

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Fakultät für Chemie und Geowissenschaften

Geographisches Institut

Abteilung Geoinformatik

Erstellung eines Routing-Profiles für Feuerwehrfahrzeuge auf Basis von OpenStreetMap-Daten

Amandus Stefan Butzer

Bachelorarbeit

Geprüft von:

1. Prof. Dr. Alexander Zipf
2. Prof. Dr. Joao Porto de Albuquerque

13. Oktober 2017

Zusammenfassung

Ziel der vorliegenden Bachelorarbeit war es ein prototypisches Routing-Profil für Löschfahrzeuge in einer Forschungs Kooperation mit der Freiwilligen Feuerwehr Lützelburg (in Schwaben) zu entwickeln.

Bisher gibt es keinen Routing-Service der die Tatsache berücksichtigt, dass Einsatzfahrzeuge im Notfall alle Vorschriften der StVO vernachlässigen dürfen. Auf Basis des Openrouteservice wurde mit OpenStreetMap als Datengrundlage ein Emergency Profil entwickelt, welches für Einsatzfahrzeuge adäquate Geschwindigkeiten verwendet sowie Radwege, Fußgängerzonen und weitere Wegtypen befahren darf. Weiterhin dürfen Einbahnstraßen in Gegenrichtung durchfahren werden.

Die Verzögerung durch Beschleunigungs- und Abbremsvorgänge wurde anhand von Zeit-Penalties für Anfahrt, Abbiegevorgänge und Ankunft berücksichtigt. Das Profil wurde durch Analytische Vergleichen zwischen den Fahrtzeiten von drei durchgeführten Testfahrten kalibriert. Die Ergebnisse zeigen, dass eine Funktion für die Steigung eingebaut werden muss. Die Differenzen der Fahrzeiten für die Minuten Marker der zwei weitgehend ebenen Teststrecken liegen nahezu komplett im 10-Sekunden-Bereich.

Der Autor möchte mit dieser Arbeit eine Grundlage für exaktere Routenfindung und Fahrzeitberechnung für Notfallfahrzeuge sowie Hilfestellung bei der Entwicklung von Bedarfsplänen zu liefern.

Abstract

The goal of the present bachelor thesis was to develop a prototypical Routing-Profile for Fireengines in a research cooperation with the voluntary fire department in Lützelburg (in swabia).

So far there is no routing-service considering that emergency vehicles are allowed to ignore the StVO¹. Based on the Openrouteservice using OpenStreetMap as data pool, an Emergency Profile was developed, which uses reasonable speeds for emergency vehicles and is allowed to drive on cycling-ways, pedestrian zones and additional waytypes. Furthermore oneways can be passed through from the opposite direction. The delay for acceleration and break processes were taken into account with time penalties for starting, turns and arrival. The Profile was calibrated through analytical comparison of three Test runs. The results show, that a function for the steepness has yet to be added. Differences in Travel times for the minute markers of the two mostly level test routes were mostly in 10 second range.

The Author wants to provide a basis for more accurate routing and travel time calculation for emergency vehicles as well as assistance in the development for demand plans.

¹german road traffic regulation

Inhaltsverzeichnis

Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
Abkürzungsverzeichnis	VI
1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	1
2 Theoretische Grundlagen	3
2.1 Graphentheorie	3
2.1.1 Gerichtete Graphen	4
2.1.2 Gewichtete Graphen	5
2.1.3 Bau eines Graphen aus OpenStreetMap Daten	6
2.2 Routing	6
2.2.1 Shortest Path Problem	7
2.2.2 Dijkstra-Algorithmus	7
2.2.3 Speedup Techniken	9
2.3 Isochronen Berechnung	10
2.3.1 Gitterbasierter Ansatz	10
2.3.2 Dreiecksvermaschung	11
2.3.3 Formenbasierter Ansatz	12
3 Methodik	13
3.1 Informations Erhebung	13
3.2 Generierung des Routing-Profiles	15
3.3 Anpassungen Frontend	19
4 Ergebnisse	21
5 Diskussion – Interpretation	26
5.1 Teststrecke 1	26
5.2 Teststrecke 2	27
5.3 Teststrecke 3	27
5.4 Benötigte Änderungen	29

6	Fazit	30
7	Ausblick	31
8	Anhang	33
8.1	Copyrights	33

Abbildungsverzeichnis

1	Ein simpler Graph G	3
2	Zwei isomorphe Graphen G und H	4
3	Ein gerichteter Graph G	5
4	Ein gewichteter Graph G	5
5	Simplifizierung eines OSM Datensatzes	7
6	Ein gewichteter und gerichteter Graph G	8
7	Kompletter Durchlauf eines Dijkstra Algorithmus	9
8	Speedup Techniken für den Dijkstra Algorithmus	10
9	Marching Squares Algorithmus	11
10	EmergencyVehicleAttributesGraphStorage (Speicherobjekt für die Fahrzeugeigen- schaften)	16
11	Entnahme der Dimensionsbeschränkungen (EmergencyVehicleGraphStorageBuilder)	17
12	Vergleich der übergebenen Dimensionen mit den Restriktionen des Graphen(EmergencyVehicleEc)	17
13	Maximalgeschwindigkeiten für unterschiedliche Wegtypen (EmergencyFlagEncoder)	18
14	Limit für 30er Zonen und Spielstraßen(EmergencyFlagEncoder)	19
15	Nutzung von speziellen Wegen(EmergencyFlagEncoder)	19
16	Nutzung von speziellen Wegen(EmergencyFlagEncoder)	20
17	Routing in die Fußgängerzone der Heidelberger Altstadt	21
19	Ergebnis Isochronen	22
20	Testfahrt 1	23
21	Testfahrt 2	24
22	Testfahrt 3	25
23	Steigung und Höhenprofil der ersten Teststrecke	26
24	Fahrt 2 – Unterführung	27
25	Fahrt 3 – Tempo-30-Zone	28
26	Fahrt 3 – Vergleich eines Turns	29
28	Fahrt 3 – Höhenprofil	29

Tabellenverzeichnis

1	W_s und K_s für jede Iteration aus Abb. 7	9
2	Standardgeschwindigkeiten für unterschiedliche highway Values	14
3	Standardgeschwindigkeiten für unterschiedlichen tracktype Values	14
4	Standardgeschwindigkeiten für spezielle Wegtypen	15
5	Fahrt 1 – 1. Auswertung	23
6	Fahrt 1 – Fahrzeit	24
7	Fahrt 2 – Fahrzeit	24
8	Fahrt 3 – Fahrzeit	25
9	Fahrt 1,2,3 – 2. Auswertung	25
10	Fahrt 3 – neue Fahrzeit	28

Abkürzungsverzeichnis

APSP All Pairs Shortest Path Problem 7

FFL Freiwilligen Feuerwehr Lützelburg 1, 2, 13

GIS Geoinformationssysteme 6

GUI Graphical User Interface 19

ORS Openrouteservice 1, 2, 12, 13, 33

OSM OpenStreetMap 6, 13, 17

SPP Single Pair Problem 7, 8

SSP Single Source Problem 7, 8, 10, 11

TSP Traveling Salesman Problem 7

1 Einleitung

1.1 Motivation

Zum Zeitpunkt der Erstellung dieser Arbeit ist der Verfasser in der Geoinformatik Abteilung des Geographischen Instituts der Ruprecht-Karls-Universität Heidelberg als wissenschaftliche Hilfskraft im Projekt Openrouteservice (ORS) tätig. Dieser bietet neben einer Fülle von Webdiensten, wie Geocoding und Routing, auch einen Isochronen-Service an. Mit Isochronen werden Orte bestimmt, die von einem Standort aus in einer bestimmten Zeit erreicht werden können. Für Unternehmen ist so eine Analyse zum Beispiel bei der Standortwahl zur Berechnung von Umsatzpotentialen interessant. Arbeitnehmer können über Isochronen geeignete Wohnorte für eine zukünftige Arbeitsstelle ermitteln. Bus und Bahn richten an Isochronen ihre Verkehrsnetze aus oder legen Tarifzonen damit fest. Für Polizei, Rettungsdienst sowie Feuerwehr geht es vor allem um das Einhalten amtlich vorgegebener zeitlicher Hilfsfristen. Diese stellen eine bedeutende Eigenschaft für die Planung und Qualität der Einsätze von Feuerwehr und Rettungsdienst dar.

Der Brandschutz ist im Gegensatz zum Rettungsdienst eine kommunale Aufgabe und unterliegt nur in manchen Bundesländern bestimmten Standards (Stein 2015). Daher bedienen sich diese Organisationen unterschiedlicher Hilfsmittel um Bedarfspläne für ihren Standort zu erstellen.

Da mit dem Isochronen-Service jene zeitlichen Erreichbarkeitsanalysen durchgeführt werden können, ist dieser für die Erstellung eines Brandschutzbedarfsplans der Feuerwehr geeignet. Jedoch kann der Dienst in seiner bisherigen Form noch nicht alle erforderlichen Anforderungen für Einsatzfahrzeuge erfüllen. Um diese Nachteile zu überwinden, wurde eine Forschungskooperation mit der Freiwilligen Feuerwehr Lützelburg (FFL) eingegangen.

§35 Abs. 1 StVO:

”Von den Vorschriften dieser Verordnung sind die Bundeswehr, die Bundespolizei, die Feuerwehr, der Katastrophenschutz, die Polizei und der Zolldienst befreit, soweit das zur Erfüllung hoheitlicher Aufgaben dringend geboten ist.”

Dieser kurze Absatz der Straßenverkehrs-Ordnung ermöglicht Einsatzfahrzeugen sich unter Benutzung von Martinshorn und Blaulicht über jede Vorschrift im Straßenverkehr hinwegzusetzen. In einem Notfall hat das schnellste Erreichen des Zielorts eine höhere Priorität als Geschwindigkeitsbegrenzungen oder Fahrverbote. Bisher gibt es trotz einer großen Anzahl an Routing-Services keinen, der diese Tatsache berücksichtigt.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist in Kooperation mit der FFL zu Ermitteln, bis zu welchem Grad diese Notstandsvollmachten im Ernstfall in Anspruch genommen werden können. Auf Ba-

sis dieser Informationen soll dann ein prototypisches Routing-Profil für Einsatzfahrzeuge entwickelt werden. Die Implementierung wird aufgrund des Umfangs einer Bachelorarbeit auf einen Fahrzeugtyp der Feuerwehr begrenzt. Dabei handelt es sich um Löschfahrzeuge der Klassen LF8, LF8/6 und MLF zwischen 3,5 und 7,5 Tonnen (je nach Beladung). Diese Typen wurden von der FFL als erste Priorität empfohlen. Allerdings wird bei der Erstellung dieses Routing-Profils darauf geachtet, dass Erweiterungen für diverse Einsatzfahrzeuge einfach möglich sind.

Als Basis wird das Profil auf dem Backend² des bereits bestehenden Routing Service des ORS aufgebaut. Zusätzlich sollen Java Funktionen implementiert werden, die speziell auf das Emergency Profil zugeschnitten sind. Zur Darstellung wird das ORS Frontend³ mit Hilfe der Java-Script Programmiersprache angepasst. Dadurch können die Ergebnisse in verständlicher und anschaulicher Weise präsentiert werden.

²Programm und Datenbanken die sich zur Berechnung von Routen auf einem über das Internet ansprechbaren Server befinden

³Graphische Benutzeroberfläche der ORS-Website mit der Anfragen an das Backend gestellt und die Antworten dargestellt werden können

2 Theoretische Grundlagen

Als Grundlage für die Berechnung kürzester Wege in einem Straßennetzwerk wird die Graphentheorie als Teilgebiet der Mathematik herangezogen.

2.1 Graphentheorie

Ein Graph ist eine abstrakte Struktur, die Objekte und deren Verbindungen untereinander modellieren kann. Die in der Graphentheorie verwendeten Termini belaufen sich dabei auf Ecken (engl: *nodes* oder *vertices*) für Objekte und Kanten (engl: *edges*) für Verbindungen. (Kurt Mehlhorn 2008, S. 49)

Mathematisch ausgedrückt ist ein Graph G die Funktion aus einer endlichen Eckenmenge V und einer endlichen Kantenmenge E (Aigner 2015, S. 4)

$$G = (V, E)$$

Für den Graphen in Abbildung 1 sehen V und E folgendermaßen aus:

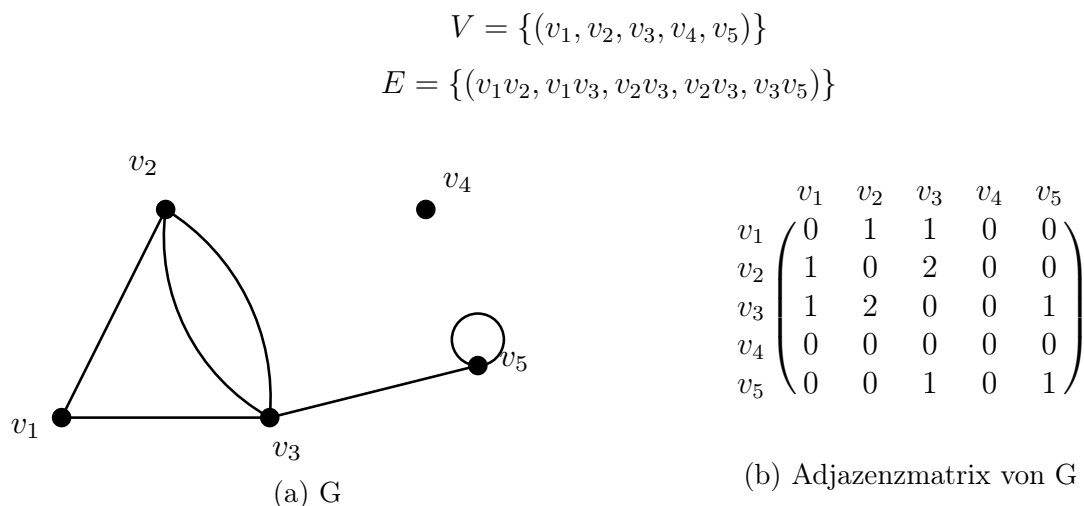


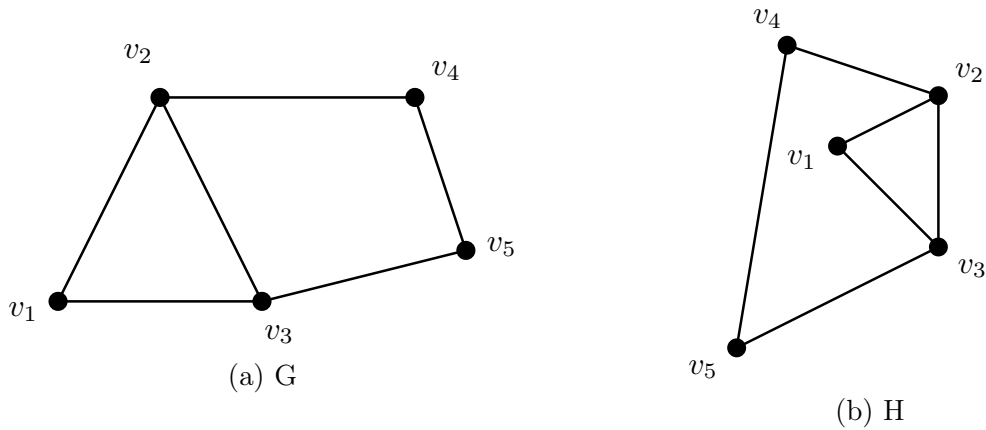
Abbildung 1: Ein simpler Graph G

Ein Vorteil von Graphen ist eine einfache Struktur. Dabei werden die Ecken als Punkte und die Kanten als Linien oder Pfeile dargestellt (Abbildung 1a) (Kurt Mehlhorn 2008, S. 49). Zwischen zwei Ecken können einfache, mehrfache oder keine Kanten bestehen. Darüber hinaus können sie mit sich selbst verbunden sein und eine sogenannte Schlinge bilden (v_5). Sind zwei Ecken durch eine Kante verbunden bezeichnet man sie als *adjazent* (benachbart). Ist eine Ecke der Start- oder Endpunkt einer Kante, werden beide Objekte als *inzident* zueinander bezeichnet. Ist eine Ecke zu keiner Kante inzident heißt sie *isoliert*. Ein Graph der keine isolierten Ecken besitzt heißt *zusammenhängend*. (Aigner 2015, S. 4 f.)

Computer können Graphen sehr gut verarbeiten, da sich alle Ecken und Kanten in Form von Matrizen speichern lassen. Die Abbildung 1b zeigt die Adjazenzmatrix des Graphen aus 1a. Darin sind die Nachbarschaften für die jeweilige Eckenkombination gespeichert. Die Reihen sind die Start- und die Spalten die Enden. Für eine existierende Kante

wird eine 1 und für keine Verbindung eine 0 eingetragen. Die 2 in Spalte v_2 und Reihe v_3 zeigt die Mehrfachkante zwischen den beiden Ecken an. Für ungerichtete Graphen ist eine Kante von v_1 nach v_2 äquivalent mit einer Kante von v_2 nach v_1 . Daher ist die Adjazenzmatrix für ungerichtete Graphen spiegelsymmetrisch entlang der Hauptdiagonale ($v_1v_1 \rightarrow v_5v_5$). Der Speicherbedarf für eine Matrix wird folglich halbiert, da die eine Hälfte mit der Anderen rekonstruiert werden kann (Abb. 2c). Besitzt der Graph keine Schlingen, besteht die Hauptdiagonale lediglich aus Nullen und kann ebenfalls eingespart werden (Sven Oliver Krumke 2012, S. 19).

Im Folgenden werden nur zusammenhängende Graphen ohne Schlingen betrachtet, da diese für viele Problemstellungen irrelevant sind (Aigner 2015, S. 4 f.).



$$\begin{array}{c}
 v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \\
 \begin{pmatrix}
 \ddots & 1 & 1 & 0 & 0 \\
 & \ddots & 1 & 1 & 0 \\
 & & \ddots & 0 & 1 \\
 & & & \ddots & 1 \\
 & & & & \ddots
 \end{pmatrix}
 \end{array}$$

(c) Adjazenzmatrix von G und H

Abbildung 2: Zwei isomorphe Graphen G und H

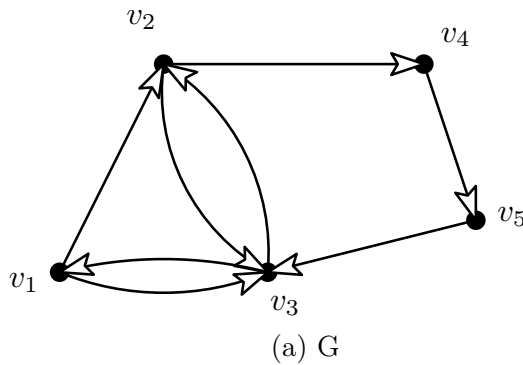
Wenn zwei Graphen bei gleichbleibenden Nachbarschaften der Ecken aufeinander abgebildet werden können, spricht man von isomorphen Graphen (Aigner 2015, S. 106). Daraus ergibt sich für isomorphe Graphen auch immer eine gleiche Adjazenzmatrix (Abb. 2).

2.1.1 Gerichtete Graphen

Im Gegensatz zu einem ungerichteten Graphen können bei einem gerichteten Graphen Kanten lediglich in einer Richtung durchlaufen werden. Die Kanten werden daher durch Pfeile anstatt Linien dargestellt.

$$G = (V, R)$$

Demnach muss in Abbildung 3a um von v_5 nach v_4 zu gelangen der Weg über die Ecken v_3 und v_2 führen.



$$\begin{matrix} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{matrix}$$

(b) Adjazenzmatrix von G

Abbildung 3: Ein gerichteter Graph G

2.1.2 Gewichtete Graphen

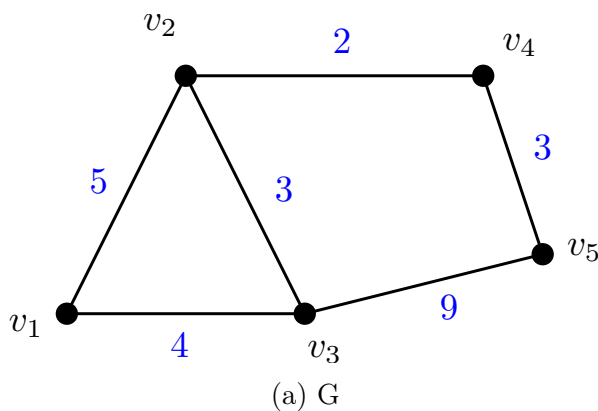
In dieser Arbeit bezeichnet der Begriff *gewichteter Graph* einen Kanten-gewichteten Graphen, bei dem jeder Kante ein Wert c zugewiesen wird.

$$G = (V, E)$$

mit

$$c : E \rightarrow \mathbb{R}$$

Neben dem Kanten-gewichteten gibt es auch Knoten-gewichtete Graphen, bei welchen entsprechend die Knoten gewichtet werden. Diese werden aber nur für wenige Problemstellungen benötigt und sind hier nicht von Belang. Gewichtete Graphen können gerichtet und ungerichtet sein. Ein klassisches Beispiel hierfür ist der Linien-Netzplan einer Straßenbahn, bei dem die Knotenpunkte einzelne Haltestellen darstellen und die Kantengewichte die benötigten Minuten beinhalten.



$$\begin{matrix} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{pmatrix} 0 & 5 & 4 & 0 & 0 \\ 5 & 0 & 3 & 2 & 0 \\ 4 & 3 & 0 & 0 & 9 \\ 0 & 2 & 0 & 0 & 3 \\ 0 & 0 & 9 & 3 & 0 \end{pmatrix} \end{matrix}$$

(b) Adjazenzmatrix von G

Abbildung 4: Ein gewichteter Graph G

Mit gewichteten Graphen können diverse Problemstellungen gelöst werden, zum Beispiel die Bestimmung maximaler (Durch-)Flüsse in Rohrsystemen oder das Berechnen kürzester

Wege. Andererseits kann ein gewichteter Graph auch für Routing-Zwecke eingesetzt werden, indem räumliche Positionen sowie Eigenschaften von Straßen in der Datenstruktur eines Graphen gespeichert werden.

2.1.3 Bau eines Graphen aus OpenStreetMap Daten

Die OpenStreetMap (OSM) Datenstruktur kann als Graph abgebildet werden. Hier werden Punktobjekte als *Nodes*(Knoten) und Linienobjekte wie Straßen als *Ways*(Wege) bezeichnet. Ein Way ist dabei die Verbindung zwischen zwei Nodes. Zusätzlich gibt es *Relations*(Relationen) die einem Set aus Nodes und Ways einen funktionalen Zusammenhang zuschreiben. Für Straßennetze kann dies durchaus hilfreich sein um beispielsweise unterschiedliche Segmente eines Autobahnsegmentes zusammenzufassen oder um Abbiegebeschränkungen an Kreuzungen zu beschreiben (Contributors 2015).

Um aus den OSM Daten einen routingfähigen Graphen zu erhalten, müssen zuerst alle benutzbaren Nodes und Ways extrahiert werden um sie dann anhand ihrer OSM-Tags zu identifizieren. Dazu gehören alle Arten von Straßen und Wegen sowie als befahrbar gekennzeichneten Ways (zum Beispiel asphaltiert, allerdings ohne Straßentyp). Für das Routing sind vor allem Verbindungspunkte wie Kreuzungen, Ab- und Auffahrten etc. sowie Sackgassen interessant. Daher werden diese *Tower Nodes* aus den importierten Daten ermittelt. Anschließend werden die Straßen anhand der Verbindungspunkte segmentiert. Danach werden die Verbindungen zwischen den Tower Nodes berechnet und anhand der Distanz gewichtet. Das Grundgerüst des eigentlichen Routing-Graphen ist damit erstellt. Einbahnstraßen und Abbiegebeschränkungen werden berücksichtigt und geben die Richtung der Edges an (Rehrl u. a. 2012). Die Punkte zwischen zwei Tower Nodes werden *Pillar Nodes* genannt. Sie werden als *WayGeometry* auf der jeweiligen Edge gespeichert, da sie nicht für den Routing Vorgang benötigt werden (Abbildung 5). Das Routing ist dadurch um ungefähr das 8-fache schneller (Karich 2016). Relevante Attribute wie Geschwindigkeit oder Straßentyp werden vereinheitlicht und als *Flags* (Markierungen) auf der Edge gespeichert. Diese sind für die individuelle Gewichtung bei der Routenfindung interessant. (Mehr zu Gewichtung später in Kapitel 3.2). Zuletzt wird der Graph abgespeichert und ist für Routing Abfragen bereit (Rehrl u. a. 2012).

2.2 Routing

Routing bezeichnet den Vorgang in einem Netzwerk Wege zu finden, auf denen Datenpakete entlang gesendet werden können. Diese Definition bezieht sich vor allem auf elektronische Datennetzwerke wie das Telefonnetz oder das Internet. Im Fachbereich der Geoinformationssysteme (GIS) werden hauptsächlich Straßennetze für Routing-Analysen verwendet (Wolfgang Kresse 2012, S. 165). Ein Weg P (*engl: path*) von einem Startecke s zu einem Zielecke z ist eine Folge von benachbarten Ecken mit s als erster Ecke und z als letzter Ecke der Folge (Abb). Die Weglänge entspricht in einem gewichteten Graphen der Summe aller Kantengewichte.

Eine der wichtigsten Netzwerk-Analyse-Operationen ist die Berechnung des kürzesten Weges zwischen zwei Knoten. Jedes Navigationssystem muss diese Aufgabe erfüllen können. Ein kürzester Weg hat demnach die Eigenschaft, dass die Summe aller Kantengewichte, in anderen Worten die Kosten des Weges, minimal gegenüber allen anderen Wegen im Graphen ist.

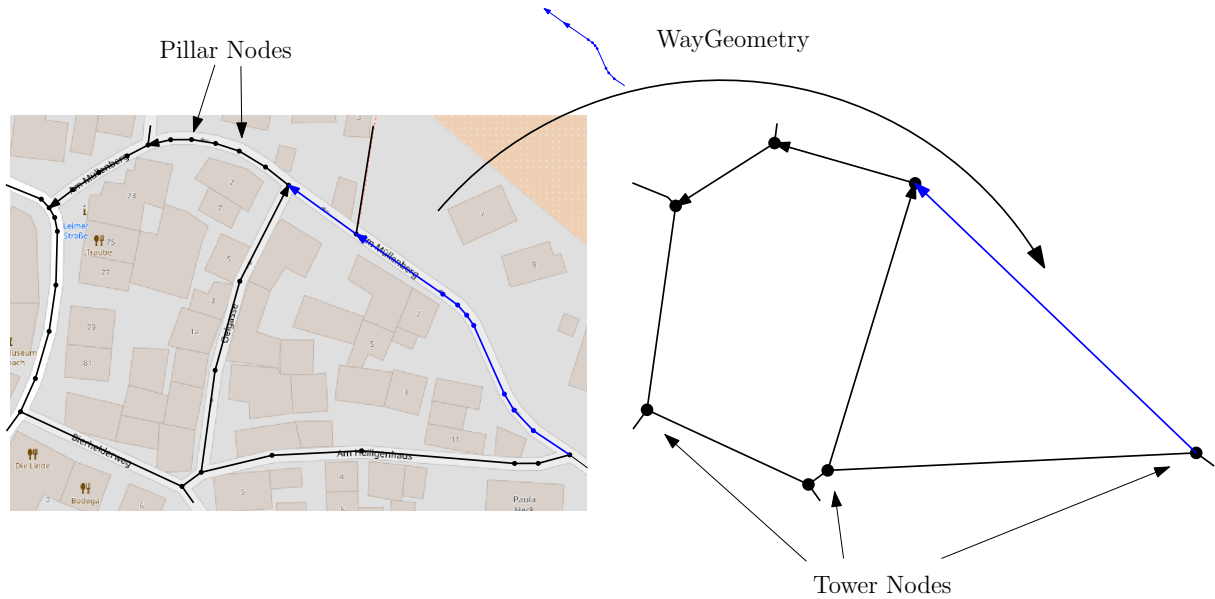


Abbildung 5: Simplifizierung eines OSM Datensatzes

2.2.1 Shortest Path Problem

Die nächstliegende zu beleuchtende Problemstellung ist das *Shortest Path Problem* welches aber nicht mit dem *Traveling Salesman Problem (TSP)* verwechselt werden sollte. Beim TSP ist die kürzeste Tour auf einem Graphen $G = (V, E) = K_n^4$ mit der Gewichtsfunktion $c : E \rightarrow \mathbb{R}_+$ gesucht, die jeden Knoten des Graphen besucht (Sven Oliver Krumke 2012, S. 135). Das Shortest Path Problem lässt sich in drei Typen untergliedern denen jeweils ein gerichteter Graph $G = (V, E)$ mit der Gewichtsfunktion $c : E \rightarrow \mathbb{R}$ zugrunde liegt. Beim *Single Pair Problem (SPP)* wird der kürzeste Weg von einer Ecke a zu einer anderen Ecke b mit $a, b \in V$ gesucht. Das *Single Source Problem (SSP)* möchte den kürzesten Weg einer Ecke a zu jeder anderen Ecke ermitteln (Formel ?). Das *All Pairs Shortest Path Problem (APSP)* sucht den kürzesten Weg von jeder Ecke zu jeder anderen Ecke in V (Sven Oliver Krumke 2012, S. 169 f.).

Für eine Route von Startecke s zur Zielecke z ist das SPP demnach die richtige Wahl. Allerdings wird dazu das SSP herangezogen, welches mit dem Algorithmus von Dijkstra gelöst werden kann.

2.2.2 Dijkstra-Algorithmus

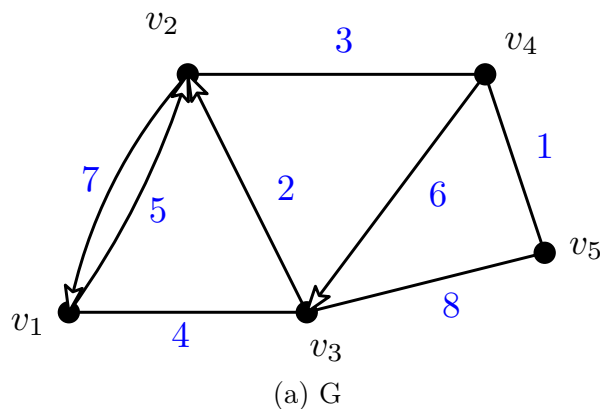
Der in 1959 von Edsger W. Dijkstra entwickelte Algorithmus (Dijkstra 1959) benötigt einen gewichteten Graphen ohne negative Kantengewichte⁵ sowie eine Startecke $s \in V$. Es gibt eine Warteliste W_s mit unmarkierten *gesichteten* Ecken. Dort sind für alle v die Kosten für den bisher kürzesten Weg von s und die jeweilige vorangehende Ecke

⁴ K_n bezeichnet einen vollständigen Graphen bei dem jede Ecke aus V mit jeder anderen Ecke verbunden ist

⁵Das Problem bei Graphen mit negativer Gewichtung entsteht, wenn diese auf einer Schlinge oder der Kante eines Rings liegen. Sobald der Algorithmus den Zyklus erreicht, werden die Kosten für die Ecken des Zyklus immer geringer. Die Kosten für den kürzesten Weg nähern sich $-\infty$ während der Algorithmus in einer endlos Schleife läuft. Deswegen ist der Dijkstra nur für positive Kantengewichte anwendbar (Kurt Mehlhorn 2008, S. 194 f.)

auf diesem Weg gespeichert. Die Kosten können sich noch ändern wenn ein noch kürzerer Weg gefunden wird. Diese Liste enthält zu Beginn nur den Startpunkt s mit den trivialen Kosten 0. Es gibt eine weitere Liste der endgültig kürzesten Wege K_s in der alle markierten Ecken gespeichert werden.

Dijkstra's Algorithmus markiert die Ecke mit den geringsten Kosten aus W_s und verschiebt diese nach K_s . Nun werden alle benachbarten Ecken gesichtet und die Kosten berechnet. Die Kosten und der Vorgänger werden in W_s gespeichert. Die Ecke mit den geringsten Kosten wird als nächstes markiert, da der Weg dorthin auf jeden Fall ein kürzester ist. Dieser Vorgang wird wiederholt bis alle Ecken aus W_s markiert wurden und W_s somit leer ist.



	v_1	v_2	v_3	v_4	v_5
v_1	0	5	4	0	0
v_2	7	0	0	3	0
v_3	4	2	0	0	8
v_4	0	3	6	0	1
v_5	0	0	8	1	0

(b) Adjazenzmatrix von G

Abbildung 6: Ein gewichteter und gerichteter Graph G

Die Funktionsweise des Dijkstra Algorithmus wird am Graphen aus Abbildung 6 mit der Startecke $s = v_1$ veranschaulicht.

Die erste Ecke der Warteschlange ist $s = v_1$. Da s der einzige Eintrag ist, sind die Kosten automatisch am geringsten. s wird von W_s nach K_s verschoben, markiert und die erreichbaren Ecken gesichtet. Das sind v_2 mit den Kosten 5 und v_3 mit den Kosten 4. Beide werden mit v_1 als Vorgänger in W_s eingetragen. Im zweiten Durchgang wird v_3 (geringste Kosten in W_s) markiert und nach K_s verschoben. Über v_3 erreichbare Ecken sind v_5 mit $4 + 8 = 12$ Kosten und v_2 mit $4 + 2 = 6$ Kosten. Da bereits ein kürzerer Weg nach v_2 besteht wird der aktuelle Pfad über v_3 verworfen. Als nächstes wird v_2 nach K_s verschoben. Die einzige neue von dort gesichtete Ecke ist v_4 mit 8 Kosten. Bei der vierten Wiederholung wird v_4 ($8 < 12$) markiert. Es werden v_3 und v_5 gesichtet. Für v_3 besteht bereits ein kürzerer Weg. Für v_5 ist der neue Weg über v_4 mit den Kosten von 9 allerdings kürzer als der Weg über v_3 . v_5 wird aktualisiert und der längere Pfad verworfen. Im fünften Durchgang wird v_5 als letzte Ecke in W_s nach K_s verschoben. Von dort sind keine neuen Ecken sichtbar. Die Warteschlange ist leer und der Algorithmus damit beendet.

Damit wurde das SSP gelöst und einen kürzeste Wege zu jeder Ecke des Graphen vom Startpunkt. Daraus ergibt sich auch die Lösung aller SPP für s zu jeder anderen Ecke aus V . Es ist jedoch nicht sinnvoll für die Lösung eines SPP jedes mal das SSP für den kompletten Graphen zu berechnen. Das liefert nicht nur viele irrelevante Ergebnisse sondern kostet auch mehr Berechnungsressourcen und Zeit. Daher gibt es unterschiedliche Möglichkeiten den Dijkstra Algorithmus zu beschleunigen.

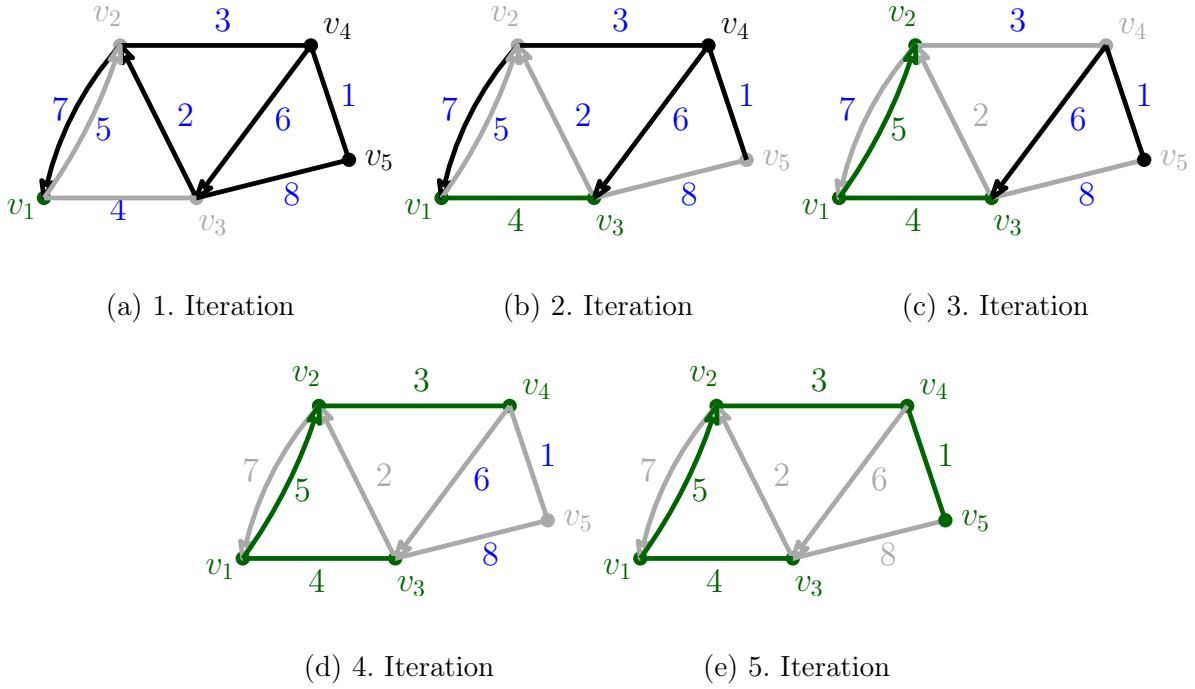


Abbildung 7: Kompletter Durchlauf eines Dijkstra Algorithmus

	W_s	K_s
1.	$v_2 : 5(v_1) \ v_3 : 4(v_1)$	$v_1 : 0(v_1)$
2.	$v_2 : 5(v_1) \ v_5 : 12(v_3)$	$v_1 : 0(v_1) \ v_3 : 4(v_1)$
3.	$v_5 : 12(v_3) \ v_4 : 8(v_2)$	$v_1 : 0(v_1) \ v_3 : 4(v_1) \ v_2 : 5(v_1)$
4.	$v_5 : 9(v_4)$	$v_1 : 0(v_1) \ v_3 : 4(v_1) \ v_2 : 5(v_1) \ v_4 : 8(v_2)$
5.	-	$v_1 : 0(v_1) \ v_3 : 4(v_1) \ v_2 : 5(v_1) \ v_4 : 8(v_2) \ v_5 : 9(v_4)$

Tabelle 1: W_s und K_s für jede Iteration aus Abb. 7

2.2.3 Speedup Techniken

Early Stopping: Der Algorithmus wird bisher jedes mal für den ganzen Graphen ausgeführt, obwohl oft nur die Route für einen kleinen Bruchteil des Graphen benötigt wird. Die einfachste Methode ist den Dijkstra zu stoppen, nachdem z erreicht wurde (8a).

Bidirectional Dijkstra: Beim *Bidirectional Dijkstra* werden zeitgleich zwei Algorithmen nebeneinander ausgeführt. Einer auf s und einer auf z (auf einem umgekehrt gerichteten Graphen). Für beide Instanzen gibt es eine separate Warteschlange W_s und W_z . Zu Beginn wird für jeden Startpunkt die initiale der umliegenden Ecken durchgeführt. Anschließend wird die Ecke mit der geringsten Distanz in beiden Warteschlangen markiert und aus der jeweiligen Warteschlange entfernt. Wird eine Ecke aus beiden Warteschlangen entfernt, werden W_s und W_z auf weitere übereinstimmende Ecken geprüft. Für jede Übereinstimmung wird die Distanz in beiden Instanzen Berechnet. Die Ecke ist Teil des kürzesten Weges wenn die Summe beider Distanzen minimal ist.

In der schematischen Abbildung 8b wird die Einsparung gegenüber dem Early Stopping deutlich. Der graue Bereich muss nicht berechnet werden. Die Bearbeitungszeit lässt sich

also theoretisch um die Hälfte reduzieren⁶.

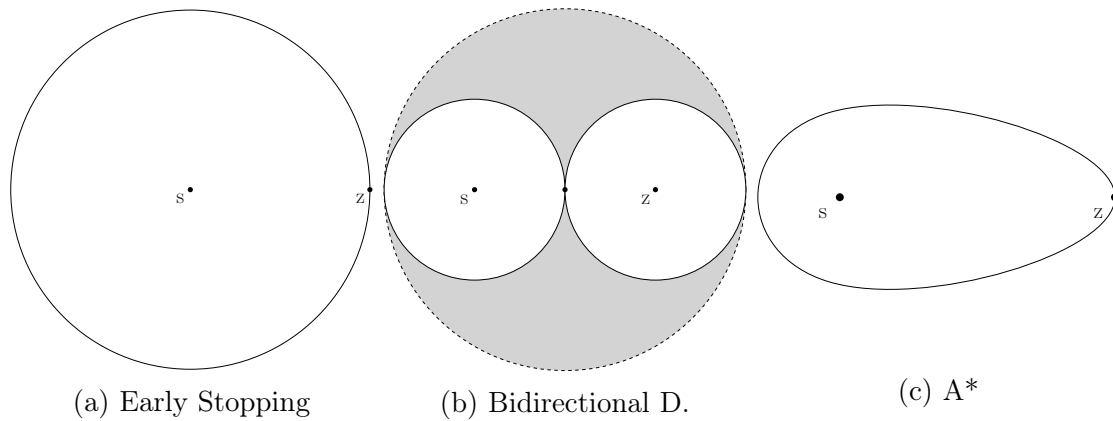


Abbildung 8: Speedup Techniken für den Dijkstra Algorithmus

A*: Der A* Algorithmus ist eine Variante des Dijkstras die den Suchraum in Richtung der Zielecke lenkt. Es wird durch eine Funktion für jede Ecke die Distanz zum Ziel geschätzt. Diese wird mit den Kantengewichten verrechnet damit Ecken in Zielrichtung früher markiert werden. Die standardmäßig kreisförmige Ausbreitung des Dijkstras wird mit dem A* zu einem Oval gestreckt. Da der Zielpunkt so früher erreicht wird, müssen weniger Iterationen durchgeführt werden.

Diese und weitere Möglichkeiten sind in Kurt Mehlhorn 2008, S. 209–213 ausführlich beschrieben.

2.3 Isochronen Berechnung

Isochronen sind Linien gleicher Zeit (griech.: *iso* = gleich + *chronos* = Zeit).

Wenn in einem gewichteten Graphen die Kanten die benötigte Zeit enthalten, um von einem Knoten zum nächsten zu gelangen, können damit Analysen zur Erreichbarkeit durchgeführt werden. Dazu wird ein SSP für eine zentrale Ecke z mit einem gegebenen Zeitlimit t gelöst. Isochronen können mit unterschiedlichen Methoden berechnet werden. Das resultierende Objekt ist immer ein Polygon, welches jeden in gegebenem Zeitlimit erreichbaren Punkt beinhaltet.

2.3.1 Gitterbasierter Ansatz

Beim *Marching Squares* Algorithmus wird um das Zentrum ein Gitter über dem Graphen gebildet (9a). Die Eckpunkte des Gitters erhalten dabei die Werte des nächsten Punktes auf dem Graphen (9b). Anschließend werden auf den Kanten des Gitters diejenigen Punkte markiert, bei denen der Wert mit dem gesuchten Zeitlimit übereinstimmt. In Abbildung 9c wurde das für die Werte $t = 5$ und $t = 10$ durchgeführt. Die markierten Punkte werden verbunden und bilden schließlich die Isochrone.

⁶In der Realität wird dieser Wert selten erreicht. Bei Graphen mit einer höheren Eckendichte in Nähe von z könnte die Bearbeitungszeit sogar größer werden. Allgemein wird aber ein signifikanter Vorteil erzielt.

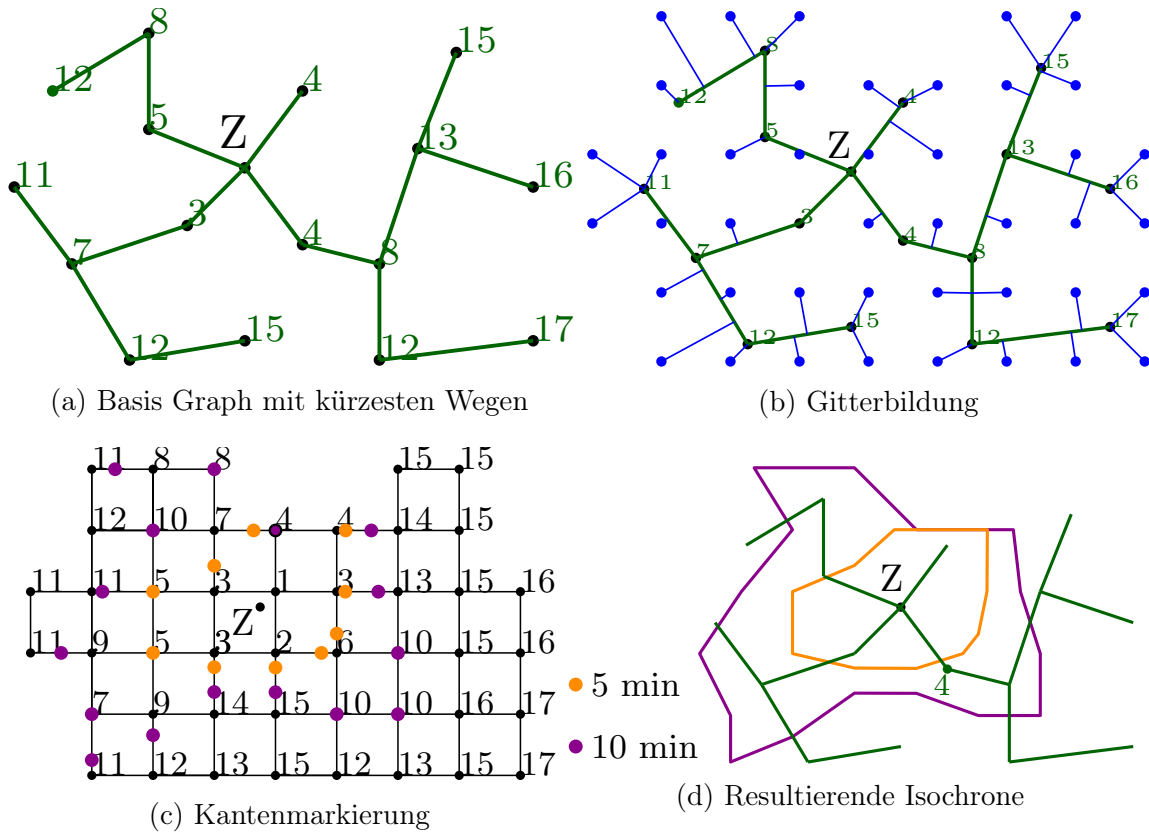


Abbildung 9: Marching Squares Algorithmus

Der Vorteil dieses Algorithmus ist, dass die Maschengröße des Gitters angepasst werden kann. Bei sehr kleinen Maschen liefert der Algorithmus ein sehr genaues Ergebnis. Allerdings werden dabei mehr Ressourcen zur Berechnung gebraucht. Daher können nur kleine Gebiete und geringe Zeitlimits berechnet werden. Bei weiten Maschen ist der Algorithmus dagegen sehr schnell und kann große Distanzen und lange Zeitspannen berechnen. Das Ergebnis ist dementsprechend aber auch ungenauer. Auch in Abbildung 9d wurde die Größe der Maschen unglücklich gewählt. Eine Ecke mit Abstand 4 Minuten vom Zentrum liegt dadurch außerhalb der 5-Minuten-Isochrone.

2.3.2 Dreiecksvermaschung

Neis u. a. 2008 beschreiben eine anschauliche Methode um Isochronen zu berechnen. Nach der Lösung des SSPs werden den Ecken des Graphen die geographische Koordinate der repräsentierten Kreuzung zugeteilt. Die Kanten werden nicht benötigt und daher entfernt. Es liegt also eine 3D-Punktwolke vor. Jeder Ecke wird nun die benötigte Zeit zugewiesen, mit der diese zu erreichen ist. Anschließend werden die Ecken nach dem Delaunay Triangulationsverfahren vermascht. Es entsteht eine Art Trichter mit der Startecke als Tiefpunkt auf der Höhe Null. Wird dieser Trichter auf Höhe des Zeitlimits geschnitten, entsprechen von oben betrachtet die Randkanten des Trichters der Isochrone.

2.3.3 Formenbasierter Ansatz

Die Implementierung des ORS zur Berechnung von Isochronen verwendet ist einen Formbasierten Ansatz. Zuerst werden mit dem in Kapitel 2.2.2 bereits ausführlich erklärten Dijkstra Algorithmus alle in gegebener Zeit erreichbaren Kanten markiert. Anschließend werden die geographischen Punkte (Pillar Nodes) aus der WayGeometry der Kante extrahiert (Siehe Kapitel 2.1.3 auf Seite 6). Um jeden der extrahierten Punkte wird ein kreisförmiger Pufferbereich gelegt. Dadurch können nahe beieinanderliegende Punkte übersprungen werden. Mit den verbleibenden Punkten wird eine Punktwolke generiert. Auf diese Punktwolke wird nun der Alpha-Shape Algorithmus(Zitat) angewandt um die Isochrone als Hülle um die erreichbaren Wegsegmente zu zeichnen. (Abb. Pictures shape based stuff) Dieser Ansatz liefert präzise Ergebnisse bei schnellen Berechnungszeiten. Die Verwendung der Alpha-Shapes verhindert allerdings die Möglichkeit der Darstellung von nicht erreichbaren 'Löchern' innerhalb der Isochronen.

3 Methodik

Die Grundsätzlichen Überlegungen beruhen, wie in der Einleitung bereits erwähnt, auf den Notstandsvollmachten durch §35 Absatz 1 der StVO für Einsatzfahrzeuge des Rettungsdienstes, der Feuerwehr und der Polizei. Demnach dürfen sich Einsatzfahrzeuge im Notfall über die Vorschriften der StVO hinwegsetzen. Die Frage ist allerdings in wie weit von diesen Rechten im Ernstfall Gebrauch gemacht werden kann. Um dieser Sache auf den Grund zu gehen, wurde ein Fragenkatalog für die FFL zusammengestellt. Dieser wurde als Datei auf Google Docs gespeichert und ist auch weiterhin verfügbar (Link im 8). Die Fragen wurden mit Hinblick auf die OSM Datenstruktur und das vorhandene ORS Backend gewählt. Die Antworten der FFL sind hier sinngemäß wiedergegeben. Die wörtlichen Antworten sind dem Google Dokument im Anhang zu entnehmen. Auf Basis der erhobenen Informationen wurden geeignete OSM-Tags gesucht mit denen die empfohlenen Änderungen verwirklicht werden können. Anschließend wurden neue Java Funktionen geschrieben um diese Tags bei der Erstellung des Graphen und bei Anfragen zu berücksichtigen. Außerdem wurde das Frontend an das neue Profil angepasst.

3.1 Informations Erhebung

1. Maße der Fahrzeuge: Die Dimensionen des Fahrzeuges sind limitierende Faktoren für bestimmte Wegsegmente. Manche Brücken halten nur ein bestimmtes Gewicht aus und ein Tunnel hat nur eine gewisse Höhe. Hier stützt sich das ORS Backend auf Restriktionen durch die OSM-Tags: `maxlength`, `maxwidth`, `maxheight`, `maxweight` und `maxaxleload`. Von der FFL wurde als wichtigstes Fahrzeug Löschfahrzeuge der Klasse LF8, LF8/6 und MLF mit folgenden Daten angegeben: - Länge: 7 Meter - Breite: 2,5 Meter - Höhe: 3 Meter - Gewicht: 7,5 Tonnen - Achslast: k.A. ask Stefan Tonnen

2. Maximale Geschwindigkeiten auf unterschiedlichen Straßentypen: Es kann nicht auf jeder Straße mit maximaler Geschwindigkeit des Fahrzeugs gefahren werden. Deswegen werden für jede Kante des Graphen Geschwindigkeitsmaxima definiert. Dabei wird auf einen Standardwert für den Straßentyp, die *Values* (Werte) des OSM *Keys* (Schlüssel) `highway`, zurückgegriffen. Tabelle 2 zeigt die bisherigen Standardgeschwindigkeiten für das Schwerfahrzeug-Profil des ORS zusammen mit den vorgeschlagenen Werten der FFL in Klammer. Es wurde darauf hingewiesen, dass die 80km/h auf Autobahnen nur gelten wenn kein Stau besteht. Im Fall eines Staus wäre laut FFL auch die Anfahrt auf der Gegenfahrbahn bzw. entgegen der Verkehrsrichtung auf der gleichen Bahn interessant.

Genaue Informationen zu einzelnen Tags können der OSM Wikipedia (Contributors 2014) entnommen werden.

Auf Wirtschafts-, Wald- und Feldwegen mit dem Key-Value-Pair `highway=track` kann zusätzlich der Zustand des Weges mit dem Tag `waytype` beschrieben werden. Die bisherigen und vorgeschlagenen Geschwindigkeit für die jeweiligen Values sind in Tabelle 3 zu sehen.

3. Dürfen vorgegebene Geschwindigkeiten (Bsp. 30er Zone/ Tempolimit 70 etc) überschritten werden? Manchmal sind besondere Geschwindigkeitsrestriktionen, wie zum Beispiel Tempo-30-Zonen vorhanden. In Tempo-30-Zonen darf 50 km/h

highway Value	km/h
<i>Autobahnen</i>	
motorway (Autobahn)	80
motorway_link (Autobahn-Zubringer)	50
motorroad (Kraftfahrtstraße)	80
trunk (Schnellstraße)	80
trunk_link (Zubringer zu Schnellstraße)	50
<i>Siedlungen</i>	
primary (Bundesstraßen)	60(80)
primary_link	50
secondary (Landes-/Staatsstraßen)	60(80)
secondary_link	50
tertiary (Kreisstraßen)	60(80)
tertiary_link	50
unclassified (Nebenstraße (oft ohne Mittellinie))	60
residential (Straße an und in Wohngebieten)	50
living_street (Spielstraße)	10
service (Erschließungsstraße)	20
road (unbekannte Straße)	20
track (Wald-/Feldweg)	15

Tabelle 2: Standardgeschwindigkeiten für unterschiedliche highway Values

tracktype Value	km/h
grade1 (Wasserfester Belag)	20(25)
grade2 (Wassergebundene Decke)	15
grade3 (Befestigter oder ausgebesserter Weg)	10(15)
grade4 (Unbefestigter Weg)	5(10)
grade5 (Unbefestigter Weg)	5

Tabelle 3: Standardgeschwindigkeiten für unterschiedlichen tracktype Values

und in Spielstraßen darf 20 km/h gefahren werden. Dabei ist immer noch auf die Unversehrtheit der anderen Verkehrsteilnehmer (vor allem Kinder) zu achten.

4. Dürfen Gewicht und/oder Achslast für Straßen mit Vorgabe überschritten werden? Es dürfen Straßen benutzt werden die zum Beispiel wegen einer folgenden Brücke auf 7,5 Tonnen beschränkt sind. Um die Brücke selbst zu nutzen ist allerdings lokales Wissen über die tatsächliche Tragfähigkeit notwendig. Allgemein müssen bei Zu- oder Durchfahrten für die Feuerwehr die Aufstellflächen und die Bewegungsflächen so befestigt sein, dass sie von Feuerwehrfahrzeugen mit einer Achslast bis zu 10 t und einem zulässigen Gesamtgewicht bis zu 16 t befahren werden können.

5. Dürfen Einbahnstraßen in entgegengesetzter Richtung durchfahren werden? Einbahnstraßen dürfen in entgegengesetzter Richtung durchfahren werden.

Key-Value-Paar	Geschwindigkeit
aeroway=runway (Start/Landebahn)	80
aeroway=taxilane (Rollweg)	80
highway=raceway (Rennstrecke)	80
highway=cycleway (Radweg)	10
tracktype=pedestrian(Fußgängerzone)	
tracktype=footway(Fußweg)	

Tabelle 4: Standardgeschwindigkeiten für spezielle Wegtypen

6. Ist 5. im Einsatz überhaupt sinnvoll? (falls z.B. die Straße durch Fahrzeug(e) blockiert ist?) Ob eine Einbahnstraße benutzt wird muss der Fahrer während dem Einsatz entscheiden. Es hängt dabei von der verfügbaren Breite und der Länge der Straße ab. Auch die Möglichkeit des Ausweichens bei Gegenverkehr spielt eine Rolle. Anhand dieser Faktoren wird entschieden ob Zeit gewonnen wird und die Einbahnstraße benutzt wird oder nicht.

7. Zu welchen weiteren Routen bzw. Verkehrsnetz bezogenen Besonderheiten kommt es im Einsatz? Es dürfen Bus-, Taxi- und Tram-Spuren verwendet werden.

8. Ist es wichtig auf der richtigen Straßenseite anzukommen ? Nein.

9. Wäre eine Suche nach Hydranten, Löschwassertanks etc. am Zielort bzw. im Einzugsgebiet sinnvoll? Vor Ort gibt es nicht immer Löschwasser, weshalb eine Suche nach Hydranten, Seen, Tanks, Flüssen und ähnlichen Quellen auf jeden Fall sinnvoll ist.

10. Dürfen folgende Wegtypen befahren werden? Wenn ja mit welcher Geschwindigkeit? Es dürfen die in Tabelle 4 gelisteten, für den normalen Straßenverkehr nicht zugänglichen Wegtypen benutzt werden.

3.2 Generierung des Routing-Profiles

Aufbauend auf den gewonnenen Informationen aus dem Fragenkatalog werden nun Java-Klassen für das Emergency Profil Erstellt. Dazu werden die bereits vorhandenen Klassen eines ORS Profils verwendet. Da die Dimensionen des Fahrzeuges mitberücksichtigt werden sollen wird das HGV-Profil als Schablone benutzt. Die eigentliche Arbeit wird von vier Klassen erledigt, die allerdings an den richtigen Stellen im Code des Backends eingebunden werden müssen.

Davon sind alleine drei für die Fahrzeugdimensionen zuständig. Die Klasse `EmergencyVehicleAttributesGra` initiiert das Speicherobjekt und ist für die Fehlerbehandlung zuständig (Abb. 10). Die Klasse `EmergencyVehicleGraphStorageBuilder` baut das Speicherobjekt zusammen. Hier werden die Beschränkungen für die Höhe, Breite, Länge, Achslast und das Gewicht aus den OSM-Daten gespeichert. Das Profil für Schwerfahrzeuge berücksichtigt hier normalerweise unterschiedliche Restriktionen für verschiedene Fahrzeugtypen. Manche Straßen sind

```

public class EmergencyVehicleAttributesGraphStorage extends
    HeavyVehicleAttributesGraphStorage {
    public EmergencyVehicleAttributesGraphStorage(boolean
        includeRestrictions) {
        super(includeRestrictions);
    }

    /* pointer for no entry */
    public void init(Graph graph, Directory dir) {
        if (edgesCount > 0)
            throw new AssertionError("The_ext_emergency_storage
            _must_be_initialized_only_once.");

        this.orsEdges = dir.find("ext_emergency");
    }

    public boolean loadExisting() {
        if (!orsEdges.loadExisting())
            throw new IllegalStateException("Unable_to_load_
            storage_'ext_emergency'._corrupt_file_or_directory?");

        edgeEntryBytes = orsEdges.getHeader(0);
        edgesCount = orsEdges.getHeader(4);
        return true;
    }
}

```

Abbildung 10: EmergencyVehicleAttributesGraphStorage (Speicherobjekt für die Fahrzeug-eigenschaften)

beispielsweise nur für Forst- andere nur für Lieferfahrzeuge freigegeben. Andere Straßen dürfen normalerweise aufgrund des OSM-Tags `access=private` oder `access=restricted` gar nicht verwendet werden. Außer den materiellen Beschränkungen durch die Fahrzeugdimensionen gelten für Einsatzfahrzeuge diese Beschränkungen nicht, daher konnte hier einiges auskommentiert werden.

Die Klasse `EmergencyVehicleEdgeFilter` vergleicht bei der Ausführung des Routing Requests für jede Kante die als Parameter angegebenen Fahrzeug-Dimensionen mit den Restriktionen aus dem im `EmergencyVehicleGraphStorageBuilder` produzierten Speicherobjekt. Liegt der übergebene Wert unterhalb der Beschränkung kann potenziell über diese Kante geroutet werden. Überschreitet der Wert die Beschränkung, wird die Kante vom Routing ausgeschlossen (Abb. 12).

Der Hauptteil der Änderungen wird in der `EmergencyFlagEncoder` Klasse vorgenommen. Hier werden zum Beispiel die Maximalgeschwindigkeiten für die unterschiedlichen Wegtypen aus der 2.Frage mit eingebaut (Abb. 13). Die Geschwindigkeiten sind bewusst höher als die zurückgegebenen Werte gewählt, mit der Absicht das Emergency Profil auch für andere Einsatzfahrzeuge, wie zum Beispiel PKW der Polizei nutzbar zu machen. Das Geschwindigkeitslimit von 80km/h hängt nämlich mit der Fahrzeugklasse zusammen und nicht mit dem Wegtyp. Fahrradwege und Fußgängerzonen werden durch die Zuweisung einer Geschwindigkeit ebenfalls für das Profil nutzbar gemacht.

Die Beschränkungen 30er-Zonen und Spielstraßen sollen mit dem Code Segment aus Abbildung 14 übergangen werden. Wichtig ist dabei, dass der OSM-Tag `highway=living_street`

```

maxheight")) {
    = VehicleDimensionRestrictions.MaxHeight;
    equals("maxweight")) {
    = VehicleDimensionRestrictions.MaxWeight;
    equals("maxweight:hgv")) {
    VehicleDimensionRestrictions.MaxWeight;
    ("maxwidth")) {
    = VehicleDimensionRestrictions.MaxWidth;
    equals("maxlength")) {
    = VehicleDimensionRestrictions.MaxLength;
    equals("maxlength:hgv")) {
    = VehicleDimensionRestrictions.MaxLength;
    ("maxaxleload")) {
    = VehicleDimensionRestrictions.MaxAxleLoad;
}

if (key.equals("
valueIndex
} else if (key.
valueIndex
} else if (key.
valueIndex =
} else if (key.equals
valueIndex
} else if (key.
valueIndex
} else if (key.
valueIndex
}
else if (key.equals
valueIndex
}

```

Abbildung 11: Entnahme der Dimensionsbeschränkungen (EmergencyVehicleGraphStorageBuilder)

Abbildung 12: Vergleich der übergebenen Dimensionen mit den Restriktionen des Graphen (EmergencyVehicleEdgeFilter)

nicht automatisch eine Spielstraße ist. In Spielstraßen darf in Deutschland nur Schrittgeschwindigkeit gefahren werden. Diese Eigenschaft muss beim taggen⁷ berücksichtigt werden. Häufig wird solchen Straßen der Tag maxspeed=7 zugewiesen.

Um Bus- und Taxispuren für das Profil nutzbar zu machen wurde das Code Segment 15 implementiert. Für mit Fahrzeugen befahrbare Trampuren gibt es in OpenStreetMap bisher leider keinen geeigneten Tag. Die Segmente der Trampspur die befahren werden können liegen sowieso meistens auf einer vorhandenen Straße.

Um Einbahnstraßen in jeder Richtung nutzen zu können wurden Teile des Codes auskommentiert, die einer Kante eine Vorwärts oder Rückwärts Richtung geben⁸.

Der EmergencyFlagEncoder beinhaltet auch die Funktion `collect` mit welcher dem Wegsegment anhand seines Wegtyps eine Priorität zugeschrieben wird. So können Straßen auf

⁷taggen: das Zuweisen eines Key-Value-Pairs zu einem OSM Objekt

⁸Jeweils aus Sicht der Startecke einer Kante gesehen

```

Map<String, Integer> defaultSpeedMap = new HashMap<String, Integer>
>();
// autobahn
defaultSpeedMap.put("motorway", 130);
defaultSpeedMap.put("motorway_link", 50);
defaultSpeedMap.put("motorroad", 130);
// bundesstraße
defaultSpeedMap.put("trunk", 120);
defaultSpeedMap.put("trunk_link", 50);
// linking bigger town
defaultSpeedMap.put("primary", 120);
defaultSpeedMap.put("primary_link", 50);
// linking towns + villages
defaultSpeedMap.put("secondary", 120);
defaultSpeedMap.put("secondary_link", 50);
// streets without middle line separation
defaultSpeedMap.put("tertiary", 110);
defaultSpeedMap.put("tertiary_link", 50);
defaultSpeedMap.put("unclassified", 60);
defaultSpeedMap.put("residential", 50);
// spielstraße
defaultSpeedMap.put("living_street", 20);
defaultSpeedMap.put("service", 20);
// unknown road
defaultSpeedMap.put("road", 20);
// forestry stuff
defaultSpeedMap.put("track", 15);
// additional available for emergency
defaultSpeedMap.put("raceway", 100);
defaultSpeedMap.put("cycleway", 10);
defaultSpeedMap.put("pedestrian", 10);
defaultSpeedMap.put("footway", 5);

```

Abbildung 13: Maximalgeschwindigkeiten für unterschiedliche Wegtypen (EmergencyFlagEncoder)

denen hohe Geschwindigkeiten gefahren werden können oder die extra für Liefer- oder Forstfahrzeuge ausgeschrieben sind bevorzugt benutzt werden und Straßen durch Wohngebiete oder Spielstraßen nur dann verwendet werden, wenn das Ziel anders nicht zu erreichen ist.

Durch das den Code in Abbildung 16 sollen die höchste Priorität Straßen mit dem Tag `service=emergency_access` erhalten. Das sind zum Beispiel Rettungszufahrten zu Gebäuden und gesonderte Autobahnauffahrten, die vom normalen Verkehr nicht benutzt werden dürfen. Da Busspuren theoretisch auch vom normalen Verkehr freigehalten werden müssen, werden diesen hier ebenfalls eine höhere Priorität zugewiesen.

Beim ausführen des Codes werden die auf diese Weise generierten Flags beim Bau des Graphen zusammen mit den geographischen Koordinaten, den Wegtypen, der Richtung, der Oberfläche und noch viele weitere Flags auf den Kanten des Graphen gespeichert. Der Graph wird, wie in Kapitel 2.1.3 bereits beschrieben aufgebaut.

Um das Profil auch für die Kooperationspartner in Lützelburg nutzbar zu machen, wurde eine Instanz des ORS Backends mit diesen Änderungen auf einem Tomcat Server der Universität Heidelberg installiert. Genaue Anweisungen zum Aufsetzen so eines Servers

```

// 30er Zone TODO restrict to waytype
if(maxSpeed == 30.0)
    maxSpeed = 50.0;
// Spielstraße
if(maxSpeed == 7.0 && highway == "living_street")
    maxSpeed = 20.0;

```

Abbildung 14: Limit für 30er Zonen und Spielstraßen(EmergencyFlagEncoder)

```

if (way.hasTag("lanes:psv") || way.hasTag("lanes:bus") || way.
hasTag("lanes:taxi") || way.hasTag("busway:left_lane") || way.hasTag("busway:
:left_lane") || way.hasTag("busway:right_lane"))
    return acceptBit;
// allow railway=tram where paved? no suitable exclusion criteria
found yet
if (way.hasTag("aeroway", "runway") || way.hasTag("aeroway", "
taxilane"))
    return acceptBit;

```

Abbildung 15: Nutzung von speziellen Wegen(EmergencyFlagEncoder)

werden in Neis 2006 ausführlich beschrieben.

Damit ist der prozessierungs Teil des Profils fertig gestellt. Es können an die Schnittstelle des ORS nun Routing-Anfragen mit dem neuen Profil gestellt werden, die eine Antwort im GeoJson Format zurückgeben.

3.3 Anpassungen Frontend

Das ORS Frontend ist ein Graphical User Interface (GUI) welches die interaktive Erstellung einer Abfrage an das Backend ermöglicht. Gleichzeitig kann das Ergebnis quasi sofort auf der gleichen Oberfläche dargestellt werden. Das Frontend für das Emergency-Profil kann unter erreicht werden.

Allgemein kann über einen Rechtsklick auf die Karte ein Start-, Weg- oder Endpunkt platziert werden. Sobald zwei Punkte vorhanden sind wird ein Routing Request für das ausgewählte Profil an das Backend gesendet und die erhaltene Antwort als schnellster Weg zwischen den Punkten dargestellt. Genauere Informationen zu der Route, wie die benötigte Zeit, die Distanz oder die einzelnen Wegstücke können der Seitenleiste entnommen werden. Um eine Erreichbarkeitsanalyse durchzuführen kann am linken Rand der Website von Routing auf Isochronen umgestellt werden. Hier kann mit einem Rechtsklick auf die Karte ein Zentrum gesetzt werden. Nach einem Klick auf den Isochronen Generieren Button wird die Antwort für den aus den Einstellungen generierten Request angezeigt.

Da der Fokus der Arbeit auf der Erstellung des Profils und nicht der Darstellung der Ergebnisse liegt, werden die Änderungen am Frontend hier nur grob umrissen. Alle Änderungen können aber im Detail nachvollzogen werden, da auch hier der Quellcode Opensource ist (Siehe Anhang S. 33).

```
if (way.hasTag("highway", "service") && way.hasTag("service", "
emergency_access"))
    weightToPrioMap.put(100d, PriorityCode.BEST.getValue());
```

Abbildung 16: Nutzung von speziellen Wegen(EmergencyFlagEncoder)

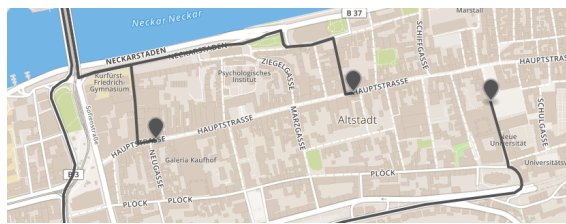
Da die Fußgänger-, das Rollstuhl- und die Fahrrad-Profile für Einsatzfahrten irrelevant sind wurden diese aus der Darstellung auskommentiert. Das Auto- und HGV-Profil wurde für direkte Vergleiche zum Emergency-Profil beibehalten. Gleichzeitig wurde das neue Emergency-Profil eingefügt für welches 2 Fahrzeugklassen vorhanden sind.

1. Profil für Löschfahrzeuge: Für dieses Profil werden automatisch die Höhe, Länge, Breite, Achslast, das Gewicht sowie die Höchstgeschwindigkeit eingestellt. Damit müssen diese Daten bei einem neuen Aufruf der Seite nicht jedes mal neu eingegeben werden. 2. Profil allgemein für Einsatzfahrzeuge: Hier werden keine Fahrzeugdimensionen eingestellt. Die Höchstgeschwindigkeit für dieses Profil ist standardmäßig auf 130km/h gesetzt. Diese kann in den weiteren Einstellungen deaktiviert werden.

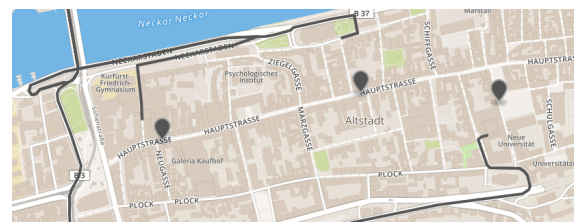
Aufgrund der Berechnungszeit für Graphen wurde der Datenbereich auf den politischen Raum Schwaben begrenzt. Abfragen können nur in diesem Bereich getätigt werden weshalb dieser als durchsichtiges Polygon über der Karte automatisch angezeigt wird. Der Datenbereich kann über eine Checkbox in der rechten unteren Ecke ausgeschaltet werden.

4 Ergebnisse

Die Verwendung neuer Wegtypen konnte erfolgreich eingebunden werden wie in Abbildung 17 zu erkennen ist. Hier wurden als Zielpunkte verschieden Stellen auf der Fußgängerzone (v.a. Hauptstraße) in der Heidelberger Altstadt⁹ gewählt. Da der ORS zur Darstellung mehrerer Routen bisher noch nicht geeignet ist, wurden die Antworten im GeoJson Format mit der Website <http://geojson.io> visualisiert. Im Gegensatz zum Auto Profil des ORS findet das Emergency-Profil seinen Weg bis zum gewünschten Zielpunkt. Durch die niedrige Geschwindigkeitsangabe des Backends wird auch nicht durchgehend in der Fußgängerzone gefahren, was zu manchen Nachtzeiten durchaus schneller gehen würde. Stattdessen werden die Zielpunkte von den schnelleren Seitenstraßen aus angefahren.



(a) Emergency Routing



(b) ORS-Car Routing

Abbildung 17: Routing in die Fußgängerzone der Heidelberger Altstadt

Im Folgenden wurden Isochronen mit der Freiwilligen Feuerwehr Gablingen als Zentrum berechnet (48.454063,10.824415 [Latitude, Longitude]). Folgende Einstellungen wurden dabei vorgenommen:

- Distanz: 5 min - Intervall: 1 min - Maximale Geschwindigkeit: 80 km/h für Löschfahrzeug und HGV; 130 km/h für PKW und Einsatzfahrzeug - HGV Einstellungen für Löschfahrzeug und HGV: Länge=7m ; Breite= 2.5m; Höhe= 3m; Gewicht= 7.5t

Die Ersten Ergebnisse lieferten zu erwartende Resultate 19. Die Emergency Profile konnten in der gleichen Zeit einen größeren Bereich abdecken.

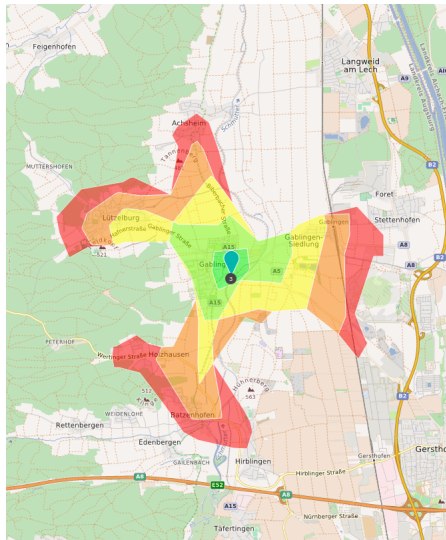
mehr blabla + bilder isos erreichbarkeits gebiet tabelle ? eingehen auf eingeschlossene gebiete (keine löcher durch alpha shapes!)?

Nach einer Testfahrt der FFL wurde allerdings schnell klar, dass die Ergebnisse nicht im realistischen Bereich liegen. In fünf Minuten konnte das Löschfahrzeug auf der Einsatzfahrt vom Startpunkt in Gablingen Richtung Muttershofen nur den nordwestlichen Rand von Lützelburg erreichen. Das Profil kam für das Löschfahrzeug sogar über Muttershofen hinaus.

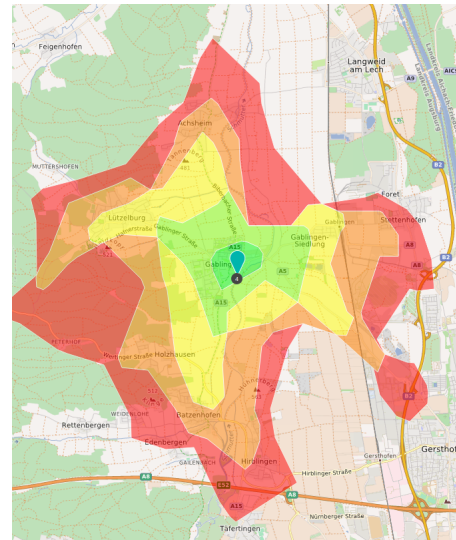
Um genaue Anpassungen am Backend vorzunehmen wurde eine weitere Testfahrt durchgeführt, bei der für jede volle Minute der Aufenthaltsort des Fahrzeuges markiert wurde (Abb. 20). Diese Angaben wurden mit den bisherigen Rückgabewerten des Profils verglichen. Wie aus der Gesamtdauer für die Strecke in Tabelle 5 ersichtlich, ist das Emergency Profil ungefähr 90 Sekunden zu schnell.

Die Vermutung liegt nahe, dass die große Differenz aufgrund der fehlenden Beschleunigungs- und Bremszeiten auftritt. Bisher wird von graphhopper bei der Berechnung der Fahrzeit

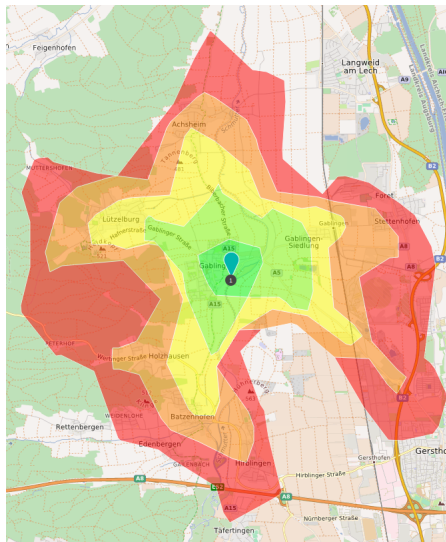
⁹Heidelberg liegt natürlich nicht in Schwaben! Für diesen Test wurde ein Gebiet um Heidelberg als Datengrundlage verwendet.



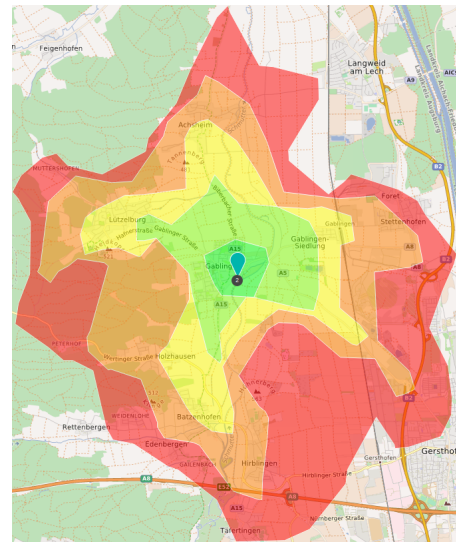
(a) HGV-Profil



(b) Car-Profil



(c) Emergency-Profil – Löschfahrzeug



(d) Emergency-Profil – Einsatzfahrzeug

■ 5 Minuten ■ 4 Minuten ■ 3 Minuten ■ 2 Minuten ■ 1 Minute

Abbildung 19: Ergebnis Isochronen

lediglich 90% des für ein Segment vorgegebenen Geschwindigkeitslimits verwendet. Damit soll sichergestellt sein, dass die Geschwindigkeitsbegrenzung auf jedenfall eingehalten wird. Dieser Faktor wurde für das Emergency Profil entfernt. Es wird also ein Straßensegment auf dem 50 km/h gefahren werden darf über die komplette Distanz bisher auch mit 50 km/h gefahren. In der Realität braucht ein Fahrzeug dieser Größenordnung aber einige Sekunden um diese Geschwindigkeit aus dem Stand zu erreichen. Die selbe Situation besteht für Bremsvorgänge. Es bestehen also drei Szenarien auf einer Fahrt, welche nach festgelegter Route einen Einfluss auf die Fahrzeit haben. Das sind der Start, ein Abbiegevorgang (engl.: Turn) und die Ankunft. Es gibt selbstverständlich noch mehr Faktoren die zum Beispiel andere Verkehrsteilnehmer betreffen, aber die drei genannten Szenarien sind bereits aus der Routenführung ersichtlich und werden definitiv bei jeder Fahrt auftreten¹⁰.

Anhand dieser Idee wurde eine weitere Java-Klasse *AccelerationWeighting* in der die zu-

¹⁰Wenn die Fahrt nur geradeaus geht ist auch kein Abbiegevorgang dabei

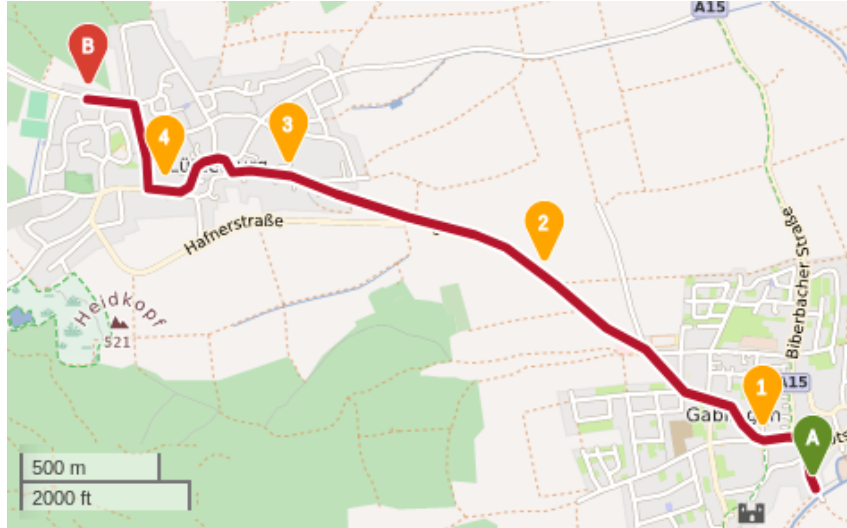


Abbildung 20: Testfahrt 1

Wegpunkt	1	2	3	4	B
Distanz [m]	363,2	1352,8	2331,9	2862,3	3387,3
Zeit (Profil) [s]	28,6	88,7	139,4	177,6	209,8
Zeit (Testfahrt) [s]	60,0	120,0	180,0	240,0	300,0
Zeit Abschnitt [s]	28,6	60,1	50,7	38,2	32,2
Zeit Abschnitt (Testfahrt) [s]	60,0	60,0	60,0	60,0	60,0
Geschwindigkeit [km/h]	45,7	59,3	69,5	50,0	58,7
Geschwindigkeit (Testfahrt) [km/h]	21,8	59,4	58,7	31,8	31,5

Tabelle 5: Fahrt 1 – 1. Auswertung

sätzliche Zeit für diese Szenarien berechnet wird implementiert (8). Diese Klasse soll zwei Funktionen erfüllen. Zum einen muss bereits bei der Suche nach dem schnellsten Weg im Gegensatz zu geraden Strecken zusätzliche Zeit für Turns mit einfließen. Zum Anderen müssen bei der Berechnung der Fahrzeit Start-, Ankunfts- und Turnzeit mit berücksichtigt werden. Es stellt sich die Frage, wie eine Turn auf dem Graphen identifiziert werden kann. Die Änderung des Straßennamens ist sicherlich eine Möglichkeit viele Abbiegevorgänge abzudecken. Allerdings ist nicht immer ein Straßename vorhanden und nicht immer bedeutet der Wechsel des Straßennamens, dass in diese Straße eingebogen werden muss. Eine bessere Lösung bietet die Ausrichtung der Straßensegmente, da hier die Daten auf jeden Fall vorhanden sind. Deshalb wird in der *AccelerationWeighting* Klasse an Tower Nodes (also Kreuzungen) der Winkel zwischen dem letzten und dem nächsten Straßensegment berechnet. Eine Abbiegung bzw. scharfe Kurve wird durch einen Winkel zwischen 50 und 140 Grad definiert. Bei einer Routing Abfrage wird in einem solchen Fall das Gewicht der auf die Kurve folgenden Kante erhöht. Ebenso wird sobald der schnellste Weg ermittelt ist an diesen Stellen die Bremszeit vor der Abbiegung und die Beschleunigungszeit nach der Abbiegung zur Gesamtzeit des folgenden Wegsegmentes addiert. Der gleiche Vorgang wird für Start und Ziel durchgeführt. Die addierte Zeit wird als *Penalty* (dt.: Strafe) bezeichnet.

Anhand der vorliegenden Testfahrt wurde das Profil derart kalibriert, dass durch die Penalties für Abbiegungen, Start und Ziel genau die fehlenden 90 Sekunden gebraucht wurden.

Für Start sowie Ankunft wurden 15 und für Abbiegungen 20 Sekunden veranschlagt.

Mit dem neu eingestellten Profil wurde nun erneut ein Request für die erste Testfahrt gesendet. Wie erwartet brauchte das Profil nun für diesen Weg ebenfalls fast genau fünf Minuten. In Tabelle 6 ist die Differenz zu den einzelnen Minutenmarkern der 1. Testfahrt für diesen Request zu sehen. Diese Ergebnisse zeigen dass das Profil den ersten Teil der Route zu langsam und den zweiten Teil zu schnell absolviert.

Wegpunkt	Sekunden	Abweichung
1min	78,3	+18,3
2min	138,5	+18,5
3min	188,5	+8,5
4min	226,7	-3,3
5min	298,9	-1,1

Tabelle 6: Fahrt 1 – Fahrzeit

Diese Kalibrierung wurde an zwei weiteren Testfahrten (Abb. 21 und 22) vom selben Ausgangspunkt überprüft.

In den Tabellen 7 und 8 ist zu sehen, dass 8 von 10 Wegpunkten um mehr als 20 Sekunden, in 3 Fällen sogar 40 Sekunden zu spät erreicht werden. Mit den Ergebnissen dieser Testfahrten steht fest: das Profil ist noch erheblich zu langsam und die Penalties zu hoch.

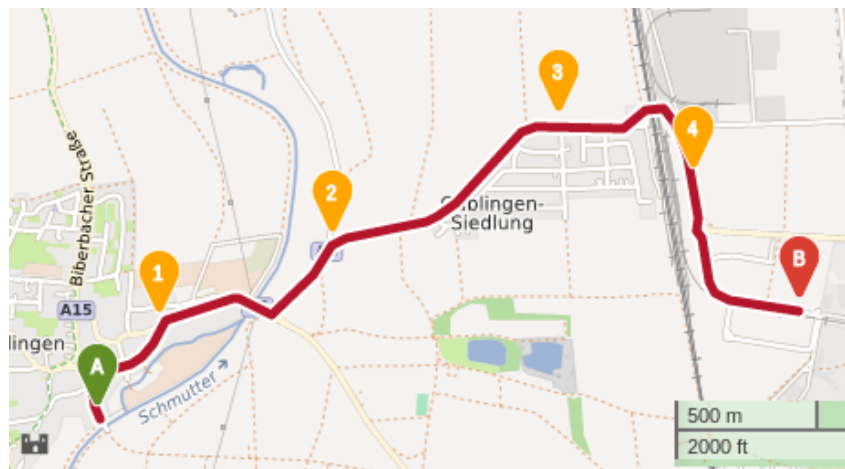


Abbildung 21: Testfahrt 2

Die Penalties für Start und Ankunft wurden für die folgenden Analysen um die Hälfte reduziert und lagen somit bei 7,5 Sekunden. Turnpenalties wurden auf 16 Sekunden reduziert.

Wegpunkt	Sekunden	Abweichung
1min	86,6	+26,6
2min	149,9	+29,9
3min	207,5	+27,5
4min	253,5	+13,5
5min	318,6	+18,6

Tabelle 7: Fahrt 2 – Fahrzeit

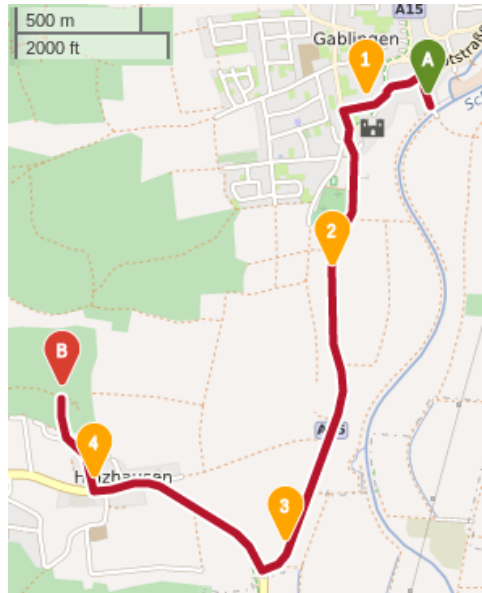


Abbildung 22: Testfahrt 3

Wegpunkt	Sekunden	Abweichung
1min:	89.8	+29.8
2min:	162.4	+42.4
3min:	213	+33
4min:	282.3	+42.3
5min:	343.9	+43.9

Tabelle 8: Fahrt 3 – Fahrzeit

Mit der neuen Gewichtung wurden die Requests für die drei Testfahrten erneut gesendet und mit den Minuten-Markierungen der wirklich benötigten Zeiten verglichen. Die Ergebnisse sind der Tabelle 9 zu entnehmen.

Um die Testfahrten genauer analysieren zu können, wurde das Backend um bisher fehlende Funktionen für die Rückgabe von zusätzlichen Informationen. Dabei wurden die Speicherobjekte für Wegtypen und Wegoberfläche hinzugefügt und zusätzlich die Höheninformationen der Punkte zurückgegeben. Im Frontend konnten dadurch nun die Wegtypen und Wegoberflächen aber auch die Steigung und somit auch ein Höhenprofil angezeigt werden. Außerdem wurde ein Feature zur Darstellung der Höchstgeschwindigkeit der einzelnen Streckensegmente eingebaut.

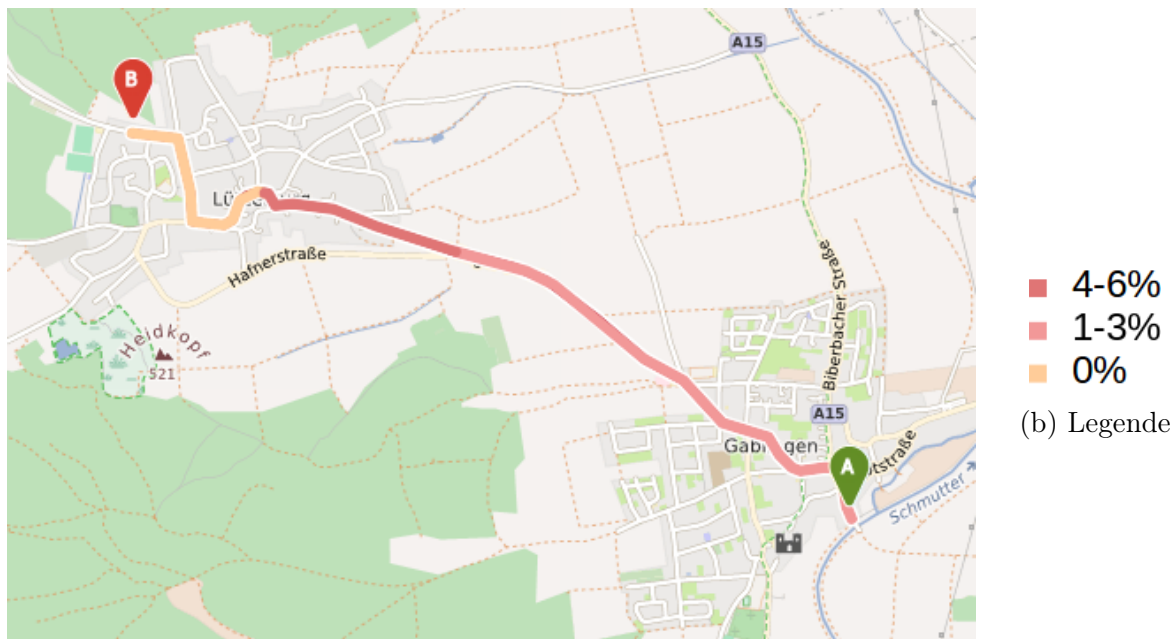
	Fahrt 1		Fahrt 2		Fahrt 3	
Wegpunkt	Sekunden	Abweichung	Sekunden	Abweichung	Sekunden	Abweichung
1min	59.5	-0.5	67.6	+7.6	70.8	+10.8
2min	119.5	-0.5	126.9	+6.9	139.4	+19.4
3min	169.5	-10.5	184.5	+4.5	190	+10
4min	207.7	-32.2	230.5	-10.5	255.2	+15.2
5min	271.9	-28.1	291.6	-8.4	312.9	+12.9

Tabelle 9: Fahrt 1,2,3 – 2. Auswertung

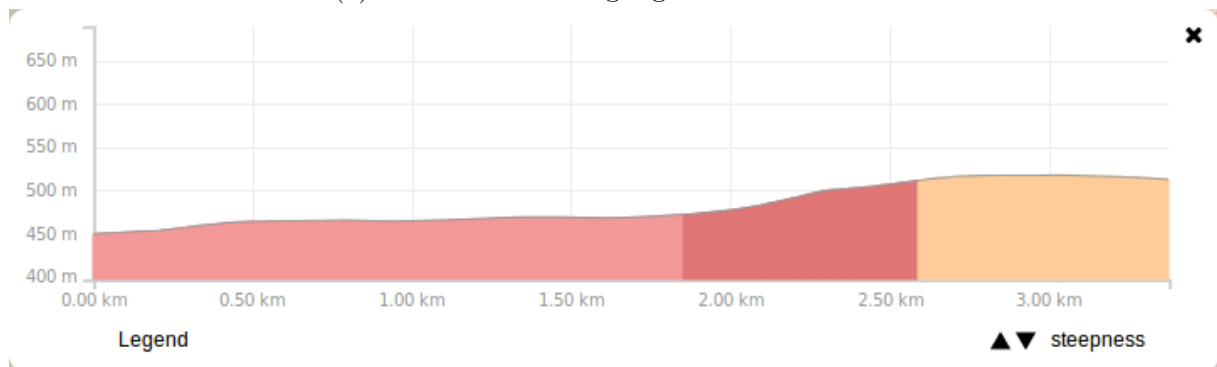
5 Diskussion – Interpretation

5.1 Teststrecke 1

Durch die eingebauten Funktionen wurde ersichtlich, dass bei der Berechnung der Zeit bisher die Steigung als entscheidender Faktor nicht betrachtet wurde. Auf der ersten Teststrecke muss das Löschfahrzeug insgesamt einen Höhenunterschied von ungefähr 70 Metern überwinden (Abb. 23c). Über die Hälfte der Distanz hat dabei eine Steigung von 1-3% und auf einem Viertel der Strecke sogar eine Steigung von 4-6%. Ein Fahrzeug dieser Gewichtsklasse kann auf dieser Strecke folglich nicht die für den größten Teil dieses Anstiegs geltende Maximalgeschwindigkeit von 80 km/h erreichen.



(a) Strecke 1 mit Steigung



(c) Höhenprofil der ersten Strecke

Abbildung 23: Steigung und Höhenprofil der ersten Teststrecke

Darüber hinaus liegen auf der Strecke im inneren Lützelburgs ein paar langegezogene aber dennoch scharfe Kurven (26). Auch hier kann das Löschfahrzeug nicht die gegebene Geschwindigkeit beibehalten und muss abbremesen. Diese Kurven werden allerdings vom Emergency Profil nicht als Turn erkannt, da der Winkel zwischen den Segmenten zu groß

ist. Daher absolviert das Profil diesen Streckenabschnitt in geringerer Zeit als das echte Fahrzeug.

Weiterhin wurde bei allen drei Fahrten beobachtet, dass auf den Segmenten mit Maximalgeschwindigkeit (80 km/h) das Profil ebenfalls konstant ein paar Sekunden gegenüber der Testfahrt gutmachen kann. Die Vermutung liegt nahe, dass das Löschfahrzeug die angegebene Maximalgeschwindigkeit nicht komplett erreichen kann. Möglicherweise stimmt die Angabe sogar mit der Tachoanzeige überein aber die effektive Geschwindigkeit ist geringer.

Die Zeitdifferenzen aus Tabelle 9 zeigen: das Emergency-Profil fährt dem echten Fahrzeug mit der derzeitigen Gewichtung über die ganze Strecke hinweg davon. Wenn die genannten Faktoren der Steigung, der nicht miteinberechneten schärferen Kurven und der Maximalgeschwindigkeit mit einbezogen werden, ist ein Ergebnis im 5 Minuten Bereich zu erwarten.

5.2 Teststrecke 2

Da die 2. Strecke nur in der Ebene verläuft sind keine Abweichungen durch die Steigung zu erwarten. Die Differenzzeiten sind auch für alle fünf Wegpunkte ungefähr im ± 10 -Sekunden-Bereich. Auffallend ist der Sprung vom 3. zum 4. Wegpunkt, bei welchem das Profil 14 Sekunden aufholt. Nach einer genaueren Betrachtung dieses Segments stellte sich heraus, dass hier eine einspurige Unterführung durchfahren werden musste auf die eine durch die Topologie unübersichtliche Kurve folgt (Abb. 24). Vermutlich kam es daher auf der Fahrt zu Verzögerungen, was das Aufholen erklären könnte.



Abbildung 24: Fahrt 2 – Unterführung

5.3 Teststrecke 3

Bei den Ergebnissen der 3. Strecke (Tab. 9) ist das Profil bereits nach dem ersten Wegpunkt um mehr als 10 Sekunden zu langsam. Bis zum Zwei-Minuten-Wegpunkt wird dieser Rückstand fast verdoppelt. Nach genauerer Überprüfung der Durchschnittsgeschwindigkeiten stellte sich heraus, dass der Weg hier durch eine 30er-Zone führt. Allerdings wird

diese nicht mit den erwarteten 50 km/h sondern nur mit den durch den maxspeed Tag festgelegten 30 km/h durchfahren. Offensichtlich ein Bug im Backend der noch behoben werden muss.

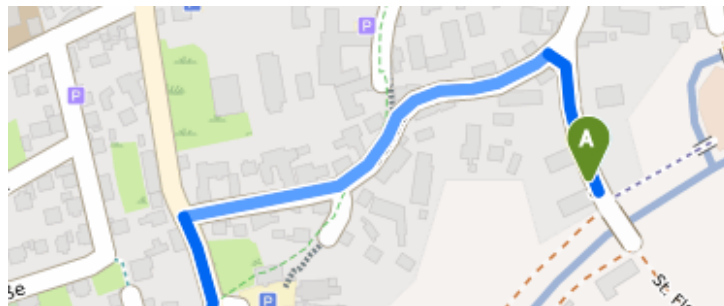


Abbildung 25: Fahrt 3 – Tempo-30-Zone

Die anfängliche 30er-Zone ist fast 350m lang. Diese Strecke kann in 25,2 Sekunden mit 50 km/h und in 42 Sekunden mit 30 km/h durchfahren werden. Wird die Differenz von 16.8 Sekunden von den bisherigen Abweichungen für die 3. Fahrt abgezogen, werden die neuen Werte aus Tabelle 10 erhalten.

Wegpunkt	Sekunden	Abweichung
1min	70.8 ^a	-1.8
2min	139.4	+2.6
3min	190	-6.8
4min	255.2	-1.6
5min	12.9	-3.9

Tabelle 10: Fahrt 3 – neue Fahrzeit

^aZum ersten Wegpunkt beträgt die Strecke nur 260m weshalb hier nur 12.6 Sekunden abgezogen werden

Vor der 2 Minuten Marke beginnt eine Überlandstraße, die bis kurz vor Holzhausen (zwischen Marker 3 und 4) reicht. Das sind 56% der Gesamtstrecke. Wie bei der Diskussion der 1. Fahrt bereits erwähnt ist zu erwarten, dass die Maximalgeschwindigkeit von 80km/h ein bisschen zu hoch gegriffen ist, was den leichten Vorsprung zur 3-Minuten-Marke erklären könnte.

Zwischen Punkt 3 und 4 fällt das Emergency Profil wieder um 5 Sekunden zurück. Zwischen diesen Punkten befindet sich ein Turn, der deutlich schneller als andere Abbiegungen gefahren werden kann. Abbildung 26 zeigt einen Vergleich dieses Turns zwischen den OSM Daten(27a) und der eigentlichen Straße(27b). Es ist zu erkennen, dass der Turn um einiges "weicher" ist als die Daten beschreiben. Daher konnte das Löschfahrzeug gegenüber dem Profil Zeit gewinnen.

Schließlich zeigte auch diese Strecke nach einer Untersuchung des Höhenprofils einen Anstieg auf den letzten 150 Metern um 7-9%(Abb. 28) der noch nicht mit einberechnet wird und dem Emergency Profil wieder einen kleinen Vorsprung gibt.



(a) Turn in den OSM Daten



(b) Turn in der Wirklichkeit

Abbildung 26: Fahrt 3 – Vergleich eines Turns

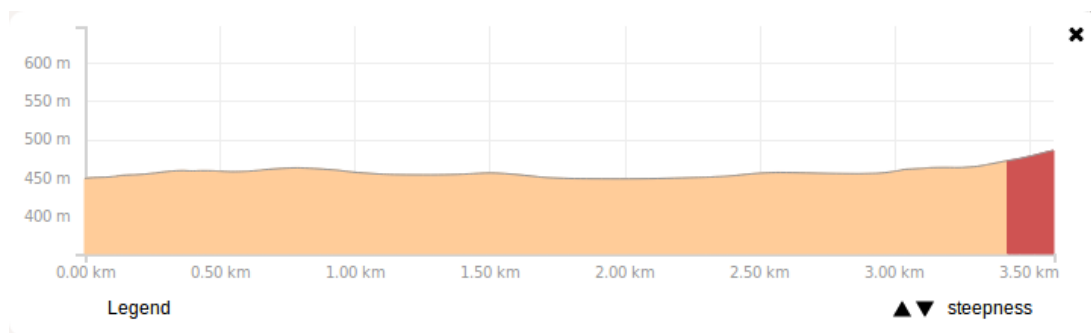


Abbildung 28: Fahrt 3 – Höhenprofil

5.4 Benötigte Änderungen

Wie durch die Vorliegende Analyse zu sehen ist, müssen noch einige Änderungen vollzogen werden, damit das Profil realistischere Ergebnisse zurückgibt.

liststyle Dass 30er Zonen noch nicht richtig erkannt werden muss repariert werden. Für die Steigung muss auf jeden Fall noch eine Funktion implementiert werden, welche diese berücksichtigt. Turns innerhalb von Routensegmenten, an Pillar Nodes, werden bisher nicht berücksichtigt. Die Zeit für einen Turn sollte sowohl anhand des effektiven Winkels als auch anhand der Geschwindigkeiten des vorherigen und folgenden Wegsegmentes bestimmt werden. Ansätze hierfür wurden bereits entwickelt(siehe Source Code) konnten aber noch nicht implementiert werden. Für die Penalties sollte die Zeit, sofern vorhanden, aus der eigentlichen Beschleunigung und ansonsten möglicherweise aus dem Gewicht des Fahrzeugs berechnet werden. Dadurch können einfacher spezifische Fahrzeugprofile erstellt und getestet werden. Da das AccelerationWeighting sowohl für das Löschfahrzeug als auch für allgemeine Einsatzfahrzeuge benutzt wird, erhalten diese bisher die gleichen Penalties. Das kann ebenfalls durch den zuletzt genannten Punkt behoben werden. Weiterhin sollte für Einbahnstraßen die in Gegenrichtung durchfahren werden ein Penalty(ca. *0.9) gesetzt werden, da diese oft nicht mit der selben Geschwindigkeit durchfahren werden können wie in der eigentlichen Fahrtrichtung (auch wenn es erlaubt ist). Das sollte ein Routing auf der Gegenfahrbahn wegen dem Gewinn von ein paar Sekunden verhindern. Ebenfalls sollten dadurch eher vorhandene Parallelstraßen benutzt werden.

6 Fazit

Das erstellte Profil für Löschfahrzeuge wird als Prototyp betrachtet und kann bereits die Grundlegenden geforderten Eigenschaften erfüllen. Es werden bei der Suche nach dem kürzesten Weg in einem Straßennetz die für andere Fahrzeuge geltenden Geschwindigkeitslimits auf geeignetere Werte angehoben. Es ist für das Profil möglich Einbahnstraßen in Beide Richtungen zu durchfahren. Bisher wird deswegen allerdings ab und zu die Gegenfahrbahn über längere Distanzen wegen einem Gewinn von nur wenigen Sekunden verwendet. Darüber hinaus würde ein Penalty dem Risiko des Gegenverkehrs gerecht werden. Interessant wäre es daher einen Penalty-Faktor einzubauen, der bei einer Nutzung entgegen der Fahrtrichtung eine Verzögerung mit einberechnet. Dadurch würden auch in der Nähe liegende Parallelstraßen vorgezogen werden. Darüber hinaus werden bisher nicht befahrbare Straßentypen wie Fußgängerzonen und Radwege für das Profil verfügbar gemacht. Eine Routenführung bis zum Zielpunkt wird so gegenüber den normalen Fahrzeug Profilen in den meisten Fällen erreicht.

Derzeit wird die Schwerfälligkeit des Löschfahrzeuges durch positive Faktorisierung¹¹ der Gewichtung für Start, Ankunft und Turns bei der Suche nach dem schnellsten Weg berücksichtigt. Für die durchgeführten Teststrecken wurde beim routen von Start bis Endpunkt jeweils genau die richtige Route zurückgegeben. Ob diese Faktorisierung allgemein gültig ist, muss noch durch weitere Analysen überprüft werden. Für die Berechnung der Zeit werden zur Berücksichtigung der Schwerfälligkeit einfache Penalties für Start, Ankunft und Turns verwendet. Hierbei wird ein Turn durch den Winkel zwischen zwei Straßen-segmenten an einer Kreuzung ermittelt. Bisher erhält jeder Turn den gleichen Penalty obwohl manche Kurven schneller gefahren werden können. Kurven innerhalb eines Seg-mentes werden bisher noch nicht registriert.

Für realistischere Ergebnisse müssen diese Kurven ebenfalls als Turn registriert werden. Darüber hinaus muss der Öffnungswinkel einer Kurve mit in die Berechnung der zusätzlichen Zeit einfließen, da Turns mit größerem Winkel auch schneller gefahren werden können. Gleichzeitig muss auch die Geschwindigkeit des vorherigen und folgenden Stra-ßensegments beachtet werden, da diese die Brems- bzw. Beschleunigungsstrecke alterieren. Die Steigung eines Straßensegments hat einen signifikanten Einfluss auf die benötigte Zeit und muss daher ebenfalls einkalkuliert werden. Es werden für die Implementierung die-ser Funktionen noch weitere Analysen zu speziellen Testfahren für das jeweilige Szenario (Steigung, Turn) benötigt.

Die Fahrzeugdimensionen und die Höchstgeschwindigkeit sind nicht im Backend festco-dierte sondern können als Parameter bei der Abfrage mitgeliefert werden. Damit ist eine Erweiterung für andere Fahrzeugklassen der Feuerwehr als auch des Rettungsdienstes oder der Polizei möglich, da für alle die selbe Grundprämisse gilt (1.1).

Im Hinblick auf OSM-Daten kann keine endgültige Sicherheit gewährleistet werden, da die Ergebnisse höchstens so gut wie die Daten sein können

Für die Erstellung von Einzugsgebieten für das Emergency Profil, muss letztlich die Ge-wichtung für den Isochrones Service verfügbar gemacht werden. Das ist in der bisherigen Implementierung noch nicht gelungen.

¹¹Die Gewichtung wird mit Werten größer 1.0 multipliziert

7 Ausblick

Add Ausblick to Fazit?

Das Analysieren der Teststrecken ist durch das einzelne Senden der Requests und das anschließende Extrahieren der benötigten Werte relativ zeitaufwändig. Es wäre hilfreich ein Werkzeug zum gleichzeitigen senden von allen Testrequests zu entwickeln mit welchem danach ebenso die Antworten adäquat dargestellt werden können. Dabei sollten zum Beispiel die Distanzen zwischen wichtigen Änderungspunkten auf einer Route berechnet werden können und direkt Tabellarisch aufgelistet werden.

Ein weiterer interessanter Aspekt den es zu beleuchten gilt, ist das unterschiedliche Verkehrsaufkommen auf einer Straße oder Fußgängerzone. So könnten Geschwindigkeiten für bestimmte Wegsegmente je nach Zeit anhand von Berufsverkehr oder Tageszeit angepasst werden. Nachts sollten zum Beispiel generell weniger Autos unterwegs sein, wodurch Zeit gespaart wird, Umgekehrt zur Rush-Hour.

Die im Fragebogen erwähnte Suche nach Löschwasser Quellen am Zielort kann und sollte verwirklicht werden. Von der Datengrundlage her existieren diverse Tags für Objekte die von Einsatzkräften genutzt werden können. Für die Feuerwehr besonders interessant ist zum Beispiel der `emergency=fire_hydrant` Tag für Hydranten. In vielen Städten sind diese durch das OpenFireMap-Projekt¹² flächendeckend abdigitalisiert und der OSM Datenbank zugefügt worden.

¹²<https://wiki.openstreetmap.org/wiki/DE:OpenFireMap>

Literatur

- [Aig15] Martin Aigner. *Graphentheorie*. 2015.
- [Con14] OpenStreetMap Contributors. *OSM Wiki*. 2014. URL: <https://wiki.openstreetmap.org/wiki>.
- [Con15] OpenStreetMap Contributors. *DE:Relationen*. 2015. URL: <https://wiki.openstreetmap.org/wiki/DE:Relationen>.
- [Dij59] Edsger W. Dijkstra. „A note on two problems in connexion with graphs“. In: *Numerische Mathematik 1*. 1959.
- [Kar16] Peter Karich. *Low Level API*. 2016. URL: <https://github.com/graphhopper/graphhopper/blob/master/docs/core/low-level-api.md>.
- [Kur08] Peter Sanders Kurt Mehlhorn. *Algorithms and Data Structures*. 2008.
- [Nei+08] Pascal Neis u. a. „Webbasierte Erreichbarkeitsanalyse – Vorschläge zur Definition eines Accessibility Analysis Service (AAS) auf Basis des OpenLS Route Service“. In: *Aktuelle Arbeiten auf dem Gebiet der informations- und Messtechnik*. 2008.
- [Nei06] Pascal Neis. „Routenplaner für einen Emergency Route Service auf Basis der OpenLS(TM) Spezifikation“. Diss. Fachhochschule Mainz, 2006.
- [Reh+12] Karl Rehrl u. a. „Evaluierung von Verkehrsgraphen für die Berechnung von länderübergreifenden Erreichbarkeitspotenzialen am Beispiel von OpenStreet-Map“. In: *Angewandte Geoinformatik 2012*. 2012.
- [Ste15] Jochen Stein. *Qualitätskriterien für die Bedarfsplanung von Feuerwehren in Städten*. 2015. URL: <http://www.agbf.de/pdf/Fortschreibung%20der%20Empfehlung%20der%20Qualitätskriterien%20fuer%20die%20Bedarfsplanung%20in%20Staedten%20Layout%20neu%202016.pdf>.
- [Sve12] Harmut Noltemeier Sven Oliver Krumke. *Graphentheoretische Konzepte und Algorithmen*. 2012.
- [Wol12] David M. Danko Wolfgang Kresse, Hrsg. *Handbook of Geographic Information*. 2012.

8 Anhang

8.1 Copyrights

Die Rechte für die mit geojson.io generierten Abbildungen(17) gehen an ©Mapbox für den Kartenstil und ©OpenStreetMap für die Daten. Die Rechte der mit dem ORS entworfenen Abbildungen(??) gehen an ©OpenStreetMap contributors für die Daten und ©Maxim Rylov für den Kartenstil. Für Abbildung ?? liegt das Copyright bei ©2017 Google für die Bilder und ©2017 GeoBasis-DE/BKG für die Kartendaten. Für Abbildung 24 liegt das Copyright bei ©DigitalGlobe.

Links

Fragenkatalog

Emergency Profil

CodeFiles

Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig verfasst, noch nicht anderweitig zu Prüfungszwecken vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt, sowie wörtlich und sinngemäße Zitate als solche gekennzeichnet habe.

Heidelberg den 13. Oktober 2017

.....

(Unterschrift)

Danksagungen

An dieser Stelle möchte ich allen Leuten danken, die dazu beigetragen haben diese Arbeit zu verwirklichen.

Ein großes Dankeschön geht natürlich an die Freiwillige Feuerwehr Lützelburg für die Bereitstellung des Fahrzeuges. Insbesondere möchte ich Stefan Witossek für die detaillierten Informationen sowie die Organisation und Durchführung aller Testfahrten danken. Ohne ihn wäre diese Arbeit nicht möglich gewesen.

Danke an meine Kollegen Tim, Max und Lu vom Openrouteservice, die mir mit motivierenden Gesprächen und hilfreichen Erklärungen zur Seite gestanden sind und mich in jeder Hinsicht unterstützt haben.

Meinem Betreuer Dr. Alexander Zipf möchte ich für das Korrigieren der Arbeit und der Möglichkeit Danken beim Openrouteservice zu arbeiten.

Außerdem möchte ich meinen Kommilitonen Angi und Marcel Danken für unterhaltsame Mittagspausen, Motivation und Korrekturlesen.

empty last page