

BACHELOR THESIS
Kalvin Döge

Bestimmung einer 'Grünen Welle' bei Lichtsignalschaltungen für Alster-Fahrradfahrer durch agentenbasierte Simulation mithilfe des **MARS-Frameworks**

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Kalvin Döge

**Bestimmung einer 'Grünen Welle' bei
Lichtsignalschaltungen für Alster-Fahrradfahrer
durch agentenbasierte Simulation mithilfe des
MARS-Frameworks**

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer Prüfer: Prof. Dr. Thomas Clemen
Zweitgutachter: Prof. Dr. Thomas Lehmann

Eingereicht am: 20. September 2023

Kalvin Döge

Thema der Arbeit

Bestimmung einer 'Grünen Welle' bei Lichtsignalschaltungen für Alster-Fahrradfahrer durch agentenbasierte Simulation mithilfe des MARS-Frameworks

Stichworte

Agentenbasierte Simulation, Alster, MARS, MARS-Framework, Grüne Welle, Ampelschaltung, Fahrradfahrer

Kurzzusammenfassung

Fahrradfahrer haben es in Großstädten noch immer schwer, sich effizient im Straßenverkehr fortzubewegen. Die Straßen sind häufig noch für Pkws und weniger auf Fahrräder ausgelegt, weshalb eine Lösung, neben der Erweiterung von Fahrradwegen, die Reduzierung von Rotphasen bei Lichtsignalschaltungen wäre. . . .

Kalvin Döge

Title of Thesis

Determining a 'Green Wave' for traffic lights for Alster-Cyclists with agent-based simulation using the MARS-Framework

Keywords

Agent based Simulation, Alster, MARS, MARS-Framework, Green Wave, Traffic Lights, Cyclists

Abstract

Cyclists still have a difficult time getting around in traffic in densely populated cities. Streets are mostly adapted to larger vehicles and less so for bikes. A solution for that problem, besides further increasing the amount of bicycle lanes, is reducing the length of red phases at light signals. . . .

Inhaltsverzeichnis

Abbildungsverzeichnis	vi
Tabellenverzeichnis	viii
1 Einleitung	1
1.1 Struktur der Bachelorarbeit	1
1.2 Inhalt der Arbeit	2
1.3 Fokus der Arbeit	3
2 Methodik	5
2.1 MARS Arbeitsgruppe und MARS Framework	5
2.2 SmartOpenHamburg	5
2.3 Simulationsmodelle	6
2.4 Stand der Wissenschaft	6
3 Konzept	19
3.1 Simulationstyp	19
3.2 Simulationsort	19
3.3 Agenten	20
3.3.1 Voraussetzungen	20
3.3.2 Modalitäten	21
3.3.3 Interaktionen mit Agenten und Entitäten	21
3.3.4 Anzahl aktiver Agenten	22
3.3.5 Agentenrouten	27
3.4 Lichtsignalschaltungen	28
3.4.1 Beschreibung der Lichtsignalanlagen	28
3.4.2 Funktionsweise von Lichtsignalschaltungen	28
3.4.3 Lichtsignalphasen	29
3.4.4 Lichtsignaltypen und Bezug zur Realität	29

3.5	Designentscheidungen	29
3.5.1	Funktionale und nichtfunktionale Anforderungen	30
3.5.2	Fachliches Datenmodell des Systems	31
3.5.3	Entwurfsmuster im System	35
3.6	Herangehensweise zur Ergebnisermittlung	36
3.7	Umfang der Arbeit	37
4	Implementierung	39
4.1	Agententen, Entitäten und Layer	39
4.2	Anbindung an das MARS-Framework	43
4.3	Konfiguration der Ein- und Ausgabedaten	45
4.4	Überprüfen der Grundfunktionalität	47
4.5	Bekannte Probleme und Einschränkungen	50
5	Evaluation und Diskussion der Ergebnisse	54
5.1	Ergebnisse der Experimente	54
5.1.1	Experimentübergreifende Eingabedaten	54
5.1.2	Ober- und Untergrenze der Simulation	55
5.1.3	Experiment 1: 70 g 20 r	55
5.1.4	Experiment 2: 140 g 10 r	55
5.1.5	Experiment 3: 280 g 5 r	55
5.1.6	Experiment 4: 467 g 3 r	55
5.1.7	Experiment 5: 350 g 4 r	55
5.2	Diskussion der Ergebnisse	55
6	Zusammenfassung	57
	Literaturverzeichnis	58
A	Anhang	60
	Selbstständigkeitserklärung	61

Abbildungsverzeichnis

2.1	Das von Kuang und andere bei einer 3-spurigen Kreuzung dargelegte 8-Phasen Modell [ZMJZ19]	8
2.2	die von Kurtc und Treiber berechneten, durchschnittlichen Kalbirungsfehler der beiden Modelle in % [KT20]	9
2.3	Eine Momentaufnahme aus dem zellulären Automatenmodell [BM19]	10
2.4	Die durchschnittliche Geschwindigkeit und Wartezeit nach Durchführung [BM19]	11
2.5	Beispielworkflow, der Daten aus den Sensorsoren und von der Stadt Hamburg bereitgestellten Daten einbaut in ein abstraktes Simulationsmodell [LAMG ⁺ 21a]	16
2.6	Karte der Fahrradverleihstationen, mit dem Simulationsbereich der Agenten hellblau hinterlegt, den Interessenspunkten rot markiert und Fahrradverleihstation als blaue Markierungen [LAMG ⁺ 21b]	18
3.1	Simulationsumgebung	20
3.2	Polynomfunktion des 4. Grades zur Annäherung	23
3.3	Grobe Flächenberechnung der bewohnbaren Bereiche	26
3.4	Route der Bicycle-Leader mit den 8 Zwischenpunkten	27
3.5	Klassendiagramm der Agenten und Entitäten	32
4.1	Das technische Datenmodell aller Agenten, Entitäten und Layer	40
4.2	Die Anbindung an bestehenden Fassaden des MARS-Frameworks	44
4.3	Der Export von OpenStreetMap in kepler.gl eingefügt	48
4.4	Lichtsignalanlagen an der Kreuzung der Hallerstraße, links in OpenStreetMap, rechts in kepler.gl eingefügt	49
4.5	Die Konsolenausgabe der aktuell eingebauten TrafficLights	49
4.6	Der Erstellbereich der HumanTraveler, links von 4 Uhr und rechts von 5 Uhr morgens	50
4.7	Die Erschaffung des BicycleLeader, links um 8 Uhr, rechts um 9 Uhr morgens	51

Abbildungsverzeichnis

4.8	Hier wird eine Ausgabe an Lichtsignalanlagen gegeben, sobald mehr als 5 Agenten in der Warteschlange einer TrafficLight sind.	51
4.9	Konsolenausgabe, bei der BicycleLeader alle 3 Stunden in Folge wegen der zu langen Rot-Signalphase die Abbruchbedingung erreicht.	52
4.10	Distanzangabe über die Konsole vom letzten, zu erreichenden Punkt, bis zum BicycleLeader.	52
4.11	Konsolenausgabe, bei der HumanTraveler nach 5 Sekunden warten an einer TrafficLight ihre Geschwindigkeit ausgeben.	53

Tabellenverzeichnis

3.1 Die Einwohnerzahl pro km ² für die genannten Stadtviertel	25
5.1	56

1 Einleitung

Mit dem Klimawandel wird es immer deutlicher, dass man selbst lieber auf den Pkw verzichten und auf den öffentlichen Personennahverkehr oder das Fahrrad wechseln sollte. Doch leider ist es nicht ganz so einfach: Gerade in der Großstadt Hamburg gehört Stau tagsüber quasi zum Alltag auf den Straßen. Pkws, Lkws und Fahrradfahrer teilen sich auf manchen Wegen immer wieder dieselbe Spur und halten auch an derselben Ampel, wenn wieder keine Fahrradspur gerade zur Verfügung steht. Doch die Stadt Hamburg fängt an, das Fahrrad im Verkehr in manchen Stadtvierteln zu bevorzugen und die Lichtsignalschaltungen nach ihnen auszurichten [Run22]. So auch gegen das Ende letzten Jahres: Im Stadtteil Eimsbüttel sollen Fahrradfahrern und Fußgängern dem Autoverkehr bevorzugt werden, indem eine Ampel an einer stark befahrenen Überquerung vor anderen vorlassen.

Auch wenn dies nur ein Anfang ist, so lässt sich daraus die Frage ableiten: Ließen sich Lichtsignalschaltungen so zeitlich einstellen, dass sie einem Fahrradfahrer den gesamten Weg eine „Grüne Welle“ geben, noch bevor man erst an der Ampel einen Knopf betätigen muss?

In dieser Arbeit soll diese Frage bei einer verkehrslastigen Rundfahrt untersucht werden und eine mögliche Zeitschaltung bereithalten können.

1.1 Struktur der Bachelorarbeit

Diese Arbeit teilt sich, nach diesem ersten Kapitel „Einleitung“, in fünf weitere Kapitel auf:

- Dem zweiten Kapitel „Methodik“, welches sich mit Erklärungen zum eigentlichen Vorgehen beschäftigt, mit der die Hypothesen vorgestellt werden.

- Dem dritten Kapitel „Konzept“, welches die Implementation textuell vorbereitet und das Simulationsmodell, als auch die Ein- und Ausgabedaten beschreibt und erklärt.
- Dem vierten Kapitel „Implementation“, welches sich mit dem Quellcode und dessen besonderen Aspekten widmet.
- Dem fünften Kapitel „Evaluation“, welches die Ausgabedaten in Bezug zu den Eingabedaten und Modelleinstellungen erläutert und darstellt, als auch den Bezug zu den Forschungsfragen und Hypothesen wieder herstellt.
- Dem sechsten und letzten Kapitel „Zusammenfassung“, in dem die Ergebnisse der Forschungsarbeit genannt und übersichtlich nochmal aufgeführt sind.

1.2 Inhalt der Arbeit

Zur Bestimmung einer „Grünen Welle“ für Fahrradfahrer um die Binnen- und Außenalster, kommen folgende Fragen zur Simulation des Szenarios auf:

- Wie sieht ein durchschnittlicher Fahrradfahrer, Fußgänger und ein Personenkraftwagen in dem Modell aus?
- Welche Strecke fährt der Fahrradfahrer um die Binnen- und Außenalster, um eine Rundfahrt unternommen zu haben?
- Was für Eigenschaften muss die Lichtsignalschaltung in dem Modell haben, damit sie geeignet für eine „Grüne Welle“ ist?
- Wie stark wirkt sich die Auslastung der Straßen auf das Lichtsignalnetz aus, wenn zu verschiedenen Uhrzeiten am Tag die Alsterrundfahrt unternommen wird?
- Wie müssen die Lichtsignalzeiten angepasst werden, wenn der Verkehr sich zu stark auf die „Grüne Welle“ auswirkt?
- Ist es überhaupt möglich, dass Fahrradfahrer in mindestens 90% der Fälle, die er um die Alster fährt, eine „Grüne Welle“ erhält, ohne anzuhalten?

Um die Aspekte genauer zu untersuchen, lassen sich aus ihnen forschungsrelevante Hypothesen aufstellen, die im Folgenden genauer definiert werden:

Für einen Fahrradfahrer ist es möglich, mit durchschnittlicher Geschwindigkeit in 90% der Fälle eine „Grüne Welle“ zu haben, in der er um die Binnen- und Außenalster fährt. Dadurch, dass in einer durchschnittlichen Arbeitswoche Fahrradfahrer von und zu der Arbeit fahren, sind zwei Fahrten um die Binnen- und Außenalster vorgesehen und zu schaffen, um die größtmögliche Menge an Fahrradfahrern abzudecken.

Die Änderung von Lichtsignalschaltzeiten wirkt sich auf die Möglichkeit einer für Fahrradfahrer erreichbaren, „Grünen Welle“ aus. Trotz hohen Verkehrs können Fahrradfahrer bei einer bestimmten Lichtsignalschaltung eine „Grüne Welle“ erreichen, sollte die Verkehrslast dafür nicht zu hoch sein und die Schaltung genügend Freiraum für die zurückgelegte Distanz lassen.

Die Veränderung der Verkehrslast wirkt sich auf die Möglichkeit einer für Fahrradfahrer erreichbaren, „Grünen Welle“ aus. Sobald im Verkehr eine zu große Menge an Personenkraftwagen, Fußgängern oder anderen Fahrradfahrern vorliegt, wird die Wahrscheinlichkeit einer „Grünen Welle“ immer geringer, da diese den Verkehr zu stark aufhalten und damit potenziell Staus verursachen können.

1.3 Fokus der Arbeit

Diese Arbeit beschäftigt sich mit der Bestimmung einer Zeitschaltung für Lichtsignalanlagen, die für alle Anlagen gleichermaßen gelten soll und mit einer hohen Wahrscheinlichkeit eine „grüne Welle“ aus der Sicht der Agenten bewirken soll. Dabei wird das MARS-Framework mit SmartOpenHamburg verwendet, um die Binnen- und Außenalster als Simulationsgebiet und Fokuspunkt des Verkehrs möglichst genau nachzustellen. Der Nutzen für die Bestimmung einer festen Zeitschaltung ist die einfache Änderung bestehender Lichtsignalanlagen in der echten Welt: Innerhalb von Hamburg gibt es mehrere hundert Lichtsignalanlagen, mit 1260 Anlagen bereits innerhalb des simulierten Bereiches. Das stetige Anpassen der Signalanlagen durch Verlängern oder Verkürzen von Phasen über, zum Beispiel, einer künstlichen Intelligenz hat nicht nur die typischen Synchronisationsprobleme eines verteilten Systemes als Schwierigkeit, sondern auch die Technik als Problem. Die Verkehrszentralstellen sind potenziell nicht ausgestattet für eine gut

1 Einleitung

ausgearbeitete, künstliche Intelligenz oder können nur eine begrenzte Menge an Anlagen häufige Phasenänderungen mitteilen. Stattdessen ist ein einmaliges Einstellen aller Anlagen mit einer bestimmten Grün-, Gelb- und Rotphasenlänge lediglich eine Abänderung der Phasenlängen und benötigt weder Synchronisation noch stabile Kommunikationswege zu den Anlagen.

Außerdem sind bisherige Forschungsarbeiten bei in Reihe geschalteten, „grünen Wellen“ stets nur auf Vehikel wie Pkws angeschaut worden, die keine flexiblen Ausweichmöglichkeiten wie ein Fahrrad einbeziehen in die Agentensimulation. Fahrräder können stets eine neue Route einschlagen oder, wenn plötzlich ein Fahrradweg aufkommt, auf diese wechseln und eine Lichtsignalanlage später oder früher erreichen. Ebenso haben Fahrräder nur auf Fahrradwegen die Sicherheit, dort fahren zu dürfen, was sie bei Straßen im Verkehr stark benachteiligt. Dadurch ist es erkennbar, dass sie teilweise auf anderen Routen als Autos fahren müssen und damit eine vorgeplante „grüne Welle“ über ein Straßennetzwerk potenziell nicht einhalten können. Spätestens dann ist eine Bindung der Lichtsignalphasen an die Distanz zu vorherliegenden Lichtsignalanlagen nicht mehr so nützlich und lässt den Fahrradfahrer anhalten.

Entsprechend ist eine zeitlich festgelegte Lichtsignalschaltung bei allen Anlagen ein wirtschaftlicher und technisch einfacherer Lösungsweg, den es in dieser Arbeit zu ermitteln gibt.

2 Methodik

In diesem Abschnitt werden im Folgenden wesentliche Grundlagen des Simulationsmodells näher ausgeführt, die zum Erforschen der Hypothesen hinzugezogen werden: die MARS-Arbeitsgruppe und deren MARS-Framework als auch SmartOpenHamburg.

2.1 MARS Arbeitsgruppe und MARS Framework

Die „MARS“-Arbeitsgruppe, mit vollem Namen „Multi-Agent Research and Simulation group“, ist ein Forschungsprojekt der HAW Hamburg und untersucht, fördert und entwickelt Simulationsszenarien mit dem gleichnamigen Framework [Gro23a]. Das MARS-Framework bietet die Simulationsgrundlage in dieser Arbeit, über die die Umwelt, Agenten und Entitäten verwaltet werden, sowie weitere hilfreiche Tools zum Nutzen bereitgestellt sind.

2.2 SmartOpenHamburg

„SmartOpenHamburg“ ist der sogenannte „digitale Zwilling“ der Stadt Hamburg [Gro23b] und ihrer Verkehrswege. Diese Simulationsumgebung ist so gestaltet, dass sie auf mikroskopischer Ebene, Multimodaler Ebene oder auf groß skalierten Szenarien angepasst werden kann, je nachdem welche Umgebung gerade für die Arbeit benötigt wird. In dieser Arbeit bildet SmartOpenHamburg die Grundlage für die Echtzeitsimulation und ihre Umgebung.

2.3 Simulationsmodelle

Um den Verkehr und deren Teilnehmer um die Binnen- und Außenalster so einstellbar wie möglich zu machen, empfehlen sich zwei Arten einer Modellsimulation:

- Eine agentenbasierte Simulation
- Eine ereignisgesteuerte Simulation

Agentenbasierte Simulation In dieser Simulationsart werden Akteure in eine Umwelt gebracht, die in Echtzeit sowohl mit sich gegenseitig, als auch mit der Umgebung interagieren und mithilfe eines festgelegten Regelsatzes Entscheidungen treffen, die andere Agenten oder die Umwelt beeinflussen können [BSC15]. Wesentlich ist dabei auch, dass die Akteure selbst Entscheidungen treffen können und ein eigenes Verhalten aufweisen. In einer Simulation mit Lichtsignalschaltungen und die Bestimmung einer optimalen „Grünen Welle“ könnten die Akteure zum Beispiel die Fußgänger, die Fahrradfahrer und die Personenkraftwagen darstellen, während die Umwelt das Straßennetz mit den Lichtsignalen wäre. Personenkraftwagen könnten gleichzeitig mit Fahrradfahrern interagieren, indem sie zum Beispiel Plätze vor Ampeln wegnehmen oder Staus verursachen, während Fußgänger zum Beispiel die Ampeln betätigen und damit Fahrradfahrern die „Grüne Welle“ verhindern könnten.

Ereignisgesteuerte Simulation Neben dem agentenbasierten Modell, gibt es ebenso auch die Möglichkeit einer ereignisgesteuerten Simulation. Diese Art der Simulation ist nicht in Echtzeit, hat aber wiederum im allgemeinen Ereignisse, die nacheinander passieren und abgegangen werden [BSC15]. Auf dem Weg von einem Ereignis zu einem anderen können äußere Einflüsse den Übergang zum nächsten Ereignis verändern. Dadurch, dass aber der Wechsel von einem Ereignis zum anderen im Vornherein bei der Implementierung bekannt ist, kann man die Übergänge der Ereignisse zum Beispiel mit Zeit bereits errechnen.

2.4 Stand der Wissenschaft

Die Idee, Lichtsignalanlagen an Kreuzungen mit einer vorherbestimmten Steuerung zu verwalten, ist aber nicht neue. Mehrere Forscher haben bereits Lösungen über

künstliche Intelligenzen, durch Fokussierung auf die Fahrradfahrer oder zeitlicher Abstimmung von den Lichtsignalphasen gefunden.

Ye Zheng, Ding Ma, Fengying Jin und Zhigang Zhao: *Intelligent Traffic Signal Control Based on Reinforcement Learning with State Reduction for Smart Cities*

In dem Forschungsbeitrag von Li Kuang und andere aus dem Jahr 2019, widmen sich die vier Autoren auf eine Lösung mithilfe eines Q-Learning-Algorithmus. Dadurch, dass die Infrastruktur der Straßen nicht weiter ausgebaut werden kann in urbanem Verkehr, bleibt nur noch das Auflösen von Staus bei Lichtsignalschaltung und Kreuzungen[ZMJZ19], weshalb sie sich mit dieser Arbeit an den Ansatz mit einer künstlichen Intelligenz setzten.

Die Aufgabe von bis entwickelten Echtzeitlösungsansätzen, so Zheng und andere, sind vier Kategorien zuzuordnen: Feste Zeitschaltungen mit Anpassungen je Tageszeit, vorhersagende Signalschaltung aufgrund von Eingabedaten aus vergangenem oder aktuellem Verkehrsfluss, Betätigungssignalschaltung mit der bei Aktivierung von Sensoren Grün- oder Rotphasen verlängert werden und die Adaptivschaltung, die mit Sensoren und mit Algorithmen den Verkehr zum Zeitpunkt des Eintretens die Schaltungen verändern[ZMJZ19].

Ihr Ansatz mit der künstlichen Intelligenz fällt unter eine neue Lösungsstrategie, mit der sie Lichtsignalphasen bei einzelnen Kreuzungen über das verstärkte Lernen abstimmen wollen[ZMJZ19]. Beispielsweise sind gegenüberliegende, geradlinig gerichtete Straßenspuren, wie in der Grafik 2.1 bei zum Beispiel den Spuren 6 und 2 zu sehen ist, miteinander verknüpfbar als eine Lichtsignalphase, da sie keine Überkreuzung ihrer Fahrbahn haben. Um diese Phaseneinteilung für die künstliche Intelligenz vorzubereiten, sollen die einzelnen Fahrbahnen in Kombination mit Fahrzeugdaten in Gruppen unterteilt werden[ZMJZ19].

Damit ließe sich der Algorithmus dann noch weiter verfeinern, als dass sie seltene oder fast nie auftretende Szenarien aus der Menge möglicher Zustände entfernen und sich nur die einzelnen Zahlen als Eingabe erhalten muss, nicht die gesamte Struktur der Kreuzung.

Im Folgenden gehen sie auf weitere Arbeiten ein, die eine andere Größenskalierungen als sie vorgenommen haben, bevor sie dann auf die genaue Implementation des Straßenmodells und dem Lernen beziehungsweise Trainieren der künstlichen Intelligenz eingehen.

Ein Problem bei der Phaseneinteilung ist aber, dass sie bei einzelnen Kreuzungen annahmen, dass jede Kreuzung je 3 eingehende Straßen hat, die in die Kreuzung münden.

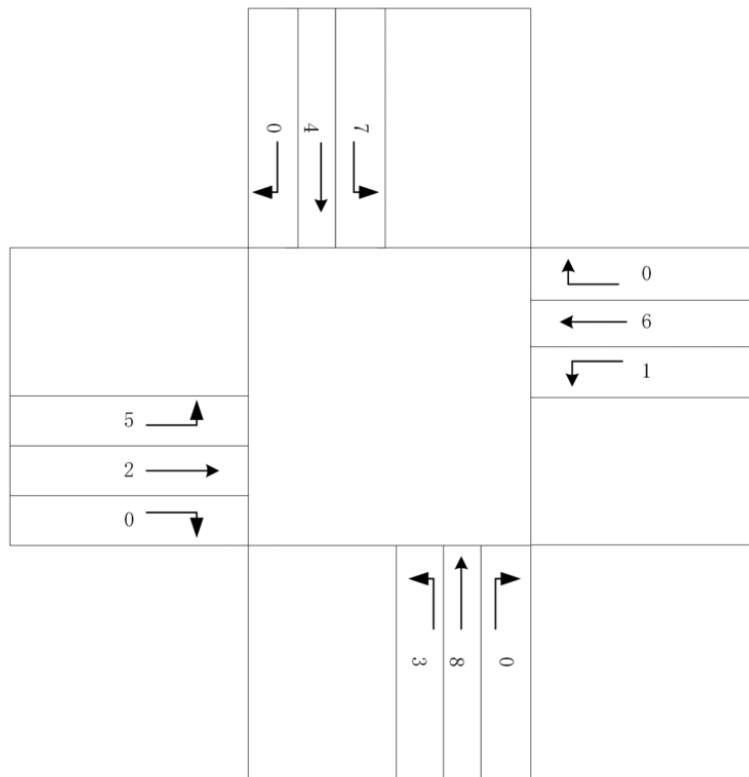


Abbildung 2.1: Das von Kuang und andere bei einer 3-spurigen Kreuzung dargelegte 8-Phasen Modell [ZMJZ19]

Auch wenn die Stauzonen meist große Kreuzungen sind, so haben Städte auch Lichtsignalschaltungen verschiedene Kreuzungen mit mehr als nur 3 oder teilweise auch nur 2 Spuren, sodass ein Q-Learning-Algorithmus nicht mit solchen Szenarien umgehen kann und bei anderen Kreuzungen von vorne konzipiert und trainiert werden muss.

Dennoch ist die Einteilung der verschiedenen Lösungsansätze ein wichtiger Aspekt, der auch in dieser Arbeit Relevanz hat, explizit der erste Typ, die feststehende Zeitschaltung.

Valentina Kurtc und Martin Treiber: *Simulating bicycle traffic by the intelligent-driver model-Reproducing the traffic-wave characteristics observed in a bicycle-following experiment*

Der Forschungsbeitrag von Kurtc und Treiber aus dem Jahr 2020 untersucht die Hypothese, dass sich die Bewegungsdynamik des Fahrzeugverkehrs qualitativ nicht unterscheidet von dem Fahrradverkehr, indem sie die Qualität eines Fahrzeugmodells beziehungsweise

Pkw-Modells ebenso für Fahrräder nutzen können. Dies beweisen sie mit einem „Intelligent Driver Model“[KT20], einem mikroskopischen Modell für Autos, und vergleichen dessen Qualität der Kalibrierung und Vorhersagefähigkeit dann mit dem „Necessary Declaration Model“[KT20], einem mikroskopischen Fahrradmodell.

Im Folgenden gehen Kurte und andere darauf ein, wie sie eine Vergleichsbasis über die Formel herstellen und das anhand der Beschleunigungsfunktionen aus den Modellen aufbauen, um dann ein Kreisverkehrsszenario mit einem Fahrraddatenset zu simulieren und die berechneten Ergebnisse zu vergleichen.

Model	Andresen et al. (2014)		Jiang et al. (2016)	
	$\sqrt{S^{\text{abs}}}$	$\sqrt{S^{\text{rel}}}$	$\sqrt{S^{\text{abs}}}$	$\sqrt{S^{\text{rel}}}$
IDM	1.92	1.95	25.64	24.85
NDM	4.53	4.58	23.40	23.30

Abbildung 2.2: die von Kurte und Treiber berechneten, durchschnittlichen Kalibrierungsfehler der beiden Modelle in % [KT20]

Das Ergebnis aus der Grafik 2.2 zeigt, dass die Unterschiede der Modelle beim Nutzen von Fahrraddaten klein sind, als dass sie keinen großen Effekt auf die Simulationen haben.

Aus Kurte und Treibers Forschungsarbeit lässt sich für diese Simulation also ableiten, dass Agenten, die auf der Straße fahren und Modalitäten wechseln, sowohl mit den Bewegungsmodellen von Fahrrad und Auto simuliert werden können, ohne an Akkuratheit einzufügen zu müssen.

Saif Islam Bouderba und Najem Moussa: *Reinforcement Learning (Q-LEARNING) traffic light controller within intersection traffic system*

In dem Thesenpapier von Bouderba und Moussa aus dem Jahr 2020 wird eine ähnliche Hypothese wie die von Kuang und andere untersucht. Sie untersuchen die Effektivität von drei Lösungsansätzen für ihr zelluläres Automatenmodell: Bei dem ersten Experiment simulieren sie einen einfachen, synchronisierten Ablauf der Lichtsignalschaltungen, beim Zweiten einen Ansatz einer „Grünen Welle“-Schaltung mit aufeinanderfolgenden Ampeln und beim Dritten eine durch Q-Learning-Algorithmus gesteuerte Kreuzungen [BM19].

Ihr Automatenmodell besteht dabei aus einer Matrix an Kreuzungen, die alle von der Position her mit ihren umliegenden Nachbarn über eine zweispurige Straße verbunden sind:

$$N \times N = 4$$

Jede Kreuzung hat also vier Eingangs- und Ausgangspunkte sowie eine Lichtsignalanlage, die mit je einer Lösungsstrategie pro Experiment gesteuert wird. Zudem ist die Simulationsumgebung aufgeteilt in Zellen, auf denen sich die Agenten entlangbewegen und zu den Kreuzungen gelangen [BM19].

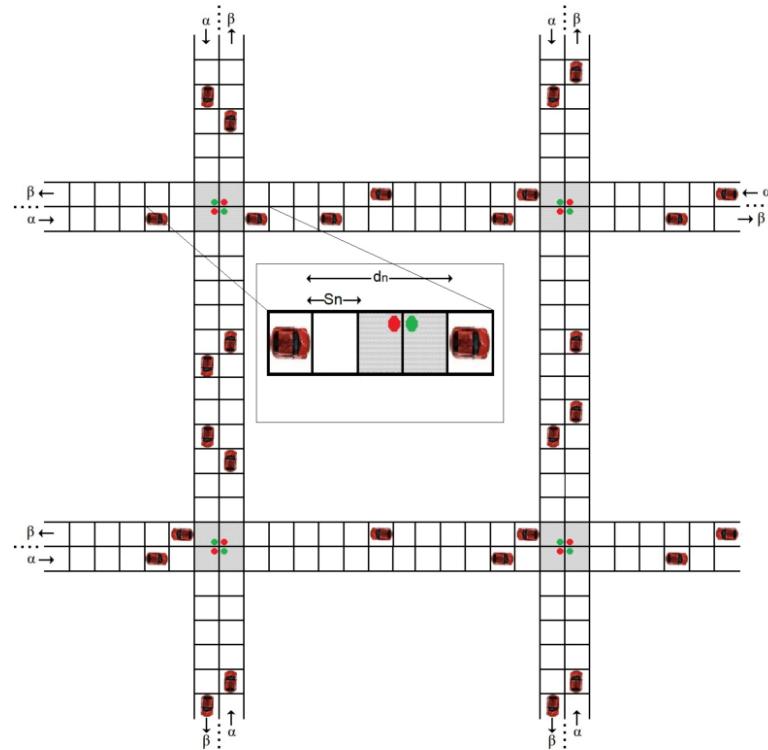


Abbildung 2.3: Eine Momentaufnahme aus dem zellulären Automatenmodell [BM19]

Mit dem Modell 2.3 als Simulationsgrundlage erläutern Bouderba und Moussa die Lösungsstrategien. Die grüne Welle Synchronisation erfolgt über eine Verschiebung der aktuellen, internen Zeituhr von umliegenden Lichtsignalen, während der Q-Learning-Alhorithmus trainiert wird, um eine passende Schaltung selbst zu erlernen.

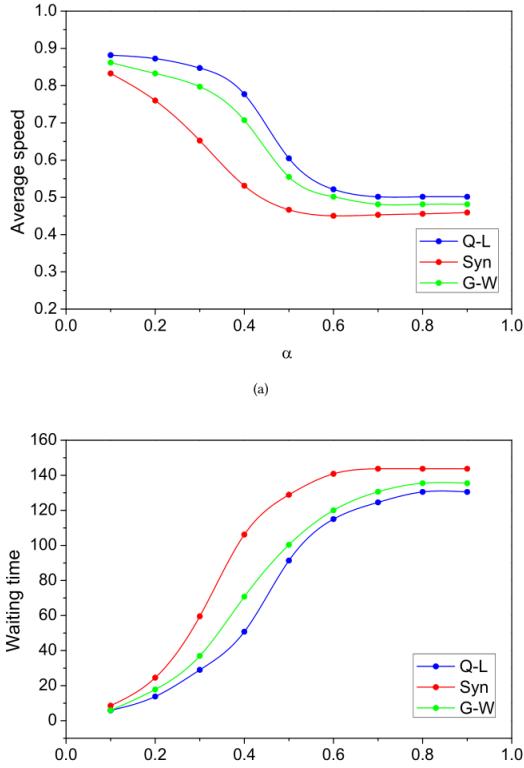


Abbildung 2.4: Die durchschnittliche Geschwindigkeit und Wartezeit nach Durchführungs [BM19]

Die Resultate in Geschwindigkeit und Wartezeit pro Eingriffsräte aus Grafik 2.4 zeigen auf, dass der einfache, synchronisierte Ansatz am schlechtesten von allen drei Algorithmen abschneidet. Die durchschnittliche Wartezeit ist höher als die von dem Q-Learning-Algorithmus und der grünen Welle, genauso wie die durchschnittlichen Geschwindigkeitsmessungen niedriger ausfällt als bei den anderen beiden Ansätzen.

Auch wenn die Grüne-Welle-Schaltung in den Experimenten besser abschnitt, so ist dieser Ansatz nicht der dieser Arbeit. Der Unterschied besteht darin, dass in Bouderba und Moussas Forschungsarbeit die Ampelphasenlängen auf ihrer Distanz basierend verlängert wurden. Dies soll in dieser Arbeit aber durch eine insgesamt geltende Phasenschaltung ersetzt werden, da Agenten hier mehr als nur Geschwindigkeit erhöhen und senken können. Sie können zum Beispiel kleine Veränderungen an Routen unternehmen, sodass sie nicht auf der Straße, sondern auf Fahrradwegen vorbeifahren und Lichtsignalschaltungen ignorieren. Diese Bewegungsfreiheit der Agenten ist in dem Modell von Bouderba und

Moussa nicht gegeben, weshalb die Forschungsarbeit nur als Motivation für die Verbesserungsmöglichkeiten einer grünen Welle angesehen werden kann.

Katharina Mulack: *Multiagenten Simulation von Fahrradfahrern im Kontext urbaner Verkehrs dynamik*

Die Masterarbeit von Katharina Mulack aus dem Jahr 2020 basiert ebenso auf dem MARS-Framework und ergänzt die bis zu dem Zeitpunkt der Arbeit in dem Framework fehlenden Fahrradfahrer. Dabei fokussiert sich die Arbeit auf die detailreiche Rekonstruktion von Fahrradfahrern mit einem eigenen Bewegungsmodell, Statistiken über Eigenschaften von Fahrrädern und deren Interaktion mit der Umgebung[Mul20]. Bei den Verkehrsflussmodellen, einem Kernaspekt der Forschungsarbeit, wird insbesondere auf vier unterschiedliche Modellierungskonzepte eingegangen, die sich in der Feinheit der Simulationsmodelle unterscheiden:

Makroskopische Flussmodelle, die bei Simulationen sich mit dem allgemeinen Verlauf beschäftigen und die Details wie die Simulation einzelner Fahrzeuge nicht simulieren, um das Gesamtverhalten und ihr Effekt der Bewegung zu beobachten[Mul20].

Mikroskopische Modelle wiederum dienen der detaillierteren Simulation und beziehen individuelle Verhaltensweisen von Agenten mit der Umwelt oder anderen Agenten ein[Mul20].

Submikroskopische Modelle wiederum sind noch spezifischer und simulieren sogar Beziehungen zwischen Agenten und Entitäten, um den Detailgrad der echten Welt zu imitieren[Mul20].

Zuletzt wird noch das Mesoskopische Modell genannt, welches eine Mischung aus Makro- und Mikroskopmodell ist. Bei diesem werden sowohl über große Distanzen und Gesamtverhalten angeschaut werden, dennoch aber die einzelnen Agenten oder Entitäten diese große Veranschaulichung ausmachen[Mul20].

Im Folgenden untersucht Mulacks Masterarbeit die Verkehrsflussmodelle und Statistiken über Fahrräder und implementiert sie zusammen mit neuen Umwelteinflüssen, den Lichtsignalanlagen.

Mulacks Forschungsarbeit selbst befasst sich mit einem hohen Detailgrad und zwischen-agentlichen Beziehung bei den Fahrradfahrern, während sich diese Arbeit hier eher mit einem gemischten, mesoskopischen Simulationsmodell beschäftigt. Zwar wird in dieser

Arbeit ebenfalls auf einzelne Agenten benutzt, die mit ihrer Umwelt und den Entscheidungen andere Agenten beeinflussen, aber ist die generelle Stau- und Warteschlangenbildung wichtiger für die Ergebnisse.

Auch beschäftigt sich die Arbeit mehr mit dem Verhalten der Fahrradfahrer selbst innerhalb des MARS-Frameworks, nicht mit der Entwicklung einer optimalen Lichtsignalschaltung. Dies macht diese Masterarbeit zu einer geeigneten Quelle für detaillierte Verhaltensweisen von Fahrrädern, weniger aber zu einem vergleichbaren Ansatz für diese Arbeit.

Thomas Clemen, Nima Ahmady-Moghaddam, Ulfia A. Lenfers, Florian Ocker, Daniel Osterholz und Jonathan Ströbele: *Multi-Agent Systems and Digital Twins for Smarter Cities*

In der Forschungsarbeit aus dem Jahr 2021 beschäftigen sich Clemen und andere mit dem „Internet of Things“, fortan als IoT abgekürzt, welches eine Großzahl an (Echtzeit-) Daten über die Stadt Hamburg via Sensoren anbietet, und entwickeln dazu ein digitales Zwillingsmodell der Stadt selbst, zur Überführung der Daten in das MARS-Framework. Im Zentrum dabei steht nicht nur die Nutzungsermöglichung der IoT-Daten, sondern ebenfalls die Frage: Wie baut man ort- und zeitlich gebundene Daten in ein Echtzeitmodell, in der sich die Simulationen und Experimente anschaulich darstellen und kontrollieren lassen sollen [CAML+21]?

Ihr Lösungsweg: Das Herunterbrechen der Akteure in der echten Welt auf Agenten und Entitäten, die damit als eine digitale Zwillingsinstanz [CAML+21] fungieren und das Verhalten der Akteure oder Simulationsflächen imitieren. Dabei wird zudem noch ein Fokus darauf gelegt, zwischen passiven, aktiven und interagierbaren Zwillingsinstanzen zu unterscheiden, da der Verhaltens- und Implementationsaufwand durch die Entscheid- und Bewegungsfreiheit steigt. Entsprechend entscheiden sie sich, in ihrer Forschung vorerst die Echtzeitdaten nur bei passiven Zwillingsinstanzen zu implementieren. Als Grundlage dafür wird das SmartOpenHamburg-Projekt genommen, was bereits die physische Repräsentation der Stadt sowie eine Reihe der städtischen Modalitäten implementiert hat, und erweitern dieses um den Zugang zu den IoT-Daten [CAML+21].

Dadurch, dass in der Forschungsarbeit von Clemen und anderen nur die passiven, digitalen Zwillinge mit Daten angereichert werden, fehlt für die Arbeit hier die Echtzeitdaten von Lichtsignalschaltungen sowie einem Echtzeitverkehrssensor. Da in der Arbeit von diesem Papier hier sich auf den Verkehrsfluss und explizit auf die Simulation einzelner

Agenten fokussiert wird, müssen diese aktiven Agenten sein, die Entscheidungen über Modalitäten, Wege und Interaktion mit Entitäten unternehmen. Damit würden sie unter die interagierbaren, digitalen Zwillinge fallen, die mit Eingabedaten bestückt werden müssten und sind entsprechend mit einem großen Mehraufwand beim Implementieren verbunden.

Entsprechend ist die Forschung der Arbeit von Clemen und andere zwar eine gute Grundlage für zukünftige Ausblicke dieser Arbeit, sollten Echtzeitdaten bereitgestellt werden für aktuellen Verkehr an Lichtsignalanlagen oder generell auf Straßen. Jedoch ist der Nutzen für diese Arbeit beschränkt auf das SmartOpenHamburg-Projekt, dass die Grundlagen zur agentenbasierten Simulation bereits bereitstellt.

Daniel Glake, Fabian Panse, Norbert Ritter, Thomas Clemen und Ulfia Lenfers: *Data Management in Multi-Agent Simulation Systems*

Daniel Glake und andere beschäftigten sich im Jahr 2021 in ihrem Forschungsbeitrag mit den Problemen des Einbauens von unterschiedlichen Daten in einem sehr groß skalierten Simulationsszenario und präsentieren dabei Lösungen für die Herausforderungen mithilfe des MARS-Frameworks. Dabei fokussieren sie sich auf fünf großen Eingabedatenkategorien, die in der MARs-Simulationsumgebung eingebaut werden und mit großskalierten Projekten zu Herausforderungen kommen können:

- Eingabedaten für Simulationen, die zum Beispiel bei einem Ortstransfer zu Kompatibilitätskomplikationen führen, wie etwa mit geografischen Karten, den lokalen Infrastrukturnetzwerken oder weitere, nicht-ortsgebundene Daten wie zeitliche Busfahrpläne [GPR⁺²¹].
- Ausgabedaten von großen Simulationen, die beim Simulieren eine Großzahl an Agenten und Entitäten nutzen, blockieren beziehungsweise verzögern beim Exportieren von Zwischenständen die aktive Simulation deutlich und schränken damit die Fähigkeit diese in Echtzeit zu analysieren ein [GPR⁺²¹].
- Eingabedaten über Streams werden beim Vorhersagen von Simulationszuständen, wie etwa den Agentattributen oder Umweltinformationen, immer ungenauer je weiter in die Zukunft berechnet wird. Dafür bieten Schnittstellen und angebundenen Systeme Möglichkeiten zum vermindern dieser Ungenauigkeit. Das Korrigieren benötigen aber dennoch bei einer großen Anzahl von Agenten einem Moment zum Überreichen, was zeitlich ebenfalls zu Verzögerungen führen und auch die Echtzeitanalysefähigkeit erschwert [GPR⁺²¹].

- Bei Schnittstellen für räumlich-zeitliche Abfragen wird stets der jeweilige Raum benötigt und zum Beispiel in MARS als Polygon angegeben. Operationen wie „beinhaltet“, „überlappt“ oder „ist adjazent von“ sind dabei häufig genutzt, was bei einer Vielzahl von Agenten, die dies gleichzeitig durchführen und sich dabei bewegen, zu zeitlichen sowie geographischen Unterschieden führt [GPR⁺21].
- Beim Planen dieser räumlich-zeitlichen Abfragen, wie das System die echte Welt auf die Modellrepräsentation abzubilden hat, darf die Migrationszeit nicht die Datenverarbeitungszeit überschreiten, damit es nicht zu Inkonsistenzen in der Simulation führt. Strategien, die das verhindern, sind aber nur Annäherungen, da die Optimierung der Migrationszeit einem NP-Problem entspricht und somit nicht mit einer konstanten Anzahl an Operationen berechnet werden kann [GPR⁺21].

Danach gehen Glake und andere genauer darauf ein, wie sie diese Problematiken im MARS-Framework mit Lösungsansätzen vermindert oder gelöst bekommen haben. Für diese Arbeit hat die Forschungsarbeit von Glake und andere aber keinen relevanten Bezug: Das System hier ist kein Echtzeitsystem auf einem großskalierten Bereich. Der Simulationsraum beschränkt sich auf die Binnen- und Außenalster und zur Ergebnisermittlung genügt es bereits, wenn die Simulation nebenbei berechnet wird. Eine Verzögerung aufgrund von groß skalierten Bereichen oder zu vielen Agenten affektiert nicht die Korrektheit des Systems und ist sich also nicht von den kategorisierten Problemen affektiert.

Dennoch wären für zukünftige Arbeiten das Papier von Glake und andere ein guter Ansatz, um Echtzeitdaten mit der gesamten Stadt Hamburg zu verbinden und damit den gesamten Raum zu simulieren, nicht nur wie hier um die Binnen- und Außenalster.

Ulfia A. Lenfers, Nima Ahmady-Moghaddam, Daniel Glake, Florian Ocker, Daniel Osterholz, Jonathan Ströbele und Thomas Clemen: *Improving Model Predictions—Integration of Real-Time Sensor Data into a Running Simulation of an Agent-Based Model*

In einem weiteren Forschungsartikel über das MARS-Framework im SmartOpenHamburg-Projekt des Jahres 2021 beschäftigen sich Ulfia A. Lenfers und andere mit der Verbesserung der Einbindungimplemetation von Echtzeitdaten und zeigen das Potenzial anhand von Fahrradverleihstationen, wie mit der Modellierung und Daten genauere Vorhersagungen zu Fahrradverleihs getroffen werden können.

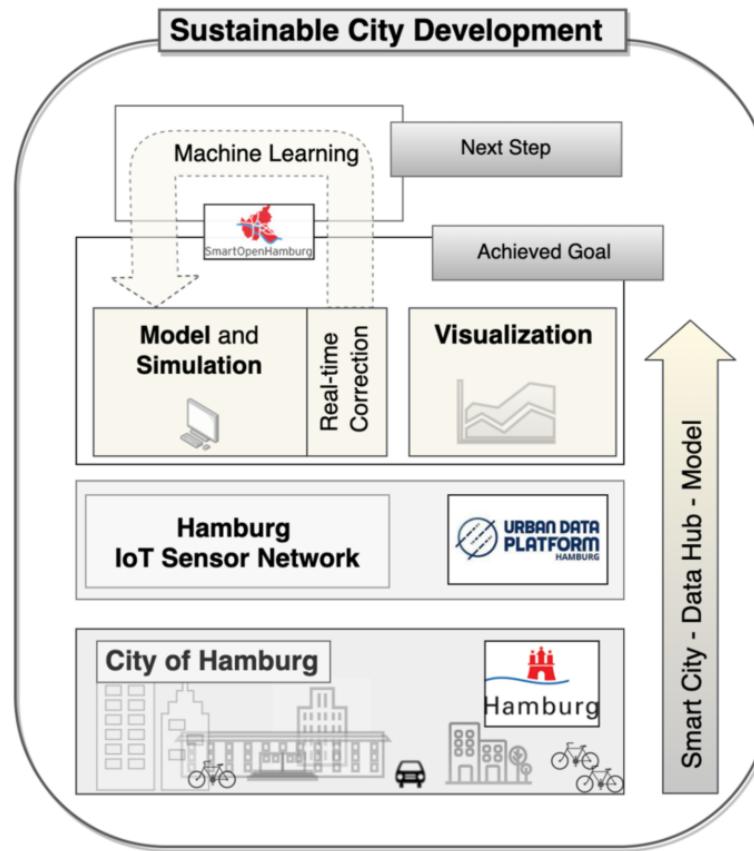


Abbildung 2.5: Beispielworkflow, der Daten aus den Sensors und von der Stadt Hamburg bereitgestellten Daten einbaut in ein abstraktes Simulationsmodell [LAMG⁺21a]

Wie aus dem Workflow von 2.5 zu entnehmen ist, fokussiert sich der Kern der Arbeit aus dem Zusammenspiel von Datenfluss des Smart-City-Data-Hub und die Dateneinbindung von Fahrradverleihstationen in die SmartOpenHamburg-Komponente. Echtzeitdaten werden bei ihrer Implementation von der „Smart City“ über ein zeitlichen Vektor-Layer in die Simulation konstant eingelesen und anhand ihrer Lebenszeit entsprechend simuliert[LAMG⁺21a]. Die Daten selbst werden dabei über eine URI von den externen Endpunkten abgefragt und mit zusätzlichen Argumenten auf die relevanten Daten reduziert . Mit diesem Datenstrom werden die derzeitigen Zustände der Fahrradverleihstationen abgefragt und in die Simulation eingebaut, wie viele Fahrräder derzeit aktuell aktiv genutzt werden und wie viele Agenten in der Simulation agieren[LAMG⁺21a].

Der Forschungsbeitrag von Lenfers und andere ist aber, wie bei der vorherigen, für diese Arbeit nur begrenzt relevant: Die statische Implementation von Fahrradverleihstationen ist relevant, da Verkehrsteilnehmer an den Stationen sich ein Fahrrad leihen können, doch ist dabei kein Echtzeitstrom vonnöten und passiert passiv nebenbei.

Dennoch bietet die Arbeit Anknüpfmöglichkeit, sollten relevanten Echtzeitdaten von der Stadt Hamburg zum Verkehr oder zu Lichtsignalanlagen über die nächste Zeit aufkommen und damit den Kern einer anderen Arbeit darstellen.

Daniel Glake, Fabian Panse, Norbert Ritter, Thomas Clemen und Ulfia Lenfers: *Incorporating Multi-Modal Travel Planning into an Agent-Based Model: A Case Study at the Train Station Kellinghusenstraße in Hamburg*

Der Forschungsartikel von Lenfers und andere aus dem Jahr 2021 erweitert ebenfalls das MARS-Framework und das SmartOpenHamburg-Projekt um nachhaltige Modalitäten wie die mietbaren Äquivalente zu Pkws und Fahrrädern oder das Zu-Fuß-Gehen. Dabei untersuchen sie, wie effizient das Wechseln der Modalitäten in einem Beispielszenario ist und ob die Optimierungsstrategien hinsichtlich der Klimaneutralität, aus finanziellen Gründen oder persönlichen Präferenzen korrekt annähern [LAMG⁺21b].

Das Beispielszenario, welches sie zur Demonstrierung der implementierten Modalitäten verwenden, umfasst dabei die Bahnhofsstation Kelinghusenstraße in Hamburg, von der aus die Agenten mit den Modalitäten in einem Kreis-Bereich zu ihrem Ziel gelangen können.

Auf dem Weg zu ihrem Endziel innerhalb des simulierbaren Bereichs aus 2.6 kommen manche Agenten durch angestrebtes Nutzen der Modalitäten erst zur nächsten Fahrrad- oder Pkwverleihstation, bevor sie ihre Reise zum Interessenspunkt fortsetzen können. Dieses angestrebte Nutzen wird in den Eingabedaten durch zum Beispiel die Variablen „hasCar“ und „usesBikeAndRide“ versinnbildlicht, wie hoch die Wahrscheinlichkeit zum Nutzen oder Besitzen einer eigenen oder gemieteten Modalität ist [LAMG⁺21b].

Auf den implementierten Modalitäten und Verleihstationen aus der Forschungsarbeit von Lenfers und andere knüpft diese Arbeit hier an und untersucht nun mithilfe der Auswahl an Modalitäten eine mögliche Lichtsignalschaltung. Damit bietet diese Arbeit eine sehr gute Grundlage, auf der nun aufgesetzt und das hier in dieser Arbeit beschriebenen Szenario aufgesetzt wird.



Abbildung 2.6: Karte der Fahrradverleihstationen, mit dem Simulationsbereich der Agenten hellblau hinterlegt, den Interessenspunkten rot markiert und Fahrradverleihstation als blaue Markierungen [LAMG⁺21b]

3 Konzept

Im Folgenden wird das Modell der Simulation konzeptioniert und ausgeführt, mit den vorherigen Begriffserklärungen als Grundlage. Dabei wird auf den Simulationsort im Modell, den Aufbau und Voreinstellungen der Agenten und Lichtsignalschaltungen, die Projektarchitektur und zuletzt noch auf den Umfang der Arbeit eingegangen.

3.1 Simulationstyp

Für diese Arbeit empfiehlt sich eine agentenbasierte Simulation, da im Vornherein die zu fahrende Route eines Agenten zwar bekannt ist, die Einflüsse von zum Beispiel Pkws aber direkt im Übergang von einer Lichtsignalschaltung zur anderen den Akteur beeinflussen und so nicht vorhergesehen werden können. Damit also keine Annahmen über dynamische, umwelt- oder agentenbedingte Einflüsse im Quellcode festgelegt werden, ist entsprechend eine agentenbasierte Echtzeitsimulation angemessener für diese Simulation.

3.2 Simulationsort

Für die Arbeit wurde die Binnen- und Außenalster als Simulationsumgebung ausgewählt.

Wie im Ausschnitt 3.1 zu erkennen ist, ist aufgrund der zentralen Lage innerhalb der Stadt, vieler anliegenden Kreuzungen und generell dichter Besiedelung um die Alster herum eine hohe Dichte an Agenten für die Simulation gegeben, die eine „Grüne Welle“ für Fahrradfahrer erschweren können. Dies gibt der Simulation auf der einen Seite eine Herausforderung, mit schwereren Vorgaben überhaupt eine Lichtsignalschaltung für die „Grüne Welle“ zu finden, während auf der anderen Seite dafür aber diese Simulation wie eine „Obergrenze“ angesehen werden kann für die Agenten- und Lichtsignalanzahlen.

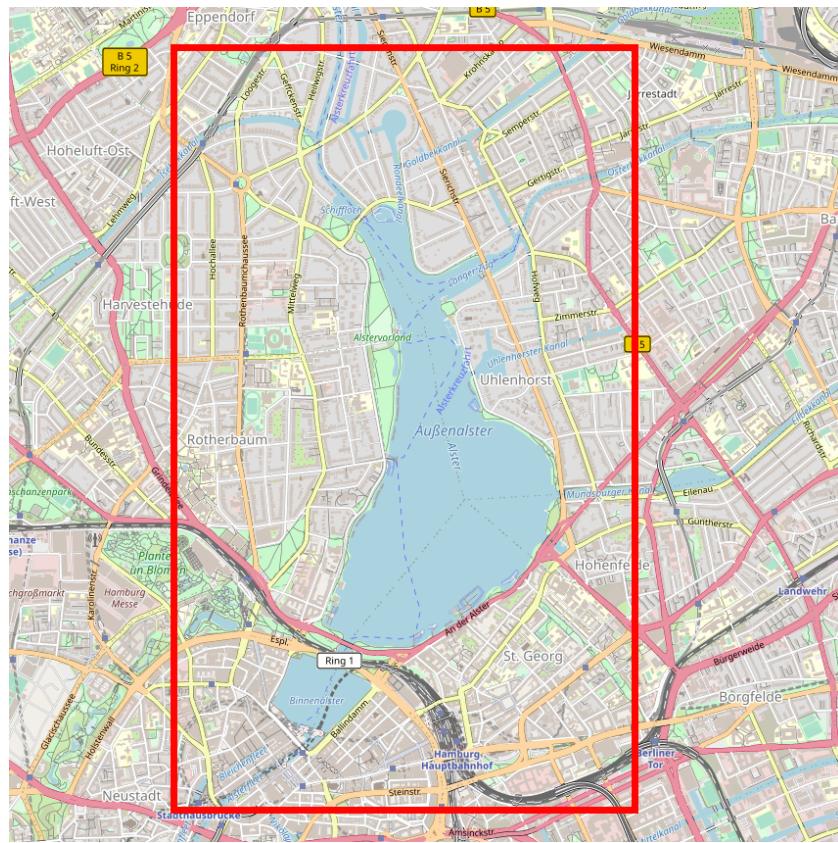


Abbildung 3.1: Simulationsumgebung

3.3 Agenten

Im Folgenden werden die Konzepte der Agenten näher erläutert, wobei besonders auf die Grundlagen der Agenten, ihre Modalitäten, die Interaktion mit anderen Agenten und Entitäten, die Anzahl an aktiven Agenten und auf die gefahrene Routen eingegangen wird.

3.3.1 Voraussetzungen

Es gibt zwei Arten von Agenten, die in der Simulation agieren: Den Nebenagenten, fortan „HumanTraveler“ genannt, und dem Hauptagenten, fortan „BicycleLeader“ genannt.

3 Konzept

Der **BicycleLeader** ist der Fokus dieser Arbeit. Dieser wird immer mit einem eigenen Fahrrad oder einem mietbaren Fahrrad ausgestattet, mit dem er vorgegebene Punkte auf einer Route um die Alster abfahren soll. Sein Ziel ist es, eine Runde um die Alster zu fahren, ohne dabei auf 0 km/h bremsen zu müssen. Langsames Fahren ist hier nicht mit einbezogen als Fehlschlagbedingung, da eine Schwelle für „zu langsames“ Fahren, sodass der **BicycleLeader** sein Gleichgewicht nicht mehr halten könne, von einer Reihe von Faktoren abhängt, die außerhalb des Rahmens dieser Arbeit wären: das Alter des **BicycleLeaders**, die Erfahrung mit dem Fahrrad, das angestrebte Fahrverhalten, Höhenprofile der Umgebung, Wetterbedingungen und noch einige Aspekte mehr.

HumanTraveler sind dabei die Einwohner, die mit Lichtsignalschaltungen interagieren und auf den Straßen dem **BicycleLeader** in die Quere kommen. Das Ziel der **HumanTraveler** ist es lediglich, ein zufällig zugewiesenes Ziel um die Alster herum zu erreichen, bevor sie aus der Simulation entfernt werden.

3.3.2 Modalitäten

Die **HumanTraveler** haben drei Arten der Transportation zur Verfügung: zu Fuß, Pkws und Fahrräder. In dem MARS-Framework ist es Agenten ebenso gestattet, bei Autos und Fahrrädern diese zu Mieten und damit sogenannte „RentalCars“ oder „RentalBikes“ zu nutzen, jedoch hat es für diese Arbeit keinen großen Einfluss, ob sie ihr eigenes Transportmittel nehmen oder einen Umweg zu den mietbaren Äquivalenten einschlagen, da die Agenten in der Simulation nur als „Störfaktor“ über Ampeln und auf Straßen mit dem Hauptagenten interagieren.

Für jede Modalität gibt es eine vorgesehene Straße zum Befahren: Pkws fahren auf Straßen, Fahrräder fahren auf Fahrradlinien und teilweise auch auf Straßen. Fußgänger können sich nur auf Fußgängerwegen bewegen, dafür können sie in andere Modalitäten wechseln. Jeder **HumanTraveler** als auch der **BicycleLeader** beginnt die Simulation als Fußgänger, bevor sie auf eine Modalität aufsteigen.

3.3.3 Interaktionen mit Agenten und Entitäten

Interaktionen zwischen Agenten sind in dieser Simulation beschränkt auf zwei Arten: dem Verlangsamen und Blockieren auf Straßen als auch dem Aufhalten von anderen Agenten an Lichtsignalschaltungen. **HumanTraveler** und **BicycleLeader** können über ihre

Pkws und Fahrräder an Ampeln einen Platz einnehmen und damit die Warteschlangen verlängern. Je länger sie wird, desto mehr Zeit benötigt die Simulation, um die Warteschlange bei einem grünen Signal zu leeren.

Auf den Straßen und Fahrradwegen selbst haben die jeweiligen Modalitäten nur dann miteinander Interaktionen, wenn sie sich zu Nahe kommen. Damit es nicht zu Kollisionen kommt, wird überprüft, wie nahe sich zwei Agenten sind, um dann zu verlangsamen oder weiterzufahren. Eine detailliertere Abhandlung dazu wird im Kapitel „Implementierung“ angegangen.

3.3.4 Anzahl aktiver Agenten

Zur Bestimmung von der Anzahl an aktiven Agenten, wurde eine angenäherte Rechnung für die Population um die Alster genommen, da direkte Statistiken zur täglichen Anzahl an Verkehrsteilnehmern in Hamburg beziehungsweise in verschiedenen Stadtvierteln fehlen. Um eine Annäherung an die Verkehrszahlen zu bekommen, wird zuerst die Verteilung über Haupt-, Neben- und Schwachverkehrszeiten untersucht.

Ein Arbeitstag, Montag bis Freitag, hat zwei große Hauptverkehrszeiten: morgens von 6 bis 9 Uhr und nachmittags von 15 bis 19 Uhr [uHHBfWVuIAfVuS15]. In diesen beiden Zeiten werden die meisten Verkehrsteilnehmer auf den Straßen unterwegs sein und damit am ehesten Staus verursachen.

Die Nebenverkehrszeit tritt von 9 bis 15 Uhr [uHHBfWVuIAfVuS15] auf, die den Übergang zwischen den beiden Hauptverkehrszeiten darstellt. Innerhalb dieser Zeit ist die Dichte an Verkehrsteilnehmern geringer als in der Hauptverkehrszeit, aber immer noch höher als in Schwachverkehrszeiten.

Die Schwachverkehrszeit ist von 20 Uhr bis 5 Uhr am nächsten Tag [uHHBfWVuIAfVuS15], in der der Verkehr am geringsten ist, die Straßen leer und der Stau am seltensten auftritt.

Um den Verlauf über einen Tag nun mit zwei Hochpunkten und drei Tiefpunkten darzustellen, während dabei der zweite Tiefpunkt höher als der erste und dritte ist, könnte man eine negierte Funktion 4. Grades benutzen.

Die Funktion aus der Grafik 3.2 lautet:

$$f(x) = -0.01x^4 + 0.45x^2 + 7.5$$

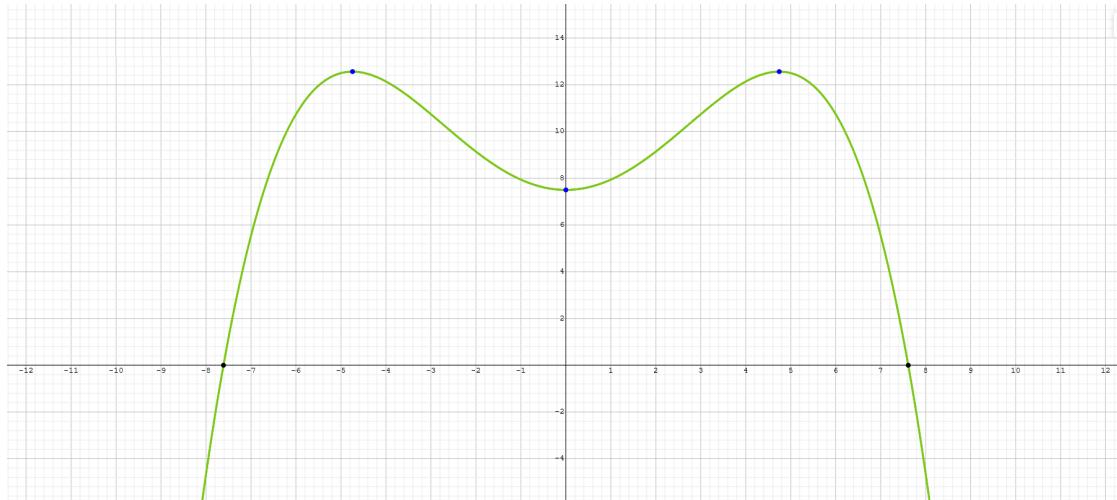


Abbildung 3.2: Polynomfunktion des 4. Grades zur Annäherung

Die Grafik 3.2 muss so interpretiert werden, als hätte die x-Achse eine Verschiebung von 12 Einheiten nach rechts noch erhalten, damit die Werte sich passend der Tagesstunden verteilen. Entsprechend ist 0 Uhr hier bei $x = -12$, 12 Uhr ist bei $x = 0$ und 24 Uhr ist bei $x = 12$.

Zum Annähern selbst ist die Funktion aber nur ansatzweise nützlich, da wenn x die Stundenanzahl und $f(x)$ der Prozentsatz aller aktiven Agenten darstellt, die Schwachverkehrszeiten bei einer Funktion 4. Grades auf der x-Achse in das Negative gehen würden. Damit wäre aber, wenn $f(x)$ Nullstellen bei

$$x = -12, x = 12$$

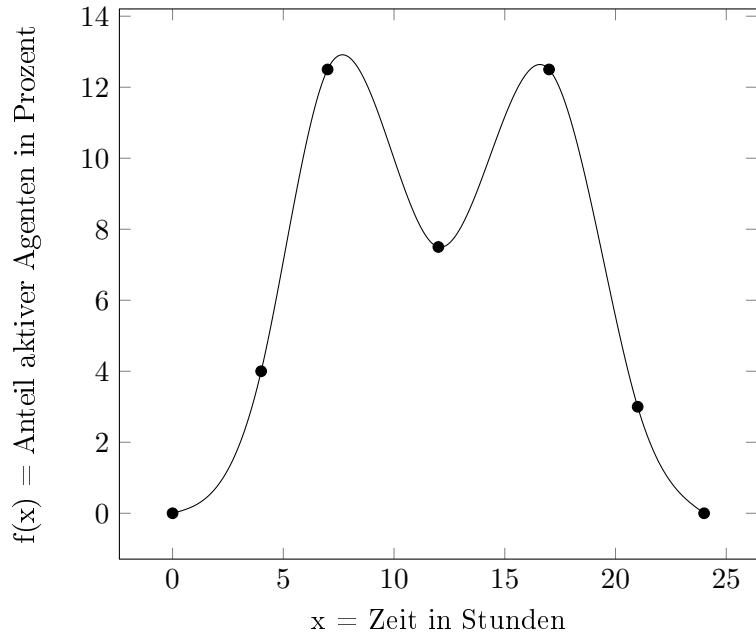
hätte, keine relativ gleiche Verteilung über die 9 Stunden, also wäre von 20 bis 5 Uhr keine relativ flache Schwachverkehrszeit. Der zweite Ansatz wäre, bei einem anderen x-Wert die Nullstelle schneiden zu lassen, wie es in der Grafik 3.2 zu sehen ist, also zum Beispiel an den Stellen

$$x = -7.6, x = 7.6$$

$x = -7.6$ wäre um circa 4 Uhr und $x = 7.6$ wäre um circa 20 Uhr. Bei diesem Ansatz werden dann aber negative Funktionswerte außerhalb der x-Werte auftreten und damit „negativen Verkehr“ verursachen. Das kann logischerweise nicht in der realen Welt auftreten, also müssen die Werte auf 0 oder einen anderen Wert gesetzt werden. Sollte der Wert auf 0 gesetzt werden, wäre das noch immer keine korrekte Darstellung für das Modell,

da in der Schwachverkehrszeit noch immer Verkehr vorliegt. Um einen besseren Funktionsverlauf zu gewährleisten, muss die Funktion interpoliert werden, um flachere Übergänge in die Schwachverkehrszeit zu bewerkstelligen. Um den Verlauf beziehungsweise die Interpolation zu konstruieren, werden kubische Polynome genutzt, die an bekannten Punkten eine Ober- beziehungsweise Untergrenze bilden und damit übergehen in die nächste, kubische Funktion. Mithilfe Timo Denks Implementation der kubischen Spline-Interpolation [Den18] lässt sich mithilfe der Datenpunkte aus der Voraabinformation von der Stadt Hamburg annähern.

$$f(x) = \begin{cases} 8.8699 \cdot 10^{-2} \cdot x^3 - 1.4163 \cdot 10^{-59} \cdot x^2 + 2.0171 \cdot 10^{-1} \cdot x + 0.0000, & \text{if } x \in [0, 3], \\ -2.0275 \cdot 10^{-1} \cdot x^3 + 2.6231 \cdot x^2 - 7.6675 \cdot x + 7.8692, & \text{if } x \in (3, 6], \\ 9.4824 \cdot 10^{-2} \cdot x^3 - 2.7333 \cdot x^2 + 2.4471 \cdot 10^1 \cdot x - 5.6408 \cdot 10^1, & \text{if } x \in (6, 12], \\ -8.5660 \cdot 10^{-2} \cdot x^3 + 3.7641 \cdot x^2 - 5.3498 \cdot 10^1 \cdot x + 2.5547 \cdot 10^2, & \text{if } x \in (12, 18], \\ 1.2958 \cdot 10^{-1} \cdot x^3 - 7.8589 \cdot x^2 + 1.5572 \cdot 10^2 \cdot x - 9.9981 \cdot 10^2, & \text{if } x \in (18, 22], \\ -1.1557 \cdot 10^{-1} \cdot x^3 + 8.3212 \cdot x^2 - 2.0025 \cdot 10^2 \cdot x + 1.6106 \cdot 10^3, & \text{if } x \in (22, 24]. \end{cases}$$



Nun muss nur noch die Einwohnerdichte pro km^2 angenähert werden, damit man eine grobe Gesamtanzahl an Agenten ableiten kann. Das Statistikamt Nord, das für Schleswig-Holstein und Stadtteile Hamburgs Statistiken sammelt, hat in ihrem Bericht aus dem Jahr 2021 die Einwohnerdichte aller Stadtteile Hamburgs aufgelistet [fHuSHSN22]. Der

3 Konzept

Stadtteil	Einw. je km ²
Neustadt	5575
St. Georg	6291
Hohenfelde	8578
Uhlenhorst	8516
Winterhude	7499
Eppendorf	9193
Harvestehude	8636
Harvestehude	6253
Gesamt	60.541

Tabelle 3.1: Die Einwohnerzahl pro km² für die genannten Stadtviertel

Simulationsort dieser Arbeit umfasst die folgenden Stadtteile: Neustadt, St. Georg, Hohenfelde, Uhlenhorst, Winterhude, Eppendorf, Harvestehude und Rotherbaum. Für diese Stadtteile wird die Einwohnerdichte pro km² ausgelesen:

Es werden absichtlich aus dieser Statistik manche Agentengruppen nicht entfernt oder hinzugefügt, da keine genauen Statistiken dazu von der Stadt Hamburg gegeben oder bei der Recherche aufgefunden werden konnten zum Verwenden in dieser Arbeit. Jene Gruppen umfassen, sind aber nicht beschränkt auf: Kinder, Eltern, die Zuhause auf Kindern aufpassen, Home-Office-Workers, Alte oder kranke Leute, Arbeitslose und noch mehr. Diese müssten theoretisch gesehen aus den Statistiken entfernt werden, während folgende Menschengruppen zu der Statistik hinzugezählt werden müssten: Pendler aus anderen Städten, Pendler aus anderen Stadtvierteln, Privatpersonen mit mehr als einem Fahrtziel, Berufsfahrer wie zum Beispiel Taxis oder Lieferanten, Touristen und so weiter. Der Einfachheit halber wurde also die durchschnittliche Einwohnerzahl der betroffenen Stadtviertel genommen, anstatt alle Umweltfaktoren aus der echten Welt einzubeziehen.

Mit der Gesamtmenge an Einwohnern pro km² lässt sich der Durchschnitt aller für dieser Simulation relevanter Stadtteile berechnen, der dann mit der Gesamtfläche des Simulationsortes multipliziert die Gesamtanzahl an aktiven Agenten ergibt. Der Flächeninhalt der Simulation ist mithilfe einer OpenStreetMap-Karte [OF23] berechenbar:

Aus der Grafik 3.3 lassen sich folgende Flächen ablesen:

3 Konzept

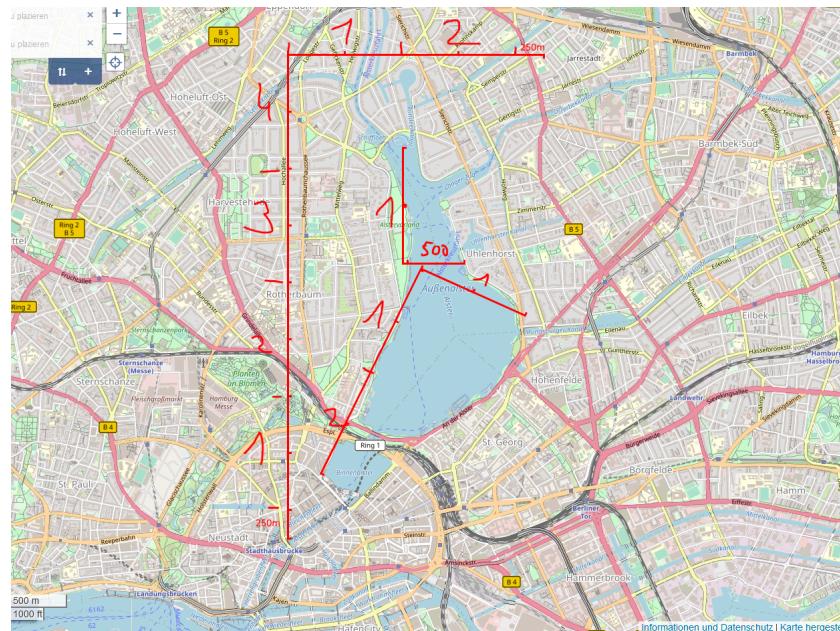


Abbildung 3.3: Grobe Flächenberechnung der bewohnbaren Bereiche

$$A = (4,25 \text{ km} * 2,25 \text{ km}) - (0,5 \text{ km}^2 + 2 \text{ km}^2) \quad (3.1)$$

$$= 9,56 \text{ km}^2 - 2,5 \text{ km}^2 \quad (3.2)$$

$$= 7,06 \text{ km}^2 \quad (3.3)$$

Zuletzt wird noch die Multiplikation der durchschnittlichen Einwohnerdichte mit dem Flächeninhalt berechnet:

$$GesamtanzahlAgenten = 7,06 \text{ km}^2 * ((60.541 \text{ Einw je km}^2)/8) \quad (3.4)$$

$$= 7,06 \text{ km}^2 * (7.567,625 \text{ Einw je km}^2) \quad (3.5)$$

$$= 53.427,4325 \text{ Einw je km}^2 \quad (3.6)$$

$$\approx 53.427 \text{ Einw je km}^2 \quad (3.7)$$

Mit den Voraussetzungen lässt sich für jeden beliebigen Zeitpunkt an einem Arbeitstag eine Annäherung mit der Interpolationsgleichung 3.3.4 berechnen. In der Simulation wird

3 Konzept

für jede Stunde, die neu angefangen wird, ein neuer Wert aus der Gleichung mit der durchschnittliche Einwohnerdichte multipliziert und damit als für diese Stunde, aktive Anzahl an Agenten festgelegt. In dieser Simulation werden nur Daten für jede Stunde genommen, da der **BicycleLeader** ebenfalls nur jede Stunde einmal die Route fährt.

3.3.5 Agentenrouten

In diesem Model haben **HumanTraveler** nur zwei Punkte auf ihrer Route, die sie erreichen müssen: Den Startpunkt, auf dem sie beginnen, und das Endziel, das ein beliebiger Punkt im Simulationsbereich ist. Diese Agenten haben, wie bei dem Unterkapitel „Modalitäten“ bereits erwähnt, dafür drei verschiedene Transportarten zur Verfügung, und sind nicht weiter eingeschränkt in der Art und Weise, wie sie zum Zielpunkt gelangen.

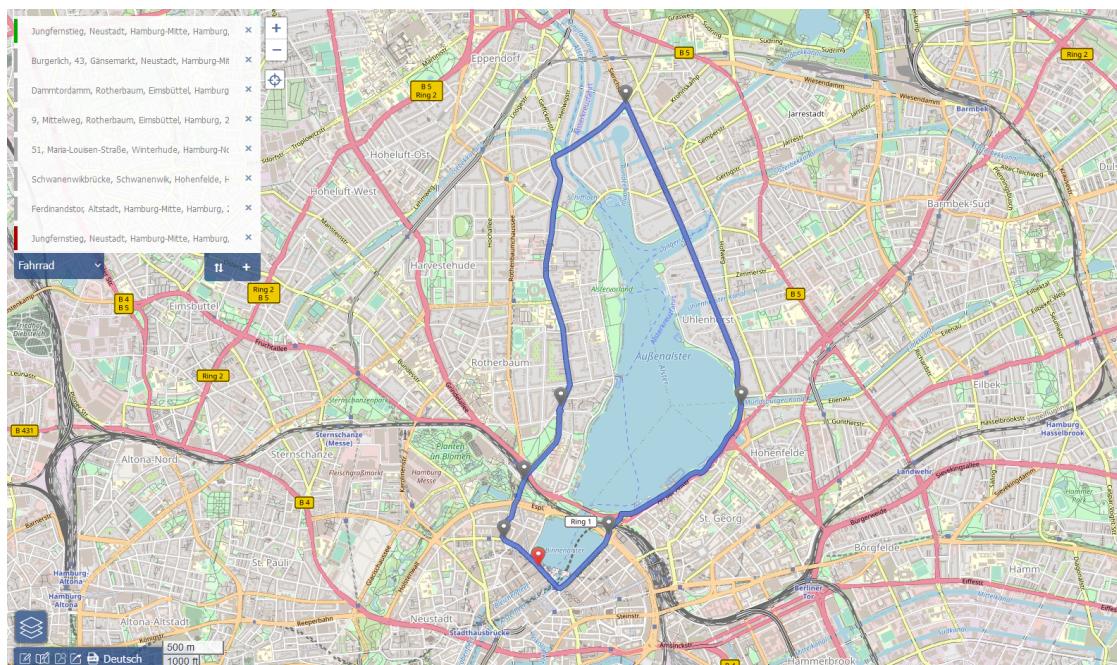


Abbildung 3.4: Route der Bicycle-Leader mit den 8 Zwischenpunkten

Der **BicycleLeader** wiederum, wie man der Grafik 3.4 entnehmen kann, hat 8 Punkte abzufahren, die um die Binnen- und Außenalster verteilt sind: Die Route beginnt und endet bei dem Café Alex, während die restlichen 6 Punkte verteilt um die Alster platziert sind, damit eine Rundfahrt bei möglichst vielen Lichtsignalschaltungen geplant sind.

3.4 Lichtsignalschaltungen

Die Lichtsignalschaltungen stellen die größte Barriere und der Fokus dieser Simulation: Ihre Aufgabe ist es, Pkws und Fahrräder aufzuhalten, wenn sie in der Rotphase sind und sie durchzulassen, wenn sie Grün sind.

3.4.1 Beschreibung der Lichtsignalanlagen

Während der Rotphase ist es möglich, dass sich mehrere Verkehrsteilnehmer an der Ampel anstauen und dadurch eine Warteschlange bilden, die sich erst bei Grün mit einer Geschwindigkeit von einem Verkehrsteilnehmer pro Sekunde leert. Bei einer Warteschlange von 20 Leuten ist es also der letzten HumanTraveler erst möglich, die Ampel hinter sich zu lassen, sobald alle 19 Verkehrsteilnehmer vor dem HumanTraveler nach 19 Sekunden durchgelassen wurden.

Die Positionsdaten der Ampeln im Simulationsbereich sind mit dem OpenStreetMap-Tool citeOSF2004 entnommen worden und dann für die Simulation aufbereitet in Form von Longitude- und Latitude-Werten.

3.4.2 Funktionsweise von Lichtsignalschaltungen

Die Funktionsweise der Lichtsignalschaltungen während der Simulation lässt sich in zwei Schritte zusammenfassen: 1. Timer fortsetzen und 2. die Warteschlange überprüfen.

Der Timer wird im ersten Schritt um eine Sekunde fortgesetzt. Ist der aktuelle Timer-Wert über der Zeitgrenze von einer Rot-, Gelb- oder Grünphase, wird die Phase gewechselt. Ist der Timer-Wert über die Summe von Rot-, Gelb- und Grünphasenlänge hinaus, setzt sich der Timer wieder zurück auf 1 verstrichene Sekunde.

Im zweiten Schritt wird die Warteschlange überprüft. Ist der Verkehrsteilnehmer nicht mehr an dieser Ampel, weil der Teilnehmer der erste in der Schlange ist und die derzeitige Ampelphase Grün ist, so wird dieser aus der Warteschlange entfernt. Das Wegfahren selbst unternimmt immer noch der Pkw-Fahrer selbst.

3.4.3 Lichtsignalphasen

Für dieses Modell wurde eine Aufnahme einer Lichtsignalschaltung getätigt, um eine Zeitangabe für die typische Rot- und Grünphasenlänge zu bekommen. Die Stadt Hamburg hat auf Nachfrage keine Daten veröffentlichen wollen oder im Internet zur Verfügung gestellt, welche Ampeln wie lange im „Normalzustand“ Grün- und Rotphasen haben, also ist die Aufnahme nicht repräsentativ für alle simulierten Lichtsignalanlagen, jedoch reichen sie für das Szenario aus, das simuliert werden soll.

In der Aufnahme haben die Lichtsignalphasen folgende Längen:

- Rotphase: 40 Sekunden
- Grünphase: 35 Sekunden
- Gelbphase: 1 Sekunde

Diese Zeiten dienen als Initialwerte, um danach iterativ sich dem Optimum von der Zeitschaltung zu nähern.

3.4.4 Lichtsignaltypen und Bezug zur Realität

Lichtsignalanlagen sind in der echten Welt nicht nur mit Zeitschalter ausgestattet. Ein Großteil derer können auch über Fernzugriffe bereits gesteuert und zu einer hintereinander geschalteten „grünen Welle“ umgestellt werden. In den Visionen von Hamburg 2030 [FOS⁺14, S. 42] aus dem Jahr 2014 wird bereits erwähnt, dass die bei Lichtsignalanlagen eingebauten Induktionsschleifen für Busse und bei Abbiegespuren zum Einsatz kommen. Doch können auch nach Tageszeit diese zur Verkehrsanpassung genutzt werden, um eine „grüne Welle“ für die Verkehrsteilnehmer zu ermöglichen. Auch wenn die Schaltung nicht für Fahrradfahrer, sondern an erster Stelle für Pkw-Fahrer gedacht ist, so soll sich diese Simulation darauf fokussieren, beiden Arten von Verkehrsteilnehmern eine „grüne Welle“ zu ermöglichen.

3.5 Designentscheidungen

In dieser Sektion wird auf die Designentscheidungen eingegangen, die für die Simulation getroffen wurden.

3.5.1 Funktionale und nichtfunktionale Anforderungen

Im Folgenden wird auf die Aufgaben des entwickelten Systems eingegangen, die es einzuhalten hat.

Funktionale Anforderungen:

- **Digitaler Zwilling als Simulationsumgebung:** Das System soll als Infrastrukturreferenz für die Simulation die Stadt Hamburg, die Straßen und ihre Lichtsignale nehmen.
- **Simulation eines ganzen Tages:** Das System soll für die Simulation die Zeit eines Wochentags von 0 Uhr morgens bis 23:59 Uhr simulieren.
- **Einbindung der HumanTraveler:** Das System bindet die vorher berechneten, aktiven HumanTraveler in die Simulation ein.
- **Zielsetzung der HumanTraveler:** Das System lässt jeden HumanTraveler ein Ziel haben, das sie mit beliebigen Modalitäten erreichen wollen.
- **Einbindung des BicycleLeaders:** Das System erstellt für jede Stunde einen BicycleLeader.
- **Zielsetzung des BicycleLeaders:** Das System lässt den BicycleLeader die acht Punkte der abzufahrenden Route nacheinander als Ziel haben.
- **Einbindung der HumanTraveler:** Das System bindet die relevanten Lichtsignalanlagen an den zugehörigen Straßen, Spuren und Kreuzungen in die Simulation ein.
- **Anhalten der HumanTraveler:** Das System soll die HumanTraveler im System an den Lichtsignalanlagen für die Rotphasen anhalten und für die Grün- sowie Gelbphasen weiterfahren lassen.
- **Anhalten des BicycleLeaders:** Das System soll das Simulieren des BicycleLeaders stoppen, sofern dieser bei einer Lichtsignalanlage hält.
- **Ausgabedaten der HumanTraveler:** Das System soll die gefahrenen Strecken der HumanTraveler nach deren Simulierung ausgeben.

- **Abbruchausgabedaten des BicycleLeaders:** Das System soll beim Anhalten des BicycleLeaders die gefahrene Strecke, den aktuellen Zeitpunkt, die Position der Lichtsignalanlage und die an der Lichtsignalanlage wartenden Agentenanzahl ausgeben.
- **Erfolgsausgabe des BicycleLeaders:** Das System soll beim Erreichen des Endziels von dem BicycleLeader den aktuellen Zeitpunkt sowie das erfolgreiche Ankommen ausgeben.

Zu den funktionalen Anforderungen gehören noch die nicht-funktionalen:

Nicht-Funktionale Anforderungen:

- **Grüne Welle für 90% der Fahrten:** Das System soll so lange weiter simulieren, bis eine Lichtsignalschaltung gefunden wurde, die dem BicycleLeader in mindestens 90% der Fälle eine „grüne Welle“ ermöglicht. Bei 24 Einträgen pro Simulationsdurchgang sind das 21,6 erfolgreiche Simulationen. Da es keine Fraktionen bei Simulationen gibt, wird auf 22 erfolgreiche Stundensimulationen aufgerundet. Das heißt also mindestens 2 oder weniger Einträge, die nicht eine „grüne Welle“ sein dürfen.
- **10 Simulationen pro Szenario:** Das System soll für jede Änderung der Lichtsignalschaltung mindestens zehn Simulationen durchführen.
- **Fahrtwege als GeoJSON:** Das System soll die Ausgabedaten der Fahrtwege als GeoJSON ausgeben, sei es von dem BicycleLeader oder von dem HumanTraveler.
- **C# als Programmiersprache:** Durch die Anknüpfung an das MARS-Framework muss das Projekt in der Programmiersprache C# entwickelt werden.

3.5.2 Fachliches Datenmodell des Systems

Das fachliche Datenmodell für die Simulation ist wie folgt:

Zu der Grafik 3.5 sind folgende Erläuterungen wichtig, um das Zusammenspiel der einzelnen Elemente zu verstehen:

HumanTraveler aus der SOHMultiModal-Komponente:

- Ist ein Agent in diesem System und kann Entscheidungen treffen zum Bewegen.

3 Konzept

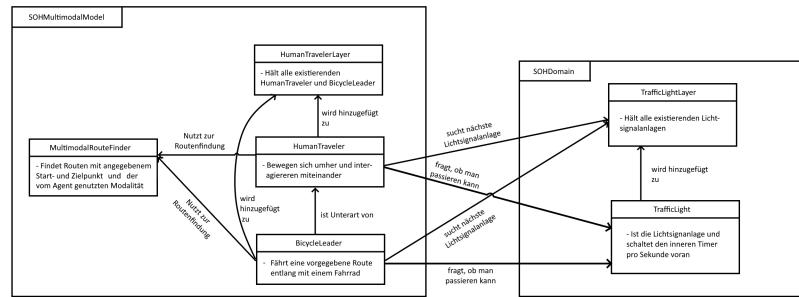


Abbildung 3.5: Klassendiagramm der Agenten und Entitäten

- HumanTravelers können eine Modalität bei sich erschaffen, zum Beispiel ein Bicycle, Car oder zu Fuß gehen. Eines dieser Modalitäten ist dann ihr Transportmittel.
- Ein HumanTraveler besitzt die folgenden Eigenschaften:
 - Eine Referenz auf ein Fahrrad, dem Bicycle, sollten sie eines besitzen
 - Eine Referenz auf einen Pkw, dem Car, sollten sie eines besitzen
 - Für alle verfügbaren Modalitäten noch eine Prozentangabe, die die Nutzungswahrscheinlichkeit der jeweiligen Modalität darstellt.
- HumanTraveler haben dazu noch folgende Funktionen zum Nutzen:
 - Tick() setzt den nächsten Agentenschritt fort, dass auch die Bewegung des HumanTravelers beinhaltet und dem Agenten die Entscheidung zum Abbiegen gibt, sollte die Route das vorschlagen.
 - IsWaitingAtTrafficLight() gibt einen Wahrheitswert zurück, ob der HumanTraveler aktuell an einer Lichtsignalschaltung wartet.

BicycleLeader aus der SOHMultimodal-Komponente:

- BicycleLeader sind identisch zu den HumanTravelers, bis auf die Einschränkung, dass sie nur Bicycle nutzen beziehungsweise zu Fuß zum Bicycle gehen können.
- Ein BicycleLeader besitzt die folgenden Eigenschaften:

3 Konzept

- Eine Referenz auf sein Fahrrad, dem Bicycle, sollte der BicycleLeader selbst eines besitzen und nicht ein gemietetes nehmen
- BicycleLeader haben dazu noch folgende Funktionen zum Nutzen:
 - Tick() setzt den nächsten Agentenschritt fort, dass auch die Bewegung des BicycleLeaders beinhaltet.
 - IsWaitingAtTrafficLight() gibt einen Wahrheitswert zurück, ob der BicycleLeader aktuell an einer Lichtsignalschaltung wartet.

RouteFinder aus der SOHMultiModal-Komponente:

- HumanTraveler und BicycleLeader nutzen für das Erreichen ihres Ziels den RouteFinder, mit der sie eine Route zum Abfahren berechnet bekommen.
- Der RouteFinder hat die folgende Funktion und bietet sie nach außen an:
 - Search(Agent, Startpunkt, Zielpunkt, Modalitätenliste) sucht für den Agenten von der Startposition zur Zielposition eine passende Route und beachtet dabei die Modalitäten, die der Agent nutzen kann.

HumanTravelerLayer aus der SOHMultiModal-Komponente:

- Der HumanTravelerLayer stellt die Umgebung für die HumanTraveler dar.
- HumanTraveler werden erschaffen und zu dem HumanTravelerLayer hinzugefügt.
- Die erschaffenen HumanTraveler bewegen sich auf dem HumanTravelerLayer, auf der sie mit anderen HumanTravelers interagieren können.
- Zu jeder vollen Stunde wird ein BicycleLeader in dem HumanTravelerLayer hinzugefügt.
- Der erschaffene BicycleLeader bewegt sich ebenfalls auf dem HumanTravelerLayer.
- Der HumanTravelerLayer besitzt dabei folgende Eigenschaften:
 - Die geographische Umwelt, die den digitalen Zwilling von Hamburg darstellt und alle Bewegungen festhält, als SpatialGraphMediatorLayer.

TrafficLight aus der SOHDomain-Komponente:

- TrafficLights selbst sind Entitäten und sind statisch auf dem zugehörigen Layer.
- TrafficLights können sowohl mit HumanTraveler als auch mit BicycleLeader in eine Warteschlange aufnehmen, sollte die aktuelle Lichtsignalphase Rot sein.
- Ein TrafficLight besitzt die folgenden Eigenschaften:
 - Die Position angegeben als Longitude und Latitude
 - Die Länge der Rot- und Grünphase als LengthPhaseRed und LengthPhaseGreen
 - Die aktuell verstrichene, interne Zeit als CurrTime
 - Die aktuelle Lichtsignalphase CurrPhase
 - Die wartenden Agenten über die Warteschlange WaitingRoadUsers.
- TrafficLights haben folgende Funktionen, die eine Relevanz für sie selbst oder für andere Agenten in der Simulation haben:
 - Tick() setzt die innere Zeitschaltung der Ampel fort und damit auch die nächste CarLightSignalPhase, zum Beispiel von Gelb zu Rot.
 - CheckQueue() überprüft die aktuelle Warteschlange und entfernt Agenten aus der Warteschlange, die nicht mehr warten müssen aufgrund von einer Grünphase.
 - Enter(Agent) fügt den angegebenen Agenten zur Warteschlange hinzu, sollte die aktuelle CarLightSignalPhase Rot sein.
 - CanPass(Agent) gibt einen Wahrheitswert zurück, ob der Agent sich überhaupt in der Nähe der Ampel befindet und wenn ja, ob dieser einfach vorbeifahren kann wegen eines grünen Lichtsignales und keinen wartenden, anderen Agenten.
 - IsQueued(Agent) überprüft und gibt einen Wahrheitswert zurück, ob der Agent noch in der Warteschlange notiert ist.

TrafficLightLayer aus der SOHDomain-Komponente:

- Die TrafficLights werden zu Beginn vollständig in die Simulation geladen und zu einem TrafficLightLayer hinzugefügt.
- Während der Simulation ist der TrafficLightLayer der Zugriffspunkt für jeden HumanTraveler, um naheliegende TrafficLights, also Lichtsignalanlagen, zu finden.
- BicycleLeader sind dabei eine Unterklasse der HumanTraveler und haben somit auch Zugriff auf den TrafficLightLayer.

3.5.3 Entwurfsmuster im System

Damit die Umsetzung im Implementationskapitel mancher Konzepte ersichtlich wird, wird im Folgenden noch auf die genutzten Entwurfsmuster des Systems eingegangen:

Agent-Entität-Umwelt-Muster:

Wie im fachlichen Datenmodell 3.5 erwähnt, basiert diese Simulation auf dem Agent-Entität-Umwelt-Muster. Dabei wird eine Simulationsebene bereitgestellt, die Umwelt, die in dieser Arbeit der HumanTravelerLayer und der TrafficLightLayer ausmachen. Für die Agenten HumanTraveler und BicycleLeader sind die beiden Layer die Umwelt, in der sie sich bewegen und mit anderen Agenten sowie Entitäten, zum Beispiel den TrafficLights, interagieren können. Agenten können selbst entscheiden, wie sie ihr Ziel erreichen, während sie dabei nur an die Regeln ihres Handels gebunden sind. In dem Fall dieser Arbeit wollen HumanTraveler zu einem zufällig festgelegten Ziel, suchen sich eine Route und nutzen dann die selbst ausgewählte Modalität zum Erreichen dieses Ziels. Die Entitäten sind in dem Rahmen dieser Arbeit die Lichtsignalanlagen, die TrafficLights, aber ebenso auch die Fahrräder, Pkws oder mietbaren Äquivalente. Entitäten selbst sind nicht in der Lage, eigene Entscheidungen zu treffen und zu „handeln“. Pkws oder Fahrräder selbst lassen sich steuern, sind dabei aber immer gleich in ihrem Verhalten: Wird die Bremse eines Fahrrads betätigt, so wird das Fahrrad entschleunigen. Die Entscheidung zu Bremsen kommt aber von dem Agenten, zum Beispiel dem BicycleLeader.

Factory-Pattern zur Agentenerstellung in der Simulationsumgebung:

Für die Erstellung der Agenten und Entitäten innerhalb der Umwelt wird je ein Factory-Pattern eingesetzt, dass die Erstellung über Parameter vereinfacht. Die benötigten Daten werden ausgelesen, in der „Factory“ verarbeitet und ein neuer Agent oder eine neue Entität wird zurückgegeben, mit den eingegebenen Eigenschaften. Das Factory-Pattern ermöglicht es, eine zentrale Stelle der Erstellung zu haben, bei der mit sich ändernden Parametern dennoch im Kern ähnliches dabei herauskommt. Beispielsweise werden HumanTraveler sich von den Daten her untereinander unterscheiden: Manche nutzen ein Fahrrad, manche ein Pkw, was dennoch unter demselben Punkt zusammengefasst wird: „Vehikel“.

Facade-Entwurfsmuster bei mehreren Unterkomponenten:

In dieser Arbeit werden Facaden beziehungsweise Interfaces benutzt, um die Handhabung verschiedener Agent-, Entitäts- oder Umweltypen zu vereinfachen. Mit dem Entwurfsmuster lässt sich die Vielzahl an Untertypen vereinheitlichen, die einer Komponente angehören, während sie nach außen hin gleiche Schnittstellen anbieten. Beispielsweise ist für die HumanTraveler und BicycleLeader eine Facade nützlich, die nach außen die Tick()-Funktion anbietet. Da beide Agenten aktiv simuliert werden, sich bei den beiden Akteuren die eigentlichen Ziele aber unterscheiden, hilft die Facade da und versteckt die verschiedenen Funktionsweisen über eine gleiche Funktion und umfasst sie damit. Außerhalb der beiden Agenten kann mit der übergreifenden Agenten-Fassade gearbeitet werden, werden die beiden Implementationen ihre Funktion beibehalten in ihrem Kontext.

3.6 Herangehensweise zur Ergebnisermittlung

Um die optimale Lichtsignalschaltung zu ermitteln, wird experimentell vorgegangen und die Lichtsignalphasen angenähert. Mit den gegebenen Daten aus den vorherigen Kapiteln lässt sich eine Ober- und Untergrenze etablieren. Die Obergrenze stellt hierbei den ungenügenden aber der Realität entsprechenden Zustand dar, den wir verbessern wollen. Dieser ist nach den Messungen an der Lichtsignalanlage mit 35 Sekunden für die Grün-, 1 Sekunde für die Gelb- und 40 Sekunden für die Rotphase ausgestattet.

Fortan wird das Format für die Phasenlängen zwischen Grün, Gelb und Rot in diesem Format angegeben, mit der eben genannten Obergrenze als Beispiel: 35 g | 40 r. Die Gelbphase wird nicht mit angegeben oder verändert, da sie von ihrem Nutzen her dem

grünen Lichtsignal ähnelt und Fahrzeugen erlaubt, weiterzufahren. Entsprechend wird sich nur auf die Veränderung der Grün- und Rotphase in dieser Arbeit beschränkt.

Die Untergrenze ist festgelegt als $35 \text{ g} + 0 \text{ r}$. Dieser Fall wäre nur in einer perfekten, realen Welt nützlich, in der bei Kreuzungen Fahrzeuge autonom sich sortieren und Straßen befahren. Dadurch, dass wir aber heutzutage Staus und Unfälle mitbekommen, ist diese Schaltung nicht wirklich praktikabel, also stellt sie damit die untere Grenze dar. Technisch hat die Länge der Grünphase bei einer Rotphase von 0 Sekunden keine Relevanz mehr, da es in der Simulation nie zu Rot beziehungsweise der Abbruchbedingung für die `BicycleLeader` kommen kann, weshalb der Einfachheit halber der Wert der Obergrenze übernommen wurde.

Bei der Annäherung selbst einer geeigneten Schaltung wird dabei antiproportional von der Obergrenze ausgegangen: Ist die Obergrenze mit $35 \text{ g} + 40 \text{ r}$ nicht ausreichend in der Erfolgsrate, so wird die Grünphasenlänge dann verdoppelt und die der Rotphase halbiert: $70 \text{ g} + 20 \text{ r}$. Ist die Annäherung nun aber ausreichend oder durchgängig positiv, so wird wieder in die andere Richtung verlängert und verkürzt. Diesmal wird aber nicht die Grünphasenlänge halbiert und die Rotphasenlänge verdoppelt, da dies wieder zu der Obergrenze führen würde: $35 \text{ g} + 40 \text{ r}$. Stattdessen wird die Rotphase mit 1.5 beziehungsweise $3/2$ und die Grünphase mit $2/3$ multipliziert, wie im folgendem Dreisatz zu erkennen ist:

$$40 \text{ r} \hat{=} 35 \text{ g} \quad (3.8)$$

$$1 \text{ r} \hat{=} 1.400 \text{ g} \quad (3.9)$$

$$30 \text{ r} \hat{=} 46.\overline{6} \text{ g} \quad (3.10)$$

$$(3.11)$$

Mit dem Ergebnis aus dem Dreisatz 3.11 hat man dann die neuen Phasenlängen: $47 \text{ g} + 30 \text{ r}$. Wichtig dabei ist zu beachten, dass Sekunden die kleinste Zeiteinheit in der Simulation sind, weshalb auf ganze Zahlen gerundet werden muss.

3.7 Umfang der Arbeit

Dieser Abschnitt beschäftigt sich mit den Grenzen dieser Arbeit und Simulation.

3 Konzept

Dadurch, dass im realen Leben Lichtsignalschaltungen verschiedenster Sicherheitsvorkehrungen unterliegen müssen, damit im Verkehr Sicherheit für alle Verkehrsteilnehmer gewährleistet werden kann, wäre eine Einbindung dieser Vorkehrungen für die Simulation ein wichtiger Realitätsaspekt. Doch aufgrund von fehlenden Daten und genauer Auskunft der Stadt Hamburg, welche Entscheidungen Lichtsignalanlagen treffen beziehungsweise nach welchem Design sie entwickelt wurden, um die Sicherheit zu ermöglichen, fällt auch dieser Realitätsaspekt aus der Simulation aus.

Zudem werden viele Umwelteinflüsse nicht erst in der Simulation eingebaut: Beispielsweise werden Ereignisse wie Wetter, Unfälle, Baustellen oder verschiedene Verkehrsteilnehmertypen, etwa wie zu langsam Fahrende oder stetig die Spur wechselnde Teilnehmer, nicht berücksichtigt.

Die für diese Simulation genutzten Werte, wie zum Beispiel die der gesamten Agentenzahl oder die der Lichtsignalphasenlänge, sind ebenfalls aufgrund fehlender Statistiken oder Daten nicht sehr realitätsnah. Beispielsweise wäre es angemessener, jede Lichtsignalschaltung auf der für `BicycleLeader` vorgesehenen Route mehrere hunderte Male zu beobachten, nur um dann eine aussagekräftigere Zeit von Rot- und Grünphasen tätigen zu können, damit statistische Abweichungen nicht potenziell das Endergebnis beeinflussen könnten.

Die Realität müsste für die Simulation über die Daten, Umwelteinflüssen und Agententypen abgebildet werden, damit ein praktischer Nutzen für die Stadt Hamburg entstehen könnte. Entsprechend bleibt der Nutzen dieser Arbeit für die Stadt Hamburg im theoretischen Bereich.

4 Implementierung

In diesem Kapitel wird die Implementation ausgeführt, wie auf dem MARS-Framework basierend die funktionalen und nichtfunktionalen Anforderungen sowie das fachliche Datenmodell eingebaut wird. Dazu wird auf die Ein- und Ausgabedaten und Implementationslimitationen eingegangen.

4.1 Agenten, Entitäten und Layer

In diesem Abschnitt wird über das Verhältnis der Agenten, Entitäten und Layer zueinander genauer eingegangen. Alle Klassen aus der Grafik 4.1 sind gleichbedeutend vom Namen her mit den Komponenten aus dem fachlichen Datenmodell:

In der Grafik 4.1 wurden einige überflüssige Eigenschaften, Methoden und Klassen weggelassen, um die Lesbarkeit und Übersichtlichkeit des Datenmodells zu erhöhen.

Agent **BicycleLeader**:

- Der **BicycleLeader** hat Modalitäten zur Auswahl `_choices`, die aber nur das Fahren eines `Bicycles`, `RentalBicycles` und das zu Fuß gehen über `WalkingShoes` erlauben.
- Dabei ist eine Referenz auf ein `Bicycle` und `RentalBicycle` gegeben von der geerbten Klasse `HumanTraveler`, die wiederum von einer Mehrzahl von Facaden erbt.
- Die zu fahrende Route ist beim `BicycleLeader` durch die Eigenschaft `stops` als Positions-Queue dargestellt.

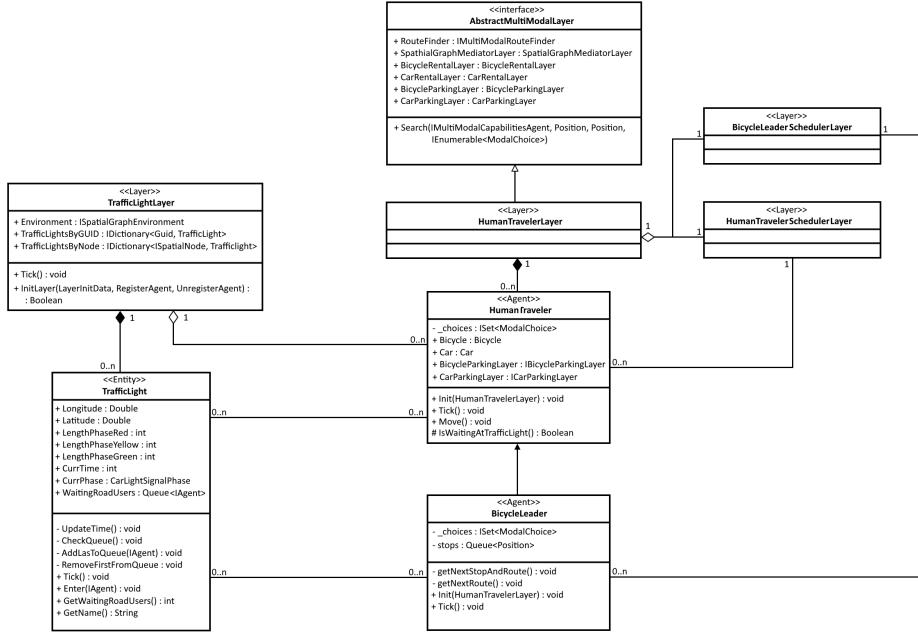


Abbildung 4.1: Das technische Datenmodell aller Agenten, Entitäten und Layer

- **BicycleLeader** haben zwei Funktionen zur Verfügung, um den nächsten Schritt ihrer Route zu berechnen: `getNextRoute()` sucht mit dem bestehenden Start- und Zielpunkt eine MultimodalRoute, während die Funktion `getNextStopAn-
dRoute()` den nächsten Punkt aus der `stops`-Queue entnimmt, dies als das neue Ziel setzt. Danach wird eine neue Route dorthin berechnet.
- Zur Routenberechnung wird der `RouteFinder` mit der Methode `Search(MultiModalCapabilitiesAgent, Position, Position, IEnumerable<ModalChoice>)` aus dem `HumanTravelerLayer` aufgerufen. Die eingegebenen Argumente meinen dabei als erstes den Agenten selbst, also den `BicycleLeader`, die Startposition, die Zielposition und die dabei zur Verfügung stehenden Modalitäten des `BicycleLeaders`.
- Die Funktion `Tick()` setzt den nächsten Simulationsschritt des `BicycleLea-
ders` in Gang. In dieser wird ebenfalls die vom `HumanTraveler` geerbte Funktion `Move()` ausgeführt.

- Die von HumanTraveler geerbte Funktion `IsWaitingAtTrafficLight()` wird ebenso in `Tick()` aufgerufen, um auf eine naheliegende TrafficLight zu prüfen: Ist ihre `CurrPhase` entweder `CarLightSignalPhase.GREEN`, also grün, oder `CarLightSignalPhase.YELLOW`, also gelb, so kann der BicycleLeader passieren. Wenn nicht, hält er hier und der BicycleLeader entfernt sich über einen `UnregisterAgent` aus der Simulation.
- Hält der BicycleLeader, so gibt er dies an zusammen mit dem Standort des TrafficLights.

Agent **HumanTraveler**:

- Der HumanTraveler hat alle Modalitäten zur Auswahl: Car, Bicycle, RentalCar, RentalBicycle und WalkingShoes.
- Die zu fahrende Route ist beim HumanTraveler durch die Funktion `Init(HumanTravelerLayer)` gegeben. Diese ruft die `Init(TLayer)`-Funktion der Elternklasse Traveler auf, in der zwei zufällige Punkte aus der Start- und Zielgeometrie genommen und als Start- und Zielposition festgelegt werden. Über ein Aufruf der `Tick()`-Funktion wird dann die Route berechnet zwischen den Punkten.
- HumanTraveler benutzen ebenfalls die `Search(...)`-Funktion des HumanTravelerLayers, um die Route berechnen zu lassen.
- `Tick()` setzt auch beim HumanTraveler über den Aufruf von `Move()` die Bewegung fort.
- Der HumanTraveler prüft ebenfalls mithilfe `IsWaitingAtTrafficLight()` vor dem Bewegen, ob er an einer TrafficLight steht. Wenn ja, betritt er die Warteschlange von ihr. Wechselt die TrafficLight die `CurrPhase` zu `CarLightSignalPhase.YELLOW` oder `CarLightSignalPhase.GREEN`, so fährt er weiter.

Entität **TrafficLight**:

- Die TrafficLight ist definiert über die Position als `Longitude` und `Latitude`, eine festgelegte `LengthPhaseRed`, `LengthPhaseYellow` und `LengthPhaseGreen` und hat eine zufällige Zahl zwischen einer dieser Phasen als `CurrTime`. Sie ist zufällig gewählt, damit keine Synchronisation mit den anderen Ampeln vorliegt.

- Die WaitingRoadUsers ist die Warteschlange der wartenden Agenten.
- Diese wird durch die CheckQueue()-Funktion immer wieder überprüft, ob ein Agent aus der Warteschlange entfernt werden kann. Ist die CurrPhase bei GREEN oder YELLOW, wird pro Tick()-Aufruf der vorderste der Warteschlange entfernt über die Funktion RemoveFirstFromQueue().
- UpdateTime() erhöht CurrTime bei jedem Aufruf von Tick() um 1. Ist die Summe aller drei PhaseLengths überschritten, so fängt CurrTime wieder bei 1 an.
- Agenten können über die Funktion Enter(IAgent) der Warteschlange beitreten, sollten sie der TrafficLight zu Nahe sein.
- Der Name des TrafficLights setzt sich aus der Longitude und Latitude zusammen und wird für die Erkennung genutzt bei der Ausgabe.

Layer HumanTravelerLayer:

- Alle HumanTraveler und BicycleLeader werden im HumanTravelerLayer festgehalten und ihre Position über den SpatialGraphMediatorLayer zusammengefasst.
- Der HumanTravelerLayer implementiert AbstractMultiModalLayer und stellt damit die verschiedenen Modalitäts-Layer HumanTravelern und BicycleLeadern bereit zum Interagieren.

Layer TrafficLightLayer:

- Alle TrafficLights werden im TrafficLightLayer festgehalten über zwei Dictionarys:
 - TrafficLightsByGUID, wenn eine GUID vorliegt und mit der die zugehörige TrafficLight entnommen werden kann
 - TrafficLightsByNode, die über einen SpatialNode, also über Longitude und Latitude-Verortung, abgespeichert werden
- Die ISpatialNodes selbst sind in der Environment vermerkt und sind befüllt mit Eingabedaten bei der Erstellung über die InitLayer(LayerInitData, RegisterAgent, UnregisterAgent)-Funktion. Die LayerInitData stellt dabei die einzelnen Einträge aus der Konfiguration dar.

Layer BicycleLeaderSchedulerLayer:

- Dieser Layer erhält Eingabedaten mit zeitgebundenen Einträgen. Diese Einträge werden dann über die Funktion `Schedule(SchedulerEntry)` der Elternklasse `AgentSchedulerLayer` hinzugefügt und erstellen den `BicycleLeader` mit den eingegebenen Eigenschaften.
- Der `BicycleLeaderSchedulerLayer` stellt damit das Factory-Pattern dar: Die Funktion `Schedule(SchedulerEntry)` wird mit Eingabedaten aufgerufen, um zu einem bestimmten Zeitpunkt einen `BicycleLeader` zu erstellen.

Layer HumanTravelerSchedulerLayer:

- Identisch zu dem `BicycleLeaderSchedulerLayer` erhält dieser Layer Eingabedaten mit zeitgebundenen Einträgen. Diese Einträge werden dann über die Funktion `Schedule(SchedulerEntry)` der Elternklasse `AgentSchedulerLayer` hinzugefügt und erstellen den `HumanTraveler` mit den eingegebenen Eigenschaften.
- Der `HumanTravelerSchedulerLayer` stellt damit ebenso das Factory-Pattern dar: Er wird mit Eingabedaten aufgerufen, um zu einem bestimmten Zeitpunkt einen `HumanTraveler` zu erstellen.

4.2 Anbindung an das MARS-Framework

Die Anbindung zum MARS-Framework geschieht bei den im technischen Modell erläuterten Klassen durch Fassaden. Im Folgen wird näher erläutert, welche Fassaden durch welche Klasse implementiert werden und wie sie damit in die Simulationsumgebung vom MARS-Framework eingebunden werden:

Auch sind hier in dem technischen Datenmodell 4.2 wieder der Übersichtlichkeit und Lesbarkeit halber überflüssige Fassaden, Eigenschaften und Funktionen weggelassen worden.

Layer TrafficLightLayer und HumanTravelerLayer:

- Der `TrafficLightLayer` und `HumanTravelerLayer` implementieren den `AbstractLayer` und erhalten dadurch Zugriff auf den `RegisterAgent`, `UnregisterAgent` und `EntityManager`, mit dem die Layer selbstständig der Simulation Entitäten und Agenten hinzufügen oder nehmen können.

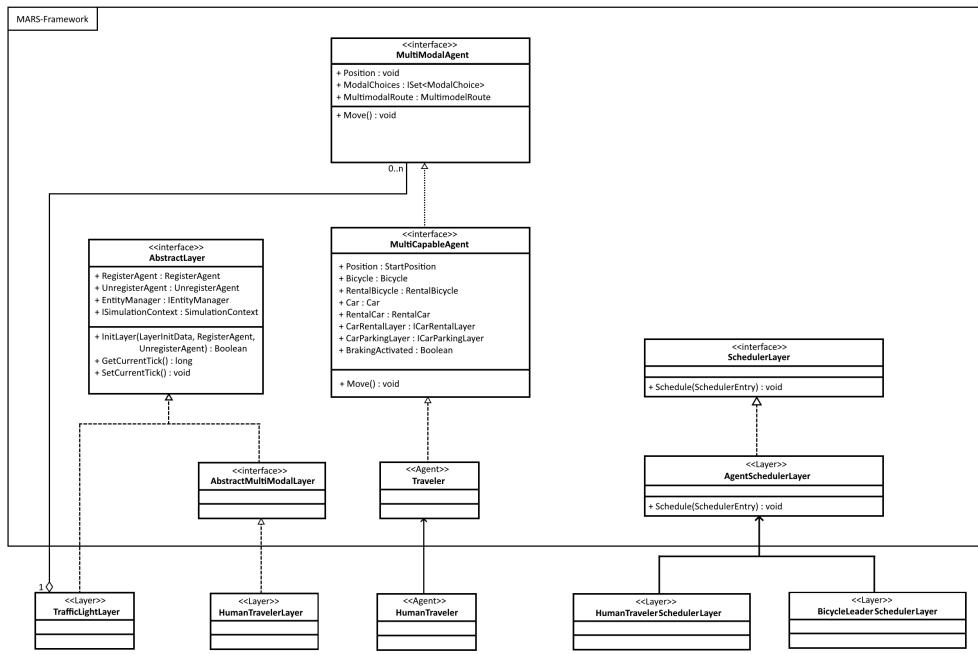


Abbildung 4.2: Die Anbindung an bestehenden Fassaden des MARS-Frameworks

- Durch das Initialisieren über die `Init(...)`-Funktion und dem `SimulationContext`, können beide Layer über die Konfiguration festgelegten Eingabedaten erhalten und alle `TrafficLights` oder `HumanTraveler` erstellen.
- Mit den Funktionen `GetCurrentTick()` und `SetCurrentTick()` lässt sich auch der derzeitige Simulationszeitpunkt feststellen, sowie die `Tick()`-Methode ausführen beim Setzen eines neuen Simulationsschrittwertes.
- Der `MultiModalAgent` benötigt eine Referenz auf den `TrafficLightLayer`, da alle erbenden Klassen auf diesen Layer zugreifen müssen. `MultiModalAgent` ist dabei die einzige Klasse, die über die Annotation `[PropertyDescription]` eine Instanz des `TrafficLightLayers` und nicht einen Null-Wert gesetzt bekommt.

Agent `HumanTraveler`

- Der `HumanTraveler` implementiert den `Traveler`, der aber über keine weiteren für diese Simulation relevanten Eigenschaften oder Funktionen verfügt.

- Zudem implementiert er den `MultiCapableAgent`, der für jede Modalität eine Eigenschaft besitzt und den dazugehörigen Layer.
- Die Referenz auf diese Modalitäten wird an die Unterklassen weitergeben, zum Beispiel den `BicycleLeader`, die mit der `Move()`-Funktion dann die Bewegung auf dem `ISpatialMediatorGraph` ausführen.
- Zuletzt implementiert `HumanTraveler` noch den `MultiModalAgent`, der mit der Position jedem Agenten seinen Standort in der Simulation angibt, mit `ModalChoices` dem Agenten die nutzbaren Modalitäten und mit `MultimodalRoute` die in der Simulation fahrende Route vererbt.

Layer `HumanTravelerSchedulerLayer` und `BicycleLeaderSchedulerLayer`

- Beide Layer implementieren den `AgentSchedulerLayer`, der eine Standard-Implementation der `Schedule(SchedulerEntry)`-Funktion bereitstellt und nur zwei angegebene Argumente benötigt beim implementieren: Den zu erschaffenden Agenten und den Layer, zu dem dieser hinzugefügt werden soll.
- Die Fassade für den `AgentSchedulerLayer`, der intern die verschiedenen `AgentSchedulerLayer` vereint, schreibt `Schedule(SchedulerEntry)` zum Implementieren vor und führt dann, sobald ein Eintrag zeitlich passt, die Funktion mit dem Eintrag aus.

4.3 Konfiguration der Ein- und Ausgabedaten

Nun folgen die Eingabedaten, die bei der Implementation in die Simulation benötigt werden:

- **SpatialMediatorGraph-Konfiguration:** Für die Simulationsumgebung ist eine GeoJSON vorgesehen, der den Simulations- und Bewegungsbereich der Agenten und Entitäten angibt.
- **HumanTraveler-Konfiguration:** Die aktiven `HumanTraveler` und `BicycleLeader` in der Simulation benötigen folgende Eingabedaten als CSV-Datei:
 - `startTime` als Startuhrzeit im Format „HH:mm“, sobald die Erstellungsphase für `HumanTraveler` beginnt

- endTime als Enduhrzeit im Format „HH:mm“, sobald die Erstellungsphase der HumanTraveler aufhört
 - spawningIntervalInMinutes als Ganzzahlangabe für die Wiederholungen der Erstellungen pro angegebener Minuten
 - spawningAmount als Anzahl der zu erstellenden Agenten
 - Wahrscheinlichkeiten im Bereich $0 \leq \text{hasBike} \leq 1$ für:
 - * hasBike zum Besitz eines eigenen Fahrrads
 - * hasCar zum Besitz eines eigenen Pkws, das höher als hasBike sein wird um den Verkehr Pkw-dominant zu machen und näher an der Realität zu simulieren
 - * prefersBike zum Bevorzugen eines Fahrrads
 - * prefersCar zum Bevorzugen eines Pkws, das höher als prefersBike sein wird um den Verkehr Pkw-dominant zu machen und näher an der Realität zu simulieren
 - * usesBikeAndRide zum Nutzen von RentalBicycles
 - * usesOwnBikeOutside zum Nutzen von dem eigenen Fahrrad
 - * usesOwnCar zum Nutzen eines eigenen Pkws
 - source als Startbereich, in dem der HumanTraveler erstellt wird
 - destination als Zielbereich, in dem der HumanTraveler ein Ziel auswählen kann
 - discriminator als Unterscheidungszahl für interne Darstellungen
- **TrafficLight-Konfiguration:** Die zu erschaffenden TrafficLights benötigen zur Erstellung folgende Eingabedaten als CSV-Datei:
 - Longitude als Longitude auf einer Weltkarte oder Simulationsumgebung
 - Latitude als Latitude auf einer Weltkarte oder Simulationsumgebung
 - LengthPhaseRed als Phasenlänge für die Lichtphase Rot
 - LengthPhaseGreen als Phasenlänge für die Lichtphase Grün

- **Simulations-Konfiguration:** Eine JSON, die alle für die Simulation relevanten Layer, Entitäten und Agenten angegeben braucht.

Zudem werden noch folgende Ausgabedaten bei der Simulation erstellt:

- **HumanTravelers Strecken-GeoJSON:** Die gefahrenen Strecken der HumanTraveler mit Longitude-Latitude-Positionsangaben
- **Bicycle-Leaders Strecken-GeoJSON:** Die gefahrenen Strecken der BicycleLeader mit Longitude-Latitude-Positionsangaben
- **Erfolgsausgabe des Bicycle-Leaders:** Wenn eine erfolgreiche „grüne Welle“ stattfand oder ob er durch eine Lichtsignalanlage aufgehalten wurde mit der jeweiligen Stunde der Simulation

4.4 Überprüfen der Grundfunktionalität

Um die Funktionalität des Modells zu bestätigen, sind die folgenden Kernelemente im Quelltext überprüft:

Einbindung und Verifikation aller TrafficLights: Zur Verifikation, ob alle 1260 TrafficLights aus der GeoJSON von OpenStreetMap [OF23] richtig exportiert sind, werden die extrahierten Daten in die Online-Anwendung „kepler.gl“ [KC18] eingegeben, die die Longitude-Latitude-Daten der GeoJSON auf einer virtuellen Weltkarte darstellt:

Die Verteilung der Lichtsignalanlagen auf Kreuzungen und Straßen, wie man sie in der Abbildung 4.3 sehen kann, entspricht der Position in OpenStreetMap, wie man anhand des Vergleichs in 4.4 erkennen kann.

Zur Überprüfung, ob bei der Initialisierung des TrafficLightLayers alle Lichtsignalanlagen richtig importiert wurden, ist nach dem Einbinden über die `InitLayer()`-Methode eine Abfrage auf die Anzahl der eingelesenen TrafficLights eingebaut, wie in der Abbildung 4.5 zu sehen ist:

Die Überprüfung, ob die GeoJSON in OpenStreetMap korrekt und vollständig ist im Vergleich zur Realität, lässt sich nur mit einem großen Aufwand überprüfen. Dafür müsste

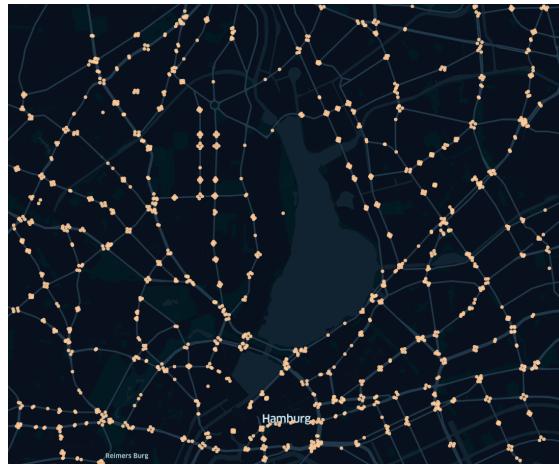


Abbildung 4.3: Der Export von OpenStreetMap in kepler.gl eingefügt

man bei jeder Schaltung selbst physisch vor Ort sein und die Position bestätigen. Dahingehend wird die Ausgabe von OpenStreetMap so hingenommen und als ausreichend für diese Simulation festgelegt.

Einbindung und Verifikation der `HumanTraveler` und `BicycleLeader`:

Zur Überprüfung, ob die `HumanTraveler` und `BicycleLeader` korrekt in der Simulation erschaffen werden, wird erneut über kepler.gl die ausgegebenen GeoJSON überprüft:

Aus der Grafik von 4.6 und 4.7 lässt sich erkennen, dass die stündliche Erschaffung korrekt funktioniert. Einerseits ist der Simulationsbereich erkennbar, die rechteckige Form, die von den Pfaden der `HumanTraveler` erzeugt wird, als auch ihr Auftreten überhaupt.

Ebenfalls ist die korrekte Interaktion mit den `TrafficLights` zu überprüfen. `HumanTraveler` und `BicycleLeader` verwenden die Funktion `IsWaitingAtTrafficLight()`, die die Agenten bei Ankunft an einer Lichtsignalanlage zur Warteschlange hinzufügt. Wie aus dem Konsolenausschnitt von 4.8 zu entnehmen ist, können die Agenten korrekt mit den `TrafficLights` interagieren.

Die Konsolenausgabe 4.8 sind alle von den `HumanTravelers`, da ein Anhalten der `BicycleLeader` separat überprüft wird und dann zur Entfernung aus der Simulation führt, wie in der folgenden Konsolenausgabe 4.9 erkennbar ist.

Hat der `BicycleLeader` erfolgreich den letzten Punkt erreicht, so wird dann nur noch geprüft, ob der letzte Punkt der zu fahrenden Route identisch zu der aktuellen Position

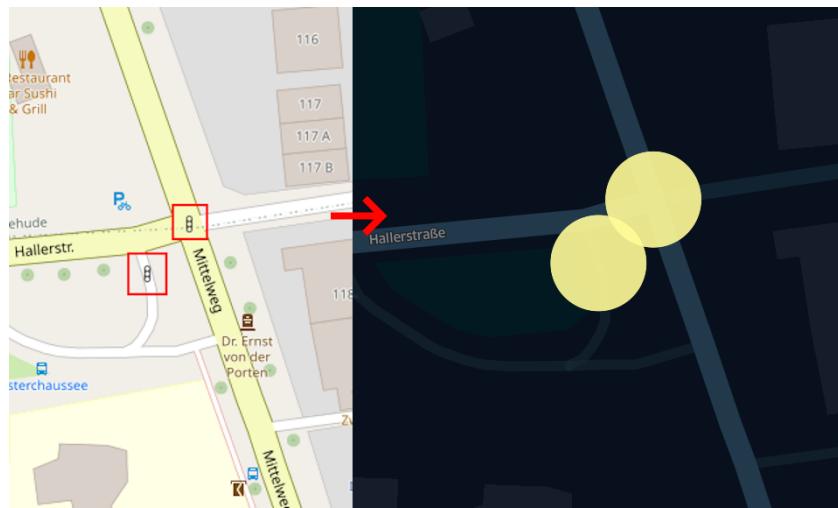


Abbildung 4.4: Lichtsignalanlagen an der Kreuzung der Hallerstraße, links in OpenStreetMap, rechts in kepler.gl eingefügt

```
C:\WINDOWS\system32\cmd.exe
C:\Users\User\Desktop\Vorlesungen\#Bachelorarbeit\project HamburgAlsterBikes.csproj
Anzahl Ampeln nach GUIDs: 1260
Anzahl Ampeln nach Knoten: 1260
```

Abbildung 4.5: Die Konsolenausgabe der aktuell eingebauten TrafficLights

des Agenten ist. Dies wird bewerkstelligt, wie man in der Konsolenausgabe aus 4.10 indem die Distanz zwischen den beiden Punkten 0 sein muss.

Zuletzt wird noch geprüft, ob die HumanTraveler anhalten, sobald sie in die Warteschlange einer TrafficLight hinzugefügt werden. Dafür wird, wie in der Konsolenausgabe 4.11 sichtbar, die Geschwindigkeit nach einer Wartezeit von 5 Sekunden beziehungsweise 5 Ticks ausgegeben, um dem HumanTraveler Zeit zu geben, die Bremsen zu aktivieren und zu entschleunigen.

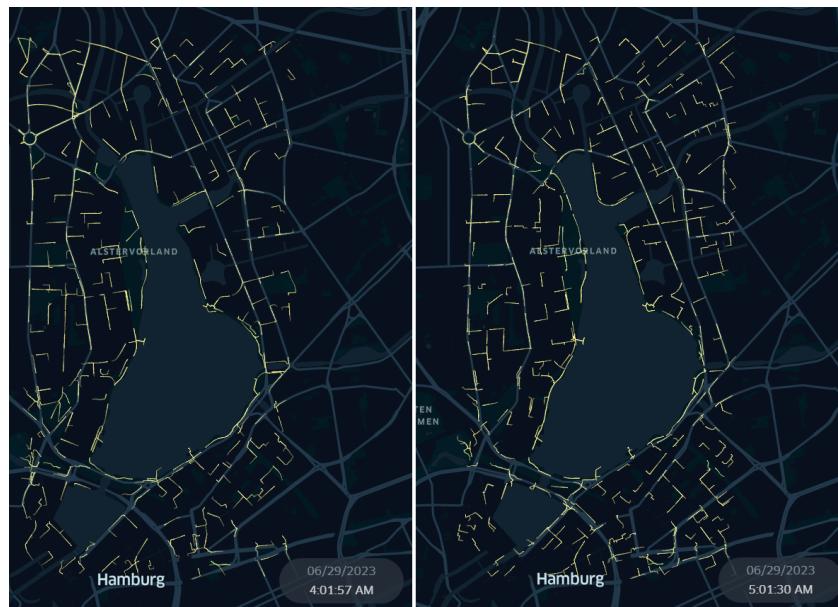


Abbildung 4.6: Der Erstellbereich der HumanTraveler, links von 4 Uhr und rechts von 5 Uhr morgens

4.5 Bekannte Probleme und Einschränkungen

In dieser Arbeit sind ebenfalls Probleme beim Simulieren aufgetreten, die im folgenden mit möglichen Gründen beziehungsweise Ansätzen näher erläutert werden:

Limitationen der Hardware

Während der Simulationen ist es zu erheblichen Performance-Problemen gekommen. Die Anzahl der aktiven Agenten ist dabei ausschlaggebend gewesen für die Verlangsamung der Simulation. Aufgefallen ist dies während der Entwicklung, als 24 Stunden am Stück mit den jeweiligen Mengen an Agenten simuliert werden sollten. Die Simulation wollte nicht enden und die vom MARS-Framework bereitgestellt Fortschrittsleiste schritt ebenfalls nur sehr langsam voran.

Durch den Hacking-Vorfall der HAW Hamburg ist ebenfalls der Zugriff auf die Cluster seit Anfang des Jahres 2023 für die Studenten verwehrt, sodass das Ausweichen auf die Rechencluster keine Alternative war für die Simulationsergebnisse. Stattdessen wurden die 24 zu simulierenden Stunden geviertelt und dann so simuliert. Das ließ die Hardware des genutzten Computers zu und so konnten die Daten dennoch entnommen werden.

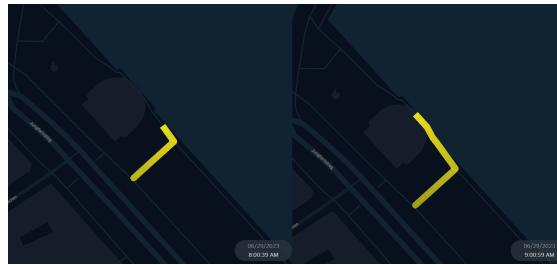


Abbildung 4.7: Die Erschaffung des BicycleLeader, links um 8 Uhr, rechts um 9 Uhr morgens

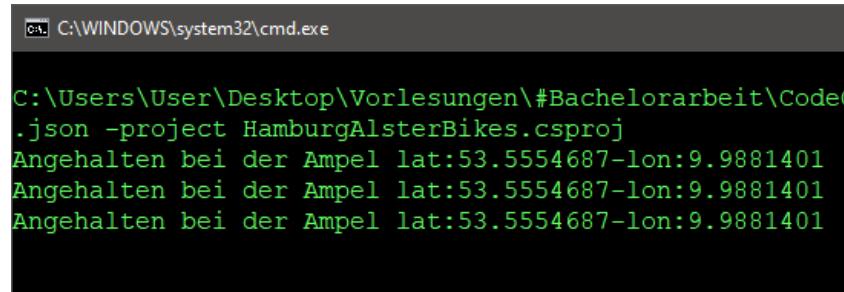
```
C:\WINDOWS\system32\cmd.exe
C:\Users\User\Desktop\Vorlesungen\#Bachelorarbeit\CodeGi
.json -project HamburgAlsterBikes.csproj
Wartende Agenten bei lat:53.5807887-lon:10.0029802: 6
Wartende Agenten bei lat:53.5807887-lon:10.0029802: 7
Wartende Agenten bei lat:53.5807098-lon:10.0034726: 6
Wartende Agenten bei lat:53.5807098-lon:10.0034726: 7
Wartende Agenten bei lat:53.5809256-lon:10.0120575: 6
Wartende Agenten bei lat:53.5744224-lon:10.0149524: 6
Wartende Agenten bei lat:53.5798222-lon:10.0129451: 6
Wartende Agenten bei lat:53.5758296-lon:10.0111494: 6
Wartende Agenten bei lat:53.5784093-lon:10.0093508: 6
Wartende Agenten bei lat:53.5781214-lon:10.0095658: 6
Wartende Agenten bei lat:53.5784093-lon:10.0093508: 7
Wartende Agenten bei lat:53.5781214-lon:10.0095658: 7
Wartende Agenten bei lat:53.5781214-lon:10.0095658: 8
Wartende Agenten bei lat:53.5781214-lon:10.0095658: 9
Wartende Agenten bei lat:53.5784093-lon:10.0093508: 8
Wartende Agenten bei lat:53.5784093-lon:10.0093508: 9
Wartende Agenten bei lat:53.5807887-lon:10.0029802: 6
```

Abbildung 4.8: Hier wird eine Ausgabe an Lichtsignalanlagen gegeben, sobald mehr als 5 Agenten in der Warteschlange einer TrafficLight sind.

Probleme mit GeoJSON-Ausgaben

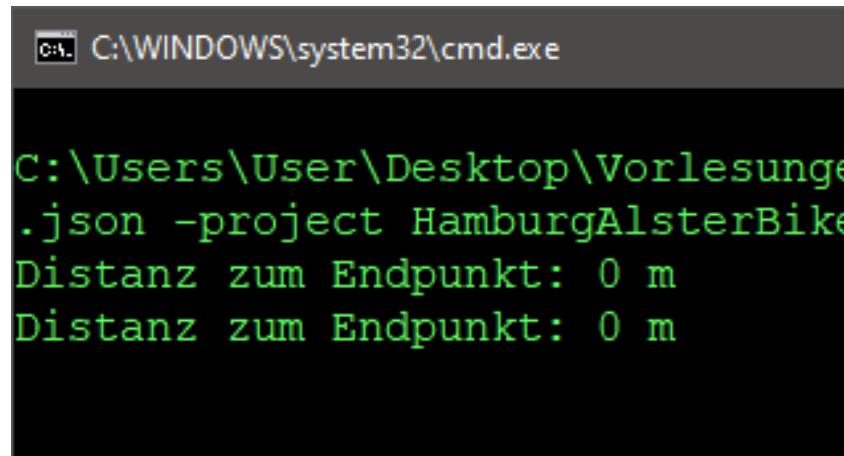
Die exportierten GeoJSONs aus dem MARS-Framework haben ebenfalls Probleme bei den Simulationen aufgezeigt. Die HumanTraveler-GeoJSON, die die Rundfahrten anzeigen sollte sowie das Anhalten an den Lichtsignalanlagen, zeigt lediglich den gefahrenen Pfad und endet beim Antreffen auf das jeweilige Ziel.

Zudem hat die GeoJSON das Setzen eines neuen Routenziels ebenfalls nicht in der `BicycleLeader.geojson` inkludieren können. Das Simulieren des BicycleLeaders ist bis zum Ziel des ersten Punktes korrekt, jedoch wird ab dann eine neue Route berechnet und damit ein neues Start- und Endziel festgelegt. Dies wird aber nicht in der GeoJSON abgelegt und lässt sich nicht darüber erkennen. Dieses Problem wurde auch mit Entwicklern von dem MARS-Framework besprochen, jedoch konnte es leider bisher nicht behoben werden. Stattdessen wird die im Testing-Kapitel und mit der Grafik 4.10



```
C:\WINDOWS\system32\cmd.exe
C:\Users\User\Desktop\Vorlesungen\#Bachelorarbeit\Code
.json -project HamburgAlsterBikes.csproj
Angehalten bei der Ampel lat:53.5554687-lon:9.9881401
Angehalten bei der Ampel lat:53.5554687-lon:9.9881401
Angehalten bei der Ampel lat:53.5554687-lon:9.9881401
```

Abbildung 4.9: Konsolenausgabe, bei der BicycleLeader alle 3 Stunden in Folge wegen der zu langen Rot-Signalphase die Abbruchbedingung erreicht.



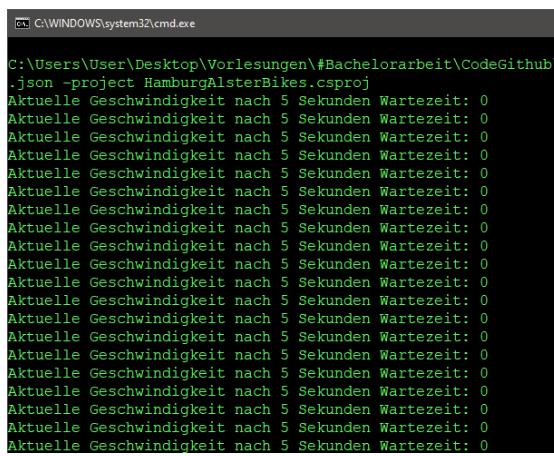
```
C:\WINDOWS\system32\cmd.exe
C:\Users\User\Desktop\Vorlesungen\#Bachelorarbeit\Code
.json -project HamburgAlsterBikes.csproj
Distanz zum Endpunkt: 0 m
Distanz zum Endpunkt: 0 m
```

Abbildung 4.10: Distanzangabe über die Konsole vom letzten, zu erreichenden Punkt, bis zum BicycleLeader.

angeführte Erfolgsüberprüfung für die BicycleLeader genommen und damit getestet, ob eine grüne Welle erreicht wurde.

Probleme bei Modalitäten mit dem RouteFinder:

In manchen Simulationsläufen kommt es zu einem Problem mit dem RouteFinder: Dieser versucht bei einem Agenten eine Route mit den möglichen Modalitäten zu finden. Dabei gibt er manchmal aber nur einen Weg an, der über genau eine Modalität nur erreicht werden kann, die der Agent genau nicht besitzt. Ein möglicher Grund könnte die Startposition mancher HumanTraveler sein, die am Anfang zufällig geschehen. Beispielsweise befindet sich der Agent nach seiner Erschaffung auf einer Autostraße, die nur von Pkws befahren werden darf. Dennoch versucht er dann die nächste Modalität oder das Interessenziel aufzusuchen und ruft den RouteFinder auf für eine passende



A screenshot of a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The window contains a series of identical green text entries repeated 20 times. Each entry consists of the prefix 'Aktuelle Geschwindigkeit nach 5 Sekunden Wartezeit: 0' followed by a blank line.

Abbildung 4.11: Konsolenausgabe, bei der HumanTraveler nach 5 Sekunden warten an einer TrafficLight ihre Geschwindigkeit ausgeben.

Route. Da aber mit dem angegebenen Startpunkt keine andere Möglichkeit vorliegt, außer eben das Befahren der Straße, wirft der RouteFinder einen Fehler und die Simulation bricht ab.

Die Lösung für das Problem war in dem Fall die Simulation neu zu starten und einer der Entwickler das Problem aufzuzeigen, auch wenn der Fehler bisher noch nicht behoben werden konnte.

Probleme in der Haupt-Konfigurationsdatei:

In der Simulation werden Agenten erschaffen, die durch eine vorher eingegebene CSV die Start- und Endzeitpunkte angeben. Diese Variablen geben den Zeitraum an, wann die Erstellphase von Agenten- oder beginnt und endet. Wenn man zu Beginn der Simulation bei 0:00 Uhr einen Agenten erschaffen und die Simulation zum selben Zeitpunkt beginnen möchte, also an einem Tag um 0:00 Uhr startet, so würde man erwarten, dass direkt zu Beginn die angegebenen Agenten erstellt werden. Das werden sie aber nicht, da die angegebene Zeit überschritten werden muss, damit die Agentenerstellung ausgelöst wird.

Um dieses Problem anhand des Beispieles mit 0:00 Uhr zu lösen, muss am vorherigen Tag die Simulation begonnen und am besten ein „Puffer“ von 10 Sekunden eingebaut werden. Beispielsweise ist das Startdatum und die Uhrzeit der Simulation nicht der 29.06.2023 um 00:00:00 Uhr, sondern der 28.06.2023 um 23:59:50 Uhr. Damit schafft die Simulation es, in der angegebenen Zeit von 0:00 Uhr der CSV die Agenten zu erschaffen.

5 Evaluation und Diskussion der Ergebnisse

5.1 Ergebnisse der Experimente

In diesem Kapitel werden nun die Ergebnisse der durchgeführten Experimente dargestellt und näher erläutert. Dabei werden die genauen Eingabedaten angezeigt, sowie die Ober- und Untergrenze der Zeiten für die grüne Lichtsignalschaltung noch einmal aufgeführt.

5.1.1 Experimentübergreifende Eingabedaten

Bei allen durchgeführten Experimenten gibt es einige, gleiche Eingabedaten. Damit sie nicht redundant wiederholt werden, sind sie hier einmal zusammengefasst.

Einige der überflüssigen Daten aus den Konfigurationen werden hier nicht aufgelistet, damit sie die Lesbarkeit und die Übersichtlichkeit nicht stören. Beispielsweise wird die Nennung aller hinzugefügten Layer oder Agenten aus der Simulationskonfiguration ausgelassen.

Simulationskonfiguration:

- "startPoint": 2023-06-28T23:59:50
- "endPoint": 2023-06-29T23:59:59
- "deltaT": 1
- "deltaTUnit": "seconds"

HumanTraveler-Konfiguration:

- hasBike: 0

- hasCar: 0.85
- prefersBike: 0.1
- prefersCar: 0.9
- usesBikeAndRide: 0.2
- usesOwnBike: 0.85

5.1.2 Ober- und Untergrenze der Simulation

Im Folgenden werden die Ober- und Untergrenzen der Simulation evaluiert. Die Obergrenze stellt hierbei die Schaltung 35 g | 40 r dar, was der Realität angenähert ist.

Damit die Spalten übersichtlich bleiben, werden folgende Abkürzungen verwendet:

- Stunde stellt hier die Stunde am Tag dar.
- #NHT meint die Anzahl der zu dieser Stunde neu hinzugefügten HumanTraveler - „(N)eue (H)uman(T)raveler“.
- E1, E2 ... E10 meint hier, ob das Experiment 1, 2 und so weiter erfolgreich war, eine grüne Welle für den BicycleLeader zu simulieren.

5.1.3 Experiment 1: 70 g | 20 r

5.1.4 Experiment 2: 140 g | 10 r

5.1.5 Experiment 3: 280 g | 5 r

5.1.6 Experiment 4: 467 g | 3 r

5.1.7 Experiment 5: 350 g | 4 r

5.2 Diskussion der Ergebnisse

Stunde	#NHT	E1	E2	E4	E5	E6	E7	E8	E9	E10
0	14									
1	51									
2	130									
3	274									
4	487									
5	707									
6	857									
7	879									
8	804									
9	684									
10	570									
11	514									
12	550									
13	650									
14	765									
15	850									
16	857									
17	755									
18	579									
19	379									
20	205									
21	94									
22	34									
23	14									
Gesamt:	11.703									

Tabelle 5.1:

6 Zusammenfassung

Literaturverzeichnis

- [BM19] Saif Islam Bouderba and Najem Moussa. Reinforcement learning (q-learning) traffic light controller within intersection traffic system. Association for Computing Machinery, 10 2019.
- [BSC15] W. Clifton Baldwin, Brian Sauser, and Robert Cloutier. Simulation approaches for system of systems: Events-based versus agent based modeling. volume 44, pages 363–372. Elsevier, 2015.
- [CAML⁺21] Thomas Clemen, Nima Ahmady-Moghaddam, Ulfia A. Lenfers, Florian Ocker, Daniel Osterholz, Jonathan Ströbele, and Daniel Glake. Multi-agent systems and digital twins for smarter cities. pages 45–55. Association for Computing Machinery, Inc, 5 2021.
- [Den18] Timo Denk. Online tools - cubic spline interpolation, 2018.
- [fHuSHSN22] Statistisches Amt für Hamburg und Schleswig-Holstein Statistikamt Nord. Vorabinformation zur beabsichtigten direktvergabe eines öffentlichen dienstleistungsauftrags der freien und hansestadt hamburg an die verkehrsbetriebe hamburg-holstein gmbh (vhhs), 10 2022.
- [FOS⁺14] Christoph Färber, Dr. Michaela Ölschläger, Joana Schleinitz, Jan-Oliver Siebrand, and Reinhard Wolf. Stadtmobilität in hamburg 2030, 2 2014.
- [GPR⁺21] Daniel Glake, Fabian Panse, Norbert Ritter, Thomas Clemen, and Ulfia Lenfers. Data management in multi-agent simulation systems from challenges to first solutions. volume P-311, pages 423–436. Gesellschaft fur Informatik (GI), 2021.
- [Gro23a] MARS Group. Mars - multi-agent research and simulation, 2023.
- [Gro23b] MARS Group. Smartopenhamburg, 2023.

- [KC18] Kepler.gl-Community. A powerful open source geospatial analysis tool for large-scale data sets., 2 2018.
- [KT20] Valentina Kurtc and Martin Treiber. Simulating bicycle traffic by the intelligent-driver model-reproducing the traffic-wave characteristics observed in a bicycle-following experiment. *Journal of Traffic and Transportation Engineering (English Edition)*, 7:19–29, 2 2020.
- [LAMG⁺21a] Ulfia A. Lenfers, Nima Ahmady-Moghaddam, Daniel Glake, Florian Ocker, Daniel Osterholz, Jonathan Ströbele, and Thomas Clemen. Improving model predictions—integration of real-time sensor data into a running simulation of an agent-based model. *Sustainability (Switzerland)*, 13, 7 2021.
- [LAMG⁺21b] Ulfia Annette Lenfers, Nima Ahmady-Moghaddam, Daniel Glake, Florian Ocker, Jonathan Ströbele, and Thomas Clemen. Incorporating multi-modal travel planning into an agent-based model: A case study at the train station kellinghusenstraße in hamburg. *Land*, 10, 11 2021.
- [Mul20] Katharina Mulack. Multiagenten simulation von fahrradfahrern im kontext urbaner verkehrsdynamik. 5 2020.
- [OF23] OpenStreetMap-Foundation. Openstreetmap, 2023.
- [Run22] Norddeutscher Rundfunk. Neue ampelschaltung: Vorfahrt für fußgänger und radfahrer, 12 2022.
- [uHHBfWVuIAfVuS15] Freie und Hansestadt Hamburg Behörde für Wirtschaft Verkehr und Innovation: Amt für Verkehr-und Straßenwesen. Vorabinformation zur beabsichtigten direktvergabe eines öffentlichen dienstleistungsauftrags der freien und hansestadt hamburg an die verkehrsbetriebe hamburg-holstein gmbh (vhv), 12 2015.
- [ZMJZ19] Ye Zheng, Ding Ma, Fengying Jin, and Zhigang Zhao. Es-band: A novel approach to coordinate green wave system with adaptation evolutionary strategies. pages 36–39. Association for Computing Machinery, Inc, 11 2019.

A Anhang

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort

Datum

Unterschrift im Original