulator additionally need to apply the appropriate pathloss formula, but it also needs to distinguish between obstructed-line-of-sight (OLOS) and LOS and between NLOS between vehicles on parallel and vehicles on orthogonal streets.

The drastically reduced simulation time while maintaining accuracy leads to the conclusion that the method of defining a maximum pathloss, subsequently transforming it to maximum Euclidean distances using the pathloss models and then

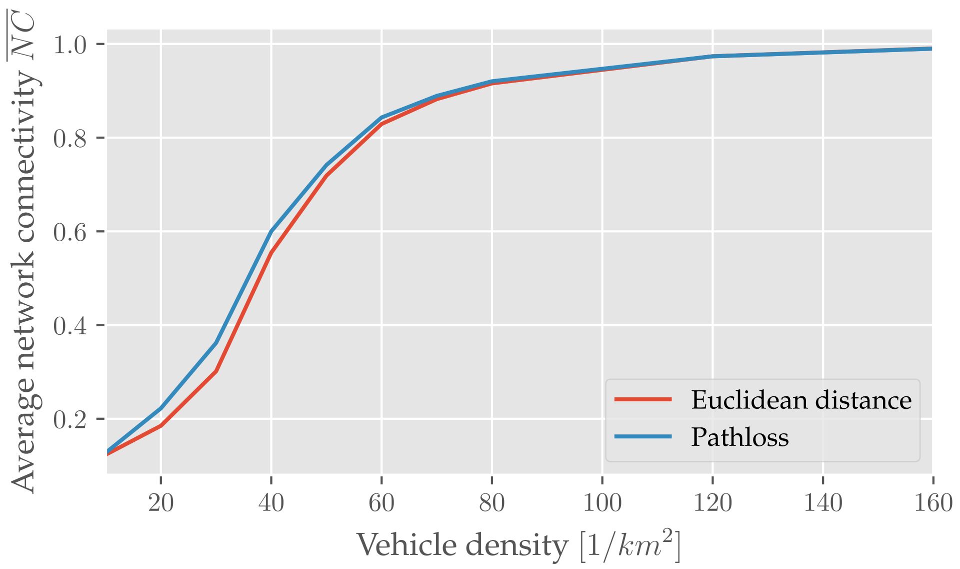


FIGURE 5.11: Network connectivity

using these distances during the simulation is a viable option. Naturally, the error resulting from this simplification, varies for different network connectivity metrics and needs to be assessed beforehand. If an agreement for all metrics is required, directly comparing the resulting connection matrices is advisable.

5.4 Building Simplification

To assess the performance of the building simplification algorithm developed in Section 3.2, we set up a set of simulations, where unmodified building data and simplified building data has been used. The resulting propagation conditions between vehicle pairs and the runtimes where compared for both scenarios.

5.4.1 Simulation Setup

The simulation was conducted using the different areas, Upper West Side, New York, USA and Neubau, Vienna, Austria. The simulations where repeated with a building tolerance of 0 meters, resulting in no simplification, 1 meter and 5 meters.

To generate the same vehicle positions for all building tolerances, the random seed has been kept constant between simulation runs. To cover most of the area by the investigation, the vehicles were uniformly distributed on the streets. Since the simulation framework delivers the link status between vehicle all vehicle pairs as output, and not the propagation condition, the NLOS range has been set to 0 while the LOS range has been set to an extremely high value. From the resulting vehicle connection matrix, the propagation condition can subsequently be derived by interpreting an established link as LOS link and an unestablished link as NLOS.

Additionally, the runtime of the whole simulation was saved, to determine the speed up by using simplified building data. All the simulation parameters are summarized in Table 5.5.

Parameter	Value
Place	{Upper West Side, Neubau}
Vehicle placement	Uniform distribution
Vehicle count	$\{10, 100, 1000\}$
Iterations	100
Building tolerances	$\{0, 1, 5\}$ m
Connection metric	Euclidean distance
Maximum distance LOS	1×10^6 m
Maximum distance NLOS	0 m

TABLE 5.5: Simulation parameters

5.4.2 Results and Conclusion

To describe the accuracy of the resulting propagation conditions using the simplified building data we define use relative error η_{simp}. It is defined as the number of links with different propagation conditions N_{error} in relationship to the total number of links $N_{\text{total}} = \binom{N_{\text{veh}}}{2}$, where N_{veh} is the vehicle count. This relationship can be expressed as $\eta_{\text{simp}} = N_{\text{error}}/N_{\text{total}}$. Furthermore, we define the gain in reduced runtime of the simulation resulting from the use of the simplified building data by δ_{simp}, which is the ratio of the runtime of the simulation using the simplified building data T_{simp} and the runtime of the simulation using the original building data T_{orig}. This can be expressed as $\delta_{\text{simp}} = T_{\text{simp}}/T_{\text{orig}}$. The resulting speedups and errors for the set of simulations are summarized in Table 5.6.

Place	N_{veh}	N_{total}	1 m tolerance η_{simp}	1 m tolerance δ_{simp}	5 m tolerance η_{simp}	5 m tolerance δ_{simp}
Upper West Side	10	45	0	0.329	0	0.363
	100	4950	4.8×10^{-5}	0.255	4.8×10^{-5}	0.244
	1000	499 500	5.7×10^{-5}	0.232	5.7×10^{-5}	0.233
Neubau	10	45	0	0.510	6.12×10^{-2}	0.417
	100	4950	1.18×10^{-4}	0.396	6.70×10^{-2}	0.389
	1000	499 500	7.92×10^{-5}	0.371	6.64×10^{-2}	0.387

TABLE 5.6: Building simplification results

The simulation results for Upper West Side show that a large speed up is possible while keeping the inaccuracy minimal. For the low vehicle count the resulting error is zero, since not enough connections have been simulated, to result in errors. For the two higher vehicle counts the error is in the region o 10^{-5}. The factor of runtime gain ranges from one quarter to one third, decreasing with increasing vehicle count. Additionally, no differences in speedup and error resulted from the two tolerances, 1 and 5 meters. This means that when running simulations in Upper West Side, the computation time can be reduced by up to a factor of 4, while keeping the error minimal.

For the area of Neubau the gain is smaller but still significant. When using 1 meter of building tolerance the factor of runtime gain ranges from 0.5 to slightly below 0.37, while the error ranges between 0 and 10^{-4}. However, when using a building tolerance of 5 meters the error is more than 6 percent, which is already too high to get accurate results. In Neubau, a building tolerance of 1 meters is therefore advisable, resulting in a speedup of factor 2, while achieving a minimal error.

Generally it can be concluded that the building simplification process is a viable approach that reduces the computational complexity of the simulation process while having very limited impact on the simulation result. However, the gain in reduced simulation time and the resulting error are dependent on the simulated place. The building tolerance should be chosen as a trade-off between accuracy and computational complexity.

5.5 Vehicle Distributions at Urban Intersections

Using the simulation framework we want to investigate a potential relation between the distance a vehicle has from the nearest intersection and its nodal degree. The nodal degree is the number of neighbors of a node and is formally defined in Section 4.3. Intuitively, one would assume that in case of uniform distribution of the vehicles, the nodal degree grows with decreasing distance to the intersection. This however, can not be assumed for more complex vehicle distributions, resulting from SUMO usage. Using the Information Bottleneck we want to discretize the road segments into coarse intervals providing each interval's empirical cumulative distribution function (ECDF) of the nodal degree.

5.5.1 Simulation Setup

We investigate the issue by running simulations using the street network of Linz, Austria. Three different vehicle densities will be investigated, $10\ \mathrm{veh/km}$ (low density), $20\ \mathrm{veh/km}$ (medium density) and $50\ \mathrm{veh/km}$ (high density). The large size of the street network paired with higher vehicle densities results in a very high number of possible connections that need to be simulated. This in turn results in long simulation time. To not increase the simulation time further, Euclidean distance instead of pathloss will be used as connection metric. To investigate the impact of vehicle distributions, we repeat the simulation with random uniform distribution and with SUMO snapshots. A comprehensive list of the general and SUMO specific simulation parameters is presented in Table 5.7 and Table 5.8, respectively.

Parameter	Value
Place	Linz, Austria
Vehicle densities	$\{10, 20, 50\}\ \mathrm{veh/km}$
Vehicle placement	{Uniform distribution, SUMO}
Connection metric	Euclidean distance
LOS range	$350\ \mathrm{m}$
NLOS range	$100\ \mathrm{m}$
Building tolerance	$1\ \mathrm{m}$

TABLE 5.7: General simulation parameters

The simulation yields six data sets $S_{\beta,v}$ for all combinations of vehicle densities β and distribution methods v. Each set consists of tuples $t_i = (d_i, n_i)$, one for every vehicle that has been simulated. The tuple consists of the distance d_i of the respective vehicle to the nearest intersection and the nodal degree n_i. The distance is then quantized finely with a resolution of five meters resulting in the discrete distances

Parameter	Value
Warmup duration	1000 s
TLS cycle time	45 s
TLS green / red light ratio	0.5
TLS yellow light duration	2 s
TLS coordination	Deactivated

TABLE 5.8: SUMO specific simulation parameters

d'_i. The estimated joint probability mass function (pmf) can therefore be expressed as

$$p_{\beta,v}(d',n) = \frac{1}{|S_{\beta,v}|} \sum_{t_i \in S_{\beta,v}} \delta(t_i,(d',n)) , \tag{5.1}$$

where δ is Kronecker delta function defined in Equation (2.9).

From the estimated joint pmf we can derive the estimated conditional pmf $p_\beta(n|d')$, that is the probability of a vehicle having n neighbors when being d' away from the nearest intersection. We want to find a coarser quantization $\tilde{d}'$ that has L steps and minimizes the information loss introduced by the quantization. This can be achieved using the Agglomerative Information Bottleneck algorithm introduced in Section 4.7. We used the modified version described by Algorithm 4, that only merges adjacent clusters. The algorithm was aborted after only L clusters remain.

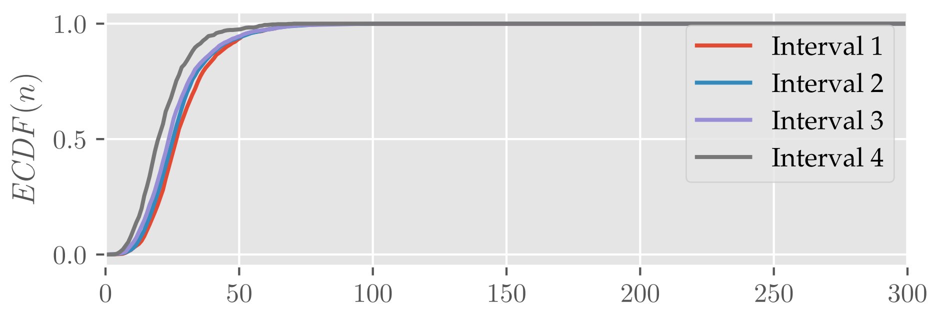
Placement	β veh/km	Boundary 1-2	Boundary 2-3	Boundary 3-4
Uniform	0.01	14 m	57 m	143 m
	0.02	39 m	97 m	251 m
	0.05	31 m	107 m	177 m
SUMO	0.01	29 m	62 m	106 m
	0.02	16 m	56 m	77 m
	0.05	37 m	69 m	123 m

TABLE 5.9: Interval boundaries

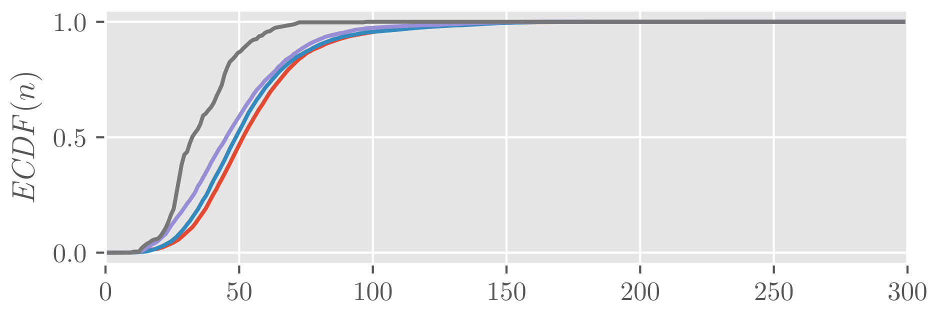
As a trade-off between simplicity of the resulting model and information loss resulting from it, we choose $L = 4$. The resulting distance region boundaries are listed in Table 5.9. For all densities and vehicle placement methods the first region is relatively small, with the largest one being 39 m. This indicates that the region close to the intersection differs in its statistics from the rest and is therefore of special interest. For farther distanced regions, the Information Bottleneck algorithm chooses larger interval sizes with uniform placement than with SUMO. A possible explanation for this is that vehicles in the SUMO case accumulate in the main streets and produce traffic congestion. Therefore, a larger amount of vehicles is encountered at smaller distances. The final interval region in the uniform distribution case is chosen well outside of the NLOS range, while it is very close to the NLOS sensing range in the SUMO case.

5.5.2 Results and Conclusion

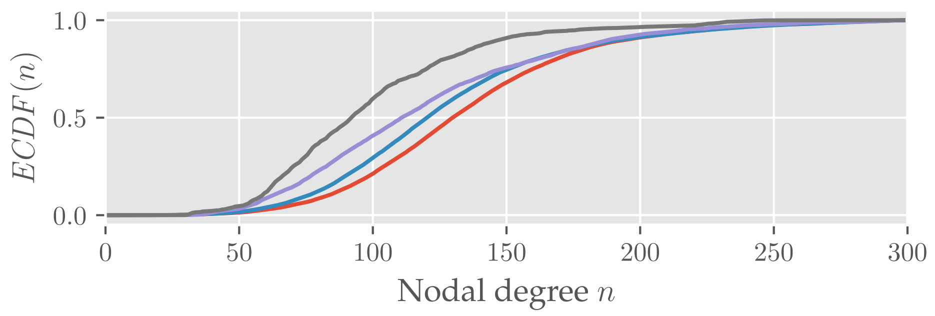
Figure 5.12 and Figure 5.13 show ECDFs of the number of neighbors a node in the VANET has for the static and the dynamic vehicle placement, respectively. They



(A) 10 veh/km

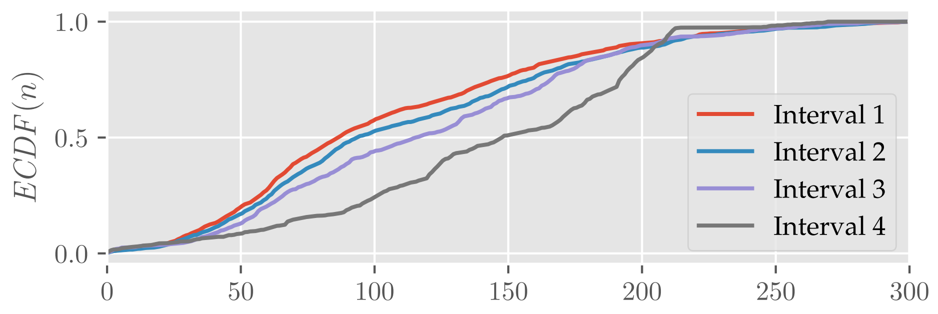


(B) 20 veh/km

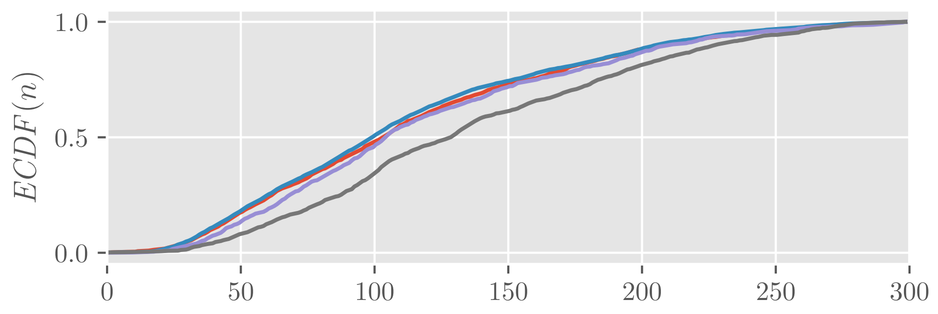


(C) 50 veh/km

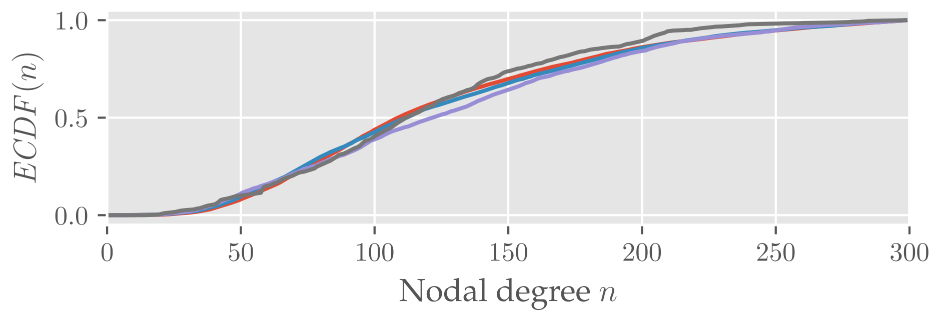
FIGURE 5.12: ECDFs of the nodal degree - static placement



(A) 10 veh/km



(B) 20 veh/km



(C) 50 veh/km

FIGURE 5.13: ECDFs of the nodal degree - dynamic placement

show that few intervals are needed to capture the statistics in dependence of the distance to the next intersection. First we investigate the case were vehicles where placed using a uniform distribution, depicted in Figure 5.12. For low and medium traffic densities the ECDFs of the nearest intervals 1 to 3 are almost perfectly overlapping. However, the farthest region 4 shows a shifted distribution, meaning a vehicle far away from the intersection has on average fewer neighbors. It is therefore important to consider this far-off region in a potential model. For higher densities the effect is similar. Interval 4 again shows a strong offset while intervals 1 to 3 differ only slightly, by at most 15 neighbors for any given probability. The general trend is that vehicles nearer to the intersection always have more neighbors than vehicles farther away. This corresponds to the intuition that nearer vehicles could potentially also connect to vehicles on the intersecting street.

Next we examine the case where SUMO snapshots have been used to determine vehicle positions. The resulting ECDFs are depicted in Figure 5.13. For medium and high densities the lower intervals 1 to 3 overlap almost perfectly and vary only slightly when using a low density. Again, the farthest distant interval 4 shows an offset in regard to the others. The general trend in the SUMO simulation is however completely the opposite as with the uniform distribution. The number of neighbors a vehicle has increases with increasing distance to the intersection. This counter-intuitive behavior can be explained by congestion at intersections. While it is likely that a vehicle at an intersection has only neighbors that are also at or near this intersection, a vehicle currently traveling between two intersections could have neighbors at two intersections. This different trends show the importance of considering distance-dependent behavior when modeling road intersections.

These results show that simple assumptions about the vehicular distributions on street networks can have a large impact on the results and the models derived from them. Additionally, it has been shown that the number of other nodes a vehicle can reach, is dependent on the distance from the nearest intersection. Furthermore, we provided a coarse quantization to capture and model this effect for different scenarios. The static and dynamic placement led to contradictory trends when investigating the ECDF of the number of neighbors. While a uniform random distribution resulted in distributions where the number of neighbors grows with the distance from the intersection, the opposite is the case with SUMO placement.

Chapter 6

Conclusion and Outlook

In this thesis we treated the topic of connectivity in vehicle-to-vehicle (V2V) networks. While the state of the art is to use strictly regular or randomly generated street networks for simulations, we proposed the usage of real-world street networks to achieve a more realistic model. Additionally, we proposed the usage of pathloss models derived from vehicular measurements instead of the Euclidean distance to determine connections between vehicles.

In the course of the thesis we developed a simulation framework that automates the process of downloading street network and building data, running V2V connectivity simulations and the subsequent analysis of simulation results by deriving connectivity metrics. The street and building data is obtained from OpenStreetMap (OSM), which offers public access to map data, that is highly accurate, especially in urban areas. The framework offers the usage of Euclidean distance and pathloss models to determine connections, where the appropriate models are used for line-of-sight (LOS), obstructed-line-of-sight (OLOS) and non-line-of-sight (NLOS) links. To place vehicles two methods were implemented. A simple static placement resulting in a uniform distribution is available, as well as using Simulation of Urban Mobility (SUMO), allowing vehicle movement while considering traffic lights and traffic congestions. The simulation framework has been made public, to allow the reproduction of the results presented in this work, as well as to allow anyone to modify or adapt the software.

In subsequent simulations using the framework, we compared the resulting network metrics of vehicular ad hoc networks (VANETs) in real-world street network with those from a regular (Manhattan) grid. The fundamentally different vehicle distributions in the two scenarios resulted in differing network connectivity metrics. Generally, the simulations using real-world street networks result in worse performance metrics than in the Manhattan grid scenario. These differences prove the importance of using real-world street networks when simulating V2V communication. Additionally, we investigated the impact of the maximum vehicle speed and traffic light period time. The chosen parameter values impacted the connectivity metrics substantially, resulting in shifted distributions for the link duration.

Furthermore, we compared the two different methods to determine links, Euclidean distance and Pathloss. While simulating with pathloss models was much more computationally expensive, the results for the investigated scenario coincided for both cases, where the maximum distances have been set to match the maximum pathloss. This gave rise to the idea to only use a pathloss to determine the respective maximum distances and use these during the simulations.

To reduce the computational complexity of simulations, we developed an algorithm to simplify building data. These simplifications lead to less time spent to determine if a link is LOS or NLOS and therefore reducing the overall simulation time. In simulations, we compared results from original and simplified building data. The results have shown, that the error resulting from the simplification is not significant, while the computational complexity is drastically reduced. Naturally, the performance depends on the chosen tolerance.

Finally, we investigated the relationship between the number of neighbors a vehicle in a VANET has, and its distance to the nearest intersection. We have shown that a uniform random distribution resulted in a growing number of neighbors with decreasing distance to the intersection. When using vehicle movement with SUMO, the opposite association could be observed. This underlines the importance of realistic vehicle placement and movement in simulations.

In conclusion, it can be stated, that the usage of real-world street networks for VANET simulations is suitable to model these communication networks realistically and merits further investigation.

Bibliography

[1] R. Viereckl, D. Ahlemann, A. Koster, *et al.*, "Connected car report 2016: Opportunities, risk, and turmoil on the road to autonomous vehicles", Strategy& PwC, research rep., Sep. 18, 2016. [Online]. Available: https://www.strategyand.pwc.com/media/file/Connected-car-report-2016.pdf.

[2] M. Bertoncello and D. Wee. (Jun. 15, 2015). Ten ways autonomous driving could redefine the automotive world, [Online]. Available: http://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world.

[3] W. G. Najm, J. Koopmann, J. D. Smith, and J. Brewer, "Frequency of target crashes for intellidrive safety systems", U.S. Department of Transportation, research rep., Oct. 1, 2010. [Online]. Available: https://www.nhtsa.gov/DOT/NHTSA/NVS/Crash%20Avoidance/Technical%20Publications/2010/811381.pdf.

[4] R. Lozano, M. Naghavi, K. Foreman, *et al.*, "Global and regional mortality from 235 causes of death for 20 age groups in 1990 and 2010: A systematic analysis for the global burden of disease study 2010", *The Lancet*, vol. 380, no. 9859, pp. 2095–2128, 2013.

[5] "Annual accident report 2016", European Road Safety Observatory, research rep., 2016. [Online]. Available: https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/pdf/statistics/dacota/asr2016.pdf.

[6] J. B. Greenblatt and S. Saxena, "Autonomous taxis could greatly reduce greenhouse-gas emissions of us light-duty vehicles", *Nature Climate Change*, vol. 5, no. 9, pp. 860–863, 2015. DOI: 10.1038/nclimate2685.

[7] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments", *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pp. 1–51, Jul. 2010. DOI: 10.1109/IEEESTD.2010.5514475.

[8] "Ieee standard for telecommunications and information exchange between systems - lan/man specific requirements - part 11: Wireless medium access control (mac) and physical layer (phy) specifications: High speed physical layer in the 5 ghz band", *IEEE Std 802.11a-1999*, pp. 1–102, Dec. 1999. DOI: 10.1109/IEEESTD.1999.90606.

[9] W. Viriyasitavat, F. Bai, and O. K. Tonguz, "Dynamics of network connectivity in urban vehicular networks", *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 515–533, Mar. 2011, ISSN: 0733-8716. DOI: 10.1109/JSAC.2011.110303.

[10] O. K. Tonguz, W. Viriyasitavat, and F. Bai, "Modeling urban traffic: A cellular automata approach", *IEEE Communications Magazine*, vol. 47, no. 5, pp. 142–150, May 2009, ISSN: 0163-6804. DOI: 10.1109/MCOM.2009.4939290.

[11] W. Viriyasitavat, O. K. Tonguz, and F. Bai, "Network connectivity of vanets in urban areas", in *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Jun. 2009, pp. 1–9. DOI: 10.1109/SAHCN.2009.5168949.

[12] M. Fiore and J. Härri, "The networking shape of vehicular mobility", in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '08, Hong Kong, Hong Kong, China: ACM, 2008, pp. 261–272, ISBN: 978-1-60558-073-9. DOI: 10.1145/1374618.1374654.

[13] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2017.

[14] J. Nuckelt, D. M. Rose, T. Jansen, and T. Kürner, "On the use of openstreetmap data for v2x channel modeling in urban scenarios", in *2013 7th European Conference on Antennas and Propagation (EuCAP)*, Apr. 2013, pp. 3984–3988.

[15] M. Boban, J. Barros, and O. Tonguz, "Geometry-based vehicle-to-vehicle channel modeling for large-scale simulation", *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4146–4164, Nov. 2014, ISSN: 0018-9545. DOI: 10.1109/TVT.2014.2317803.

[16] M. Boban, X. Gong, and W. Xu, "Modeling the evolution of line-of-sight blockage for v2v channels", in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Sep. 2016, pp. 1–7. DOI: 10.1109/VTCFall.2016.7881090.

[17] Z. H. Mir and F. Filali, "Simulation and performance evaluation of vehicle-to-vehicle (v2v) propagation model in urban environment", in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, Jan. 2016, pp. 394–399. DOI: 10.1109/ISMS.2016.56.

[18] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks", *CoRR*, vol. abs/1611.01890, 2016. DOI: 10.2139/ssrn.2865501.

[19] T. Abbas, F. Tufvesson, and J. Karedal, "Measurement based shadow fading model for vehicle-to-vehicle network simulations", *International Journal of Antennas and Propagation*, vol. abs/1203.3370, May 17, 2015. DOI: 10.1155/2015/190607.

[20] T. Mangel, O. Klemp, and H. Hartenstein, "5.9 ghz inter-vehicle communication at intersections: A validated non-line-of-sight path-loss and fading model", *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, p. 182, 2011, ISSN: 1687-1499. DOI: 10.1186/1687-1499-2011-182.

[21] H. M. El-Sallabi, "Fast path loss prediction by using virtual source technique for urban microcells", in *VTC2000-Spring. 2000 IEEE 51st Vehicular Technology Conference Proceedings (Cat. No.00CH37026)*, vol. 3, 2000, 2183–2187 vol.3. DOI: 10.1109/VETECS.2000.851659.

[22] I. Stepanov, J. Hahner, C. Becker, J. Tian, and K. Rothermel, "A meta-model and framework for user mobility in mobile networks", in *The 11th IEEE International Conference on Networks, 2003. ICON2003.*, Sep. 2003, pp. 231–238. DOI: 10.1109/ICON.2003.1266195.

[23] F. Bai, N. Sadagopan, and A. Helmy, "The important framework for analyzing the impact of mobility on performance of routing protocols for adhoc networks", *Ad Hoc Networks*, vol. 1, no. 4, pp. 383–403, 2003, ISSN: 1570-8705. DOI: `10.1016/s1570-8705(03)00040-4`.

[24] S. Krauß, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics", PhD thesis, Deutsches Zentrum für Luft– und Raumfahrt, Apr. 6, 1998.

[25] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility", *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, Dec. 2012.

[26] C. Backfrieder, C. F. Mecklenbräuker, and G. Ostermayer, "Traffsim – a traffic simulator for investigating benefits ensuing from intelligent traffic management", in *2013 European Modelling Symposium*, Nov. 2013, pp. 451–456. DOI: `10.1109/EMS.2013.76`.

[27] E. F. Moore, "The shortest path through a maze", in *Proceedings of the International Symposium on the Theory of Switching*, Harvard University Press, 1959, pp. 285–292.

[28] E. W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. DOI: `10.1007/bf01386390`.

[29] R. Bellman, "On a routing problem", *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958. DOI: `10.1090/qam/102435`.

[30] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.

[31] E. Jones, T. Oliphant, P. Peterson, *et al.*, *SciPy: Open source scientific tools for Python*, [Online; accessed 2017-08-03], 2001. [Online]. Available: `http://www.scipy.org/`.

[32] C. F. F. Karney, "Transverse mercator with an accuracy of a few nanometers", *J. Geodesy 85(8), 475-485 (Aug. 2011)*, Feb. 8, 2010. DOI: `10.1007/s00190-011-0445-3`. eprint: `1002.1417v3`.

[33] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973. DOI: `10.1002/9780470669488.ch2`.

[34] A. Molisch, *Wireless Communications*. Wiley-IEEE Press, 2005, ISBN: 047084888X.

[35] K. Menger, "Zur allgemeinen kurventheorie", *Fundamenta Mathematicae*, vol. 10, no. 1, pp. 96–115, 1927. [Online]. Available: `http://eudml.org/doc/211191`.

[36] P. Elias, A. Feinstein, and C. Shannon, "A note on the maximum flow through a network", *IRE Transactions on Information Theory*, vol. 2, no. 4, pp. 117–119, Dec. 1956, ISSN: 0096-1000. DOI: `10.1109/TIT.1956.1056816`.

[37] D. R. White and M. E. J. Newman, "Fast approximation algorithms for finding node-independent paths in networks", *Santa Fe Institute Working Papers Series*, May 9, 2011. DOI: `10.2139/ssrn.1831790`.

[38] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method", *arXiv preprint physics/0004057*, 2000.

[39] S. Kullback and R. A. Leibler, “On information and sufficiency”, *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, Mar. 1951. DOI: `10.1214/aoms/1177729694`.

[40] N. Slonim and N. Tishby, “Agglomerative information bottleneck”, in *Advances in neural information processing systems*, 2000, pp. 617–623.

Declaration of Authorship

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Ort und Datum:

____

Unterschrift:

____

Markus GASSER, BSc