Hochschule Karlsruhe
University of
Applied Sciences

HKA

Institut für
Ubiquitäre
Mobilitätssysteme

Masterarbeit

Entwicklung und Evaluation eines auf künstliche Intelligenz gestützten Systems zur Betriebslenkung von Linienbussen im Störungsfall

31. Januar 2023

Sebastian Knopf
Mat.-Nr.: 68390

Betreuung
M. Sc. Waldemar Titov,
Dirk Weißer (VDV)

Verantwortlicher Hochschullehrer
Prof. Dr.-Ing. Thomas Schlegel

Aufgabenstellung

(DE) Kurzfristige Streckensperrungen oder erhöhtes Verkehrsaufkommen gefährden die Fahrplanstabilität und sorgen dadurch für Frust und Hemmnisse im öffentlichen Personenverkehr. Besonders kleine und mittelständische Verkehrsunternehmen verfügen in den meisten Fällen nicht über eine Betriebsleitstelle, die geeignete Umleitungsfahrwege anweisen und dadurch den Betrieb aufrechterhalten kann. In dieser Masterarbeit soll untersucht werden, inwieweit die Anordnung von Umleitungen durch ein Betriebsleitsystem unterstützt durch künstliche Intelligenz automatisiert werden kann. Nach einer Literaturrecherche zu den Themen Betriebsleitsystem und KI werden passende Verfahren ausgewählt und anhand von Daten aus der Praxis miteinander verglichen. Eine Evaluation prüft anhand ausgewählter Kennzahlen und im Vergleich mit Expertenentscheidungen prüfen Leistungsfähigkeit und Praxistauglichkeit der gewählten Verfahren. Eine kritische Diskussion, eine Zusammenfassung und ein Ausblick runden die Masterarbeit ab.

(EN) Unplanned service interruptions or traffic jams affect the public transport service stability negatively and thus customers feel frustrated and act self-consciously regarded to public transport services. Most smaller and medium-sized traffic companies do not have a control center which may arrange a re-routing and keep the service running. The aim of this work is to find procedures which enable a vehicle control system to arrange re-routing of public transport vehicles automatically based on artificial intelligence (AI). To achieve this, different approaches of AI are selected based on literature research and compared to each other based to datasets from the operation of several bus agencies. An evaluation confirms performance and productivity of the AI procedures found. A critical discussion and a summary show possible commitments in practice and complement the whole work.

Betreuender Hochschullehrer

Prof. Dr.-Ing. Thomas Schlegel

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit mit dem Titel

Entwicklung und Evaluation eines auf künstliche Intelligenz gestützten Systems zur Betriebslenkung von Linienbussen im Störungsfall

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht zu haben.

Karlsruhe, den 31. Januar 2023

Sebastian Knopf

Inhaltsverzeichnis

Abkürzungsverzeichnis

API *Application Programming Interface*
CSV *Comma Separated Values*
DQN *Deep Q-Network*
GTFS *General Transfer Feed Specification*
ITCS *Intermodal Transportation Control System*
KI *Künstliche Intelligenz*
KNN *Künstliches neuronales Netz*
MC *Monte-Carlo-Methode*
MEP *Markov-Entscheidungsprozess*
ML *Machine Learning, Maschinelles Lernen oder Maschinenlernen*
ÖPNV *Öffentlicher Personennahverkehr*
OSM *OpenStreetMap*
RBL *Rechnergestütztes Betriebsleitsystem*
RL *Reinforcement Learning*
TD *Temporal-Difference-Methode*
VDV *Verband Deutscher Verkehrsunternehmen*
XML *Extensible Markup Language*

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Häufig gilt der öffentliche Personenverkehr noch immer als unzuverlässig, wenig flexibel und zu kompliziert. Unvorhergesehene Ereignisse und Streckensperrungen beeinträchtigen die Fahrplanstabilität und bestätigen damit unnötigerweise das Bild des unzuverlässigen, unflexiblen öffentlichen Personenverkehrs. Auswertungen nach dem Ende des im Sommer 2022 von der Deutschen Bundesregierung initiierten 9-Euro-Tickets zeigen, dass neben dem Fahrpreis besonders auch die Angebotsqualität sowohl bezogen auf die Verfügbarkeit als auch auf die vorhandene Fahrgastinformationen erheblichen Einfluss auf die Verkehrsmittelverlagerung haben (vgl. Krämer 2022).

Durch die Marktöffnung im öffentlichen Personenverkehr sind insbesondere in Busnetzen immer häufiger mittelständische Verkehrsunternehmen mit der Betriebsdurchführung betraut. Bedingt durch den hohe und stetig wachsenden Kostendruck leisten sich kleine und mittelständische Verkehrsunternehmen selten eine Betriebsleitstelle, sodass insbesondere bei kurzfristigen Störungen nicht zeitgerecht reagiert und beispielsweise Umleitungen von zentraler Stelle angeordnet werden können. In den meisten Fällen kommt der Betrieb im Störungsfall kurzzeitig zum Erliegen. Selbst nach Ende der Störung kann es mitunter Stunden dauern, bis alle Fahrzeuge bedingt durch fehlende Wendezeiten in den Wagenumläufen die geplanten Fahrten wieder planmäßig durchführen können. Eine zeitgemäße Information der Fahrgäste unterbleibt in den meisten Fällen komplett, was wiederum zu Frustration und Abneigung auf Seiten der Fahrgäste führt.

Im Rahmen dieser Masterarbeit soll untersucht werden, inwieweit sich Daten aus der Planung als auch aus dem Betrieb als Entscheidungsgrundlage für dispositive Maßnahmen eignen und wie ein Betriebsleitsystem unterstützt durch künstliche Intelligenz damit selbstständig in die Lage versetzt werden kann, den Betrieb auch im Störungsfall möglichst aufrecht zu erhalten.

1.1 Motivation

Aus technischer Sicht betrachtet sind viele Lösungsansätze für Teilprobleme bereits verfügbar. Besonders in größeren Verkehrsunternehmen, die im Regelfall dann auch über eine über die gesamte Betriebszeit besetzte Betriebsleitstelle verfügen, sind Lösungen zur Unterstützung des Leitstellenpersonals im Störungsfall vielfach auch implementiert.

Bei kleineren und mittelständischen Verkehrsunternehmen ohne Betriebsleitstelle werden wertvolle Daten aus den Bordrechnern der Fahrzeuge hingegen nicht genutzt, obwohl genau diese eine gute Basis für zukünftige Entscheidungen sein könnten. Diese Entscheidungen könnten sowohl im Umfeld der Betriebslenkung als auch bei der Fahrgastinformation eine Basis zur stabilen Aufrechterhaltung des Betriebes sein. Was dies konkret bedeuten könnte, soll an einem selbst erlebten Beispiel aus dem Alltag verdeutlicht werden:

An einem Samstagvormittag möchte ich öffentliche Verkehrsmittel zu einer großen Veranstaltung nutzen. Erfreulicherweise fährt der kombinierte Stadt- und Regionalverkehr auch am Wochenende im 30-Minuten-Takt. Am Busbahnhof angekommen, sehe ich wie sich eine Busfahrerin etwas aufgeregt mit einem Kollegen unterhält. Beim Einstieg in den Bus erzählt mir die motivierte Busfahrerin, dass meine Zielhaltestelle wegen dem hohen Verkehrsaufkommen nicht mehr angefahren wird und sie mich stattdessen an einer alternativen Haltestelle absetzen wird, die zu Fuß aber dieselbe Entfernung bedeutet. Beim Ausstieg zeigt sie mir noch die Haltestelle, an der ich in den Bus zur Rückfahrt einsteigen solle.

Stunden später finde ich mich pünktlich zur Rückfahrt an der besagten Haltestelle ein, doch auch 20 Minuten nach der planmäßigen Abfahrtszeit taucht kein Bus auf. Einschlägige Informationsmedien und die lokale Fahrplanauskunft helfen mir kein Stück weiter, da sie weder Echtzeitdaten noch eine vernünftige Information über eine eventuelle Umleitung enthalten. Erst 30 Minuten später taucht ein Kleinbus auf, der dann noch einen stark von der Fahrplanauskunft abweichenden Weg zurück zum Busbahnhof fährt und mich schließlich mit einer Verspätung von rund 40 Minuten an den Busbahnhof zurückbringt.

Das hier beschriebene Problem gehört oftmals zum Alltag einer jeden Person, die den öffentlichen Personenverkehr täglich nutzt.

Was ist hier passiert? Die folgende stichpunktartige Analyse soll eine wahrscheinliche Antwort auf diese Frage liefern.

- Ausschlaggebend für die Umleitung war das starke Verkehrsaufkommen an einer Stelle im Verlauf der Buslinie.

- Eine Einhaltung des regulären Linienweges hätte dafür gesorgt, dass sich je Folgefahrt die Verspätung mit einem Delta von +10 Minuten erhöht hätte, da Wendezeiten in diesem Ausmaß im geplanten Wagenumlauf nicht enthalten sind.

- Da es keine Betriebsleitstelle, die korrigierend von zentraler Stelle eingreifen kann und auch kein Konzept für einen solchen Störfall gibt, entschied das Fahrpersonal vor Ort eigenständig, dass die Verspätung durch eigenständige Änderung des Linienweges vermieden wird. Zwar erfolgte eine Absprache und der den beiden aktuell diensthabenden Personen, ob diese aber auch an die Ablösung weitergegeben wurde, bleibt fraglich. Auch andere Fahrzeuge, die an dieser Absprache nicht beteiligt waren, wurden nicht informiert.

- Durch das eigenständige Entscheiden des Fahrpersonals kamen unterschiedliche Umleitungswege zur Anwendung, die jedoch weder einheitlich noch für Fahrgäste kommuniziert wurden. Die Folge davon war, dass die Fahrpersonale jeweils ihre eigenen, nicht einheitlichen Umleitungswege bedienten, die jedoch nicht aufeinander abgestimmt waren.

- Das Fehlen einer zentralen Anweisung sorgte für den willkürlichen Ausfall eines Linienabschnittes oder gar einer ganzen Fahrt. Die ausbleibende Fahrgastinformation machte das Chaos perfekt.

Das Wissen über technische Möglichkeiten und das eigene Interesse, Lösungen zur Behebung dieser längst bekannten Alltagsprobleme bei der Nutzung öffentlicher Verkehrsmittel zu liefern, bilden die Motivation zu dieser Masterarbeit.

1.2 Zielsetzung

Im Rahmen dieser Masterarbeit sollen Möglichkeiten erörtert werden, die ein Betriebsleitsystem in die Lage versetzen, selbstständig Umleitungen im Störungsfall anzuordnen. Dabei sollen möglichst Eingangsdaten verwendet werden, die ohnehin vorhanden sind oder mit wenig Aufwand erzeugt werden können. Ziel ist ein funktionaler Prototyp, welcher nach Kenntnis über eine Störung selbstständig eine möglichst optimale Umleitung der betroffenen Fahrzeuge anordnet und darüber hinaus auch für die Fahrgastinformation eingesetzt werden kann. Im Kern sollen dabei folgende Forschungsfragen beantwortet werden:

- Welche Daten müssen in welcher Qualität verfügbar sein, um diese zum Training einer künstlichen Intelligenz verwenden zu können?

- Welche Strategien eignen sich für den Einsatz zur automatischen Anordnung einer Umleitung von Linienbussen in einem Betriebsleitsystem?

- Bietet ein solches auf künstliche Intelligenz gestütztes Betriebsleitsystem Potenzial für einen zeitnahen Einsatz in der Praxis?

1.3 Vorgehensweise

Für das grundlegende Verständnis der Arbeit werden in **Kapitel 2** zunächst Grundlagen der künstlichen Intelligenz ermittelt. Darüber hinaus werden Begriffe aus dem täglichen Betrieb eines Verkehrsunternehmens definiert, die für das Verständnis der Arbeit wichtig sind.

Die folgenden **Kapitel 3** und **4** beschäftigen sich jeweils mit der Auswahl zweier zum Vergleich geeigneter ML-Algorithmen und passenden Anwendungsbeispielen dazu und der Implementierung des Funktionsprototypen.

Eine Evaluation in **Kapitel 5** basierend auf qualitativen Daten stellt die beiden ML-Algorithmen auf den Prüfstand und zeigt, ob diese überhaupt geeignet sind und welches Optimierungspotenzial sie bieten. Meinungen von Fachleuten verschiedener Verkehrsunternehmen fließen darüber hinaus als subjektive Einschätzung in die Evaluation ein und schärfen gewonnene Erkenntnisse bezogen auf die Praxistauglichkeit.

In **Kapitel 6** werden die Ergebnisse der Masterarbeit zusammengefasst und zur kritischen Diskussion gestellt.

2 Grundlagen

In diesem Kapitel sollen zunächst theoretische und technische Grundlagen der künstlichen Intelligenz (KI) erörtert und zusammengefasst werden. Neben den wichtigsten Grundlagen zu KI und Machine Learning sind zum Verständnis der Arbeit darüber hinaus auch Begriffe und Abläufe aus dem Betrieb des öffentlichen Personennahverkehrs (ÖPNV) nötig. Im Zweiten Teil des Kapitels werden bestehende Lösungen bei verschiedenen Verkehrsunternehmen beleuchtet. Der Blick geht hierbei bewusst über den Horizont der Linienbusse hinaus, um eventuell gesammelte Erfahrungen aus verwandten Bereichen des Verkehrs und der Logistik transferieren zu können.

2.1 Verwandte Arbeiten

Zur Einordnung der vorliegenden Arbeit werden in diesem Unterkapitel einige Arbeiten beleuchtet, die sich mit einem ähnlichen Forschungsgegenstand beschäftigen.

Das aktuell laufende Forschungsprojekt KARL (KARL 2021) beschäftigt sich mit dem zukunftsweisenden Einsatz von künstlicher Intelligenz in der Arbeit und beim Lernen. In der Anwendungsdomäne *Mobilität und autonomes Fahren* ist unter Anderem eine Arbeitspaket enthalten, bei dem ein KI-gestützter Vorschlagsassistent dem Leitstellenpersonal geeignete Maßnahmen zur Betriebsstabilisierung vorschlagen soll. Hierdurch erhofft man sich insbesondere Zeitersparnis, eine Verbesserung der Qualität des ÖPNV im Störungsfall und eine Entlastung der Mitarbeitenden. Verglichen mit dieser Arbeit zielt KARL jedoch darauf ab, dass eine Betriebsleitstelle generell vorhanden ist. Der Vorschlagsassistent soll keine Maßnahmen selbstständig anordnen, sondern lediglich dem Leitstellenpersonal verschiedene Maßnahmenpakete vorschlagen. Die Entscheidung über die Einleitung einer Maßnahme liegt bis zuletzt beim Leitstellenpersonal selbst. Da sich das Projekt noch in der Anfangsphase befindet, stehen noch keine detaillierten Informationen zu diesem Arbeitspunkt zur Verfügung, die eine Einordnung erleichtern würden.

Vergleichbare Forschungsschwerpunkte sind im Projekt U-THREAT (U-THREAT 2021) zu finden. Dieses Projekt beschäftigt sich mit der Frage, wie die Resilienz von U-Bahnsystemen bei störenden Einflüssen wie Überschwemmungen, Bränden oder Bombendrohungen so weit wie möglich aufrechterhalten werden kann. In solchen Fällen hängt die Betriebsstabilität maßgeblich von der Erfahrung des Leitstellenpersonals ab. Forschungsgrundlage ist ein

einheitliches Bewertungsschema für Stationen und Streckenabschnitte, welches als Basis für einen Maßnahmenkatalog dienen soll, mit dem das U-Bahnsystem zunächst mit Einschränkungen weiter betrieben wird, um schließlich wieder in den Normalbetrieb überzugehen. Hierbei wird das vorhandene Wissen aus der Angebots- und Betriebsplanung um künstlich klassifiziertes Wissen aus Daten über vergangene Störungen ergänzt. Ziel ist es, sowohl die Planungs-, als auch Dispositionsprinzipien mathematisch und maschinell verwertbar zu machen und auf bestehende Netze anzuwenden.

Eine dritte Arbeit von Levy et al. widmet sich einer ähnlichen Forschungsfrage. In dieser Arbeit geht es darum, mit Hilfe eines Machine Learning Algorithmus eine optimierte Route durch ein Stadtstraßennetz unter Berücksichtigung von Kriminalitätshotspots zu finden. Hierbei werden nicht vordefinierte Laufwege beurteilt, sondern mittels Machine Learning eine Route komplett dynamisch berechnet. Das Ergebnis lässt sich mit den bekannten Routingpräferenzen „Kürzester Weg“ oder „Schnellste Route“ vergleichen. Die Arbeit basiert jedoch auf der Annahme, dass sich die Umgebung nicht dynamisch verändern kann, sondern Kriminalitätshotspots bis zu einem neuen Trainingsdurchlauf stets gleichbleiben. In der vorliegenden Arbeit wird jedoch eine hochdynamische Umgebung behandelt, in der Störfälle kurzfristig auftreten und auch genauso wieder abklingen.

2.2 Definition künstlicher Intelligenz

Der Begriff der künstlichen Intelligenz spielt in dieser Masterarbeit eine Schlüsselrolle. Sie soll als Entscheidungsträger in einem Betriebsleitsystem eingesetzt werden. Doch wann genau agiert ein System *intelligent*? Nachfolgend werden einige Definitionsansätze beleuchtet, welche diese Frage beantworten und eine Arbeitsdefinition für künstliche Intelligenz im Kontext der Betriebslenkung im öffentlichen Personenverkehr liefern sollen. Grundsätzlich problematisch hierbei ist, dass der Begriff der Intelligenz selbst nicht definiert ist. So existiert keine einheitliche Definition von Intelligenz, welche als Referenz zur Beurteilung eines Systems dienen kann (vgl. Rimscha 2008, S. 105; Mainzer 2019, S. 2).

Die wohl bekannteste und gleichwohl älteste Definition für den Begriff der künstlichen Intelligenz stammt von Turing aus den 1950er Jahren. Dem sogenannten Turing-Test zur Folge ist ein System dann als intelligent anzusehen, wenn eine beobachtende Person nicht mehr in der Lage ist zu unterscheiden, ob sie mit einer Maschine oder einer natürlichen Person interagiert (vgl. Mainzer 2019, S. 10). Diese Aussage wurde entsprechend oft auch hinterfragt. Bekannt wurde die Kritik von Ada Lovelace, die behauptete, eine Maschine könne ausschließlich de Zweck dienen, für den sie letztendlich programmiert wurde (vgl. Liggieri und Müller 2019, S.

304). Noch eindrucksvoller wird die Kritik am Beispiel des „Chinesischen Zimmers“ nach Searle (1980). Hierbei handelt es sich um ein gedankliches Experiment mit Ursprung in der Psychologie, bei dem eine Person in einem Raum steht und Fragen auf Chinesisch gestellt bekommt. Als einziges Hilfsmittel verfügt die Person über ein Regelbuch, mit dessen Hilfe sie die Fragen ebenfalls auf Chinesisch beantworten muss. Eine außenstehende Person könnte zu dem Schluss kommen, dass die Person im Raum die Sprache Chinesisch beherrscht, obwohl sie in Wahrheit nur einen fest definierten Regelsatz abarbeitet und überhaupt nicht über Kenntnisse der chinesischen Sprache verfügt (vgl. Liggieri und Müller 2019, S. 304). Diese beiden Gegenargumente zeigen deutlich, dass die fehlende äußere Unterscheidbarkeit zwischen einem Menschen und einer Maschine noch nicht zwangsläufig eine Interaktion mit einem scheinbar intelligenten Individuum von bedeuten. Im letztgenannten Beispiel wäre dann Intelligenz im Spiel, wenn die Person innerhalb des Raumes sich tatsächlich Kenntnisse der chinesischen Sprache angeeignet hätte.

Einen weiteren, deutlich konkreteren Definitionsansatz liefert Mainzer (2019). Er betrachtet Intelligenz nicht als absolute, binäre Eigenschaft, die durch ein Individuum entweder erfüllt oder nicht erfüllt sein kann, sondern relativiert den Begriff. Statt einem absoluten Vorhandensein oder Nichtvorhandensein der Eigenschaft Intelligenz, wird diese in Graden angegeben und bereits nicht mehr auf beliebige Individuen, sondern schon auf „[Computer]Systeme“ (Mainzer 2019, S. 3) bezogen. Demzufolge hängt der Intelligenzgrad eines Systems „vom Grad der Selbstständigkeit, vom Grad der Komplexität des Problems und dem Grad der Effizienz der Problemlösung“ (Mainzer 2019, S. 3) ab. Diese Definition eröffnet in Bezug auf ein Betriebsleitsystem neue Möglichkeiten. So können Selbstständigkeit, Komplexität des Problems und Effizienz gemessen und in alternativen Systemen miteinander verglichen werden.

Noch kürzer fasst sich Rimscha (2008) bei seiner Definition. Diese besagt sinngemäß, dass ein System dann als intelligent angesehen werden kann, wenn selbstständig Probleme lösen kann, ohne vorher von einem Menschen gesagt zu bekommen, wie genau das Problem gelöst werden soll (vgl. Rimscha 2008, S. 105). Dieser Auffassung ähnelt auch die Definition von Nahrstedt (2012). Ein System im Kontext der künstlichen Intelligenz ist demnach als sogenanntes Expertensystem anzusehen, wenn die gelieferten Lösungen „von der Qualität her von denen eines menschlichen Experten nicht zu unterscheiden sind“ (Nahrstedt 2012, S. 246).

Der Begriff des Expertensystems taucht auch in der historischen Forschung zur künstlichen Intelligenz auf. Während bereits im Barockzeitalter versucht wird, Gedankengut und erlerntes Wissen allein auf Rechnungen zurückzuführen, wächst der Wunsch nach einer Universalsprache. Diese *Lingua Universalis* soll es unabhängig vom Inhalt möglich machen, Wissen in einem mathematischen Kalkül auszudrücken (vgl. Mainzer 2019, S. 8). Die Ergebnisse dieser Forschungen waren jedoch bis Ende der 1960er Jahre kaum praktisch einsetzbar, weshalb

sich in den 1970er Jahren der Begriff des Expertensystems etablierte. Hierunter versteht sich die Abgrenzung vom „allgemeinen Problemlöser“ (Mainzer 2019, S. 11) hin zu Expertenwissen in einem in sich geschlossenen und überschaubaren Bereich. Ein Expertensystem verfügt damit über begrenztes Wissen zu einem spezifischen Sachgebiet und kann mit diesem Wissen entsprechend automatisch Schlussfolgerungen ziehen. Erst diese Eingrenzung ermöglicht erste praxistaugliche Anwendungen basierend auf künstlicher Intelligenz (vgl. Mainzer 2019, S. 12).

Die Kernaussagen dieser Definitionen lassen sich zu einer Arbeitsdefinition für den Begriff des intelligenten Betriebsleitsystems zusammenfassen, die eine Bewertung des Funktionsprototypen aus Kapitel 3 ermöglichen und damit die Überprüfbarkeit sicherstellt.

Ein intelligentes Betriebsleitsystem ist demnach ein System, dass selbstständig in der Lage ist, komplexe Probleme effizient zu lösen, ohne dabei ausschließlich vordefinierte Ergebnisse aus einer endlichen Menge auszuwählen, wobei die gelieferten Ergebnisse hinsichtlich ihrer Qualität nicht von denen eines Menschen unterscheidbar sein dürfen.

Die folgenden Stichpunkte beschreiben beispielhaft, wie ein solches intelligentes Betriebsleitsystem in der Praxis ausgeprägt sein könnte:

- Die Menge der Ergebnisse ist nicht von vorneherein endlich, sondern richtet sich nach messbaren, relevanten Einflussgrößen und bildet dabei auch Schnittmengen von möglichen Kombinationen von Fahrwegen (Eigenständige Lösungsfindung)

- Die Erkennung einer Störungssituation soll vom Betriebsleitsystem automatisch basierend auf bekanntem Wissen erfolgen (Selbstständigkeit)

- Die erwartete Lösung bezieht sich auf die Anordnung von Umleitungen, nicht jedoch auf weitere Dispositive Maßnahmen wie beispielsweise Kurzwenden. Umleitungen sollen hingegen von nahezu beliebigem Umfang sein können (Komplexität)

- Die Lösung soll im Optimalfall schneller als von einem Menschen, zumindest aber in angemessen kurzer Zeit geliefert werden (Effizienz)

2.3 Überwachtes und unüberwachtes Lernen

Insgesamt wird im Machine Learning (ML) zwischen dem *überwachten Lernen* und dem *unüberwachten Lernen* unterschieden. Beim überwachten Lernen werden in *Trainingsdaten* enthaltene Eigenschaften, sogenannte *Features,* extrahiert und einem Ergebnis, dem sogenannten *Label,* zugeordnet. Um zu verhindern, dass das Modell ausschließlich mit den Trainingsdaten funktioniert, wird das Modell nach Abschluss des Trainings auf einen *Testdatensatz* angewandt. Auf diesem Weg kann die Performance ermittelt werden, welche das Modell mit unbekannten Daten erreicht. Die Überanspassung des Modells an die Trainingsdaten wird auch als *Overfitting* bezeichnet (vgl. Niebler 2018, S. 11).

Das unüberwachte Lernen verarbeitet hingegen grundsätzlich unbekannte Daten mit dem Ziel, bislang unbekannte Zusammenhänge innerhalb dieser Daten zu finden oder Inhalte nach verschiedenen Merkmalen zu ordnen. Ein nachgelagerter Test mit Testdaten erfolgt nicht. (vgl. Niebler 2018, S. 11).

Das konstruierte Beispiel der Haltestellensuche kann anhand dieser Definitionen nicht exakt eingeordnet werden. Tendenziell trifft die Beschreibung des überwachten Lernens eher zu. Die Nutzung des Systems kann dabei als fortlaufendes Training betrachtet werden. Die Trigramme der Eingabezeichenfolge werden dem jeweils von den Nutzenden tatsächlich gewählten Suchvorschlag zugeordnet. Die Kombination aus Trigrammen entspricht dabei den Features, der tatsächlich gewählte Suchvorschlag dem Label. Das typischerweise beim überwachten Lernen folgende Testing entfällt an dieser Stelle allerdings. Die verarbeiteten Daten werden direkt gespeichert und beim nächsten Suchvorgang genutzt. Eine Überanspassung ist dann möglich, wenn die Trigramme der Eingabezeichenfolge bereits aus anderen Eingaben enthalten und dadurch einem anderen Suchvorschlag zugeordnet sind als dem eigentlich gewünschten. Entsprechend könnte dem Beispiel auch der k-Nearest-Neighbours-Algorithmus zu Grunde liegen, welcher dem unüberwachten Lernen zuzuordnen ist (vgl. Niebler 2018, S. 11). Der k-Nearest-Neighbours-Algorithmus ordnet Daten dabei anhand ihrer Features in Dimensionen ein und sucht entsprechend die k Nachbarn, deren Features am nächsten am Ausgangsdatensatz liegen. Anhand dieser k Nachbarn wird dann versucht, eine Aussage über den Ausgangsdatensatz zu treffen.

Die folgende Abbildung verdeutlicht die Funktionsweise des k-Nearest-Neighbours-Algorithmus anhand einem Datensatz über den eine Aussage basierend auf den nächsten k Datensätzen aus zwei verschiedenen Klassen getroffen werden soll:

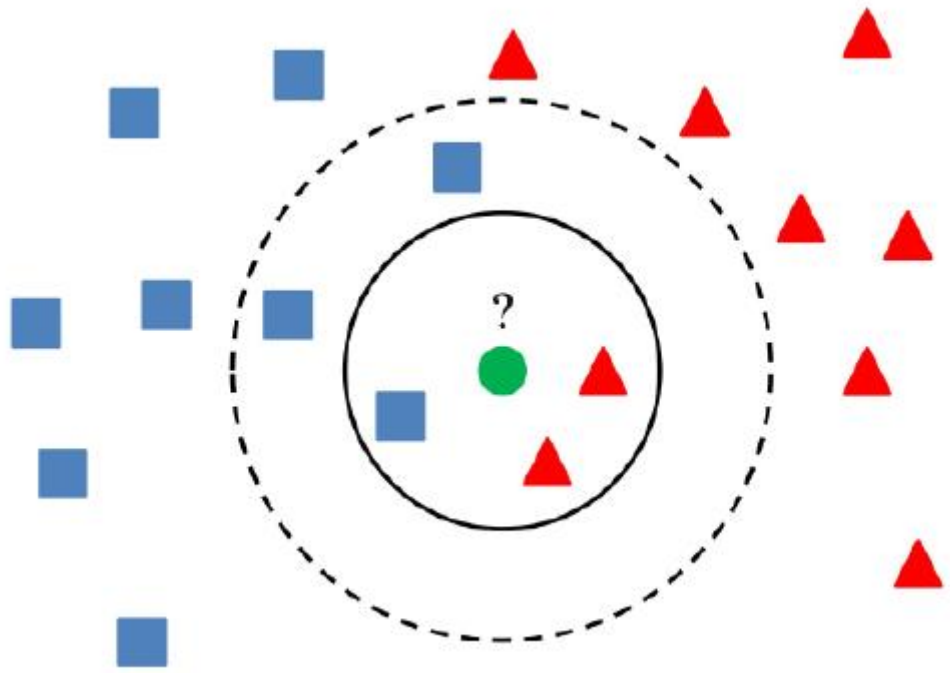


Abbildung 1: Graphische Darstellung des k-Nearest-Neighbours Algorithmus (Quelle: Alaliyat 2022, S. 38)

In grün dargestellt ist der Datensatz, über den eine Aussage getroffen werden soll. In rot und blau respektive als Dreieck und als Quadrat dargestellt, sind die bereits klassifizierten Datensätze. Der k-Nearest-Neighbours-Algorithmus ermittelt nun die k Datensätze, die am nächsten liegen und verwendet diese als Grundlage für eine Aussage über den neuen Datensatz (vgl. Alaliyat 2022, S. 38). Je größer k ist, desto größer ist im Regelfall auch der Suchraum. Am Beispiel der Haltestellensuche käme eine Zählung der Trigramme der Eingabezeichenfolge mit anschließendem Abgleich zwischen bereits einmal gesuchten Trigrammen und anschließender Suche mittels k-Nearest-Neighbours-Algorithmus am nächsten.

2.4 Bestärkendes Lernen

Im vorhergehenden Kapitel wurde Machine Learning bereits in beiden verbreiteten Verfahrensarten des überwachten und unüberwachten Lernens unterteilt. In diesem Kapitel wird entkoppelt vom Beispiel der Haltestellensuche das sogenannte *bestärkende Lernen* betrachtet.

Üblicherweise wird das bestärkende Lernen auch als Reinforcement Learning (RL) bezeichnet. Im Gegensatz zum überwachten und unüberwachten Lernen benötigt RL keine Datenbasis mit entsprechenden Features und Labels (vgl. Schmitz 2017, S. 12). RL liefert als Ergebnis eine Folge von *Aktionen*, die beim Eintreten eines gewissen Falles optimalerweise durchgeführt werden sollen. Das Training eines sogenannten *Agenten* erfolgt nach einem „Trial-and-Error" Verfahren in einer Simulation des späteren Einsatzgebietes, wobei eine virtuelle *Belohnung* zur Bewertung einer Aktion dient. Ziel des RL ist es, in einer *Umgebung* ausgehend von einem gewissen *Zustand* durch eine bestimmte Strategie bestehend aus geeigneten Aktionen ein Maximum an Belohnung zu erhalten (vgl. Chan et al. 2022, S. 47). Die Verfahrensweise von RL ähnelt dem Lernprozess bestimmter Lebewesens sehr. Ein bekanntes Beispiel dazu ist ein Baby, dass Laufen lernt (vgl. Schmitz 2017, S. 12). Schafft es das Baby, eine bestimmte Strecke zu laufen, wird es gelobt. Es kommt zu einem Erfolgserlebnis, welches in diesem Fall die positive Belohnung repräsentiert. Fällt das Baby hingegen hin, führt dies zum Scheitern und damit zu einer negativen Belohnung. Das Beispiel lässt sich ähnlich auch auf ein autonomes Fahrzeug übertragen, welches ein bestimmtes Ziel erreichen soll. Das Fahrzeug kann abbiegen oder geradeaus fahren. Erreicht das Fahrzeug das geforderte Ziel ohne einen Unfall, folgt eine positive Belohnung. Kommt es hingegen zu einem Zusammenstoß mit einem anderen Fahrzeug, folgt daraus eine negative Belohnung (vgl. Chan et al. 2022, S. 47). Generell werden im RL eine *Policy-* und eine *Valuefunktion* definiert. Während die Policyfunktion eine Wahrscheinlichkeit angibt, mit der eine bestimmte Aktion ausgeführt wird, definiert die Valuefunktion, ob ausgehend von einem bestimmten Zustand beim Ausführen einer Aktion eine negative oder positive Belohnung zu erwarten ist (vgl. Schmitz 2017, 12 ff.).

Die folgende Grafik zeigt das Zusammenspiel zwischen Agenten und Umgebung im Reinforcement Learning:

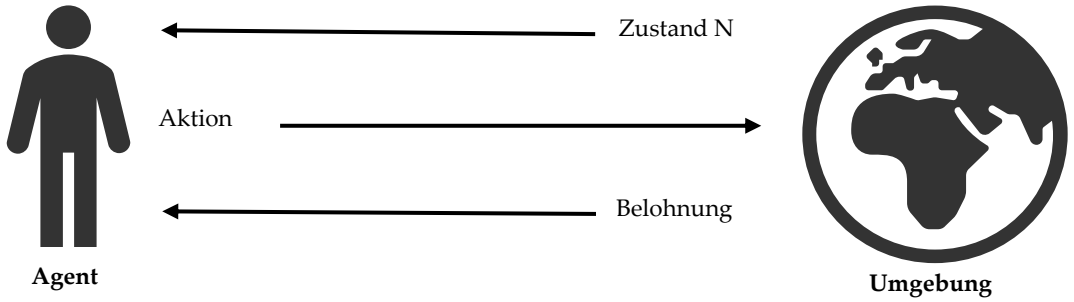


Abbildung 2: Graphische Darstellung des Reinforcement Learning (in Anlehnung an Schmitz 2017, S. 12)

Ausgehend von einem bestimmten Zustand der Umgebung entscheidet sich der Agent für eine verfügbare Aktion. In Abhängigkeit der Valuefunktion erhält der Agent dafür eine positive oder negative Belohnung, welche wiederrum Einfluss darauf nehmen wird, welche Aktion beim nächsten Durchgang mit demselben Ausgangszustand gewählt wird. Je nach Dauer und Detaillierungsgrad es Trainings handelt es sich also um eine Kombination aus „Trial and Error" und bereits erlerntem Wissen, welches der Agent für seine Entscheidungsfindung heranzieht (vgl. Lüth 2019, S. 2). Daraus folgt, dass der Agent umso sicherer in seiner Entscheidung wird, je häufiger er einen bestimmten Ausgangszustand mit darauffolgender Bewertung trainiert hat.

Üblicherweise wird die Umgebung als sogenannter Markov-Entscheidungsprozess (MEP) modelliert (vgl. Lüth 2019, S. 2; Schmitz 2017, S. 7). Der nach dem russischen Mathematiker Markov benannte MEP dient zur Modellierung eines Entscheidungsproblems, bei dem ausgehend vom einem gewissen Zustand eines Systems beim Ausführen einer Aktion mit einer gewissen Wahrscheinlichkeit ein bestimmter Folgezustand erreicht wird. Jeder Zustandsübergang wird darüber hinaus durch eine Kostenfunktion bewertet (vgl. Böhm 2016, 14 f.). Außer der Beschreibung der Umgebung als Simulationsmodell ist darüber hinaus kein zusätzliches, menschliches Fachwissen in aufbereiteter Form für das Training erforderlich (vgl. Schmitz 2017, S. 12). Das Simulationsmodell muss darüber hinaus kein vollständig mathematisches oder stochastisches Modell sein, was insbesondere in komplexen Umgebungen mit vielen Einflussfaktoren zum Vorteil wird (vgl. Lüth 2019, S. 3).

Durch die Anwendung des „Trial-and-Error" Prinzips folgt, dass das Training vorab auch entfallen und stattdessen im laufenden Betrieb einer Umgebung durchgeführt werden kann. In der Praxis ist das jedoch allenfalls für unkritische, unterstützende Anwendungen möglich, bei der der Nutzen durch Einsatz von RL einen eventuellen Schaden während der Trainingsphase eliminiert. Man stelle sich jedoch im Gegenzug vor, wo die Forschung rund um das autonome

Fahren nach heutigem Stand wäre, hätte man ein RL-Training für ein Fahrzeug wie im letzten Beispiel tatsächlich im öffentlichen Straßenverkehr durchgeführt.

2.5 Künstliche Intelligenz und maschinelles Lernen

Die Begriffe künstliche Intelligenz und maschinelles Lernen oder auch Maschinenlernen genannt (ML) tauchen oft im selben Kontext auf. In den letzten Jahren werden diese Schlüsselwörter dem Anschein nach besonders häufig dann verwendet, wenn ein Projekt nach außen als besonders innovativ oder forschungsintensiv dargestellt werden soll. Stellenweise werden beide Begriffe auch als Synonym verwendet. In diesem Kapitel werden KI, ML und verwandte Begriffe aus diesem Fachgebiet voneinander abgegrenzt.

Zur Verdeutlichung der Abgrenzung zwischen KI und ML sei das folgende Beispiel konstruiert:

In einem Eingabefeld können Nutzende Haltestellennamen eingeben und aus einem Dropdown mit Vorschlägen die gewünschte Haltestelle auswählen. Dabei werden Tippfehler in gewissen Grenzen toleriert und Vorschläge auch dann angezeigt, wenn keine exakte Übereinstimmung gefunden wurde. Werden Abkürzungen verwendet, die zu keiner direkten Übereinstimmung führen, merkt das System sich, welche Haltestelle die Nutzenden nach ihrer Eingabe aus den Vorschlägen auswählen und setzt diesen Vorschlag beim nächsten Suchvorgang im Ranking nach oben.

Die nachfolgend ausgeführten Leitfragen lauten nun:

- Handelt es sich bereits um eine Form der KI?
- Kommen hierbei Ansätze aus dem ML zum Einsatz?

Handelt es sich bereits um eine Form der KI?

Das System ist in der Lage, kleinere Schreibfehler zu korrigieren und den Nutzenden trotzdem passende Suchvorschläge anzuzeigen. Wird so beispielsweise „Kallsruh Hbf“ statt „Karlsruhe Hbf“ eingegeben, ist das System dennoch in der Lage, die korrekte Haltestelle „Karlsruhe Hbf“ zu ermitteln. Würde einem Menschen mit entsprechendem Fachwissen dieselbe Aufgabe stellen, wäre auch dieser in der Lage trotz des offensichtlichen Fehlers passende Ergebnisse vorzuschlagen.

Eine Möglichkeit, um solche Ergebnisse zu erhalten, ist die Verwendung sogenannter *unscharfer Mengen*. Im Gegensatz zu einer klassischen Menge, welche bezogen auf das obige Beispiel eine

einfache Liste aller verfügbaren Haltestellennamen im Klartext wäre, würde eine unscharfe Menge die Haltestellennamen unterteilt in Buchstabenblöcken zu jeweils drei Buchstaben enthalten. Diese Buchstabenblöcke werden auch als *Trigramme* bezeichnet. Aus dem Wort „Karlsruhe Hbf“ würden sich demnach die Trigramme

KAR ARL RLS LSR SRU RUH UHE HBF

ergeben.

Der durch die Nutzenden eingegebene Text wird ebenfalls in seine Trigramme zerlegt und für jedes Element in der unscharfen Menge ermittelt, wie viele Trigramme aus der Eingabezeichenfolge im jeweiligen Element der unscharfen Menge enthalten sind. Das Ergebnis ist ein Überdeckungsgrad auf dem Intervall [0; 1].

Am Beispiel der Eingabezeichenfolge „Kallsruh Hbf“ ergäben sich die Trigramme

KAL ALL LLS **LSR SRU RUH HBF**

wovon die fett markierten Trigramme auch im entsprechenden Element der unscharfen Menge enthalten sind. Durch den entstehenden Überdeckungsgrad > 0 kann mit einer gewissen Wahrscheinlichkeit davon ausgegangen werden, dass mit der Eingabezeichenfolge eigentlich das korrespondierende Element der unscharfen Menge gesucht ist, wenngleich keine hundertprozentige Übereinstimmung mit der Eingabezeichenfolge vorliegt.

Die folgende Abbildung zeigt die graphische Darstellung einer klassischen und einer unscharfen Menge am Beispiel von Temperaturen im Vergleich:

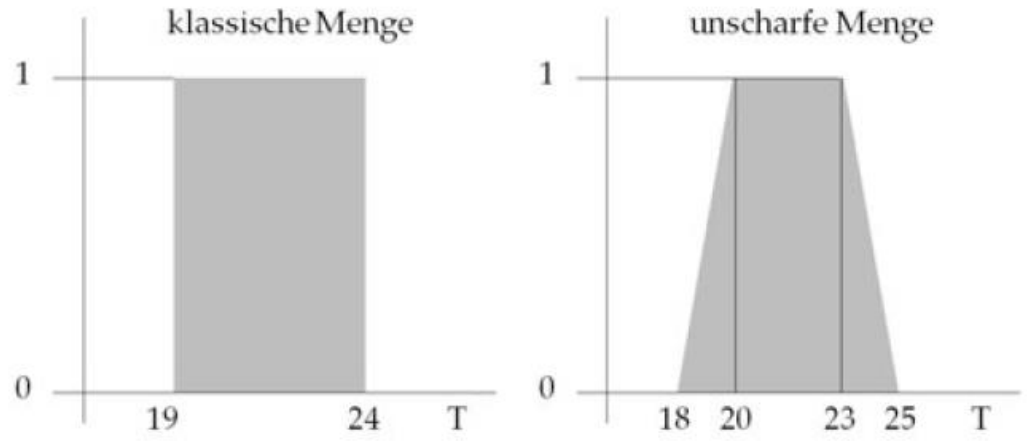


Abbildung 3: Darstellung einer klassischen und einer unscharfen Menge(Quelle: Nahrstedt 2012, S. 237)

Die Temperaturen werden in der Klassifikation „tief“, „normal“ und „hoch“ durch Verwendung der unscharfen Mengen nicht exakt abgegrenzt, sondern bilden das tatsächliche

Empfinden verschiedener Menschen ab. So liegt eine „niedrige“ Temperatur bei einer Person schon bei 15° C, bei einer anderen Person aber erst bei 5° C. Umgekehrt empfindet eine Person eine Umgebungstemperatur von 20° C bereits als „warm“, während eine andere Person hier erst im Bereich von „angenehm“ ankommt.

Diese Überdeckung von verschiedenen Temperaturempfinden ist in der folgenden Abbildung graphisch dargestellt:

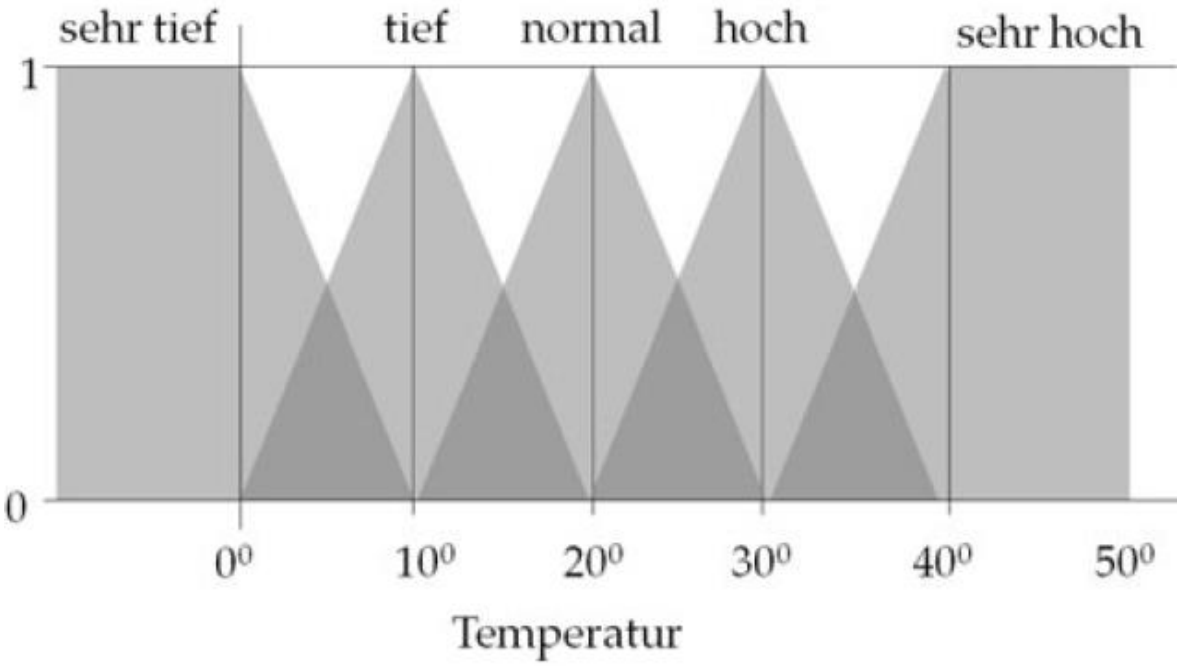


Abbildung 4: Graphische Darstellung der Überdeckung unscharfer Mengen (Quelle: Nahrstedt 2012, S. 238)

Dasselbe Prinzip lässt sich auf die Suche nach Haltestellennamen übertragen. Die Zerlegung der Eingabezeichenfolge und der tatsächlichen Haltestellennamen bilden mehrere unscharfe Mengen, die sich zu einem gewissen Grad überdecken können und damit eine teilweise Übereinstimmung feststellbar machen. Wäre die Suche hingegen in einer klassischen Menge durchgeführt worden, hätte dies zu keinem Ergebnis geführt, da die Eingabezeichenfolge mit keinem der Elemente exakt übereinstimmt. Dieses Verfahren wird auch als *unscharfe Suche* oder *Fuzzy-Suche* bezeichnet und ist eine bekannte KI-Anwendung (vgl. Engfer 2002, S. 3).

Die erste Bedingung, um von einer KI gemäß der zuvor festgelegten Arbeitsdefinition ausgehen zu können ist damit erfüllt. Die Qualität der Ergebnisse der KI ist im Nachhinein nicht von der Qualität der Ergebnisse eines Menschen mit entsprechendem Fachwissen zu unterscheiden. Folglich darf beim verwendeten Beispiel von einer Form der KI ausgegangen werden.

Kommen hierbei Ansätze aus dem ML zum Einsatz?

Grundsätzlich ist ML eine Möglichkeit, die „Programmen die Möglichkeit gibt, mit Hilfe von Daten zu lernen, ohne explizit programmiert zu werden“ (Niebler 2018, S. 10). Diese Bedingung ist im Beispiel der Haltestellensuche durch die Zuordnung von möglichen Eingaben zu tatsächlich existenten Haltestellen gegeben.

Die Haltestellensuche lässt sich mit diesen Erkenntnissen jedoch nicht exakt dem überwachten oder unüberwachten Lernen zuordnen. Das bestärkende Lernen scheidet per Definition bereits aus. Vielmehr hängt eine genaue Einordnung von der Implementierung der Haltestellensuche innerhalb eines Systems ab. Generell lässt sich jedoch festhalten, dass die Haltestellensuche aus dem Beispiel durchaus lernfähig ist, ohne dabei in die Implementierung einzugreifen. Angelehnt an die Definition von Niebler (2018) kann die Haltestellensuche zumindest klar dem Machine Learning zugeordnet werden.

2.6 Zusammenfassender Vergleich von ML-Verfahren

In den vorangegangenen Kapiteln wurde der Begriff des Machine Learning von der künstlichen Intelligenz abgegrenzt und in überwachtes, unüberwachtes und bestärkendes Lernen aufgeteilt. In diesem Kapitel werden die drei ML-Verfahren einander gegenübergestellt und zusammenfassend miteinander verglichen.

Überwachtes Lernen

Das überwachte Lernen arbeitet mit einem Eingangsdatensatz, in dem bestimmte Eigenschaften ihren korrekten Werten, den sogenannten Labels, zugeordnet sind. In der Trainingsphase wird das Modell mit dem Eingangsdatensatz trainiert. Nach Abschluss des Trainings wird das Modell mit einem zweiten Datensatz geprüft. Ziel ist die Evaluation des Modells und eine Überanpassung zu vermeiden. Das überwachte Lernen eignet sich besonders für Anwendungen, über die eine Vielzahl an möglichst umfangreichen Daten vorliegen. Optimalerweise sind diese Daten bereits gelabelt. Andernfalls muss zunächst menschliches Wissen eingebracht werden, um dem Modell eine Trainingsgrundlage zu schaffen

Unüberwachtes Lernen

Im Gegensatz dazu enthält der Trainingsdatensatz beim unüberwachten Lernen keine Labels, die Evaluation mittels eines zweiten Datensatzes entfällt ebenso. Das unüberwachte Lernen eignet sich damit besonders, um unbekannte Zusammenhänge zwischen Daten zu finden oder Daten in bestimmte Cluster einzuordnen. Menschliches Vorwissen ist folglich nicht erforderlich.

Bestärkendes Lernen

Abweichend von den beiden zuvor genannten ML-Verfahren benötigt das bestärkende Lernen, auch Reinforcement Learning genannt, gar keinen Trainingsdatensatz. Stattdessen findet das Training in einer Simulation des späteren Einsatzgebietes statt, das Modell trainiert sich selbst anhand eines "Trial-and-Error" Verfahrens. Damit eignet sich das bestärkende Lernen besonders für Anwendungen mit komplexen Einflussfaktoren, ohne dass zuvor alle möglichen Rückschlüsse durch menschliches Vorwissen ausreichend trainiert wurden.

2.7 Algorithmen aus dem Reinforcement Learning

In Kapitel 2 wurde der Begriff des Machine Learning bereits in die drei Kategorien überwachtes, unüberwachtes und bestärkendes Lernen - oder Reinforcement Learning - eingeordnet. Reinforcement Learning scheint für die Anwendung in dieser Arbeit besonders geeignet, da es weder einen gelabelten Eingangsdatensatz noch menschliches Vorwissen voraussetzt. Lediglich die Modellierung der Umgebung mit ihren wichtigsten Eigenschaften wird benötigt. Daher wird das Reinforcement Learning und die dem RL ungeordneten Algorithmen in diesem Kapitel einer tiefergreifenden Betrachtung unterzogen. Ziel ist es, zwei zur Implementierung im Prototyp und anschließendem Vergleich geeignete Algorithmen zu identifizieren.

Typisch für RL-Algorithmen ist, dass sie ihr Wissen selbstständig durch ein "Trial-and-Error" Verfahren in einer Simulation ihrer späteren Umgebung erarbeiten. Das Modell der Umgebung ähnelt dabei einem Markov-Entscheidungsprozess (vgl. Lüth 2019, S. 2). Für eine Umgebung mit einer überschaubaren Anzahl an Zuständen könnten die Ergebnisse des Trainingsprozesses einfach in einer Lookup-Table gespeichert und bei Bedarf abgerufen werden. Reelle Umgebungen können jedoch schnell eine unüberschaubare Anzahl an Zuständen annehmen, wie das folgende Beispiel verdeutlichen soll.

Im Beispieldatensatz in dieser Arbeit, welcher alle mit einem Linienbus befahrbaren Strecken im Einzugsgebiet zwischen der Gemeinde Engelsbrand und der Stadt Pforzheim enthält, sind

insgesamt 6416 Knotenpunkte vorhanden, auf denen mögliche Routen berechnet werden können. Die Anzahl der Knotenpunkte sei fortan als $|N|$ bezeichnet. Ein Systemzustand wird durch die aktuelle Linie, die gewählte Umleitung und den gesperrten Knotenpunkt ausgedrückt. Die Anzahl möglicher Zustände ergibt sich demnach angenähert aus:

$$|R| * |D| * |N|$$

wobei $|R|$ die Anzahl der Linien und $|D|$ die Anzahl der insgesamt bekannten Umleitungen sind.

Für ein durchschnittliches Netz aus 15 Linien mit jeweils 4 üblichen Umleitungsfahrwegen und rund 10.000 Knotenpunkten ergäbe dies eine Zustandsmenge von 9.000.000 Zuständen.

2.7.1 Wert- und Strategieapproximation

Weiter gilt es zu unterscheiden zwischen Wert- und Strategie-Approximationsalgorithmen. Diese werden respektive auch als Off-Policy- und On-Policy-Algorithmen bezeichnet. Während On-Policy-Algorithmen dieselbe Funktion zur Bewertung und Steuerung des Agenten eingesetzt wird, können diese Funktionen bei Off-Policy-Algorithmen diese beiden Funktionen getrennt betrachtet werden (vgl. Lorenz 2020, S. 62). In diesem Fall gibt es also jeweils eine Value- und eine Policy-Funktion. Während erstere für die Bewertung der Ausführung einer Aktion in einem bestimmten Zustand (vgl. Schmitz 2017, S. 13) verantwortlich ist, steuert letztere das Verhalten des Agenten in der Umgebung durch die Auswahl einer Aktion in einem bestimmten Zustand anhand ihrer Auswahlwahrscheinlichkeit (vgl. Schmitz 2017, S. 16). In diesem Unterkapitel werden die wesentlichen Unterschiede beider Approximationsarten erläutert.

Bei Off-Policy-Algorithmen wird eine Wert-Funktion definiert, welche einzelne mögliche Zustände oder mögliche Aktionen innerhalb eines Zustandes bewertet. Hierzu wird das Ursprungsproblem in mehrere Teilprobleme zerlegt, deren optimale Teillösungen dann zur einer optimalen Gesamtlösung führen. Nach Ausführen einer Aktion und dem Erreichen eines Folgezustandes wird die erwartete und die tatsächlich erhaltene Belohnung des Agenten überprüft und die Parameter der Wert-Funktion entsprechend angepasst (vgl. Dammann o.D., S. 6). Durch die Initialisierung der Wert-Funktion mit niedrigen Bewertungen bleiben die die Bewertungen von Zuständen oder Aktionen in bestimmten Zuständen nach jedem Schritt entweder gleich oder werden verbessert, bis sie schließlich gegen eine Maximalbewertung konvergieren. Unterschreitet die maximale Bewertungsänderung nach einem Schritt einen festzulegenden Schwellenwert, kann das Training als beendet betrachtet werden. Das Ausführen der Aktionen, welche zur entsprechend höchsten Bewertung führen, beschreibt dann die optimale Strategie eines Agenten (vgl. Lorenz 2020, S. 23).

Im Gegensatz dazu wird bei On-Policy-Algorithmen versucht, direkt die Parameter der Strategie-Funktion anzupassen. Der Weg über die Bewertungsfunktion entfällt dabei. Die Strategie-Funktion beschreibt eine Wahrscheinlichkeit dafür in einem Zustand eine bestimmte Aktion zu wählen (vgl. Schmitz 2017, S. 16). Bei der Strategieapproximation gilt es, die Parameter der Strategie-Funktion derart anzupassen, dass Aktionen mit möglichst hohem erwartetem Gewinn gewählt werden. Das Parameterupdate kann dazu nach jedem Einzelschritt, aber auch erst nach Abschluss einer gesamten Episode durchgeführt werden (vgl. Schmitz 2017, S. 16). In gefährlichen Umgebungen, in denen auch kleine Fehler schwerwiegende Folgen haben können, sind On-Policy-Algorithmen besser geeignet, da sie Verluste aus dem Training berücksichtigen und generell Aktionen mit höherer Bewertung bevorzugen (vgl. Williams 1992, S. 288).

Jeder RL-Algorithmus nutzt zumindest eine Wert- oder Strategiefunktion zur Steuerung seines Agenten (vgl. Schmitz 2017, S. 18). Einer der Hauptunterschiede beider Strategien ist, dass Off-Policy-Algorithmen zunächst den erwarteten Gewinn des Agenten schätzen, jedoch erst nach dem tatsächlichen Ausführen einer Aktion Kenntnis über den tatsächlichen Gewinn erlangen. On-Policy-Algorithmen arbeiten hingegen ausschließlich auch den Erfahrungswerten tatsächlich ausgeführter Episoden.

2.7.2 Monte-Carlo- und Temporal-Difference-Methoden

Außerdem können RL-Algorithmen in den Monte-Carlo-Methoden (MC-Methoden) oder Temporal-Difference-Methoden (TD-Methoden) zugeordnet werden.

MC-Algorithmen basieren auf einer zufallsbasierten Exploration. Zunächst wird eine hinreichend große Anzahl Episoden ausgehend von einem bestimmten Zustand durchgeführt, welche im Nachgang bewertet werden. Die Bewertungen der unterschiedlichen Episoden werden anschließend gemittelt (vgl. Wagner 2018, S. 31). Die Bewertung führt anschließend zu einem Update der Bewertungsfunktion (vgl. Lorenz 2020, S. 56). Entsprechend setzen MC-Algorithmen voraus, dass es mindestens einen Terminalzustand gibt (vgl. Wagner 2018, S. 31). Da MC-Algorithmen ausschließlich auf der Erhebung und Auswertung von Zufallsexperimenten basieren, kommen sie gänzlich ohne ein mathematisches Modell aus. Man spricht daher auch von *modellfreien Algorithmen*.

Im Gegensatz dazu stehen TD-Algorithmen, welche anstelle von ganzen Episoden jeweils Einzelschritte in zeitlichen Abständen betrachten und bewerten (vgl. Lorenz 2020, S. 59). Die Bezeichnung rührt aus der Annahme, dass die Einzelschritte zeitlich betrachtet nacheinander ausgeführt und bewertet werden. TD-Algorithmen nutzen die Eigenschaft der Bellmanngleichung aus, der zufolge nach die Bewertung eines Folgezustandes direkt von der Bewertung des aktuellen Zustandes abhängt (Wagner 2018). Folglich ist ausgehend von einem

bestimmten Zustand möglich, die erwartete Belohnung des Agenten für alle möglichen Folgezustände abzuschätzen. Wie MC-Algorithmen auch, können TD-Algorithmen komplett ohne ein Modell der Umgebung auskommen und zählen damit ebenso zu den modellfreien Algorithmen (vgl. Arnold 2021, S. 13). Durch Berücksichtigen einer Lernrate können die Wahrscheinlichkeiten aus einem Markov-Modell mit einbezogen werden. Im Verlauf des Trainings wird die Lernrate beispielsweise mit jedem Besuch eines Zustandes verkleinert werden, sodass die Bewertung schließlich stabilisiert wird (vgl. Lorenz 2020, S. 60). Die Berücksichtigung von Wahrscheinlichkeiten aus einem Markov-Modell setzt allerdings voraus, dass die Umgebung zumindest mit ihren relevantesten Eigenschaften als solches definiert ist.

2.7.3 Einordnung bekannter RL-Algorithmen

In den vorhergehenden Unterkapiteln wurden zwei Klassifizierungsarten für RL-Algorithmen eingeführt. In diesem Unterkapitel werden ausgewählte RL-Algorithmen vorgestellt und in ihre entsprechende Klasse eingeordnet. Diese Einordnung ermöglicht später eine gezielte Auswahl zweier zum Vergleich geeignete RL-Algorithmen.

2.7.3.1 Q-Learning

Der Q-Learning-Algorithmus, nachfolgend abgekürzt als Q-Learning bezeichnet ist der älteste RL-Algorithmus aus dem Jahr 1989 und kommt ohne Modell einer Umgebung aus (vgl. Lorenz 2020, S. 61). Q-Learning basiert auf einer Bewertung durch eine sogenannte *Action-Value-Funktion* aus der Klasse der Wertfunktionen. Die Action-Value-Funktion gibt an, welchen Gewinn der Agent zu erwarten hat, wenn er ausgehend von einem bestimmten Zustand eine bestimmte Aktion ausführt. Ein trainierter Agent ist in der Lage, eine optimale Strategie für Handlungen in einem MEP zu finden. Voraussetzung dafür ist eine hinreichend große Anzahl an Trainingsversuchen mit entsprechend ausführlicher Exploration in der Trainingsphase (vgl. Arnold 2021, S. 14). Nach dem Training wird in jedem Zustand jene Aktion gewählt, die dem Agenten den höchsten erwarteten Gewinn einbringt (vgl. Schmitz 2017, S. 13). Das gierige Verhalten entspricht einer Greedy-Strategie (vgl. Gass und Fu 2013, S. 666).

Das Training in Q-Learning kann beispielsweise durch Hinzufügen eines zusätzlichen Zufallsparameters in die Strategie-Funktion erfolgen. Mit diesem Zufallsparameter wird festgelegt, dass der Agent mit der Wahrscheinlichkeit des Zufallsparameters die Aktion mit der höchsten Bewertung auswählt. Alternativ wird eine beliebige, andere Aktion mit einer niedrigeren Bewertung gewählt. Je größer der Zufallsparameter ist, desto höher ist die Explorationsrate, durch die der Agent verschiedene Aktionen auswählt und letztendlich trainiert wird (vgl. Dammann o.D., S. 8). Der Zufallsparameter kann zu Beginn des Trainings sehr groß gewählt werden, um die Exploration zu fördern und mit fortschreitendem Training

schrittweise verkleinert werden. Bedingt durch den Zufallsparameter ist es dadurch auch möglich, dass die tatsächlich gewählte Aktion nicht jene mit dem höchsten erwarteten Gewinn ist. Die Episode mehrerer Aktionen spielt für die im Q-Learning erarbeitete Strategie daher keine Rolle (vgl. Schmitz 2017, S. 14).

Da Wert- und Strategiefunktion in diesem Fall getrennt betrachtet werden können, kann Q-Learning also den Off-Policy-Algorithmen zugeordnet werden. Die schrittweise, nicht-episodische Bewertung macht außerdem eine Einordnung des Q-Learning bei den TD-Algorithmen möglich.

2.7.3.2 Deep Q-Learning

Eine Sonderform des Q-Learning stellt das Deep Q-Learning dar. Nachfolgend wird das Deep Q-Learning auch als Deep Q-Network (DQN) bezeichnet. Hierbei werden klassisches Q-Learning und Deep Learning kombiniert. Ein künstliches neuronales Netz (KNN) wird dabei als Funktionsapproximator für die Wert-Funktion eingesetzt (vgl. Larsson 2018, S. 19). Als Eingangsdaten erhält das KNN den aktuellen Zustand, als Ergebnis erhält man die Bewertungen der einzelnen Aktionen (vgl. Dammann o.D., S. 11; Lüth 2019, S. 8). Anstelle einer initial definierten Wert-Funktion wird lediglich eine Bewertungs-Funktion benötigt, welche „beurteilt, wie gut die gewählten Input-Output-Paare sind“ (Huber 2018, S. 23). Eine Anpassung der gesonderten Strategie-Funktion durch das Training erfolgt nicht, weshalb auch das Q-Learning den Off-Policy-Verfahren zugeordnet werden kann (vgl. Huber 2018, S. 27). Die Updates erfolgen jeweils nach jedem Einzelschritt, weshalb eine Zuordnung zu den TD-Algorithmen nahe liegt.

Das Training von KNN mit sequenziellen Updates, wie es beim RL der Fall ist, kann bedingt durch unerwünschte Korrelationen zwischen den Daten zu einer Verfälschung durch Fehlanpassung des KNN führen (vgl. Larsson 2018, S. 19). Der DQN-Algorithmus kompensiert dieses Verhalten zum einen durch die Verwendung zweier KNN mit unterschiedlichen Parametern und zum anderen durch die regelmäßige Einbindung von Erfahrungswerten während des Trainings. Die beiden KNN werden jeweils als *Online Network* und als *Target Network* bezeichnet. Zwar führt diese Verfahrensweise zu einem langsameren, dafür stabileren Training des KNN und folglich zu einer exakteren Approximation der Wert-Funktion (vgl. Larsson 2018, 19 f.).

2.7.3.3 SARSA

Der SARSA-Algorithmus lernt im Vergleich zum Q-Learning nicht aus dem nächsten beobachteten Zustand, sondern aus dem tatsächlich erreichten Gewinn, der durch den Übergang ein einen Zustand erreicht wurde (vgl. Lorenz 2020, S. 62). Der wesentliche

Unterschied zum Q-Learning besteht in der Steuerung des Agenten. Im SARSA-Algorithmus wird der Agent ausschließlich über seine Strategie-Funktion gesteuert, die gleichzeitig explorativ ausgeprägt ist (vgl. Lorenz 2020, S. 62). SARSA ist folglich den On-Policy-Algorithmen zuzuordnen.

SARSA ist ein TD-Algorithmus, das heißt, die Updates folgen jeweils nach dem nächsten Schritt (vgl. Arruda et al. 2020, S. 4). Alternativ besteht beim SARSA-Algorithmus auch die Möglichkeit, eine sogenannte *Aktionshistorie* zu führen. In dieser Aktionshistorie werden alle Aktionen, die Teil einer bestimmten Episode sind, rückwirkend gespeichert. Über einen Parameter wird geregelt, wie viele Episoden rückwirkend berücksichtigt werden. Hat dieser Parameter den Wert 1, wird exakt eine rückwirkende Episode berücksichtigt. SARSA wird dadurch zu einem MC-Algorithmus (vgl. Lorenz 2020, S. 70). Je nach Ausprägung des Parameters werden die Eigenschaften von MC- und TD-Algorithmen kombiniert. Oft genutzte Aktionen, welche Teil vieler erfolgsversprechender Episoden sind, bremsen allerdings die Exploration aus, da diese Aktionen nur schwer wieder verlassen werden (vgl. Lorenz 2020, S. 71). Letztendlich kommt es also auf die genaue Implementierung des SARSA-Algorithmus an, in welcher Klasse SARSA eingeordnet werden kann.

Verglichen mit Q-Learning kann SARSA Ergebnisse näher an der Realität erzielen, da nicht ausschließlich die Aktionen mit dem erwarteten höchsten Gewinn gewählt werden, sondern insbesondere auch Erfahrungswerte aus vergangenen Episoden mit in die Entscheidungsfindung der Strategie-Funktion mit einbezogen werden. Außerdem findet auch nach dem Training Exploration statt, die jedoch besonders in gefährlichen Umgebungen kritisch zu beurteilen ist.

Wie auch Q-Learning kann SARSA durch Nutzung eines KNN umgesetzt werden. In diesem Fall spricht man von Deep SARSA, bei dem dann die Strategie-Funktion durch ein KNN approximiert wird. Die Funktionsweise ist dabei gleich, wie auch beim Q-Learning.

Bei Expected SARSA handelt es sich um eine weitere, bekannte Sonderform von SARSA. Wie auch SARSA selbst, nutzt Expected SARSA die Strategiefunktion zum Training und gehört daher zu den On-Policy-Algorithmen. Der Unterschied zu SARSA besteht in der Updateregel. Diese verwendet statt dem tatsächlich erreichten Gewinn aus dem Übergang in den Folgezustand den erwarteten Gewinn unter Anwendung der Strategiefunktion. Expected SARSA kann damit gewissermaßen als On-Policy Variante von Q-Learning bezeichnet werden (vgl. van Seijen et al. 2009, S. 2)

2.7.3.4 REINFORCE

Im Jahr 1992 von Williams definiert, arbeitet der klassische REINFORCE-Algorithmus Ergebnisse aus dem Sampling der Gesamtbewertung verschiedener Episoden (vgl. Schmitz 2017, S. 16). Er ist damit klar den MC-Algorithmen zuzuordnen. Prinzipiell ist die Funktionsweise des REINFORCE-Algorithmus ähnlich wie beim Q-Learning. Wesentliche Unterschiede bestehen allerdings zum einen darin, welche Rückschlüsse aus der approximierten Funktion gezogen werden und andererseits in der Art und Weise des Trainings (vgl. Schmitz 2017, 17 f.).

Die approximierte Strategie-Funktion gibt an, mit welcher Wahrscheinlichkeit eine bestimmte Aktion in einem bestimmten Zustand gewählt wird (vgl. Schmitz 2017, S. 16). Entsprechend handelt es sich nicht um eine deterministische, sondern um eine stochastische Strategie-Funktion. Eine Wert-Funktion gibt es bei REINFORCE nicht, daher zählt REINFORCE zu den On-Policy-Algorithmen. Beim Training wird statt einzelner Schritte jeweils eine ganze Episode ausgeführt, weshalb REINFORCE zweifelsfrei den MC-Algorithmen zuzuordnen ist. Zwar könnte das Update auch nach jedem Schritt erfolgen, allerdings steht der zu berücksichtigende, erreichte Gesamtgewinn erst nach Abschluss einer Episode fest (vgl. Schmitz 2017, S. 17). Wie auch bei SARSA kann bei REINFORCE eine Aktionshistorie geführt werden, die dann in der Strategie-Funktion fortlaufend eingebunden wird (vgl. Schmitz 2017, S. 17).

2.8 Grundbegriffe aus dem ÖPNV-Betrieb

In diesem Unterkapitel werden der Reihe nach einige wichtige Grundbegriffe aus dem Betrieb des öffentlichen Personennahverkehrs definiert, welche für das spätere Verständnis der Arbeit von Nöten sind.

2.8.1 Betriebstag

Generell betrachtet geht man davon aus, dass ein Tag 24 Stunden hat. Diese Sichtweise stößt dann an ihre Grenzen, wenn Fahrten abgebildet werden sollen, die über 23:59 Uhr hinaus verkehren. Im Fachjargon spricht man daher von *Betriebstagen*, die auch länger als 24 Stunden sein können. Entsprechend gibt es im betrieblichen Sinne auch Zeiten wie 25:08 Uhr. Gemeint ist damit beispielsweise der folgende Kalendertag um 01:08 Uhr morgens. Der *Betriebsschluss* gibt die Zeit an, zu der die letzte Fahrt endet. Zur Vereinheitlichung wurde der Betriebsschluss

weitestgehend einheitlich auf 03:00 Uhr festgelegt. Ein Betriebstag hat demzufolge 26 Stunden und 59 Minuten.

2.8.2 Linie und Linienvariante

Typischerweise sind ÖPNV-Netze in *Linien* eingeteilt. Eine Linie ist per Definition eine Streckenführung mit einem regelmäßig in beide Richtungen verkehrenden Fahrtenangebot entlang fest definierter Haltestellen (vgl. Reinhardt 2018, S. 456). Einzelne Fahrten können in Teilen von der Linienführung abweichen und beispielsweise nur einen Abschnitt der Linie bedienen. Ein typisches Beispiel hierfür sind Schnellbuslinien, die als Zu- und Abbringer zum nächstgrößeren Verkehrsknotenpunkt dienen. Die einzelnen Fahrwege aller Fahrten einer Linie werden in diversen Planungssystemen als *Linienvariante* hinterlegt. Auch im Fachjargon unter Verkehrsunternehmen hat sich dieser Begriff durchgesetzt. Entsprechend besteht eine Linie grundlegend aus einer Linienvariante für die Hin-Richtung und einer Linienvariante für die Rück-Richtung. Weitere Linienvarianten kommen für jeden abweichenden oder auch verkürzten Fahrweg respektive hinzu.

2.8.3 Umlauf- und Dienstplan

Jede Fahrt innerhalb einer Linie muss mit mindestens einem Fahrzeug besetzt sein. Alle Fahrten, die ein Fahrzeug an einem Betriebstag durchführt, werden durch den sogenannten *Umlauf* zusammengefasst. Anhand eines Umlaufes können alle Fahrten ermittelt werden, die von einem bestimmten Fahrzeug planmäßig durchgeführt werden. Muss ein Umlauf bedingt durch Abweichungen im Betriebsablauf neu besetzt werden, gilt es dabei eine Vielzahl an Parametern zu beachten. Auszugsweise genannt seien hier Wendezeiten an den Endhaltestellen, eventuelle Folgefahrten- und Umläufe eines Fahrzeuges und zugewiesene Stellplätze in Betriebshöfen und Abstellanlagen (vgl. Reinhardt 2018, 496 f.). Während der Umlauf den geplanten Einsatz eines Fahrzeuges vorgibt, regelt ein *Dienst* die Abfolge von Tätigkeiten, die das Fahrpersonal über einen Betriebstag hinweg verteilt durchführt. Analog dazu gilt es einen noch größeren Umfang an Parametern bei der Planung von Diensten und eventuellen Eingriffen im Tagesverlauf zu beachten. Allen voran seien hier das Arbeitszeitgesetzt, die Lenk- und Ruhezeiten und eventuell anzuwendende Regeln aus einem Tarifvertrag genannt. In bestimmten betrieblichen Konstellationen können Dienst- und Umlaufplan gleich sein.

2.8.4 Betriebsstabilität und dispositive Maßnahme

Das Ziel während einem Betriebstag ist es, den Sollfahrplan so gut wie möglich einzuhalten. Man spricht von einem entsprechend *stabilen Betrieb*. Dennoch kommt es im Verlauf des Betriebstages immer wieder zu Abweichungen, die ein Eingreifen notwendig machen. Optimalerweise erfolgt dieser Eingriff durch die Betriebsleitstelle oder eine andere befugte Person. Vielfach angewandt wird der sogenannte *Verspätungsausgleich*, bei dem eine Fahrt gegen Ende vorzeitig beendet und auf die Folgefahrt gewechselt wird. Dadurch entfällt zwar ein Teil einer der ursprünglichen Fahrt, dafür kann die Folgefahrt im Optimalfall pünktlich gestartet werden. Wird hingegen ein Streckenabschnitt kurzfristig über längere Zeit gesperrt, muss eine *Umleitung* angeordnet werden. Optimalerweise erfolgt die Umleitung so, dass ersatzweise Haltestellen bedient werden, die geographisch möglichst nahe an den planmäßigen Haltestellen einer Fahrt liegen, zur Sicherung der Betriebsstabilität können aber einzeln zu bestimmende Halte gezielt entfallen. Ziel der Umleitung ist es, eine größere Verspätung des Fahrzeuges abzufedern, was jedoch nicht immer erreicht werden kann. Solche und vergleichbare Eingriffe in den Betriebsablauf werden als *Dispositionsmaßnahmen* bezeichnet (vgl. Schranil 2013, S. 124). In Extremfällen kann es notwendig sein, mehrere dispositive Maßnahmen nacheinander anzuordnen, um die Betriebsstabilität zu gewährleisten. Zusätzlich zu den Beispielen von Umlauf- und Dienstplanung seien hier das Fahrgastaufkommen und mögliche Anschlüsse an Knotenpunkten genannt. Letztendlich hängt es von einer Vielzahl von Einflussfaktoren ab, welche Maßnahmen in welchem Fall geeignet sind. Im Fokus dieser Arbeit steht die automatische Suche nach Umleitungen für Linienbusse im Störungsfall, andere dispositive Maßnahmen werden daher nicht weiter betrachtet.

2.8.5 Betriebsstörung

Als Arbeitsdefinition für den Begriff der Störung kann zunächst jede Veränderung eines Systems betrachtet werden, welche eine Abweichung vom Plan erforderlich macht oder unmittelbar zur Folge hat. Zu unterscheiden ist dabei in *technische* und *betriebliche Störung*. Während erstere eine bestimmte Funktion außer Kraft setzt, hat sie nicht zwangsläufig einen Einfluss auf den laufenden Betrieb. Letztere ist für diese Arbeit von Relevanz und beschreibt eine Störung des planmäßig geregelten Betriebsablaufs (vgl. Schranil 2013, S. 16). So handelt es sich beispielsweise beim Ausfall der Haltestellenbeleuchtung um eine technische Störung, die jedoch betrieblich keine unmittelbare Auswirkung auf den laufenden Betrieb hat. Störungen auf der Strecke machen hingegen die Umleitung einzelner Fahrten erforderlich und sind damit den betrieblichen Störungen zuzuordnen. Eine Fahrzeugstörung kann sowohl den technischen als auch den betrieblichen Störungen zugeordnet werden, da die zunächst technische Störung

unmittelbare Auswirkungen auf den Betrieb hat. In dieser Arbeit wird stellvertretend für alle betrieblichen Störungen der Begriff *Betriebsstörung* festgelegt.

2.8.6 Innerbetriebliche und öffentliche Information

Der *innerbetriebliche und öffentliche Informationsfluss* ist essenziell zum Aufrechterhalten der Betriebsstabilität. Bei dispositiven Maßnahmen wollen Fahrgäste erwiesenermaßen zeitnah über die Störung selbst und mögliche Alternativen informiert werden (vgl. Brezina et al. 2012, S. 11). Die innerbetriebliche Information stellt darüber hinaus einen geregelten und gleichmäßigen Betriebsablauf sicher (vgl. Scherm et al. 2001, 44 f.). Die Pünktlichkeit und damit einhergehend die Betriebsstabilität ist ähnlich hoch zu bewerten (vgl. Findl et al. 2022, S. 6). Qualifiziertes und erfahrenes Leitstellenpersonal ist in der Lage, Informationen zielgerichtet und korrekt einzubringen und trägt damit einen erheblichen Teil zu einer stabilen Betriebsabwicklung und damit nicht zuletzt auch zur wahrgenommenen Qualität des Verkehrsangebotes auf Seiten der Fahrgäste bei.

2.8.7 Rechnergestütztes Betriebsleitsystem

Das *rechnergestützte Betriebsleitsystem* (RBL) ist die zentrale Steuereinheit zwischen Betriebsleitstelle und Fahrzeug im ÖPNV-Betrieb. Sie umfasst „die Steuerung der Informations- und Kommunikationsmöglichkeit zwischen Fahrzeugen und Leitstelle, die Steuerung des Fahrbetriebs und die Aktualisierung der Fahrgastinformation in den Fahrzeugen und an den Haltestellen“ (Reinhardt 2018, 520 f.). Zwischenzeitlich ist der Begriff *Intermodal Transportation Control System* (ITCS) als Synonym anstelle des RBL getreten. Zur Vereinheitlichung wird in dieser Arbeit jedoch ausschließlich letztere Bezeichnung verwendet. Über das ITCS besteht eine bidirektionale Kommunikationsschnittstelle zwischen Betriebsleitstelle und Fahrzeug beziehungsweise Fahrpersonal. Durch die Ortung im Fahrzeug hat die Betriebsleitstelle stets Kenntnis vom aktuellen Standort eines Fahrzeugs. Ist zudem ein Soll-Fahrplan im System hinterlegt, kann die Betriebsleitstelle auf einen Blick Fahrzeuge sehen, die vom Soll abweichen. Das betrifft sowohl Fahrzeiten als auch Fahrweg. Über eine Mobilfunkschnittstelle kann die Betriebsleitstelle dem Fahrpersonal Anweisungen erteilen und das Fahrpersonal die Betriebsleitstelle außerdem über Störungen im Betriebsablauf informieren. Ebenso können über das ITCS Fahrgäste über die Informationsbildschirme im Fahrzeug und an den Haltestelle über Abweichungen informiert werden (vgl. Reinhardt 2018, 520 f.).

2.8.8 Bordrechner

Der *Bordrechner*, oftmals auch als Bord-Unit bezeichnet, ist das Gegenstück zum ITCS auf Fahrzeugebene. Der Bordrechner hält den Sollfahrplan vor, sodass selbst bei unterbrochener Mobilfunkverbindung ein autonomer Betrieb mit funktionierenden Haltestellenansagen und Fahrastinformation möglich ist und bündelt darüber hinaus alle Informationen, die im Fahrzeug zusammenlaufen (vgl. Reinhardt 2018, S. 521). Hierzu zählen unter anderem Standort, Wegzählerimpuls und Protokolle über Handlungen des Fahrpersonals. Durch den Bordrechner werden all diese Daten schließlich dem ITCS übergeben.

3 Theoretische Modellierung und Konzeption

In diesem Kapitel werden zunächst die theoretischen Grundlagen zur Modellierung des Problems erörtert. Hierzu werden zunächst praxisnahe Beispielszenarien konstruiert, welche im Nachgang als Grundlage zur Evaluation dienen sollen. Im folgenden Unterkapitel wird ermittelt, welcher Umfang an Daten zumindest benötigt wird. Anschließend werden basierend auf der Verfügbarkeit der Daten und den ersten Erkenntnissen aus **Kapitel 2** zwei geeignet erscheinende Algorithmen aus dem Bereich des Reinforcement Learning ausgewählt. Das Kapitel schließt mit der Aufstellung eines theoretischen Modells, welches im folgenden Kapitel prototypisch für Versuchszwecke umgesetzt wird.

3.1 Aufstellung geeigneter Beispielszenarien

Im Rahmen der Arbeit sollen die Ergebnisse verschiedener ML-Algorithmen miteinander verglichen werden. Hierzu ist es zunächst erforderlich, geeignete Szenarien festzulegen, welche einen praxisnahen objektiven und subjektiven Vergleich der Ergebnisse ermöglichen. Die Szenarien sollen zumindest folgende Anforderungen erfüllen:

- Es sollen sowohl städtische als auch ländliche Gebiete erfasst werden
- Stadt- und Regionalbuslinien sollen gleichermaßen inkludiert sein
- Je Szenario soll ein Störfall trainiert werden

Basierend auf diesen Maßregeln werden drei Szenarien aus dem Raum Pforzheim und Umland ausgewählt. Betroffen sind jeweils Regional- und Stadtbuslinien. Bedingt durch die unterschiedliche Haltepolitik der Linien ergibt sich je nach geographischem Betrachtungsraum ein unterschiedlicher Fahrweg mit entsprechend voneinander abweichenden Haltestellenfolgen.

Stadt-Szenario

Betrachtet werden soll der Abschnitt der Kaiser-Friedrich-Straße in der Stadt Pforzheim. Auf diesem Abschnitt verkehrt die Stadtbuslinie 2 jeweils in beide Fahrtrichtungen im 15min-Takt. Aufgrund eines schweren Verkehrsunfalls ist die Kaiser-Friedrich-Straße im östlichen Teil für

mindestens zwei Stunden voll gesperrt, sodass die Buslinien weiträumig umgeleitet werden müssen. Die folgende Abbildung zeigt den betrachteten Raum als Kartenansicht:

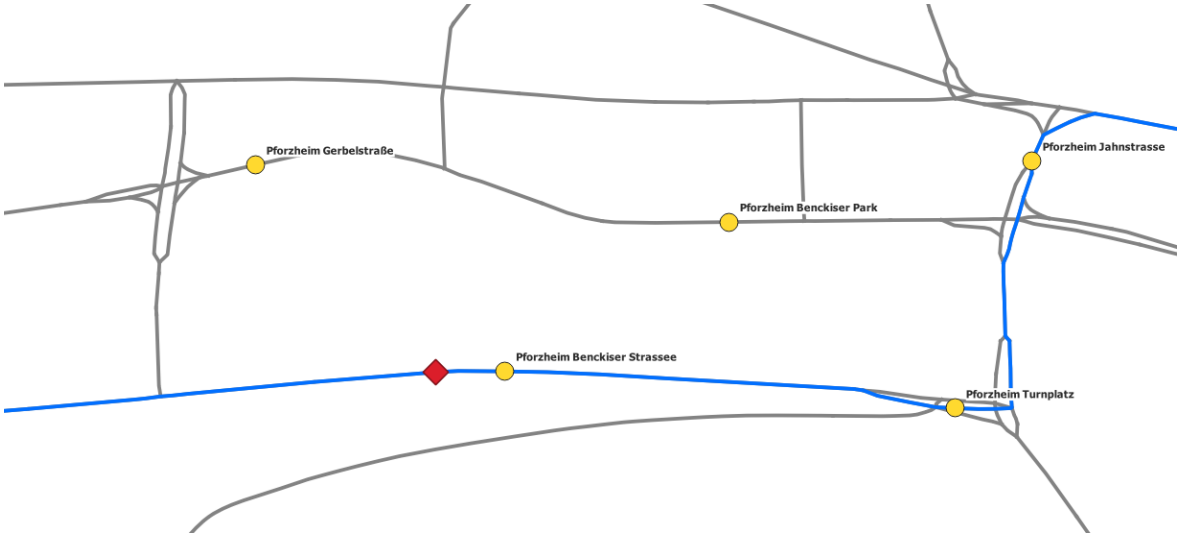


Abbildung 5: Beispielszenario für Linienbusse im Stadtverkehr im Maßstab 1:1837

In blau markiert sind der Fahrweg der Stadtbuslinie. Darüber hinaus sind die Haltestellen im Stadtgebiet durch gelbe Punkte dargestellt. Die rote Raute markiert die Unfallstelle. Die Karte ist in vergrößerter Ansicht im Anhang zu finden.

Land-Szenario

Betrachtet werden soll der Abschnitt zwischen den Gemeinden Langenbrand im Südwesten und Engelsbrand mit dem Ortsteil Salmbach. Auf diesem Abschnitt verkehren die Regionalbuslinien 743 und 744. Die Linie 743 ist eine Schnellbuslinie und starten von Langenbrand aus im Südwesten, die Linie 744 eine normale Regionalbuslinie und verkehrt von der Gemeinde Kapfenhardt in Richtung Engelsbrand im Norden. Die Linie 743 verkehrt im 60min-Takt, die Linie 744 im 30min-Takt jeweils in beide Fahrtrichtungen. Aufgrund eines umgestürzten Baumes ist die Landstraße zwischen den Orten Langenbrand und Salmbach in Richtung Salmbach gesperrt, sodass die Busse umgeleitet werden müssen.

Die folgende Abbildung zeigt den betrachteten Raum als Kartenansicht:

Abbildung 6: Beispielszenario für Linienbusse im Regionalverkehr im Maßstab 1:7361

Zu sehen sind die Regionalbuslinien 743 in orange, die Linie 744 in grün. Ab der Abzweigung an der Haltestelle Brückenäcker verkehren beide Linien auf demselben Fahrweg. Die Haltestellen sind durch gelbe Punkte markiert. Die rote Raute markiert die Stelle, an der die Straße unterbrochen ist. Die Karte ist in vergrößerter Ansicht im Anhang zu finden.

Vorort-Szenario

Das dritte Szenario bildet eine Art Kombination aus dem Land- und dem Startszenario. Es wird die Regionalbuslinie 715 betrachtet, die im 30min-Takt von Pforzheim im Norden kommend durch den Vorort Birkenfeld nach Neuenbürg im Südwesten verkehrt. Der Unterschied zum Land-Szenario besteht darin, dass in dem kleinen Vorort eine wesentlich höhere Haltestellendichte vorliegt und entsprechend auch eine größere Anzahl möglicher Umleitungsfahrwege zur Verfügung steht. Durch eine nicht kommunizierte, kurzfristige Baumaßnahme ist der Streckenabschnitt zwischen den Haltestellen Birkenfeld Kirchplatz und Birkenfeld Gräfenhäuser Straße gesperrt, sodass alle Linienbusse umgeleitet werden müssen.

Die folgende Abbildung zeigt den betrachteten Raum als Kartenansicht:

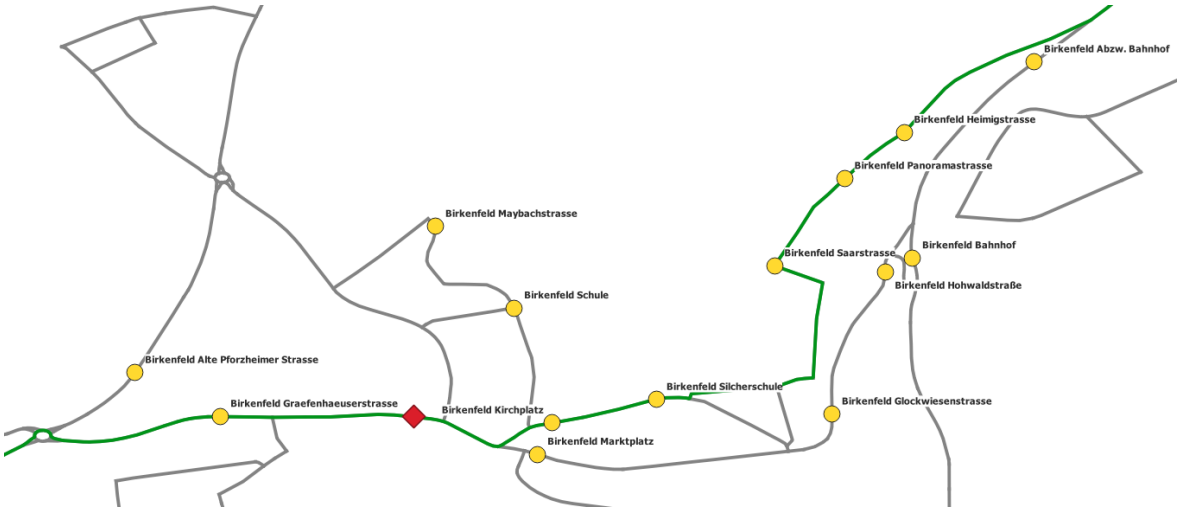


Abbildung 7: Beispielszenario für Linienbusse im Vorortverkehr im Maßstab 1:3675

Der Fahrweg der Regionalbuslinie 715 ist in grün dargestellt. Die Haltestellen sind durch gelbe Punkte markiert. Die rote Raute markiert die Stelle, an der die Straße unterbrochen ist. Die Karte ist in vergrößerter Ansicht im Anhang zu finden.

3.2 Auswahl verfügbarer Eingangsdaten

Anstatt eine vollständige Auflistung aller teils sogar frei unter OpenData-Lizenzen verfügbaren Daten anzustreben, ist es sinngemäßer zunächst nach dem gewünschten Zweck zu kategorisieren und dann dazu passende Daten auszuwählen. In diesem Unterkapitel werden zunächst die Zwecke aufgelistet und passend dazu jeweils verfügbare Datenquellen erläutert. Zudem werden notwendige Transformationen und Anpassungen an den Daten beschrieben.

3.2.1 Betriebliche Daten für Fahrplan und Liniennetz

Grundlage des klassischen Linienverkehrs ist ein Fahrplan (vgl. Reinhardt 2018, S. 456). Dieser bildet darüber hinaus auch den Maßstab für die Arbeit des Leitstellenpersonals.

Ein bekanntes Format zum Austausch von Soll-Fahrplandaten wurde vom Verband Deutscher Verkehrsunternehmen (VDV) mit der VDV-Schrift 452 veröffentlicht. Seitdem gilt VDV 452 als de-facto zum Datenaustausch von Fahrplandaten zwischen verschiedensten Systemen im ÖPNV und ist als Schnittstelle zwischen zahlreichen Systemen auch im produktiven Einsatz.

Ein weiteres Fahrplandatenformat ist das von der EU spezifizierte Format NeTEx. Dabei handelt es sich um XML-Dateien, die Daten gemäß der NeTEx-Spezifizierung enthalten. Bedingt durch den großen Anwendungsraum von NeTEx wurden zwischenzeitlich auch für einige Länder nationale Dialekte, sogenannte Länderprofile, erstellt. Diese Länderprofile ermöglichen wiederrum eine Abbildung von länderspezifischen Eigenschaften. Bei NeTEx handelt es sich damit um ein alternatives Format zu VDV 452, welches aber im Regelfall gesondert aus einem System exportiert beziehungsweise in ein weiteres System importiert werden muss.

Seit der Änderung des Personenbeförderungsgesetz sind unter Anderem auch Verkehrsunternehmen dazu verpflichtet, Fahrplan- und Netzdaten öffentlich zur Verfügung zu stellen (vgl. Bundesministerium für Verkehr, §3a). In den meisten Fällen werden die Daten im von Google definierten *General Transfer Feed Specification* (GTFS) Format bereitgestellt. Dabei handelt es sich um CSV-ähnliche Textdateien mit Informationen zu Fahrzeiten, Fahrten, Linien und Haltestellen, wie wiederrum in einer ZIP-Datei verteilt werden (vgl. Google 2022). Für betriebliche Anwendungen, zu denen ein ITCS nebst zugehöriger Bordrechner gehört, ist GTFS nur eingeschränkt geeignet, da wichtige Betriebsdaten wie Umläufe in der Spezifikation nicht vorgesehen sind.

Um Zwischenschritte für möglicherweise nötige Datenkonvertierungen zu sparen, wird für den Einsatz in einem produktiven ITCS das Format VDV 452 empfohlen. Für die Verwendung im Prototyp sind GTFS-Daten aber allemal zu gebrauchen. Aufgrund der einfachen Zugänglichkeit und des gut verarbeitbaren Datenformates werden GTFS-Daten als Grundlage für den Prototyp und dessen Evaluation genutzt.

3.2.2 Kartendaten und Routing

An dieser Stelle muss zunächst der spätere Einsatzzweck differenziert werden. Der Fokus liegt auf der Umleitungssuche für Linienbusse im Störungsfall. Hierbei ist die gegebenenfalls zu wählende Umleitungsroute von essenzieller Bedeutung, es gilt eine möglichst optimale Route im Sinne einer Stabilisierung des Betriebes im Störungsfall zu wählen. An dieser Stelle bietet sich die Verwendung offener Geodaten an. Diese enthalten nicht nur geographische Informationen, die später wieder für die Anzeige in einer Karte genutzt werden können, sondern bieten darüber hinaus auch die Möglichkeit, mit gängigen Frameworks Informationen und Daten zur weiteren Verarbeitung extrahieren zu können.

Der wohl bekannteste Anbieter für offene Geodaten im deutschsprachigen Raum dürfte OpenStreetMap (OSM) sein. Die Nutzungsbedingungen lassen fast alle Verwendungsarten, darunter auch kommerzielle, zu. Einzige Voraussetzung ist, dass die OSM-Mitwirkenden als Quelle genannt werden. Der Anbieter Geofabrik bietet darüber hinaus täglich aktualisierte,

vorab zugeschnittene Datensätze für einzelne Städte, Landkreise oder Bundesländer zum Download als OSM-Datei an. Eine OSM-Datei enthält zunächst jedoch alle auf einem Kartenausschnitt enthaltenen Daten in einer XML-Struktur. Bedingt durch den damit einhergehenden Overhead können selbst Datensätze von kleinen Kartenausschnitten beachtliche Mengen an Daten enthalten, die dann weiterverarbeitet werden müssten. Die Daten in einem OSM-Datensatz sind generell unterteilt in Punkte (Nodes), Wege (Ways) und sogenannte Relationen (Relations). Eine Relation fasst bestehende Punkte und Wege zu neuen Informationen zusammen und baut damit hierarchisch auf dem OSM-Datenmodell auf.

Für das Routing werden nur Straßendaten benötigt, Daten über Häuser und POIs sind nicht notwendig. Eine nennenswerte Alternative zu den fertig verfügbaren Datensätzen der Geofabrik bietet die sogenannte Overpass-API mit dem Webinterface Overpass-Turbo. Mittels einer skriptähnlichen Beschreibungssprache können gezielt Daten selektiert und in alle gängigen Formate exportiert werden. Neben dem Eingabefeld für die Abfrage enthält Overpass-Turbo außerdem eine Vorschau, in der nach Ausführung einer Abfrage die selektierten Daten hervorgehoben werden. Somit ist auf einfachem Weg möglich zu prüfen, ob die Abfrage zum gewünschten Ergebnis führt. Die von Straßendaten lässt sich über das Attribut *highway* eingrenzen, in dem die jeweilige Straßenkategorie abgelegt ist.

Die folgende Tabelle gibt eine Übersicht über die in OSM vergebenen Straßenkategorien:

Attribut (highway=)	**Straßenkategorie**
motorway / motorway_link	Autobahn
trunk / trunk_link	Schnellstraße
primary / primary_link	Bundesstraße
secondary / secondary_link	Kreisstraße / Landstraße
tertiary / tertiary_link	Vorfahrtstraße (Innerorts)
residential	Innerortsstraße
living_street	Spielstraße
pedestrian	Fußgängerzone

Tabelle 1: Straßenkategorien und deren Schlüssel in OSM-Daten (Quelle: OSM-Highway 2022)

Bei dieser Tabelle handelt sich nicht um eine vollständige Auflistung aller verfügbaren Werte für das Attribut highway, sondern ausschließlich um jene Straßenkategorien, die mit Linienbussen befahren werden können. Feldwege, Behelfs- und Privatstraßen sind bewusst nicht berücksichtigt. Spielstraßen und Fußgängerzonen sind deshalb mit inbegriffen, weil diese in einigen Fällen durchaus für den Linienverkehr freigegeben sind. Beispiele sind in nahezu jeder größeren Stadt zu finden. Für Fußgängerzonen wird in OSM-Daten bei einer entsprechenden Freigabe für den Linienverkehr das Attribut *PSV* für *Public Services Vehicle* gesetzt (vgl. OSM-PSV 2022).

Die folgende Abfrage liefert zunächst alle Straßendaten aus dem aktuell angezeigten Bereich, der sogenannten *BoundingBox* in Overpass-Turbo:

```
[out:xml][timeout:120];

(
  way["highway"="motorway"]({{bbox}});
  way["highway"="motorway_link"]({{bbox}});
  way["highway"="trunk"]({{bbox}});
  way["highway"="trunk_link"]({{bbox}});
  way["highway"="primary"]({{bbox}});
  way["highway"="primary_link"]({{bbox}});
  way["highway"="secondary"]({{bbox}});
  way["highway"="secondary_link"]({{bbox}});
  way["highway"="tertiary"]({{bbox}});
  way["highway"="tertiary_link"]({{bbox}});
);


out meta;
>;
out meta qt;
```

Abbildung 8: Overpass-Abfrage für Straßendaten im Primärnetz

Enthalten sind dabei alle Straßen mit einer Kategorie zwischen Autobahn und Innerorts-Vorfahrtstraße. Im weiteren Verlauf wird dieses Straßennetz als *Primärnetz* bezeichnet. Bestätigt wird diese Selektion durch die Vorschau in Overpass-Turbo, wie in der folgenden Abbildung zu sehen ist:

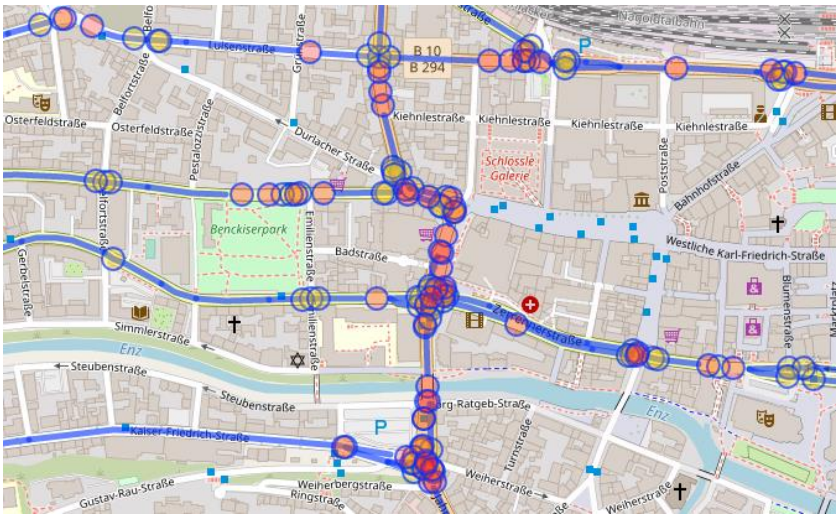


Abbildung 9: Overpass-Vorschau für Straßendatendaten im Primärnetz

In der Auswahl fehlen jedoch wichtige, ebenfalls vom Linienverkehr genutzte Innerortsstraßen und die für den Linienverkehr freigegebene Fußgängerzone. Während letztere durch Einbeziehung des OSM-Attributes PSV mit selektiert werden könnte, wird diese Eingrenzung für Innerortsstraßen schwieriger, da diese nicht weiter kategorisiert oder attribuiert sind. Werden generell Innerortsstraßen mit in die Abfrage einbezogen, sind automatisch alle Innerortsstraßen enthalten, selbst wenn diese mit einem Linienbus gar nicht befahren werden

können. Auch im Routing wäre eine Unterscheidung bestenfalls anhand der verfügbaren Straßenbreite möglich, was jedoch eine adäquate Versorgung der OSM-Daten mit diesen notwendigen Informationen erforderlich macht. Bei offenen, gemeinschaftlich gepflegten Daten aus der OpenStreetMap-Gemeinschaft ist das generell kritisch zu sehen (vgl. Josi 2020, S. 10), daher eignet sich der direkte Export des gesamten Straßennetzes nicht für das zuverlässige Routing eines Linienbusses.

Ziel muss es sein, das Primärnetz generell zu selektieren und alle darunter liegenden Straßenkategorien nur dann mit einzubeziehen, wenn diese bekanntermaßen für Linienbusse befahrbar sind. Diese zusätzlichen Straßen werden fortan unter dem Begriff des *Sekundärnetzes* zusammengefasst. Hierzu bietet sich ein zweistufiges Verfahren an.

In der ersten Stufe werden zunächst Relationen als Selektor gewählt, welche eine Buslinie in den OSM-Daten abbilden und dadurch auf die entsprechenden Wege verweist. Dann werden alle Wege selektiert, die zu diesem festgelegten Selektor passen. Hierdurch sind alle Straßen unabhängig von ihrer Kategorie enthalten, die durch die hinterlegten Linienwege folglich auch mit einem Linienbus befahren werden können. Was jedoch nach wie vor fehlt, sind Innerorts- und Spielstraßen und Fußgängerzonen, welche zwar mit Linienbussen befahrbar sind, jedoch planmäßig nicht befahren werden.

Hier kommt die nächste Stufe ins Spiel, bei der die von den Bussen aufgezeichneten GPS-Trajektorien mit Hilfe eines sogenannten MapMatching-Algorithmus einer entsprechenden Straße im Sekundärnetz zugeordnet werden. Zum MapMatching bietet sich beispielsweise der ST-Algorithmus an, da dieser auch für das Matching von GPS-Trajektorien mit geringerer Samplingrate eine hinreichende Genauigkeit liefert (vgl. Simeonov 2017, S. 21). Dass die GPS-Trajektorien der Busse immer erst am Ende eines Betriebstages und nicht in Echtzeit dem Matching zugeführt werden, trägt der Genauigkeit bei (Simeonov 2017, S. 22). Da bei diesen GPS-Trajektorien auch Leer- und Betriebsfahrten enthalten sind, ergeben sich auf diesem Weg neue Fahrwege auch außerhalb der planmäßigen Linienwege. Anschließend wird die Vereinigungsmenge aus dem bestehenden Primärnetz und dem Sekundärnetz gebildet. Hierdurch erhält man ein vollständiges Straßennetz im Geodatenformat, welche dann von Geoinformationssystemen oder zum Routing weiterverwendet werden kann.

Die folgende Abbildung fasst schematisch die Ableitung eines für das Routing von Linienbussen geeigneten Straßennetzes aus OSM-Daten zusammen:

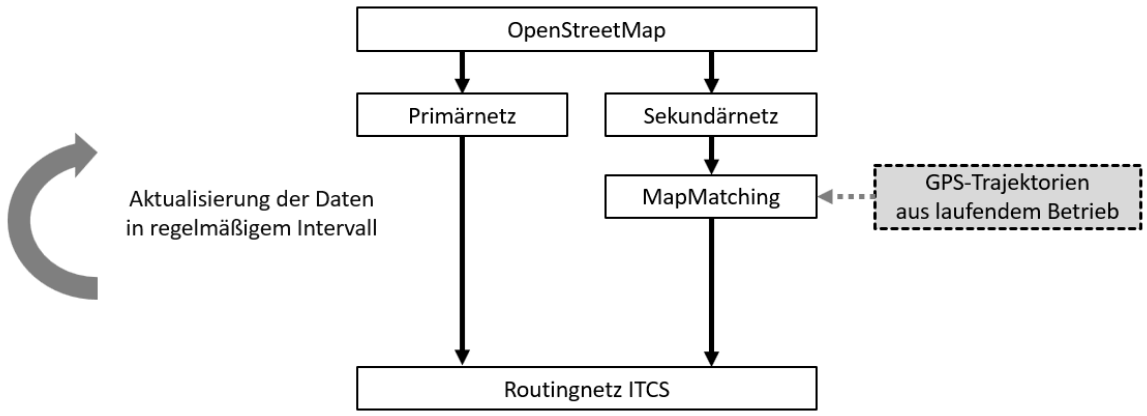


Abbildung 10: Ableitung eines für Linienbusse geeigneten Netzes aus OSM-Daten

Die Aktualisierung des Straßennetzes sollte in regelmäßigen Abständen durchgeführt werden, um stets ein aktuelles Abbild der tatsächlich verfügbaren Verkehrsinfrastruktur vor Ort zur Verfügung zu haben. In einem Produktivsystem könnte die Aktualisierung beispielsweise nachts zur Betriebsruhe durch einen automatischen Hintergrundprozess durchgeführt werden.

3.2.3 Regelfahrwege und Umleitungsstrecken

Um feststellen zu können, ob eine Fahrt von einer Störung betroffen ist, muss bekannt sein, ob der Ort der Störung von dem Fahrzeug im Verlauf der Fahrt überhaupt erreicht wird. Sowohl in VDV 452 als auch in GTFS kann einer Fahrt ein planmäßiger, geographischer Fahrweg zugewiesen werden. Im Regelfall handelt es sich dabei um eine Sequenz von GPS-Koordinaten, die den Fahrweg repräsentiert. Anhand dieser GPS-Koordinaten kann mit Hilfe der Haversine-Funktion berechnet werden, ob der Fahrweg eine Störung tangiert oder nicht.

Neben den vorliegenden Regelfahrwegen können Umleitungsfahrwege aus den GPS-Daten abgeleitet werden, welche die Fahrzeuge kontinuierlich aufzeichnen. Das Verfahren wurde zum Erzeugen des Sekundärnetzes bereits im vorhergehenden Unterkapitel erläutert. Der Unterschied zwischen der Erzeugung des Sekundärnetzes und dem Ableiten eines Umleitungsfahrweges besteht darin, dass letzterer immer bezogen auf eine Linie oder Linienvariante ist. Wird während der Fahrt oder im Nachgang durch Auswertung der aufgezeichneten GPS-Daten erkannt, dass das Fahrzeug den Linienweg verlassen hat, kann diese Strecke im Kontext der aktuell angemeldeten Linie als Umleitungsfahrweg gespeichert werden. Hierbei sollten jeweils alle GPS-Koordinaten ab Verlassen bis zum Wiedererreichen des regulären Fahrweges als Umleitungsfahrweg gespeichert werden.

Da Rechen- und Vergleichsoperationen mit GPS-Koordinaten sehr rechenaufwendig sind, bietet es sich im Rahmen der Weiterverarbeitung an, die GPS-Koordinaten in die entsprechenden OSM-Knotenpunkte umzurechnen. Dabei wird jedem GPS-Punkt ein Knotenpunkt aus den OSM-Knotenpunkten zugeordnet. Ergebnis ist eine Sequenz von Ganzzahlen, welche den IDs der OSM-Knotenpunkte in der geographischen Reihenfolge entspricht. Wird diese Konvertierung einmalig für Regel- und Umleitungsfahrwege durchgeführt, können im Anschluss neue Kombinationen aus Fahrwegen ressourcenschonender errechnet werden. Zeitgleich kann ein gefundener, optimaler Fahrweg jederzeit wieder in GPS-Koordinaten zurück konvertiert werden, um die GPS-Koordinaten beispielsweise zur graphischen Anzeige des Fahrweges in einer Karte zu nutzen. Die Kombination von Regel- und Umleitungsfahrweg aus OSM-Knotensequenzen wird am nachfolgenden Beispiel aus dem Stadt-Szenario verdeutlicht:

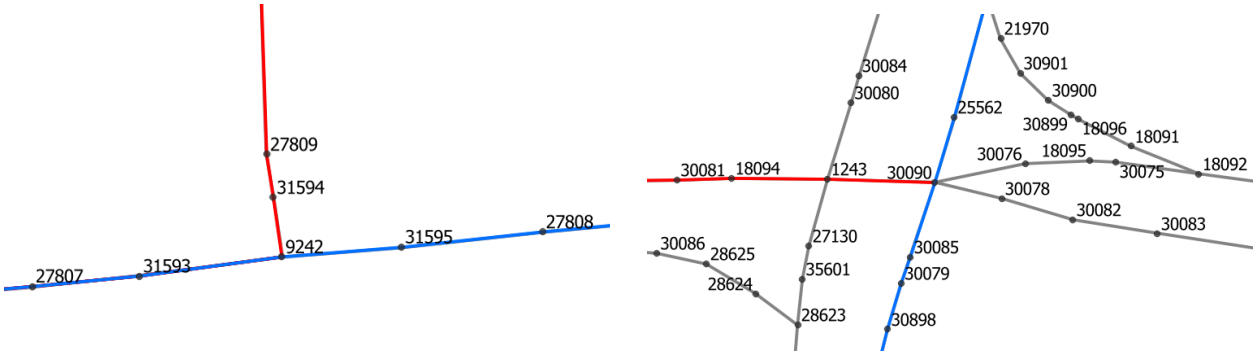


Abbildung 11: Abzweigung und Einmündung eines Umleitungsfahrweges

Auf der linken Seite ist die Einmündung zu sehen, an der das Fahrzeug den blau markierten Regelfahrweg verlässt und auf den rot markierten Umleitungsfahrweg übergeht. Auf der rechten Seite ist die Kreuzung zu sehen, an der das Fahrzeug den roten Umleitungsfahrweg wieder verlässt und auf den blauen Linienweg zurückkehrt. Die Kombination des Regel- und Umleitungsfahrweges in drei Schritten wird in der folgenden Abbildung dargestellt:

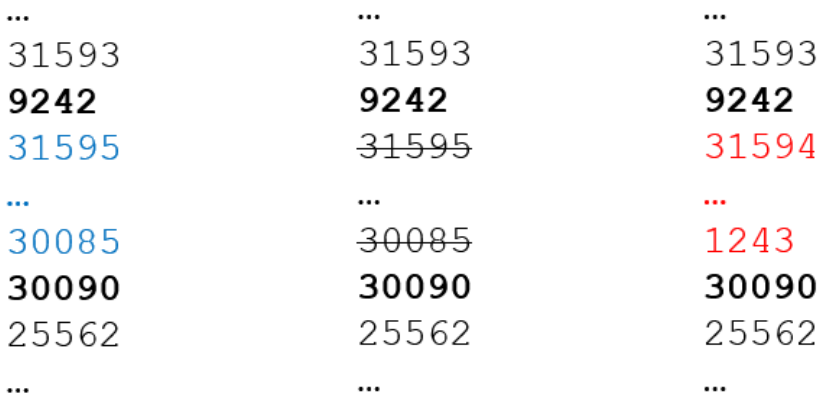


Abbildung 12: Dreistufige Kombination von Regel- und Umleitungsfahrweg

Ganz links zu sehen ist die ursprüngliche Knotenpunktsequenz des Regelfahrweges im ersten Schritt. In fett gedruckt sind die Knotenpunkte, an denen der Umleitungsfahrweg beginnt und endet. Im zweiten Schritt werden alle Knotenpunkte zwischen Beginn und Ende des Umleitungsfahrweges entfernt, um im dritten Schritt durch die entsprechenden Knotenpunkte des Umleitungsfahrweges ersetzt zu werden. Als Ergebnis erhält man die Knotenpunktsequenz, welche das Fahrzeug unter Anwendung der Umleitung zurücklegen wird.

3.2.4 Störungsmeldungen und Daten zur Verkehrssituation

Elementar für das automatische Einleiten einer Dispositionsmaßnahme ist für das ITCS die Kenntnis über eine Störung. Anders als bei den Betriebs- und Netzdaten stellt sich die Suche nach geeigneten Datenquellen hier komplexer dar.

Zunächst setzt jede Störungsinformation ein Ereignis mit entsprechender Diagnose voraus (vgl. Schranil 2013, 69 f.). Durch die Kenntnis über das zu Grunde liegende Ereignis können dann die Relevanz eingeschätzt und Prognosen zur Dauer der Störung getroffen werden. Die voraussichtliche Störungsdauer ist spätestens dann wichtig, wenn entschieden werden muss, ob eine einzelne Fahrt umgeleitet werden muss, oder die Störung aller Voraussicht nach beim Eintreffen des Fahrzeuges schon behoben sein wird.

Im Sinne der Digitalisierung von Kommunen kommt damit schnell der Begriff der *Smart City* auf. In diesem Zusammenhang wird regelmäßig auch auf den Verkehrssektor verwiesen (vgl. Soike und Libbe 2018, S. 11; Herzner und Schmidpeter 2022, S. 70). So sind führen Meier und Portmann Lösungsansätze an, die „in Echtzeit flexibel die aktuelle Situation“ (Meier und Portmann 2016, S. 261) am Beispiel von Staus berücksichtigen können. Schaaf und Wilke (2015) konstruieren ein noch umfangreicheres Beispiel, in dem als Folge eines Verkehrsunfalls beispielsweise Rettungskräfte automatisch alarmiert und zur Einsatzstelle geschickt werden und Linienbusse, die durch den Einsatz von Echtzeitdaten selbstständig planmäßige Anschlüsse über die entstehende Verspätung informieren können (vgl. Schaaf und Wilke 2015, S. 563). Alle regelmäßigen ÖV-Nutzenden wissen aus Erfahrung, dass solche Showcase-Beispiele von der Realität der meisten Verkehrsbetriebe und deren Umfeld weit divergieren. Smart City-Lösungen setzen erwartungsgemäß die Aufnahme und Verteilung einer entsprechenden Meldung über ein Ereignis mit einem sichergestellten Wahrheitsgrad voraus. Falschmeldungen könnten insbesondere in Bezug auf das Beispiel mit den Rettungskräften drastische Folgen haben. Ferner ist es für ein flächendeckend funktionierendes System eine hinreichend große Ausstattung mit aktueller Sensorik erforderlich, um notwendige Datenmengen sammeln und in adäquater Latenzzeit verarbeiten zu können (vgl. Meier und Portmann 2016, S. 266). Inwiefern Daten aus vorhandenen Smart City-Plattformen für den

Einsatz als Störungsmelder in einem ITCS dienen kann, hängt also maßgeblich vom Umsetzungsgrad entsprechender Systeme ab. Dieser wurde 2018 vom Deutschen Institut für Urbanistik für die 200 Städte mit dem höchsten Bevölkerungsgrad in Deutschland umfangreich erhoben. Die folgende Grafik zeigt den Umsetzungsgrad verschiedener Smart City-Projekte in diesen Städten:

Es ist zu erkennen, dass gut zwei Drittel aller betrachteten Städte mit Stand 2018 keine Umsetzung von Smart City-Projekten forciert haben. Vom verbleibenden Drittel sind wiederrum nur die Hälfte von umfangreicherer Natur und spiegeln konkrete Ansätze zur intermodalen Einbindung in kommunale Dienste vor (vgl. Soike und Libbe 2018, S. 7). Die Stadt Pforzheim belegt dabei immerhin den 63. Platz (vgl. Statistisches Bundesamt 2020; zitiert nach de.statista.com) und bietet nach aktuellem Stand sogar öffentliche Smart City-Projekte für die Bürgerschaft an. Bezug zu Verkehrsstörungen hat jedoch nach aktuellem Stand (2022) keines dieser Projekte. Zusammenfassend lässt sich festhalten, dass die Mehrheit der deutschen Städte bislang keine Bestrebungen zum Ausbau von Smart City-Projekten zeigt und sich die aktiven Städte in der Hauptsache auf Großstädte reduzieren lassen (vgl. Soike und Libbe 2018, S. 8). Beachtet man zusätzlich noch die fehlende, gesellschaftlich durchgesetzte Definition des

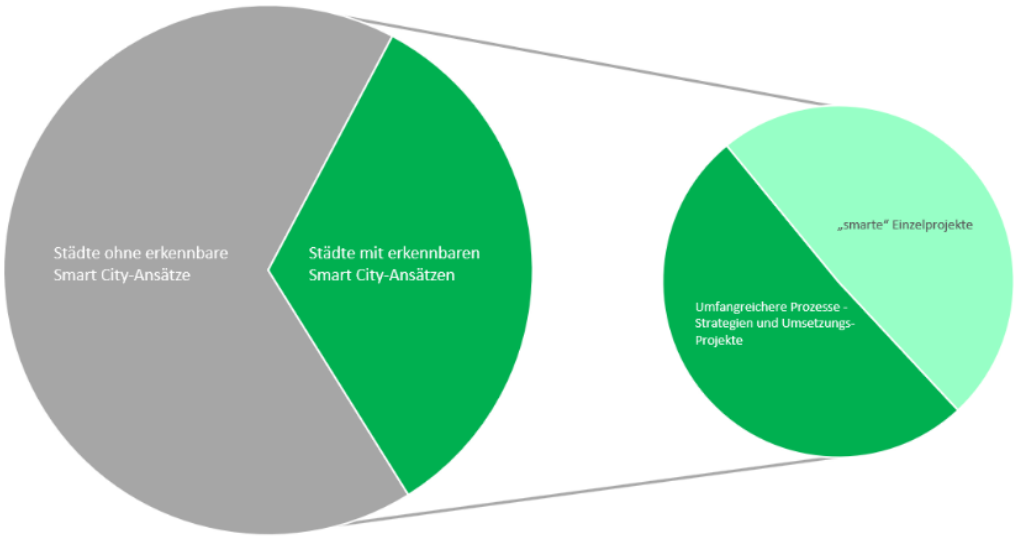


Abbildung 13: Umsetzungsgrad von Smart City Projekten in den 200 größten Städten (Nach: Soike und Libbe 2018, S. 6)

Begriffes der Smart City (vgl. Herzner und Schmidpeter 2022, S. 67), erschwert dies die Beurteilung des Umsetzungsgrades in deutschen Städten zusätzlich. Smart City-Daten scheiden daher als alleinige Quelle für Störungsinformationen in einem ITCS aus und können bestenfalls als zusätzliche Datenquelle angesehen werden. Die Konzentration auf Großstädte verhindern außerdem eine flächendeckende Nutzung in einem ITCS, welches auch Regionalbusse überwachen soll. Ob in den verteilten Daten Informationen zum zu Grunde liegenden Ereignis

oder alternativ direkt zur voraussichtlichen Störungsdauer enthalten sind, hängt von der konkreten Implementierung einer Smart City-Plattform ab. Es bleibt offen, inwieweit sich weitere Smart City-Projekte in naher Zukunft etablieren und welchen Informationsgehalt diese mitbringen werden.

Ohne die direkte Initiative einer Kommune kommt hingegen das deduktive Monitoring aus. Nach diesem Prinzip ermittelt beispielswese der Anbieter Google Daten zur aktuellen Verkehrssituation. Hierbei werden anonymisierte Standortdaten von Android-Mobilgeräten ausgewertet und auf das Verkehrsaufkommen umgelegt (vgl. Exner 2012, S. 25). Google stellt diese Daten gegen Entgelt auch über Programmschnittstellen (API) zur Verfügung. Gerade kommerzielle Anbieter mit deduktivem Monitoring in ihrem Portfolio stehen insbesondere wegen datenschutzrechtlicher Belange immer wieder in der Kritik (vgl. Exner 2012, S. 26). Erfahrungsgemäß ist die Qualität der Verkehrsdaten generell zwar zur Navigation im Individualverkehr zu gebrauchen, allerdings sind in der Praxis oft sehr hohe Latenzzeiten beim Auflösen einer Verkehrsblockade zu beobachten. So wird beispielsweise ein Stau, der bereits seit 15min abgeklungen ist, noch immer als solcher in der Karte ausgewiesen und in der Navigation berücksichtigt. Für die Anwendung im Linienverkehr würde das bedeuten, dass je nach Taktdichte zahlreiche Fahrten ihren planmäßigen Fahrweg verändern würden, obwohl dies gar nicht notwendig wäre.

Eine Alternative zum Abgriff von Daten Dritter aus Smart City-Systemen oder Datenbrokern bieten viele ITCS bereits in Form sogenannter *kodierter Meldungen* an (vgl. DELFI 2020, S. 122). Unter diesem zunächst undurchsichtigen Begriff verstehen sich vordefinierte Textnachrichten, welche das Fahrpersonal an die Leitstelle zum Austausch von Informationen senden kann. Der Inhalt dieser Nachrichten kann in jedem System von der Administration festgelegt werden, sodass nahezu alle Formen von Nachrichten ausgetauscht werden können.

Gängige Beispiele sind in der folgenden Abbildung zu sehen:

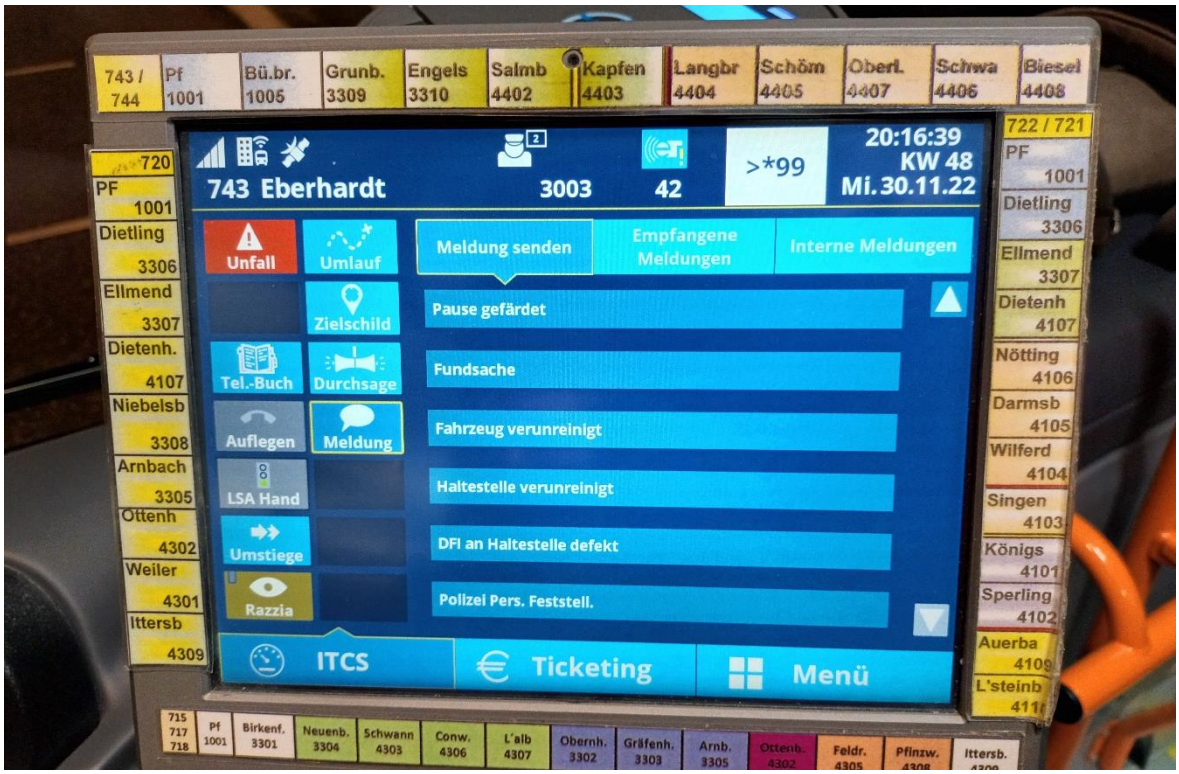


Abbildung 14: Vordefinierte Meldungen auf einem Bordrechner in einem Linienbus

Anstatt in solch einem Fall eine Sprechfunk- oder Telefonverbindung zur Leitstelle herzustellen, genügt es beim Eintreten eines entsprechenden Ereignisses eine entsprechende Meldung abzusetzen, um die Leitstelle über einen bestimmten Zustand zu informieren. Dem Leitstellenpersonal wird die Meldung dann zusammen mit Informationen zur Linie, Verspätungslage und Standort und anderen betrieblich relevanten Daten angezeigt. Diese Funktion ist für Meldungen vom Fahrpersonal an die Leitstelle initiiert, die „keine zeitsynchrone Bearbeitung“ erfordern und um den Sprechfunk zu entlasten (vgl. VDV-Schrift 730, S. 56). Es stellt sich also die Frage, wie diese Informationen aus erster Hand gewinnbringend als Datenquelle für eine automatische Anordnung von Umleitungen eingesetzt werden können.

Es sind zumindest folgende Gesichtspunkte in Betracht zu ziehen:

- **Richtigkeit:** Meldungen können auch versehentlich abgesetzt werden. Gründe hierfür können von einer ein Versehen, eine fehlerhafte Unterweisung oder aber auch eine Fehlbedingung aufgrund von Sprachbarrieren des Fahrpersonals sein. In jedem Fall muss also die Richtigkeit der Meldung sichergestellt werden, um darauf basierend weitere Entscheidungen zu treffen.
- **Genauigkeit** Es müssen so viele Ereignisse wie möglich als Meldung hinterlegt werden, um möglichst viele Details zur Störung herleiten zu können. Zeitgleich darf das Fahrpersonal aber auch nicht mit Informationen überfrachtet werden, da andernfalls die Richtigkeit und Qualität der Meldungen leiden könnte. Für die richtige Bedienung sind Einfachheit, Verständlichkeit und Intuitivität essenziell.
- **Dauer:** Die Dauer der Störung oder zumindest eine Prognose der Dauer ist nötig, um festzulegen, welche Fahrten von der Störung betroffen sein werden. Ebenso ist es nötig, ein eventuell früheres Störungsende erkennen zu können, um den Regelbetrieb gegebenenfalls wieder früher herstellen zu können. Auf die Notwendigkeit zusätzlicher Handlungen des Fahrpersonals sollte hierbei verzichtet werden, da die Erkennung eines früheren Störungsendes sonst im Zweifel bei Unterlassen dieser Handlung nicht möglich ist.

Eine mögliche Umsetzungskonzeption soll in Bezug zum Stadt-Szenario aus Kapitel 3.1 wir nachfolgend vorgestellt.

Die erste Fahrt, welche von Westen kommend in den gesperrten Abschnitt einfährt, kommt wenige Fahrzeuge hinter der Unfallstelle zum Stehen. Nachdem das Fahrpersonal die Situation überblickt hat, setzt es eine entsprechende Meldung vom Typ „Polizeieinsatz“ ab. Die Folgefahrt nähert sich im zeitlichen Abstand von 10min von hinten an.

Nach dem Eintreffen der Meldung geht das ITCS schrittweise folgendermaßen vor:

1. Zunächst werden – falls verfügbar – weitere Quellen hinsichtlich einer ähnlichen Störungsmeldung bezogen auf Ort und Zeit durchsucht. Gibt es an dieser Stelle bereits signifikante Übereinstimmungen, ermittelt das ITCS die voraussichtlich betroffenen Fahrten und ordnet für diese eine geeignete Umleitung an.
2. Sind hingegen keine weiteren Datenquellen verfügbar, überwacht das ITCS zunächst die Position des Fahrzeuges, von dem die Meldung abgesetzt wurde. Da von einer Streckensperrung ausgegangen wird, darf sich die Position des Fahrzeuges im Überwachungszeitraum von beispielsweise einer Minute nicht mehr signifikant verändern. Ist diese Bedingung erfüllt, ermittelt das ITCS die voraussichtlich betroffenen Fahrten und ordnet für diese eine geeignete Umleitung an.

Die folgende Abbildung zeigt die Validierung einer Meldung als Flussdiagramm:

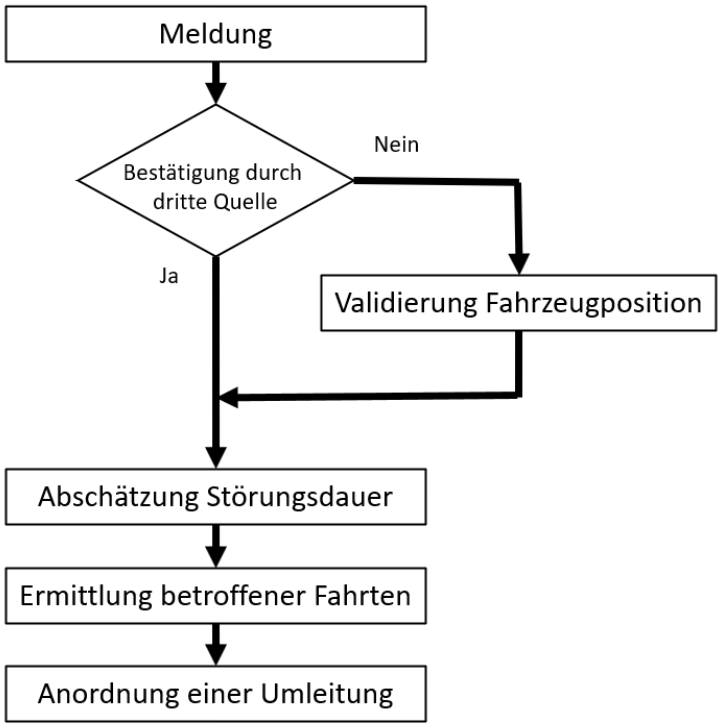


Abbildung 15: Validierung einer systemseitigen Meldung als Flussdiagramm

Nachdem entweder die statistisch betrachtet zu erwartende Störungsdauer erreicht ist oder alternativ eine Positionsänderung des ersten Fahrzeuges zu verzeichnen ist, geht das ITCS folgendermaßen vor:

1. Das erste betroffene Fahrzeug wird erneut hinsichtlich der GPS-Position überwacht. Ist hier noch immer keine signifikante Positionsänderung zu verzeichnen, wird auch für die folgenden Fahrten eine Umleitung angeordnet.
2. Kann hingegen eine Fortbewegung des Fahrzeuges festgestellt werden, wird die Umleitung für Fahrzeuge, die sich bereits auf der Umleitungsstrecke befinden, beibehalten und für alle anderen Fahrten zurückgenommen. Eine gesonderte Handlung des Fahrpersonals ist damit nicht erforderlich.

Bezüglich der Genauigkeit bietet es sich an, Störungsursachen aus der Vergangenheit zu analysieren und in Klassen einzuordnen. Oft kristallisieren sich dabei einige wenige Störungsursachen heraus, welche dann dem Fahrpersonal gezielt zur Auswahl angeboten werden können (vgl. Schranil 2013, S. 188). Die Störungsdauer lässt sich basierend auf statistischen Verfahren ebenfalls aus Störungsdaten aus der Vergangenheit abschätzen (vgl. Schranil 2013, S. 193). Einzige Voraussetzung hierzu ist, dass vergangene Störungen in ausreichendem Maß dokumentiert wurden. Für die Richtigkeit, Genauigkeit und Abschätzung

der Dauer ist damit systemseitig gesorgt. Wurde eine Meldung aus dem ITCS selbst erfolgreich validiert, kann diese Störungsmeldung auch über Datendrehscheiben an weitere ITCS von anderen Verkehrsunternehmen kommuniziert werden. Hierzu eignet sich der SIXI-SX-Dienst. Mit Hilfe dieses Dienstes lassen sich innerhalb einer Störungsmeldung auch direkt Haltestellen oder Linien referenzieren (vgl. VDV-Schrift 736-2, S. 34), sodass die Informationen in einer Störungsmeldung übertragbar zwischen verschiedenen ITCS sind.

Ohne die Verfügbarkeit weiterer Datenquellen mit Detailinformationen, beispielsweise zu gesperrten Fahrtrichtungen, muss entweder grundsätzlich von einer Vollsperrung ausgegangen oder eine Fahrt pro Richtung eine entsprechende Meldung absetzen. Andernfalls würden Umleitungen für Fahrten angeordnet, die betrieblich betrachtet gar nicht umgeleitet werden müssen. Als weitere Datenquelle kann beispielsweise auch ein manuelles Eingreifen des Leitstellenpersonals angesehen werden. An dieser Stelle muss im Einzelfall ein betrieblich vertretbarer Trade-Off durch entsprechende Systemkonfiguration gefunden werden.

3.3 Auswahl geeigneter RL-Algorithmen

In Kapitel 2.7.3 wurden wichtige Algorithmen aus dem Bereich des RL vorgestellt. Wie die folgende Abbildung zeigt, handelt es sich jedoch nur um einen kleinen Teil der erforschten oder bekannten RL-Algorithmen:

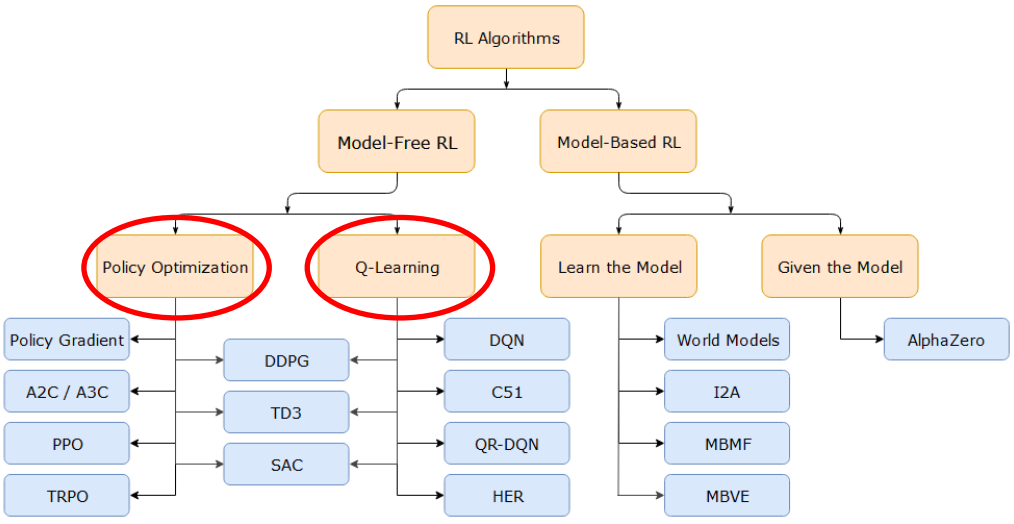


Abbildung 16: Bekannte RL-Algorithmen in der Übersicht (Quelle: OpenAI 2018)

Alle bisher vorgestellten Algorithmen sind der Klasse der modellfreien Algorithmen zuzuordnen. Problematisch ist grundsätzlich bei modellfreien Algorithmen, dass sie

ausschließlich zahlenbasiert auf der Gewinnfunktion arbeiten. Modellbasierte Algorithmen sind hingegen in der Lage, auch eine Vielzahl im Modell berücksichtigter Informationen aus der Umwelt zu berücksichtigen. Einbußen sind dafür insbesondere bei der Laufzeitkomplexität modellbasierter Algorithmen zu verzeichnen (vgl. Pong et al. 2018, 1 f.). Modellbasierte Algorithmen bringen außerdem ein zusätzliches Maß an Komplexität mit, da die Umwelt zuerst in einem Modell zusammengefasst werden muss. Fragwürdig ist an dieser Stelle, inwieweit ein Gesamtvorteil für die ursprüngliche Problemstellung, die Umleitungssuche für Linienbusse, durch Nutzung eines modellbasierten Algorithmus erreicht werden kann. Diese Frage ist jedoch nicht Teil der vorliegenden Arbeit. Vielmehr liegt der Fokus auf dem Vergleich von On- und Off-Policy-Algorithmen, die auch in **Abbildung 16** rot markiert sind. Bei beiden Varianten handelt es sich um modellfreie Algorithmen. Zur Implementierung und zum anschließenden Vergleich der Performance und erzielten Ergebnisse gilt es nun, zwei der vorgestellten Algorithmen auszuwählen.

Zur besseren Übersicht werden die vorgestellten Algorithmen aus Kapitel 2.7.3 in der folgenden Grafik nochmals zusammenfassend nach On- und Off-Policy, sowie nach MC- und TD-Algorithmen differenziert:

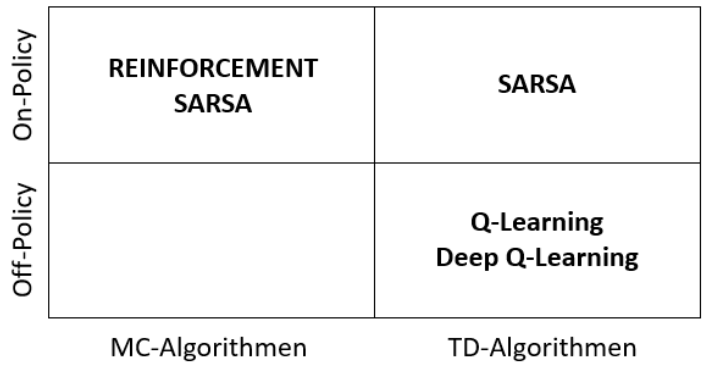


Abbildung 17: Einordnung der vorgestellten RL-Algorithmen

An dieser Stelle ist anzumerken, dass es sich auch hierbei keineswegs um eine annähernd vollständige Auflistung aller Subvarianten dieser Algorithmen handelt. Zu nahezu jede der vorgestellten Algorithmen wurden bereits eine Vielzahl von Kombinationen mit anderen Strategien oder statistischen Verfahren erforscht. Der Algorithmus REINFORCE steht als On-Policy-Algorithmus klar dem Q-Learning und der daran ansetzenden Variante des Deep Q-Learning als Off-Policy-Algorithmus gegenüber. SARSA nimmt aufgrund seiner in Kapitel 2.7.3.3 näher erläuterten Möglichkeit zur Einbindung einer Aktionshistorie eine Sonderrolle ein und kann sowohl als TD-, als auch als MC-Algorithmus implementiert werden.

Diese Sonderrolle macht SARSA zu einem interessanten Algorithmus im Hinblick auf den Einsatz in einem Betriebsleitsystem. Die Variante Expected SARSA verspricht in der Literatur

eine geringere Varianz (vgl. van Seijen et al. 2009, S. 3) als SARSA selbst. Q-Learning ist als ältester und zeitgleich wohl auch am weitest entwickelter, modellfreier RL-Algorithmus prädestiniert für einen Vergleich.

Im weiteren Verlauf der Arbeit werden also die Algorithmen

SARSA, *Expected SARSA* und *Q-Learning*

in einem Prototyp zur Betriebslenkung von Linienbussen im Störungsfalls eingesetzt und hinsichtlich ihrer Ergebnisse und Performance miteinander verglichen. Stellenweise wird Expected SARSA auch als E-SARSA abgekürzt.

3.4 Modellierung der Umwelt zur Simulation

Zur Implementierung des Prototyps ist zunächst notwendig, die Umgebung zu modellieren. Das mag zunächst verwirrend erscheinen, wurden doch im vorhergehenden Unterkapitel SARSA und Q-Learning erst den modellfreien Algorithmen zugeordnet. Zwar wurden im vorhergehenden Unterkapitel explizit modellfreie Algorithmen für die weitere Betrachtung ausgewählt, das Training erfolgt für gewöhnlich jedoch nicht in einer reellen Umgebung, sondern in einer Simulation derselben (vgl. Huber 2018, S. 23). Zunächst muss also die Umgebung modelliert werden. Mit Modellierung im engeren Sinne ist hier die Überführung aller relevanten Bedingungen aus der Praxis in eine allgemeine, mathematisch verwertbare Form gemeint. Diese Transformation komplexer, reeller Eigenschaften einer Umgebung in eine vereinfachte mathematische Darstellung entspricht per Definition einem mathematischen Modell (vgl. Rellensmann 2019, S. 5; Nahrstedt 2012, S. 201).

3.4.1 Anlehnung an Markov-Entscheidungsprozesse

Zur Modellierung der Simulationsumgebung für den Agenten eignet sich ein MEP (vgl. Lorenz 2020, S. 15; Witt 2019, S. 7). Sie werden durch das Tupel (S, A, p, r, γ) beschrieben, wobei

- S die Menge der möglichen Zustände
- A die Menge der möglichen Aktionen
- p die Übergangswahrscheinlichkeitsfunktion
- r die Gewinnfunktion
- und γ der Diskontierungsfaktor

sind.

Der aktuelle Zustand ergibt sich aus dem Fahrweg, dem aktuell gewählten Umleitung und dem eingetretenen Störfall. Diese drei Informationen werden numerisch abgebildet und zu einem Tupel zusammengefasst. Ausgehend von vier betrachteten Linien mit jeweils vier bekannten Umleitungsfahrwegen und einem Streckennetz mit 6416 Knotenpunkten ein möglicher Zustandsraum von

$$|R| * |D| * |N| = 4 * (4 * 4) * 6416 = 410624$$

Zuständen, wobei $|R|$ die Anzahl der Linien, $|D|$ die Anzahl der bekannten Umleitungsfahrwege und $|N|$ die Anzahl der Knotenpunkte im Streckennetz sind.

In der Praxis dürfte die relevante Zustandsmenge S jedoch viel kleiner sein, da die theoretische Annahme jeden einzelnen Knotenpunkt für alle Linien in Betracht zieht. In der Realität sind jedoch nur die Zustände interessant, in denen der gestörte Knotenpunkt auch innerhalb der gewählten Linie liegt. Außerdem könnten Knotenpunkte nach Streckenabschnitt gruppiert werden, was eine weitere Verkleinerung der Zustandsmenge zur Folge hat. Die Bedeutung dieser Gruppierung soll anhand des folgenden Kartenausschnittes verdeutlicht werden:

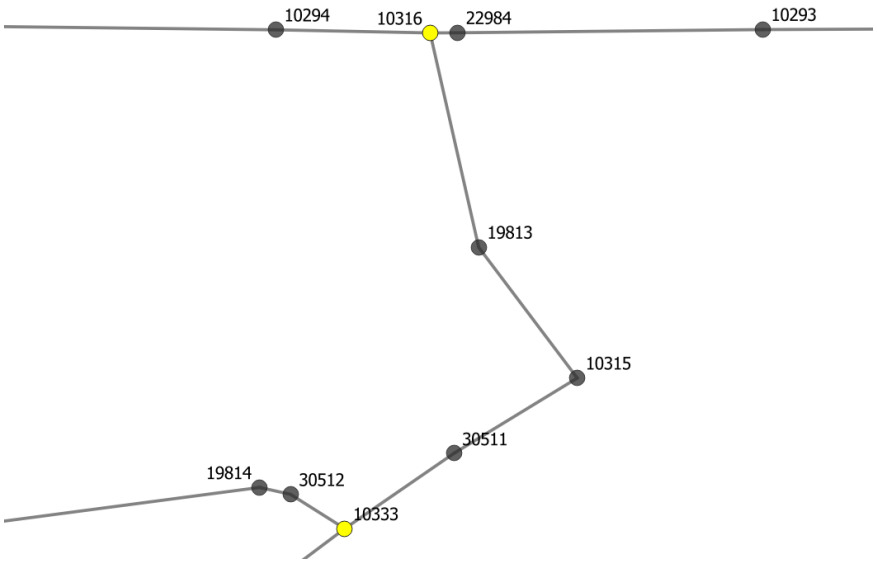


Abbildung 18: Relevanz von Knotenpunkten zur Beurteilung der Zustandsmenge

Zur Beurteilung der Befahrbarkeit des Streckenabschnittes zwischen den beiden gelb markierten Knotenpunkten 10316 und 10333 ist es irrelevant, ob nun einer der auf dem Abschnitt liegenden Knotenpunkte 30511, 10315 oder 19813 nicht befahrbar ist. Sobald einer der Knotenpunkte nicht befahrbar ist, kann der komplette Streckenabschnitt nicht mehr befahren werden. Folglich könnten die drei letztgenannten Knoten auch zu einem Knotenpunkt zusammengefasst werden, der als „gesperrt“ markiert wird, sobald der zugehörige

Streckenabschnitt nicht mehr befahrbar ist. Anders sieht es hingegen an Knotenpunkten aus, die eine Verbindung von zwei oder mehr Streckenabschnitten darstellen.

In dieser Form der Modellierung wird davon ausgegangen, dass pro Zustand jeweils nur ein Knotenpunkt gesperrt ist. Tatsächlich kann es in der Praxis aber auch vorkommen, dass mehrere Knotenpunkte zeitgleich gesperrt sind. Dies würde wiederrum zu einer Vergrößerung der Zustandsmenge S führen, soll aber in dieser Arbeit nicht weiter betrachtet werden.

Innerhalb der Zustandsmenge S kann weiter nach verschiedenen Zustandstypen differenziert werden. Folgende Zustandstypen lassen sich direkt ableiten:

- *Startzustände* $S_S \subseteq S$ sind Zustände, in denen sich das Fahrzeug auf dem regulären Fahrweg befindet
- *Abweichungszustände* $S_D \subseteq S$ sind Zustände, bei denen zwar eine Umleitung gewählt wurde, diese aber entweder den regulären Fahrweg des Fahrzeuges nicht tangiert oder nicht dafür sorgt, dass das Fahrzeug den gesperrten Knotenpunkt umfährt
- *Terminalzustände* $S_T \subseteq S$ sind alle Zustände, bei denen das Fahrzeug durch Auswahl einer geeigneten Umleitung den nicht befahrbaren Knotenpunkt umfährt

Startzustände sind folglich alle Zustände, bei denen keine Umleitung gewählt wurde und das Fahrzeug sich auf seinem regulären Fahrweg befindet. Terminalzustände sind hingegen alle Zustände, bei denen einen Umleitung gewählt wurde, welche dafür sorgt, dass das Fahrzeug den gesperrten Knotenpunkt umfährt.

Neben Zuständen erfordert ein MEP auch die Definition von Aktionen. Mögliche Aktionen sind in der Aktionsmenge A zusammengefasst. Eine Aktion beschreibt den Übergang von einem beliebigen Zustand s_t in einen Folgezustand s_{t+1}. Ein Zustand ist dabei ausschließlich abhängig vom vorhergehenden Zustand und der darin gewählten Aktion. Folgezustände oder weitere Vorgängerzustände haben keinen Einfluss auf den Zustand, der durch die Auswahl einer Aktion erreicht wird. Diese Bedingung entspricht der Markov-Eigenschaft (vgl. Witt 2019, S. 7) und ist Voraussetzung dafür, dass überhaupt von einem MEP gesprochen werden kann. Letztendlich ermöglicht erst die Annahme eines MEP eine derartige Vereinfachung der Realität im Modell (vgl. Witt 2019, S. 7). Die Zustandsmenge A ergibt sich aus allen bekannten Umleitungsfahrwegen und der *Nullaktion*, in der keine Umleitung gewählt, sondern der reguläre Fahrweg beibehalten wird. Alle gewählten Aktionen des Agenten, die der Reihe nach von einem bestimmten Startzustand $s_s \in S_s$ zu einem Terminalzustand $s_t \in S_t$ führen, bilden zusammen eine Episode (vgl. Lorenz 2020, S. 16). Bedingt durch die Definition der Zustände ist der Agent bereits nach einem Schritt in der Lage, einen Terminalzustand zu erreichen, indem er eine Aktion auswählt, die zu einer Umleitung des Fahrzeuges um den gesperrten Knotenpunkt führt.

Die Übergangswahrscheinlichkeitsfunktion p beschreibt die Wahrscheinlichkeit ausgehend von einem Zustand s_t unter Anwendung einer Aktion a_t einen Zustand s_{t+1} überzugehen als

$$p(s_{t+1}, r_{t+1}|s_t, a_t)$$

wobei r_{t+1} der erwartete Gewinn nach Übergang in den Zustand s_{t+1} ist (vgl. Witt 2019, S. 7). Unterschiedliche Übergangswahrscheinlichkeiten können entstehen, da ein Zustand jeweils immer von der aktuellen Fahrzeugposition und der Befahrbarkeit der Strecken abhängt. Durch Ausführen einer Aktion wechselt das Fahrzeug seine Position im Netz, es kommt daher in jedem Fall zu einem Zustandsübergang. Während der Folgezustand in den meisten Fällen einfach durch die geänderte Fahrzeugposition ausgemacht wird, könnte es sein, dass sich mit demselben Zeitschritt auch die Befahrbarkeit eines Streckenabschnittes ändert. In diesem Fall käme es zu einem Übergang in einen anderen Zustand. Der erwartete Gewinn wird wiederrum definiert durch die Gewinnfunktion $r(s_t, s_{t+1})$, die den Gewinn beim Übergang von einem Zustand in den nächsten definiert.

Zur Angabe der Gewinnfunktion ist es zunächst notwendig, sich mit relevanten Zustandsübergänge vertraut zu machen. Von Interesse sind besonders jene Zustandsübergänge, die zu einer Messbaren Veränderung des laufenden Betriebes führen. Das können sowohl negative als auch positive Veränderungen sein. Folgende Fälle sind theoretisch möglich:

- Das Fahrzeug, verbleibt durch Auswahl der Nullaktion in einem Startzustand $s_t \in S_S$ und befährt damit keinen gesperrten Knotenpunkt; der Agent erhält dafür einen positiven Gewinn, da in diesem Fall keine Umleitung erforderlich war und die Nullaktion daher optimal war
- Das Fahrzeug, geht aus einem beliebigen Zustand $s_t \in S$ in einen Abweichungszustand $s_{t+1} \in S_D$ über, ohne den gesperrten Knotenpunkt zu umfahren; der Agent erhält dafür einen negativen Gewinn, da keine zielführende Umleitung des Fahrzeuges erreicht wurde
- Das Fahrzeug, geht aus einem beliebigen Zustand $s_t \in S$ in einen Terminalzustand $s_{t+1} \in S_T$ über; der Agent dafür einen positiven Gewinn, das Fahrzeug erfolgreich umgeleitet wurde

Die für die beiden erstgenannten Zustandsübergänge werden mit einem festen positiven oder negativen Gewinn bewertet. Von besonderem Interesse ist der letztgenannte Zustandsübergang, da genau in diesem eine zielführende Umleitung gefunden wurde, die näher bewertet werden kann.

Die aus betrieblicher Sicht wichtigsten Bewertungskriterien sind zunächst die Differenz zwischen der Länge des Umleitungs- und des Regelfahrweges und die Anzahl eventuell

ausgelassener, planmäßiger Haltestellen. Zusätzlich können weitere Besonderheiten berücksichtigt werden. Befindet sich im erreichten Abweichungszustand $s_{t+1} \in S_D$ eine Haltestelle, deren Luftlinienabstand einen bestimmten Grenzwert zu einer Haltestelle des regulären Fahrwegs unterschreitet, erhält der Agent dafür einen positiven Gewinn. So wird erreicht, dass das Fahrzeug – sofern vorhanden – entlang möglicher Ersatzhaltestellen und nicht auf beliebigem Weg umgeleitet wird. Weiter kann das Verkehrsaufkommen auf einem Umleitungsfahrweg berücksichtig werden, um gegebenenfalls doch eine etwas längere, dafür stabiler befahrbare Umleitung zu wählen.

Ziel des Agenten muss es sein, jeweils das Maximum an Gewinn bis zum Erreichen eines Terminalzustands zu erhalten (vgl. Mainzer 2019, S. 119). Dabei werden ausgehend von einem Zustand $s_t \in S$ alle möglichen Aktionen und Folgezustände rekursiv betrachtet, bis ein Terminalzustand $s_t \in S_T$ erreicht wird. Jener Aktionspfad mit dem höchsten erwarteten Gewinn wird dann zur Bewertung eines Zustands oder einer Aktion ausgehend von einem Zustand in der State-Value-Funktion herangezogen. Der Gewinn berechnet sich durch Addition der einzelnen Gewinne nach jedem Zustandsübergang (vgl. Witt 2019, S. 8) als

$$R = r(s_t, s_{t+1}) + \gamma r(s_{t+1}, s_{t+2}) + \gamma^2 r(s_{t+2}, s_{t+3}) + \cdots + \gamma^k r(s_{t+k}, s_{t+k+1}) = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, s_{t+k+1})$$

Durch festlegen des Diskontierungsfaktors γ auf $0 < \gamma < 1$ wird erreicht, dass zeitlich nähere Aktionen mit positivem Gewinn stärker gewichtet werden, als Aktionen, die in größerem zeitlichen Abstand liegen (vgl. Witt 2019, S. 8; Dammann o.D., S. 4). Zur Veranschaulichung stelle man sich vor, dass der Agent im Training zunächst alle möglichen Umleitungswege ausprobiert, bevor er einen Terminalzustand erreicht, obwohl der Terminalzustand auch bereits nach einem Schritt erreicht werden kann, indem direkt zu Beginn die optimale Umleitung gewählt wird. Ein Diskontierungsfaktor von 1 bedeutet entsprechend, dass alle Gewinne aus nacheinander gewählten Umleitungen gleich gewichtet werden, ein Diskontierungsfaktor nahe 0 würde hingegen fast ausschließlich die ersten Umleitungen zur Bewertung ins Gewicht fallen lassen. Da der Agent in der Praxis bereits durch Auswahl der besten Aktion im ersten Schritt einen Terminalzustand erreichen kann, spielt der Diskontierungsfaktor hierbei kaum mehr eine Rolle. Nach dem Training ist für jeden Startzustand die beste Aktion bekannt, um direkt im nächsten Schritt einen Terminalzustand zu erreichen.

Im Training erfolgt die Anpassung der Q-Werte in Anlehnung auf der *Bellmann-Eigenschaft*. Diese, dass die Lösung eines Problems dann optimal ist, wenn die Lösung aller Teilprobleme optimal ist (vgl. Schmitz 2017, S. 14). Umgekehrt bedeutet das also, dass eine optimale Lösung dann erreicht werden kann, wenn auf dem Weg zur Gesamtlösung stets optimale Teillösungen gewählt werden. Ausgehend von den bisherigen Annahmen kann also zu jeder Aktion ausgehend von einem bestimmten Zustand ein skalarer Wert ermittelt werden. Man spricht

vom sogenannten Q-Wert, in diesem Fall ein *State-Action*-Wert, der für das Q-Learning definiert ist als

$$Q(s,a) = r + \gamma * \max_a Q(s_{t+1}, a)$$

wobei r der bisher erreichte Gewinn ist und über zum Q-Learning gehörende Strategiefunktion $\max_a Q(s_{t+1}, a)$ stets die Aktion ausgewählt wird, deren Q-Wert bezogen auf den Folgezustand am höchsten ist.

Für SARSA ist die Q-Funktion als

$$Q(s,a) = r + \gamma * Q(s_{t+1}, a_{t+1})$$

Der Unterschied liegt in der Strategiefunktion und in der Betrachtung des Folgezustandes. Abweichend vom Q-Learning kann die Auswahl der Aktion auch mit einer anderen Funktion als der Greedy-Strategie ausgewählt werden. SARSA nutzt dieselbe Strategiefunktion zum Training wie auch später Dadurch wird ausschließlich die tatsächlich gewählte Aktion berücksichtigt, selbst dann, wenn es möglicherweise eine Aktion mit einem höheren State-Action-Wert gibt. Abhängig von der gewählten Strategiefunktion ist ein gewisses Maß an Exploration im SARSA-Algorithmus sichergestellt.

Durch diese Grundkonstruktion wird erreicht, dass der final trainierte Agent den Linienweg nicht verlässt, wenn kein Erfordernis dazu besteht und andernfalls stets jene Umleitung mit dem geringsten Gesamtverlust wählt.

3.4.2 Exploration mit der $n\epsilon$-Greedystrategie

Da sowohl SARSA als auch Expected SARSA während dem Training dieselbe Strategiefunktion nutzen, wie auch nach Abschluss des Trainings, führen beide Algorithmen auch nach dem Training eine Exploration durch. Wie in den meisten Fällen wird auch in dieser Arbeit als Strategiefunktion die ε-Greedyfunktion verwendet. Wie stark ein Algorithmus exploriert, hängt also vom Hyperparameter ϵ ab. Während dieser in Q-Learning durch die Zerfallsrate mit steigender Anzahl an Episoden gegen 0 strebt, bleibt ϵ in SARSA und Expected SARSA gleich oder wird nach unten hin auf einen bestimmten Wert begrenzt, sodass ϵ niemals exakt 0 wird.

Dieses Verhalten hat zur Folge, dass auch bei hinreichend erfolgten Training mit einer Wahrscheinlichkeit von ϵ immer wieder eine Aktion gewählt wird, die nicht zwangsläufig die optimale Aktion darstellt. SARSA kann durch seine Exploration daher als instabil eingestuft werden (vgl. van Seijen et al. 2009, S. 8). In einem produktiven ITCS wäre das allerdings fatal, da das Fahrzeug im schlimmsten Fall wissentlich auf einen gesperrten Knotenpunkt zu geleitet würde. Zwar wäre dann durch Exploration sichergestellt, dass die gewählte Aktion noch immer

keinen besseren Wert erreicht hätte, die Betriebsstabilität wäre damit jedoch massiv eingebrochen und das Vertrauen des Fahrpersonals, des Leitstellenpersonals und nicht zuletzt auch der Fahrgäste gebrochen.

Statt die Auswahl einer alternativen Aktion komplett dem Zufall zu überlassen werden, soll die Auswahl der in Frage kommenden Aktionen in dieser Arbeit auf die n besten Aktion begrenzt werden. Es wird also zunächst eine Vorauswahl auf jene Aktionen getroffen, die nach einer gewissen Grundexploration erfahrungsgemäß am die höchsten Gewinne erzielten. Erst im nächsten Schritt wird dann per Zufallsprinzip eine Aktion aus dieser Liste ausgewählt. Dadurch wird sichergestellt, dass bei hinreichendem Training keine Aktion gewählt wird, welche zu einem massiven Einbruch der Betriebsstabilität führen würde, aber dennoch auch alternative Aktionen durch den Agenten ausprobiert werden.

Durch die Nutzung dieser $n\epsilon$-Greedystrategie wird außerdem eine weitere, wichtige Eigenschaft erreicht. Kommt nach einer hohen Anzahl bereits durchgeführter Episoden in Q-Learning eine weitere Aktion durch Aufzeichnung eines neuen, möglichen Umleitungsfahrweges hinzu, würde diese mit zunehmender Anzahl an Episoden nie gewählt werden, da im Q-Learning bedingt durch die Greedystrategie und dem zeitgleich gegen 0 strebenden ϵ stets nur jene Aktion mit dem höchsten erwarteten Gewinn gewählt wird und die neue Aktion in der Q-Tabelle mit dem Wert 0 initialisiert würde. Selbst dann, wenn die neue Umleitung aus betrieblicher Sicht die beste wäre, würde sie in Q-Learning nie erreicht. Werden in die eingangs beschriebene $n\epsilon$-Greedystrategie nun auch noch alle Aktionen einbezogen, welche zuvor nie exploriert wurden, werden diese früher oder später auch durch den Agenten ausgewählt.

3.5 Beschreibung der Gesamtarchitektur

Aufbauend auf den Entscheidungen der bisherigen Unterkapitel zur theoretischen Modellierung wird in diesem Kapitel zusammenfassend die Gesamtarchitektur in einer möglichen Systemumgebung beschrieben.

4 Prototypische Implementierung

Aufbauend auf der vorangegangenen theoretischen Modellierung wird das im vorhergehenden Kapitel erarbeitete Grundkonzept als Prototyp implementiert. Die Implementierung ist dabei auf das Machine Learning selbst beschränkt. Eingangsdaten und weitere Schnittstellen werden berücksichtigt, sind aber im Prototyp ausschließlich simuliert. Dieses Kapitel beschreibt die Erzeugung notwendiger Eingangsdaten und geht auf die technische Umsetzung des Prototyps, sowie die Aufbereitung der Ergebnisdaten ein.

4.1 Allgemeine technische Aspekte

Zur Implementierung des Prototyps wird die Sprache *Python* verwendet. Python ist die mit Abstand am weitesten entwickelte Sprache in Sachen ML, sodass hier ein optimales Angebot an verschiedenen Frameworks zur Umsetzung des Prototyps zur Verfügung steht. Zwar beinhalten diese Frameworks auch fertig implementierte Varianten von Q-Learning und SARSA und passenden Simulationsumgebungen, allerdings sind diese zumeist auf spieltheoretische Anwendungsfälle ausgelegt und daher für die Umleitungssuche für Linienbusse nur bedingt geeignet. Aus diesem Grund werden die beiden verglichenen Algorithmen Q-Learning und SARSA, sowie die Simulationsumgebung unter Zuhilfenahme des Frameworks *OpenAI Gym* umgesetzt. Weitere Module zur Datenerzeugung- und Aufbereitung werden als Einzelskripte implementiert.

Der gesamte Prototyp steht als GitHub-Repository öffentlich unter der Adresse

https://github.com/sebastianknopf/ITCS

zur Verfügung. Dieses Repository ist befristet bis zum 30.04.2023, anschließend kann der Prototyp auf Anfrage zur Verfügung gestellt werden.

Die folgende Abbildung zeigt die vollständige Projektstruktur des Prototyps:

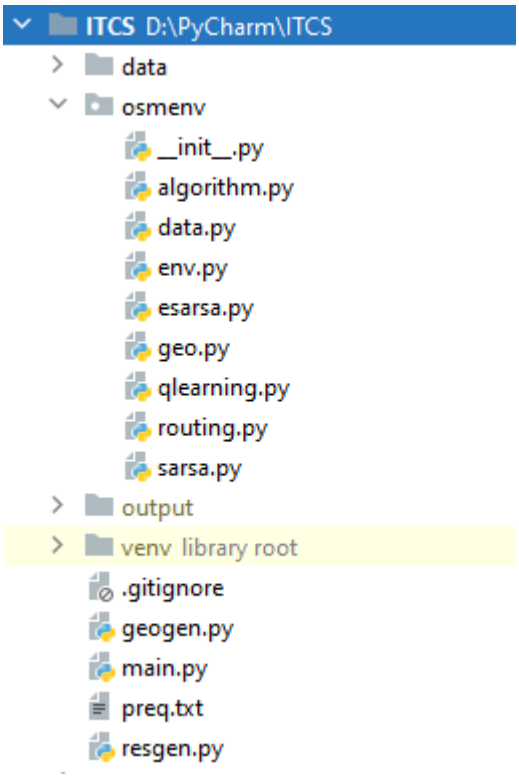


Abbildung 19: Projektstruktur des Prototyps

Das Verzeichnis `data` enthält alle erzeugten Eingangsdaten, in `output` werden Ergebnisdaten gespeichert.

Als Laufzeitumgebung wird eine *Virtuelle Umgebung* für Python 3.9 genutzt. Diese befindet sich im Ordner `venv` und ist nicht Teil des Prototyps selbst. Der Prototyp ist als Package im gleichnamigen Ordner `osmenv` enthalten. Die drei Module `main`, `geogen` und `resgen` dienen jeweils zur Ausführung des Prototyps, zum Erzeugen von Regel- und Umleitungsfahrwegen und zur Aufbereitung der Ergebnisse aus den erzeugten Q-Tabellen.

Die übrigen Dateien sind nicht Teil des Prototyps und stehen in keinem funktionalen Zusammenhang mit demselben.

Die Funktion der einzelnen Module innerhalb von `osmenv` wird in der nachfolgenden Tabelle beschrieben:

Modul	Funktion / Implementierung
`algorithm.py`	Basisklasse für Q-Learning und SARSA
`data.py`	Hilfsfunktionen zum Laden von Daten aus JSON
`env.py`	Implementierung der Simulationsumgebung
`esarsa.py`	Implementierung Expected SARSA
`geo.py`	Hilfsklasse zum Erzeugen von GeoJSON-Dateien
`qlearning.py`	Implementierung Q-Learning
`routing.py`	Hilfsklasse für das Routing auf OSM-Daten
`sarsa.py`	Implementierung SARSA

Tabelle 2: Funktion der Module im Package osmenv

4.2 Erzeugung von Regel- und Umleitungsfahrwegen

Da keine im Echtbetrieb aufgezeichneten GPS-Daten von Linienbussen zur Verfügung stehen, müssen die Regel- und Umleitungsfahrwege für die in Kapitel 3.1 vorgestellten Beispielszenarien synthetisch erzeugt werden. Dieser Schritt käme im Echtbetrieb dem MapMatching der aufgezeichneten GPS-Daten aus den Bussen am Ende eines Betriebstages gleich.

Die Erzeugung der Eingangsdaten geschieht durch Ausführung des Moduls `geogen`. Die Fahrwege werden dabei durch die Berechnung einer Route aus den OSM-Daten entlang dem eigentlichen Verlauf der Linie erzeugt. Die Umleitungsfahrwege werden basierend auf bekannten, vielfach genutzten Umleitungsstrecken ebenfalls durch Berechnung einer Route aufgezeichnet. Zum Routing wird ein manuell erzeugtes Sekundärnetz zusammen mit der Bibliothek `pyroutelib3` genutzt. Dieses Vorgehen bringt mehrere Vorteile mit sich. Zum einen entsprechen die erzeugten Fahrwege den reellen Fahrwegen unter Berücksichtigung von Einbahnstraßen, Kreuzungen und Verkehrsregeln, die auch tatsächlich so von einem Linienbus gefahren werden könnten. Zum anderen wird als Ergebnis direkt eine Liste von befahrenen OSM-Knotenpunkten geliefert, die eine einfache Weiterverarbeitung ermöglicht.

Die erzeugten Fahrwege werden jeweils als GeoJSON-Datei und als JSON-Datei abgespeichert. Während die GeoJSON-Datei zum Anzeigen der errechneten Fahrwege im Geoinformationssystem QGIS dienen, werden die JSON-Dateien anschließend manuell um planmäßige, auf dem entsprechenden Fahrweg liegende Haltestellen ergänzt. Die finale Struktur einer solchen JSON-Datei ist in der folgenden Abbildung zu sehen:


```
{
  "id": "Stadtbus 2 / Hinfahrt",
  "length": 3.4566005104272004,
  "stops": [
    {"name": "Pforzheim Buechenbronner Strasse"...},
    {"name": "Pforzheim Broetzinger Bruecke"...},
    {"name": "Pforzheim Hans-Sachs-Strasse"...},
    {"name": "Pforzheim Benckiser Strasse"...},
    {"name": "Pforzheim Turnplatz"...},
    {"name": "Pforzheim Jahnstrasse"...},
    {"name": "Pforzheim Leopoldplatz"...},
    {
      "name": "Pforzheim Bahnhofstrasse",
      "node": 27893
    }
  ],
  "nodes": [
    21233,
    21234,
    28560,
    9534,
```


Abbildung 20: JSON-Struktur eines erzeugten Regelfahrweges

Zu erkennen ist die Liste von OSM-Knotenpunkten und die nachträglich manuell eingefügten planmäßigen Haltestellen. Diese verweisen jeweils auf den Knotenpunkt, an dem sie sich innerhalb des Fahrweges befinden. Auf diese Weise kann leicht geprüft werden, ob eine Haltestelle in einem veränderten Fahrweg noch bedient wird, oder nicht.

Für das Stadtszenario werden vier Umleitungsfahrwege erzeugt. Diese sind in den nachfolgenden Abbildungen jeweils in rot zu sehen, während die Regelfahrwege in blau dargestellt sind:

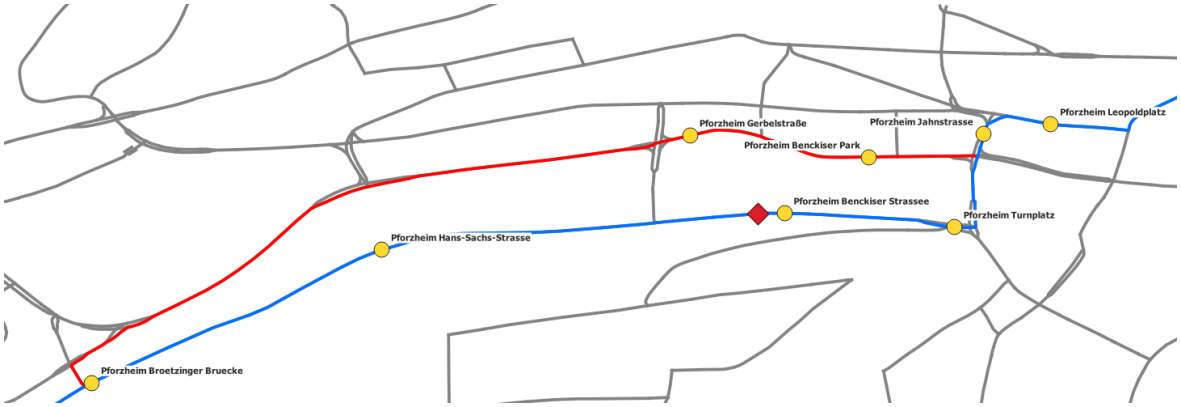


Abbildung 21: Umleitungsfahrweg #1 für das Stadt-Szenario

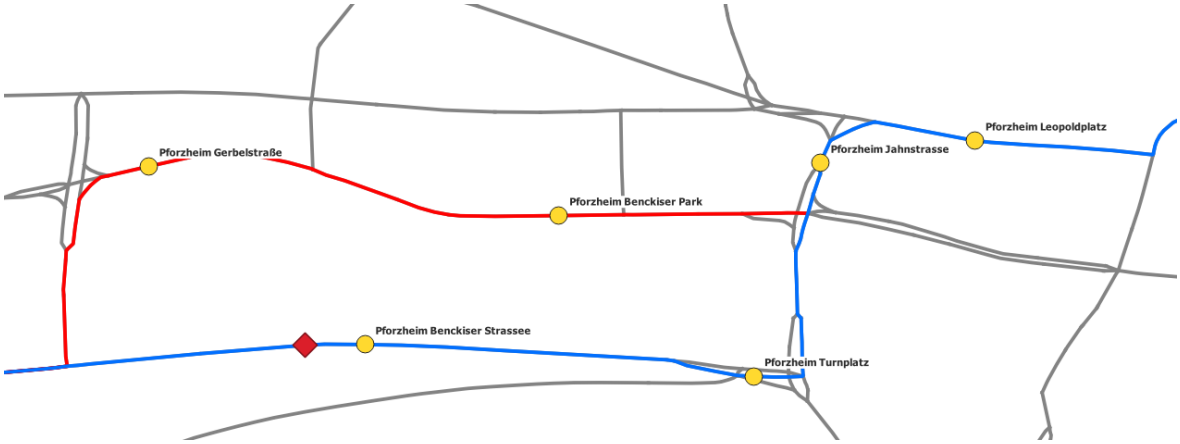


Abbildung 22: Umleitungsfahrweg #2 für das Stadt-Szenario

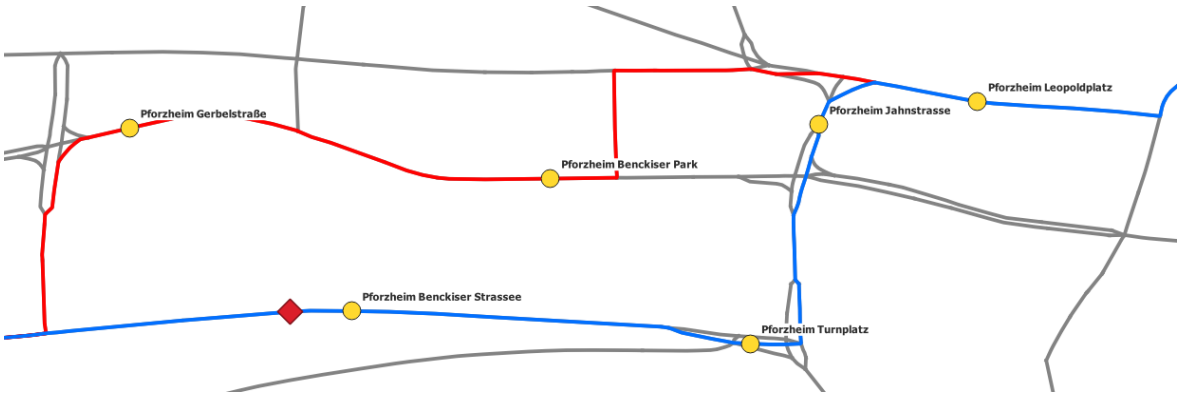


Abbildung 23: Umleitungsfahrweg #3 für das Stadt-Szenario

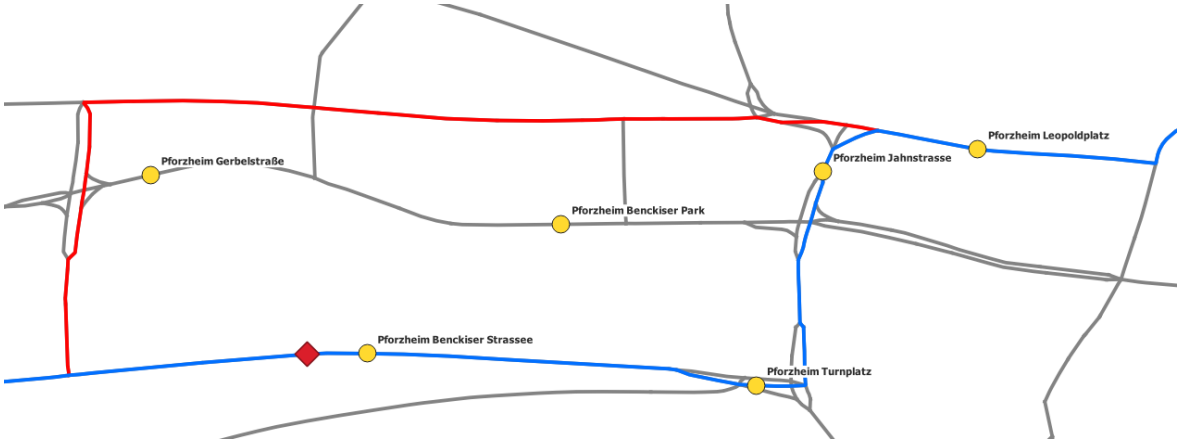


Abbildung 24: Umleitungsfahrweg #4 für das Stadt-Szenario

Für das Land-Szenario werden fünft Umleitungsfahrwege erzeugt. Zwei davon beziehen sich auf die Linie 743, drei auf die Linie 744. Die Umleitungsfahrwege sind in den folgenden Abbildungen in rot und die Regelfahrwege in orange für die Linie 743 und in grün für die Linie 744 zu sehen:

Abbildung 25: Umleitungsfahrweg #1 für das Land-Szenario

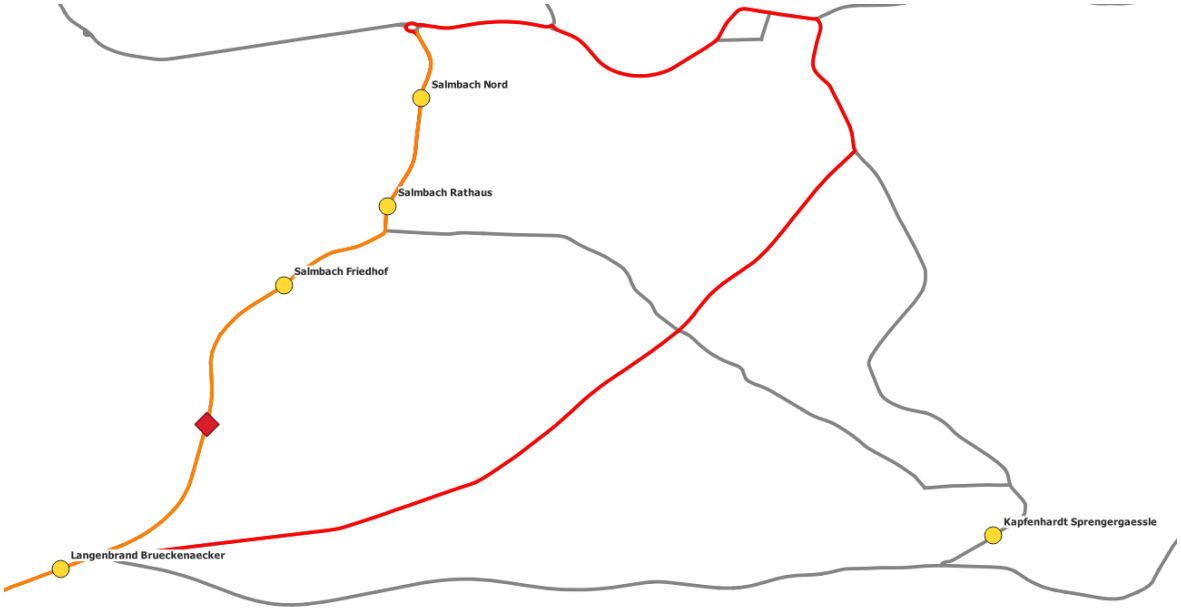


Abbildung 26: Umleitungsfahrweg #2 für das Land-Szenario

Abbildung 27: Umleitungsfahrweg #3 für das Land-Szenario

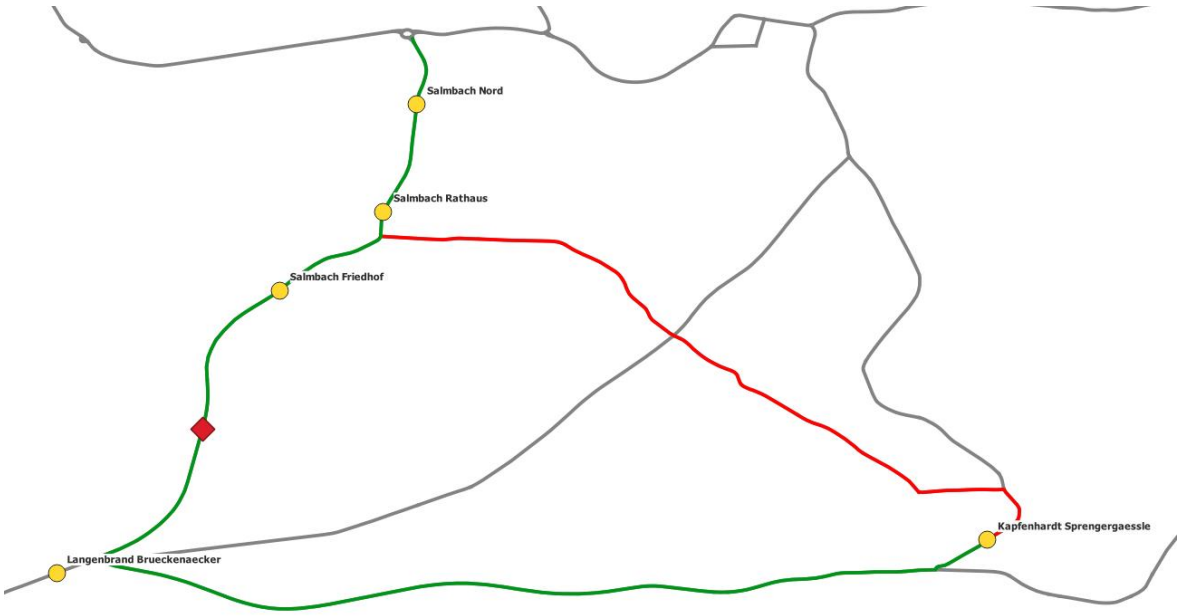


Abbildung 28: Umleitungsfahrweg #4 für das Land-Szenario

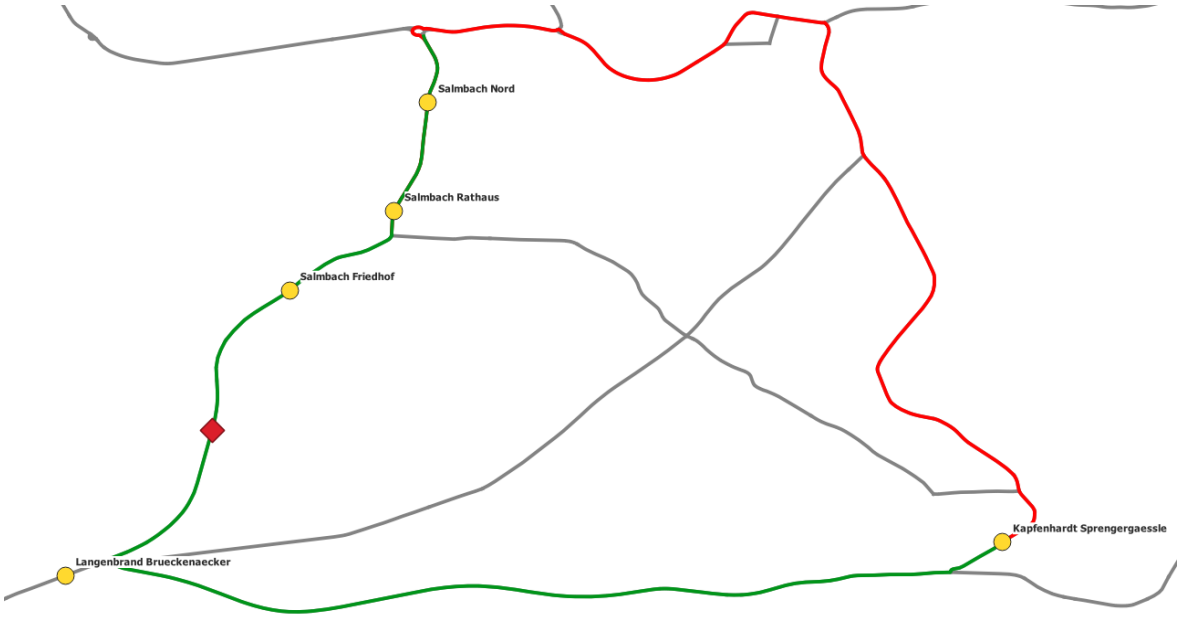


Abbildung 29: Umleitungsfahrweg #5 für das Land-Szenario

Auch für das Vorort-Szenario werden vier Umleitungsfahrwege erzeugt. In den folgenden Abbildungen sind die Regelfahrwege in grün, die Umleitungsfahrwege in rot zu sehen:

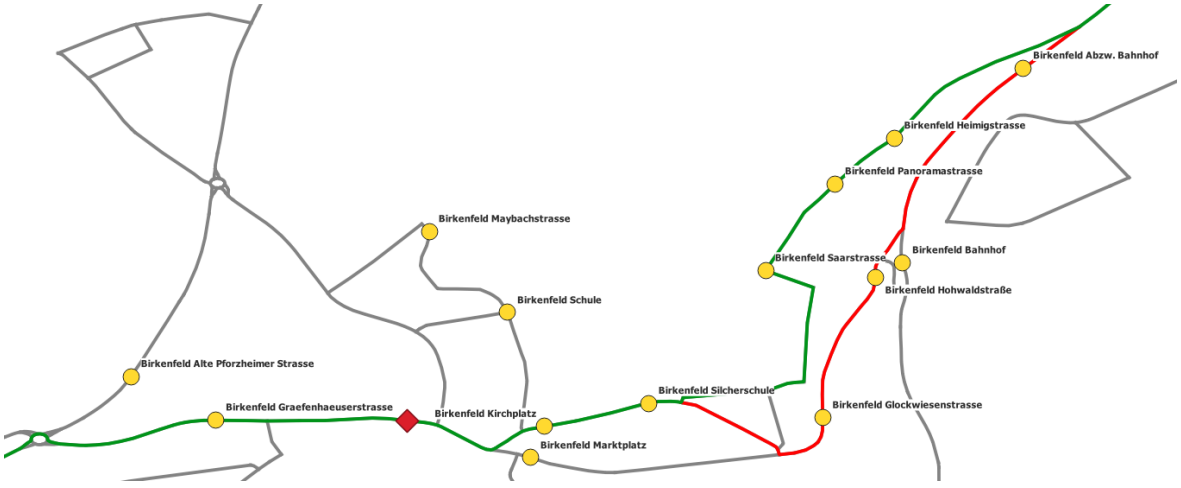


Abbildung 30: Umleitungsfahrweg #1 für das Vorort-Szenario

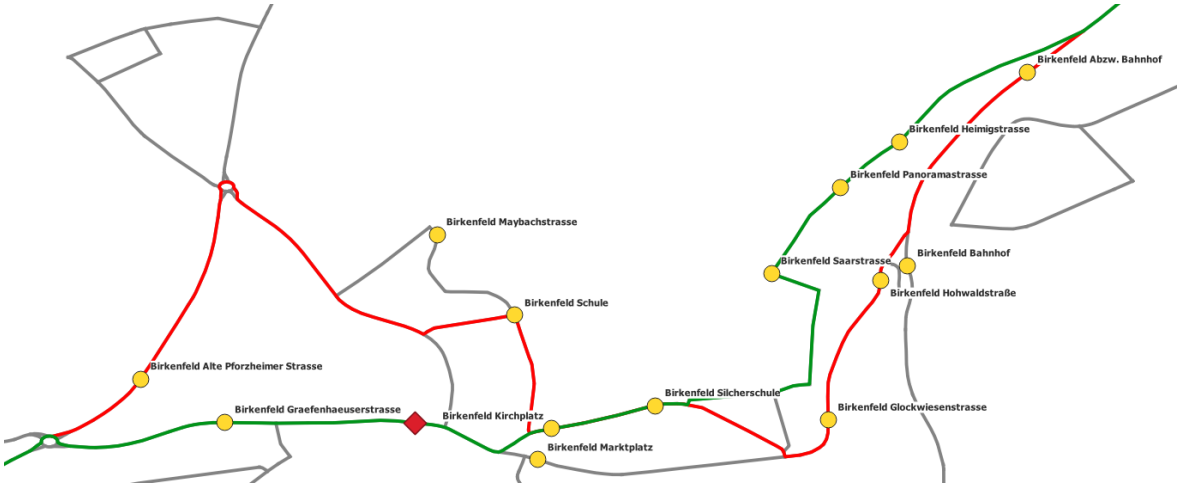


Abbildung 31: Umleitungsfahrweg #2 für das Vorort-Szenario

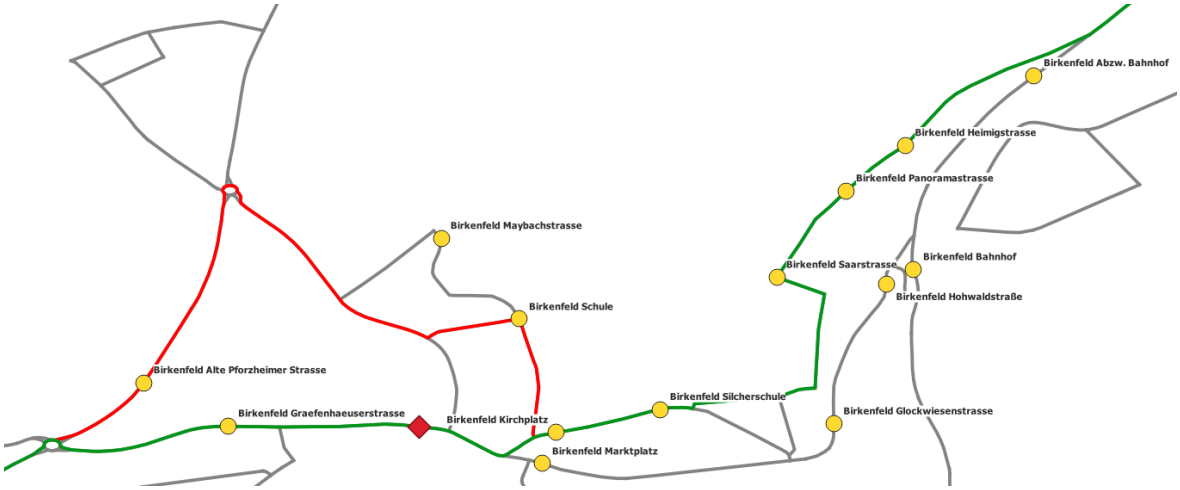


Abbildung 32: Umleitungsfahrweg #3 für das Vorort-Szenario

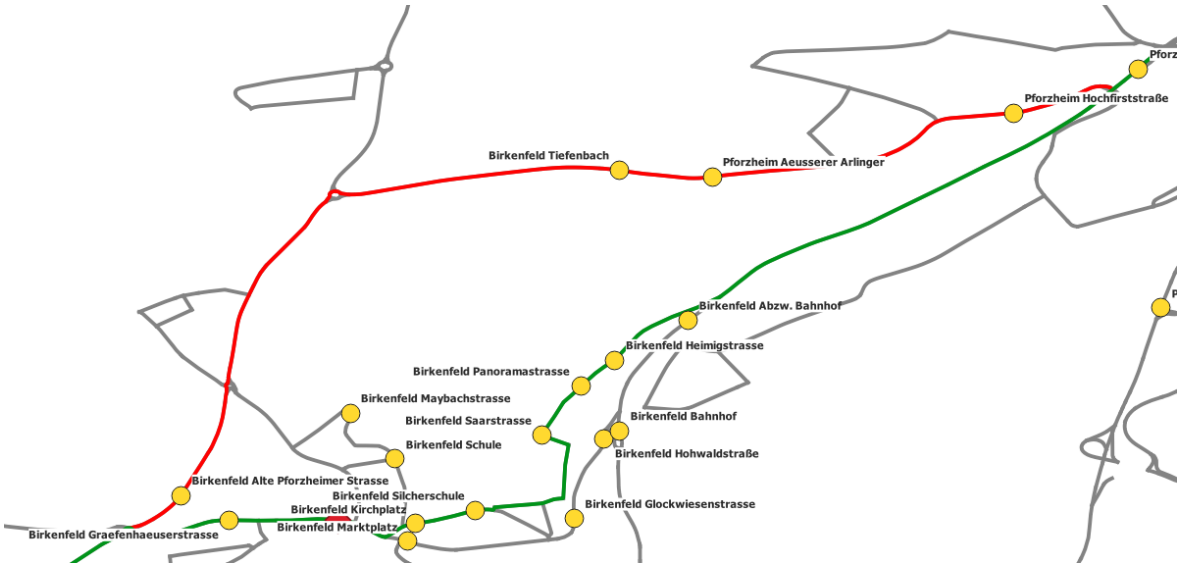


Abbildung 33: Umleitungsfahrweg #4 für das Vorort-Szenario

Mit diesen insgesamt 13 synthetisch erzeugten Umleitungsfahrwegen ist je Szenario eine optimale und eine weiträumige Umleitung enthalten, die während des Trainings beurteilt werden kann. Zusätzlich sind weitere Umleitungsfahrwege enthalten, welche sich in ihrer Haltepolitik oder Streckenlänge unterscheiden und folglich zu einer unterschiedlichen Bewertung führen müssen.

Alle erzeugten Umleitungsfahrwege sind zur besseren Übersicht im Anhang nochmals enthalten.

4.3 Simulationsumgebung und Bewertungsfunktion

Bekanntermaßen basiert das Training eines RL-Agenten auf einem „Trial-and-Error“ Verfahren. Dass ein solches Training nicht von Grund im Produktivbetrieb eines ITCS durchgeführt werden kann, zeigen die Beispiele aus Kapitel 2.4. Das Training wird daher in einer Modellumgebung durchgeführt, welche die relevanten Gegebenheiten aus der reellen Umgebung simuliert. Außerdem ist in der Simulationsumgebung die Bewertungsfunktion definiert, welche den negativen oder positiven Gewinn des Agenten nach Ausführung einer Aktion erhält. Die Verwendung einer so modellierten Simulationsumgebung bringt auch eine deutliche Beschleunigung des Trainings mit sich, da die einzelnen Schritte um ein Vielfaches

schneller durchgeführt werden können. Limitierender Faktor ist dann vielmehr die Rechenleistung der verwendeten Hardware und dem Betriebssystem.

Zur Implementierung der Simulationsumgebung wird das Framework OpenAI Gym als Basis genutzt. Die Simulationsumgebung wird in der Klasse `Environment` implementiert, welche wiederrum von der OpenAI Klasse `Env` erbt. Das folgende UML-Diagramm zeigt die vereinfachte Klassenstruktur beider Klassen:

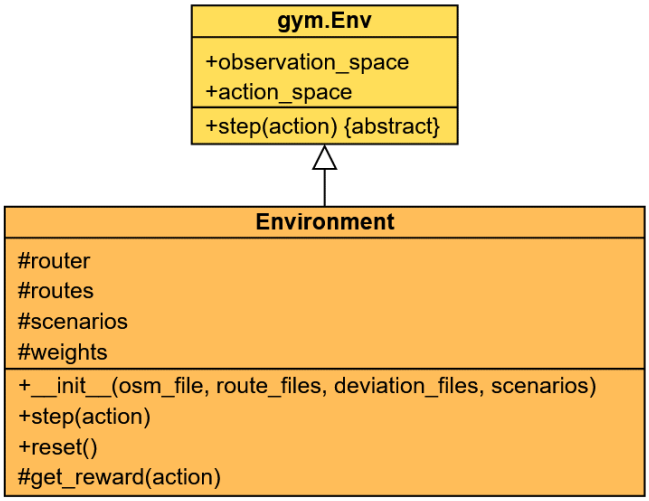


Abbildung 34: Vereinfachtes UML-Diagramm der Simulationsumgebung

Die beiden Eigenschaften `observation_space` und `action_space` werden aus der Klasse `Env` vererbt. Mit Hilfe diesen beiden Eigenschaften werden die Dimensionen des Zustands- und Aktionsraum definiert. Der Zustandsraum entspricht dabei allen möglichen Kombinationen aus Linien, gewählten Umleitungen und Störungsszenarien. Für die Beispiele in dieser Arbeit ist für vier Linien, insgesamt 13 Umleitungen und drei Störungsszenarien ist der Zustandsraum folglich auf 156 Zustände festgelegt. Der Aktionsraum entspricht exakt der Anzahl Umleitungen und der Nullaktion und nimmt in dieser Arbeit daher den Wert 14 an. Sowohl Zustände als auch Aktionen werden außerhalb der Simulationsumgebung generell als nullbasierter Index ausgedrückt. Diese Umrechnung bringt auf den ersten Blick nur zusätzliche Komplexität in die Implementierung, bringt aber den entscheidenden Vorteil, dass alle Zustände und Aktionen eine eindeutige, fortlaufende Nummer haben und damit auch auf einfachem Weg durch einen Zufallsgenerator gewählt werden können.

Die Klasse `Environment` empfängt im Konstruktor jeweils einen Dateinamen für die OSM-Datei, die das Sekundärnetz enthält, in den beiden folgenden Parametern jeweils eine Liste mit Dateinamen der JSON-Dateien, welche jeweils einen Regel- oder Umleitungsfahrweg enthalten und zuletzt eine Liste mit jenen OSM-Knotenpunkten, für die ein Störfall trainiert werden soll. Hieraus ergeben sich auch die Indices innerhalb der Zustände oder den Aktionen. Folglich

entspricht die erstgeladene Linie dem Index 0. Da die Nullaktion den Index 0 hat, entspricht der erstgeladene Umleitungsfahrweg dem Index 1.

Die Eigenschaft `router` enthält dabei eine Router-Objektinstanz aus `pyroutelib3`, mit deren Hilfe Streckenlängen aus OSM-Knotenpunktsequenzen berechnet und gegebenenfalls die OSM-Knotenpunkte auch wieder in GPS-Punkte zurückkonvertiert werden können. Die Eigenschaften `routes` und `scenarios` enthalten die geladenen Regelfahrwege und Knotenpunkte, für die eine Umleitung trainiert werden soll. Die letzte Eigenschaft mit dem Namen `weights` enthält die Gewichtungen, wie stark welches Kriterium bei der Beurteilung einer Umleitung jeweils gewichtet werden soll.

Die abstrakte Methode `step` übernimmt die gewählte Aktion als Parameter und wird in der Klasse `Environment` aus der Elternklasse überschrieben. Diese Methode beschreibt einen Einzelschritt, der durch Ausführen einer Aktion in der Simulationsumgebung innerhalb einer Episode ausgeführt wird. Innerhalb dieser Methode wechselt die Simulationsumgebung durch Auswahl einer Umleitung in den nächsten Zustand, bewertet diesen Zustandsübergang und gibt die Werte zurück. Die Umleitung wird durch die entsprechende Aktion verkörpert.

Die Bewertung eines Zustandsübergangs erfolgt in der geschützten Methode `get_reward`. In dieser Methode wird zunächst geprüft, ob überhaupt eine Umleitung für die aktuell betrachtete Fahrt notwendig gewesen wäre oder ob die gewählte Umleitung dazu führt, dass das Fahrzeug einen gesperrten Knotenpunkt umfährt. Entsprechend erhält der Agent einen positiven oder negativen Gewinn, der entweder exakt dem Wert -1 oder 1 entspricht. Wurde eine passende Umleitung gefunden, wird diese mit dem Grundwert 1 bewertet und anhand zweier Faktoren abgemindert. Diese beiden Faktoren sind im implementierten Prototypen

- die durch die Umleitung zusätzliche Streckenlänge
- die Anzahl der nicht bedienten, planmäßigen Haltestellen

Diese beiden Faktoren bilden in der Praxis die Entscheidungsgrundlage für die Anordnung einer dispositiven Maßnahme. Der Längenfaktor wird im Prototyp berechnet nach:

$$f_l = \left(\frac{Länge\ Regelfahrweg}{Länge\ Regelfahrweg\ mit\ Umleitung}\right)^{\frac{g_l}{4}}$$

wobei f_l der Abminderungsfaktor für die Länge und g_l die Gewichtung desselben ist.

Analog dazu wird der Haltestellenfaktor im Prototyp berechnet nach:

$$f_s = \left(\frac{Anzahl\ erreichte\ Haltestellen}{Anzahl\ planmäßige\ Haltestellen}\right)^{\frac{g_s}{4}}$$

wobei f_s der Haltestellenfaktor und g_s die Gewichtung desselben ist.

Die Bewertung für eine erfolgreiche Umleitung errechnet sich dann aus:

$$r = 1 * f_s * g_s$$

Im Zuge der Bewertung stellt die Methode `get_reward` außerdem fest, ob ein Terminalzustand erreicht wurde. Dies ist genau dann der Fall, wenn der Regelfahrweg kombiniert mit der gewählten Umleitung dafür sorgt, dass der gesperrte Knotenpunkt umfahren wird. Das entsprechende Flag für den Terminalzustand wird zusammen mit dem errechneten Gewinn zurückgegeben.

4.4 Umsetzung von Q-Learning, SARSA und E-SARSA

Die Module `qlearning`, `sarsa` und `esarsa` enthalten die Implementierung für Q-Learning, SARSA und Expected SARSA. Im Modul `algorithm` ist zunächst die Basisklasse `TemporalDifferenceAlgorithm` definiert, welche alle wichtigen Basisfunktionalitäten enthält. Die Klassen `QLearning`, `Sarsa` und `ExpectedSarsa` sind entsprechend in einer Klassenhierarchie angeordnet, welche im folgenden UML-Diagramm zu sehen ist:

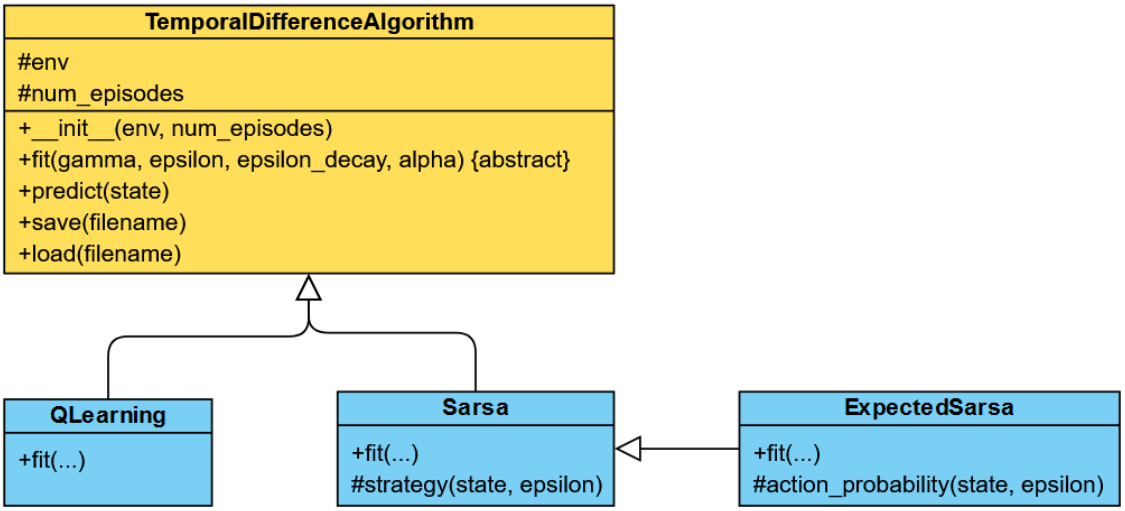


Abbildung 35: Vereinfachtes UML-Diagramm der implementierten ML-Algorithmen

Die Klasse `TemporalDifferenceAlgorithm` ist insbesondere für die Initialisierung der Q-Tabelle verantwortlich. Außerdem wird zentral eine Instanz der Simulationsumgebung vorgehalten, welche über den Konstruktor zugewiesen wird. Darüber hinaus besteht die Möglichkeit, eine feste Anzahl von durchzuführenden Episoden festzulegen. Geschieht dies nicht explizit, läuft das Training, bis eine Konvergenz festgestellt werden kann. Die abstrakte

Methode `fit` empfängt die Hyperparameter, die zum Training verwendet werden. Die Methode `predict` dient nach Abschluss des Trainings dazu, die optimale Umleitung ausgehend von einem Zustand zu ermitteln. Über die Methoden `load` und `save` kann eine bestehende Q-Tabelle gespeichert oder erneut geladen werden.

In der Klasse `QLearning` ist der eigentliche Q-Learning Algorithmus in der überschriebenen Methode `fit` implementiert. Die Implementierung folgt dabei strikt dem nachfolgend abgebildeten Schema von Sutton und Barto:

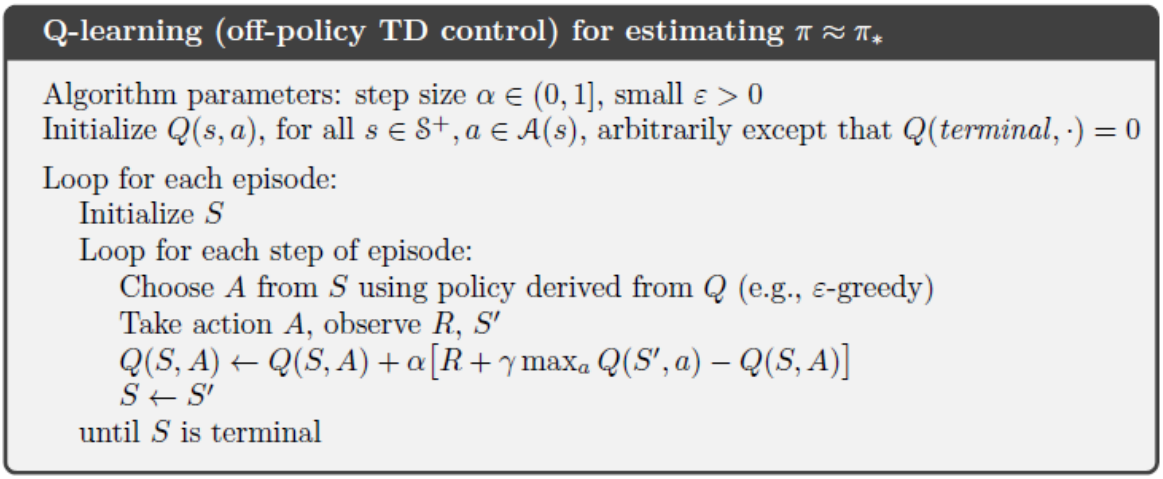


Abbildung 36: Allgemeine Formulierung von Q-Learning (Quelle: Sutton und Barto 2018, S. 131)

Die Strategiefunktion ist bei Q-Learning stets eine Greedystrategie, daher ist diese nicht als gesonderte Methode implementiert. Q-Learning wählt nach Abschluss des Trainings stets die Aktion mit dem höchsten, erwarteten Gewinn.

Die Klasse Sarsa enthält hingegen eine explizit definierte Strategiefunktion, im Fall dieser Arbeit als *nε*-Greedyfunktion, die in Kapitel 3.4.2 eingehend beschrieben wurde. Im Gegensatz zu Q-Learning wird diese Strategiefunktion auch nach Abschluss des Trainings zur Aktionsauswahl genutzt.

Ansonsten folgt die Implementierung dem nachfolgend abgebildeten Schema von Sutton und Barto:

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Abbildung 37: Allgemeine Formulierung von Q-Learning (Quelle: Sutton und Barto 2018, S. 130)

Da sich Expected SARSA nur in der Updatefunktion von SARSA unterscheidet, erbt die Klasse `ExpectedSarsa` direkt von Sarsa und definiert keine eigene Strategiefunktion. In der Methode `action_probability` wird hingegen die in Expected SARSA verwendete Auswahlwahrscheinlichkeit errechnet, die dann in der erneut überschriebenen `fit`-Methode für das Update der Q-Tabelle genutzt wird.

4.5 Ausführung des Prototyps

Um den Prototyp auszuführen und damit ein erstes Training zu starten, müssen zunächst die Eingangsdatensätze erzeugt sein. Die eigentliche Ausführung des Prototyps geschieht dann durch Aufruf des Modules `main`. In diesem Modul wird zunächst eine Instanz der Simulationsumgebung erzeugt. In dieser Instanz werden alle verfügbaren Regel- und Umleitungsfahrwege, das Sekundärnetz und drei Störungsszenarien geladen. Anschließend wird jeweils eine `QLearning`-, eine `Sarsa`- und eine `ExpectedSarsa`-Instanz erzeugt, welche wiederrum die Simulationsumgebung als Parameter entgegennehmen. Nach der Ausführung werden die Q-Tabellen und die während des Trainings aufgezeichneten Daten in einer Excel-Arbeitsmappe zur Auswertung gespeichert.

Das Modul `main` führt für alle ML-Algorithmen 5000 Episoden durch und speichert die Ergebnisse im Ordner `output` ab.

4.6 Aufbereitung der Ergebnisdaten

Im letzten Schritt werden die Ergebnisse der drei Algorithmen aufbereitet. Die vorliegenden Q-Tabellen enthalten jeweils die Einträge für alle Zustände. In den übrigen Aufzeichnungen sind die Anzahl Schritte, der erreichte Gewinn und der durchschnittliche Fehler je Episode aufgezeichnet.

In den Q-Tabellen sind auch Zustände enthalten, die eine Linie betreffen, welche sich schon auf einem Umleitungsweg befindet. Diese Zustände sind für ein ITCS im Produktivbetrieb nicht von Interesse. Das ITCS soll ausgehend von einer Linie und einem Störungsszenario in der Lage sein, direkt im ersten Schritt eine optimale Umleitung empfehlen. Von Interesse sind daher nur jene Zustände, bei denen explizit keine Umleitung gewählt wurde. Das Modul `resgen` bereinigt die Q-Tabellen entsprechend und gibt die Werte jeder Aktion ausgehend von einer Linie und einem Störungsszenario für alle drei verglichenen Algorithmen aus. Diese Ausgabe kann anschließend zum gezielten Vergleich genutzt werden.

Die Aufzeichnungen in den Excel-Arbeitsmappen enthalten unter Anderem den erreichten Gewinn je Episode. Bedingt durch die Exploration des Agenten kommt es bei diesen Gewinnen besonders zu Beginn des Trainings zu starken Schwankungen, welche in einem Liniendiagramm eher ein Rauschen, statt aufschlussreicher Informationen zeigen. Aus diesem Grund wird zur Bereinigung über die erreichten Gewinne jeweils ein gleitender Durschnitt über die letzten 50 Werte gebildet. Die folgende Abbildung zeigt auf der linken Seite die Originalgewinne, auf der rechten Seite die bereinigten Werte für das Q-Learning als Beispiel:

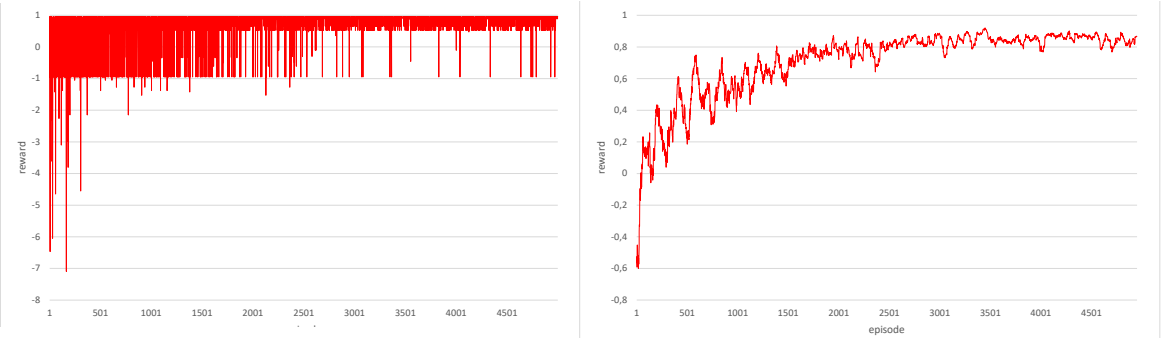


Abbildung 38: Vergleich zwischen originaler und bereinigter Lernkurve

Zwar kann bereits in der linken Darstellung eine Lernkurve erahnt werden, zum Vergleich mit den Lernkurven anderer Algorithmen eignet sich die rechte Darstellung jedoch besser. Einzig die durch die Exploration entstehenden Negativspitzen sind in der linken Darstellung besser zu erkennen.

5 Empirische Evaluation

In den vorherigen Kapiteln wurde die theoretische und prototypische Modellierung eines auf ML basierenden ITCS für Linienbusse beschrieben. In diesem Kapitel werden nun die drei ausgewählten Algorithmen Q-Learning, SARSA und Expected SARSA sowohl objektiv als auch subjektiv auf Basis der Ergebnisse aus dem Prototyp empirisch miteinander verglichen.

5.1 Vergleich der Lernkurven

Im ersten Schritt werden die Lernkurven der drei ML-Algorithmen miteinander verglichen. Als Lernkurve wird in diesem Fall der durchschnittliche Gewinn je Episode betrachtet. Auf diesem Weg kann schnell abgelesen werden, wie schnell ein Algorithmus lernt, welches Gewinnmaximum er erzielt und wie stabil die Ergebnisse sind. Das folgende Diagramm zeigt die Lernkurven von Q-Learning, SARSA und Expected SARSA:

Abbildung 39: Vergleich der Lernkurven von Q-Learning, SARSA und Exp-SARSA

Es ist zu erkennen, dass alle drei Algorithmen in der Lage sind, ein stabiles Ergebnis auf ähnlichem Weg zu erreichen. Ab der 3000. Episode sind die Ergebnisse von Q-Learning, SARSA und Expected SARSA auf etwa gleich hohem Niveau. Während Q-Learning weitestgehend stabil bleibt, kommt es bei SARSA und Expected SARSA zu leichten Schwankungen, die durch die Exploration zu erklären sind. Insgesamt lässt sich festhalten, dass sowohl das Gewinnlevel als auch die Lerngeschwindigkeit bei allen Algorithmen weitestgehend gleich sind.

In der nachfolgenden Detailansicht aus den ersten 2500 Episoden sind insbesondere bei der Lerngeschwindigkeit Unterschiede erkennbar:

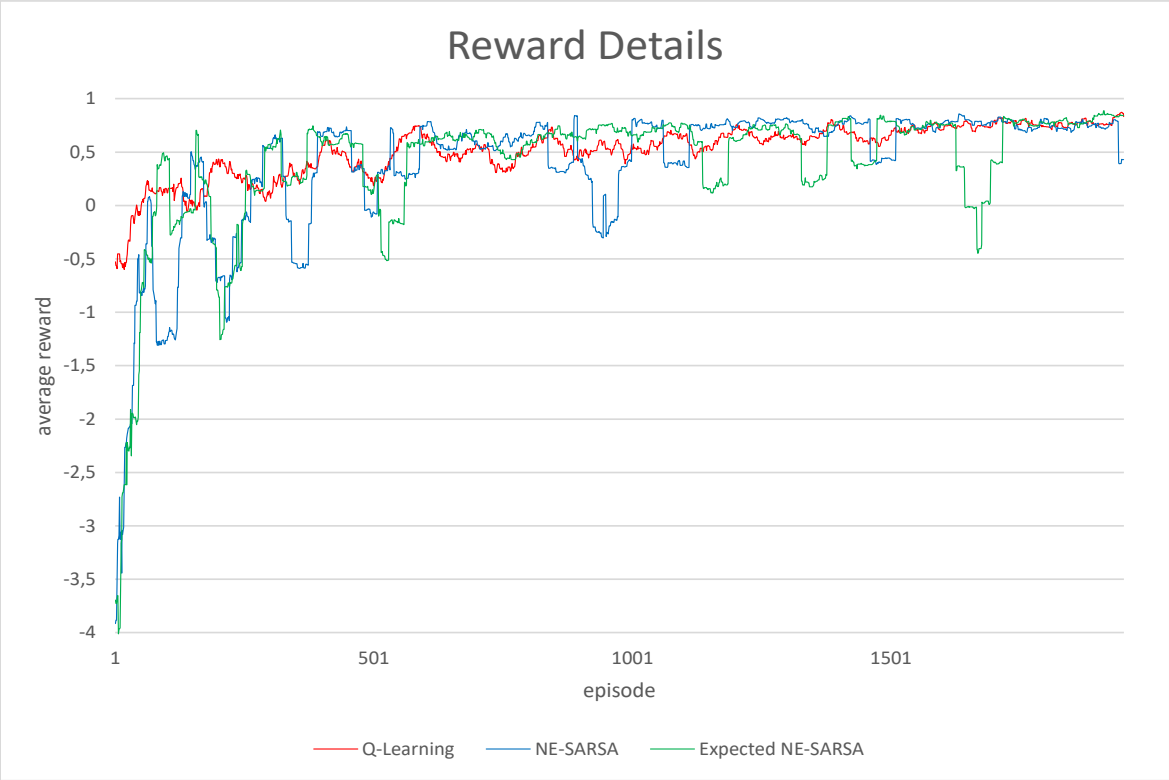


Abbildung 40: Detailansicht der Lernkurven von Q-Learning, SARSA und Exp-SARSA

Während Q-Learning direkt auf einem sehr hohen Gewinnlevel zu beginnen scheint, starten SARSA und Expected SARSA auf einem deutlichen tieferen Gewinnlevel in den ersten Episoden. Dieses Verhalten ist durch die vergleichsweise hohe Exploration in SARSA und Expected SARSA zu erklären. Bezogen auf die Geschwindigkeit ist die Lernkurve bei SARSA und Expected SARSA hingegen auch steiler. Die Exploration erklärt auch die starken Schwankungen im Vergleich zu Q-Learning. Es zeigt sich, dass Q-Learning im Trainingsprozess einer stetigen Verbesserung gleicht, während SARSA und Expected SARSA zwar auch einen deutlich erkennbaren Lernfortschritt zeigen, allerdings wesentlich stärker Explorieren. Ab etwa der 600. Episode erzeugen SARSA und Expected SARSA kurzfristig schneller stabilere

Ergebnisse als Q-Learning, dieser Unterschied ist aber eher unerheblich und könnte auch zufallsbedingt auftreten.

5.2 Vergleich der Aktionsbewertungen

Die in der Q-Tabelle gespeicherten Bewertungen der einzelnen Aktionen geben den Gewinn an, welcher bei Ausführen einer Aktion erwartet werden kann. In diesem Unterkapitel werden die Aktionsbewertungen, die von den Algorithmen erzeugt wurden, für die Störfallszenarien aus Kapitel 3.1 miteinander verglichen.

Die erste Tabelle zeigt die Aktionsbewertungen aus dem Stadt-Szenario für die Stadtbuslinie 2:

Aktion	Q-Learning	SARSA	Expected SARSA
Keine Umleitung	-0,5950	-1,2500	-1,3601
Stadt #1	0,4660	0,4697	0,4697
Stadt #2	0,6893	0,6893	0,6893
Stadt #3	0,0000	0,4384	0,4384
Stadt #4	0,6207	0,6217	0,6217
Land #1	0,0000	-1,0000	-1,0000
Land #2	-0,7809	-1,0000	-1,0000
Land #3	-0,9544	-1,0000	-1,0000
Land #4	-0,6198	-1,0000	-1,0000
Land #5	-1,0000	-1,0000	-1,0000
Vorort #1	-0,8716	-1,0000	-1,0000
Vorort #2	-1,24	-1,0000	-1,0000
Vorort #3	-0,6564	-1,0000	-1,0000
Vorort #4	-0,8155	-1,0000	-1,0000

Tabelle 3: Vergleich der Aktionsbewertungen für das Stadt-Szenario der Linie 2

Die geeigneten Umleitungen für das Stadtgebiet wurden von allen Algorithmen nahezu gleich bewertet. Bei der zweiten Aktion handelt es sich tatsächlich um jene Umleitung, welche aus betrieblicher Sicht auch am günstigsten ist. Es werden lediglich zwei Haltestellen ausgelassen und der kürzeste Weg zum Wiederreichen des regulären Linienweges gewählt. Eine eventuelle Verspätung wird dadurch minimiert. Weiter fällt auf, dass die Streuung der Bewertungen bei den ungeeigneten Umleitungen bei SARSA und Expected SARSA im Vergleich zum Q-Learning deutlich größer ist. Zu begründen ist dies mit der Exploration in SARSA und Expected SARSA.

Diese ist in Q-Learning nicht derart ausgeprägt, sodass beispielsweise die dritte Umleitung für das Stadtgebiet nie besucht und bewertet wurde. Die folgende Abbildung zeigt zum Abgleich den Regelfahrweg in blau und den gewählten Umleitungsfahrweg in rot:

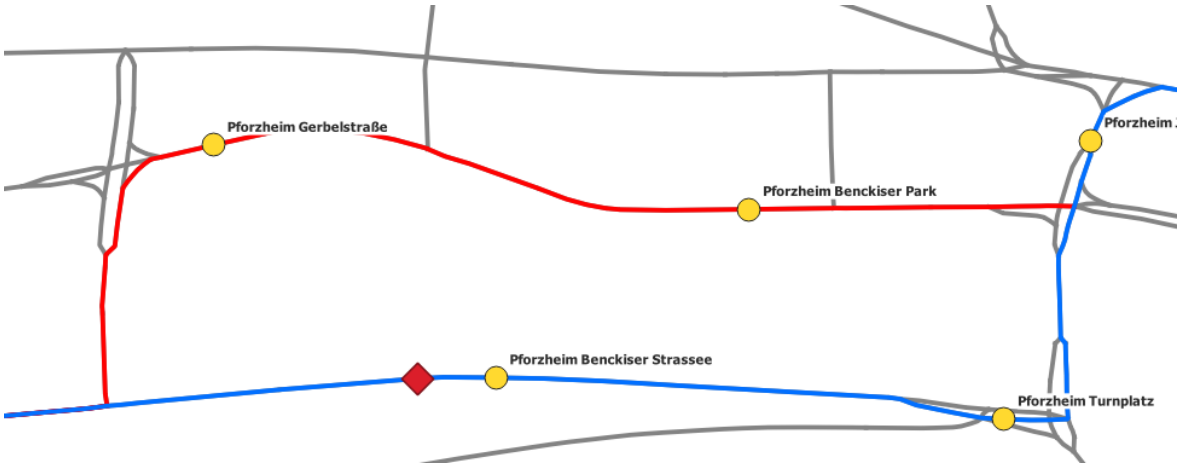


Abbildung 41: Optimale Umleitung der Linie 2 im Stadt-Szenario

Zusammenfassend lässt sich festhalten, dass alle Algorithmen in der Lage sind, den optimalen Umleitungsfahrweg für das Stadt-Szenario zu ermitteln, wenngleich Q-Learning eine zu geringe Exploration zeigt.

Die nächste Tabelle zeigt die Aktionsbewertungen für das Land-Szenario für die Regionalbuslinie 743:

Aktion	Q-Learning	SARSA	Expected SARSA
Keine Umleitung	-0,5044	-1,2500	-1,0709
Stadt #1	-0,6234	-1,2000	-1,0000
Stadt #2	-0,4434	-1,2000	-1,0000
Stadt #3	-0,4677	-1,2000	-1,0000
Stadt #4	-0,4398	-1,2000	-1,0000
Land #1	0,8582	0,8582	0,8582
Land #2	0,6857	0,6857	0,6857
Land #3	-1,2400	-1,2000	-1,2017
Land #4	-1,2000	-1,2000	-1,2018
Land #5	-1,2480	-1,2000	-1,2031
Vorort #1	-0,5738	-1,2000	-1,2033
Vorort #2	-0,6310	-1,2000	-1,2034
Vorort #3	-1,2000	-1,2000	-1,2110
Vorort #4	-1,0000	-1,2000	-1,2148

Tabelle 4: Vergleich der Aktionsbewertungen für das Land-Szenario der Linie 743

Auch in diesem Fall werden alle geeigneten Umleitungsfahrwege von den Algorithmen gleich gut bewertet. Korrekterweise wurde auch die erste Umleitung für das Überlandszenario für die Linie 743 gewählt. Dies entspricht dem kürzesten Fahrweg, es wird lediglich eine Haltestelle ausgelassen. Die folgende Abbildung zeigt zum Abgleich den Regelfahrweg in orange und den gewählten Umleitungsfahrweg in rot:

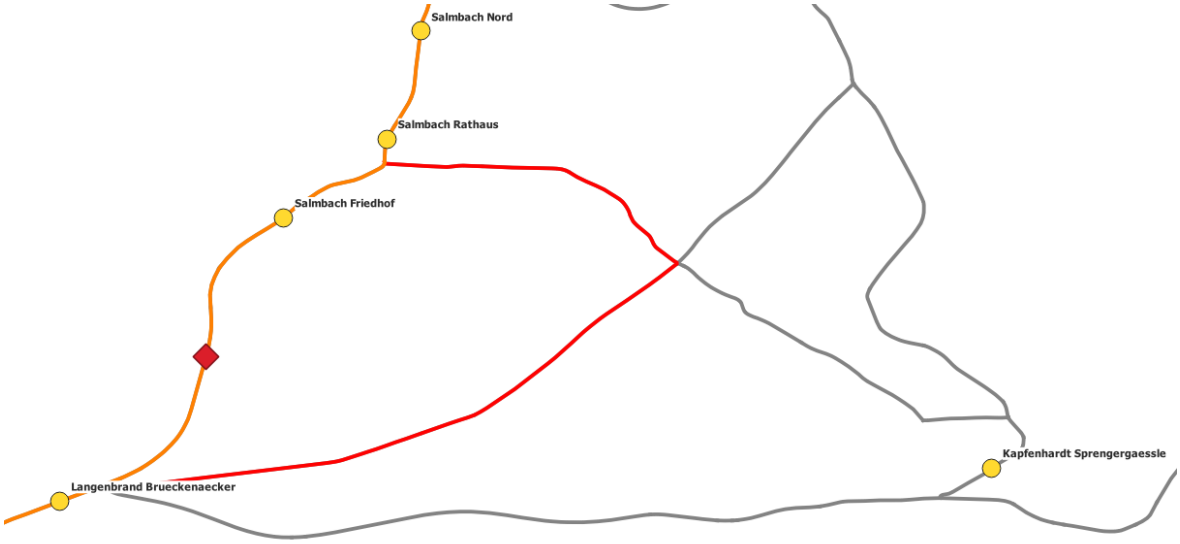


Abbildung 42: Optimale Umleitung der Linie 743 im Land-Szenario

Auch hier lässt sich festhalten, dass alle Algorithmen in der Lage sind, den optimalen Umleitungsfahrweg für das Land-Szenario zu ermitteln. Die stärkere Exploration führt bei SARSA und Expected SARSA wie erwartet hier zum eindeutigeren Ergebnis.

Die zweite Tabelle aus dem Land-Szenario zeigt die Aktionsbewertung für die Regionalbuslinie 744:

Aktion	Q-Learning	SARSA	Expected SARSA
Keine Umleitung	-0,1600	-0,5993	-1,3223
Stadt #1	-0,1503	-1,0000	-1,0000
Stadt #2	-0,3500	-1,0000	-1,0000
Stadt #3	-0,5861	-1,0000	-1,0000
Stadt #4	-1,2400	-1,0000	-1,0000
Land #1	-0,1589	-1,0000	-1,0000
Land #2	-0,4704	-1,0000	-1,0000
Land #3	0,8414	0,8414	0,8414
Land #4	1,1650	1,1650	1,1650
Land #5	0,8894	0,8966	0,8966
Vorort #1	-1,2500	-1,0000	-1,0000
Vorort #2	-1,0000	-1,0000	-1,0000
Vorort #3	0,0000	-1,0000	-1,0000
Vorort #4	-0,1450	-1,0000	-1,0000

Tabelle 5: Vergleich der Aktionsbewertungen für das Land-Szenario der Linie 744

Auffällig ist hier, dass die beste Aktion mit einem Wert > 1 bewertet wird. Korrekterweise handelt es sich dabei um die vierte Umleitung für das Land-Szenario. Der Regelfahrweg wird dabei schon nach der ersten Haltestelle in Kapfenhardt verlassen. Das stellt allerdings kein großes Problem dar, da trotzdem nur eine einzige Haltestelle ausgelassen wird, welche auch mit anderen verfügbaren Umleitungsfahrwegen nicht erreicht werden könnte. Da die gewählte Umleitung die gefahrene Strecke im Vergleich zum Regelfahrweg sogar verkürzt, kommt es zu einer Überbewertung der Umleitung, die jedoch keinen weiteren, negativen Einfluss hat. Wichtig ist in diesem Fall nur, dass die gewonnene Fahrzeit an der ersten geeigneten Haltestelle abgewartet wird, sodass es nicht zu einer Verfrühung kommt.

Die gewählte Umleitung ist zum Abgleich in der folgenden Abbildung in rot dargestellt, der Regelfahrweg ist in grün zu sehen:

Abbildung 43: Optimale Umleitung der Linie 744 im Land-Szenario

Wie auch bei beiden vorhergehenden Anwendungsfällen ist festzuhalten, dass alle Algorithmen in der Lage sind, die optimale Umleitung zu ermitteln.

Die letzte Tabelle zeigt die Aktionsbewertungen aus dem Vorort-Szenario für die Regionalbuslinie 715:

Aktion	Q-Learning	SARSA	Expected SARSA
Keine Umleitung	-0,3244	-1,1596	-1,0105
Stadt #1	-0,7765	-1,0000	-1,0000
Stadt #2	-1,0885	-1,0000	-1,0000
Stadt #3	-1,2499	-1,0000	-1,0000
Stadt #4	-1,0855	-1,0000	-1,0000
Land #1	-0,7764	-1,0000	-1,0000
Land #2	-0,8840	-1,0000	-1,0000
Land #3	-1,1318	-1,0000	-1,0000
Land #4	-1,0000	-1,0000	-1,0000
Land #5	-1,0809	-1,0000	-1,0000
Vorort #1	-0,7773	-1,0000	-1,0000
Vorort #2	0,4989	0,4989	0,4989
Vorort #3	1,0040	1,0040	1,0040
Vorort #4	0,1709	0,1780	0,1780

Tabelle 6: Vergleich der Aktionsbewertungen für das Vorort-Szenario der Linie 715

Korrekterweise wird hier die dritte Umleitung für das Vorort-Szenario mit einer ebenfalls leichten Überbewertung am höchsten bewertet. Die erste Umleitung aus dem Vorort-Szenario ist gänzlich ungeeignet, da sie nicht für eine Umfahrung des gesperrten Knotenpunktes sorgt. Folgerichtig wird sie auch entsprechend negativ bewertet. Auch im Übrigen zeigt sich dasselbe Ergebnis wie bei den drei vorhergehenden Vergleichen. Die folgende Abbildung zeigt den Regelfahrweg in grün, den Umleitungsfahrweg in rot:

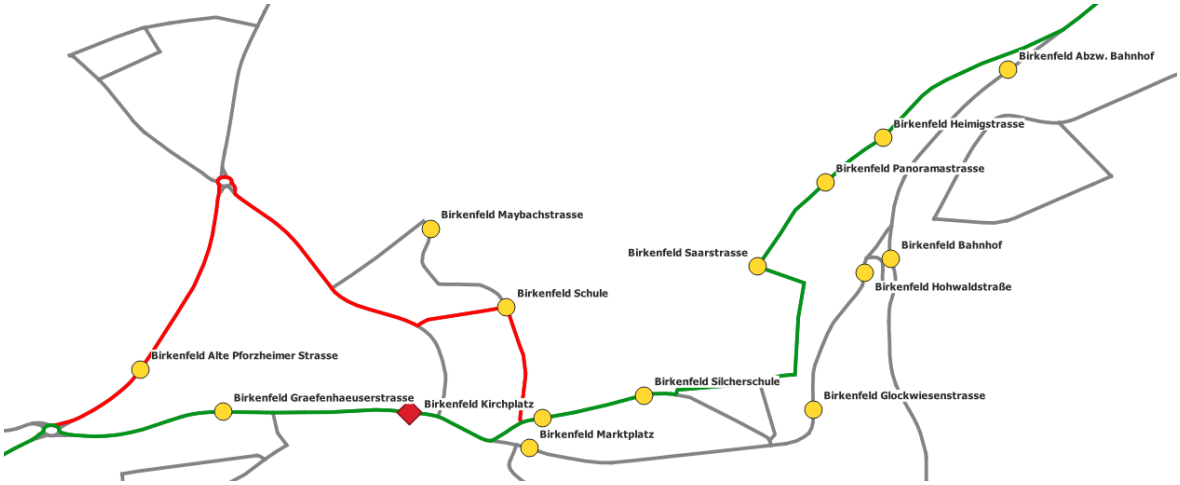


Abbildung 44: Optimale Umleitung der Linie 715 im Vorort-Szenario

Zusammenfassend lässt sich sagen, dass auch in diesem Szenario alle Algorithmen in der Lage sind, die den optimalen Umleitungsfahrweg auszuwählen.

5.3 Einfluss der $n\epsilon$-Greedystrategie

Im letzten Teil der Evaluation wird der Einfluss der $n\epsilon$-Greedystrategie auf das Explorationsverhalten von SARSA und Expected SARSA betrachtet. Die $n\epsilon$-Greedystrategie wird genutzt, um das Explorationsverhalten von SARSA und Expected SARSA dahingehend zu beeinflussen, dass nur noch die n besten Aktionen und all jene Aktionen, die noch nie genutzt wurden, in die Exploration der Algorithmen mit einbezogen werden. Ziel ist es also, auch im Rahmen der Exploration besonders schlechte Aktionen, respektive Umleitungsfahrwege, zu vermeiden.

Das folgende Diagramm zeigt den Verlauf von ϵ über die durchgeführten 5000 Episoden von Q-Learning und beiden SARSA-Implementierungen im Vergleich:

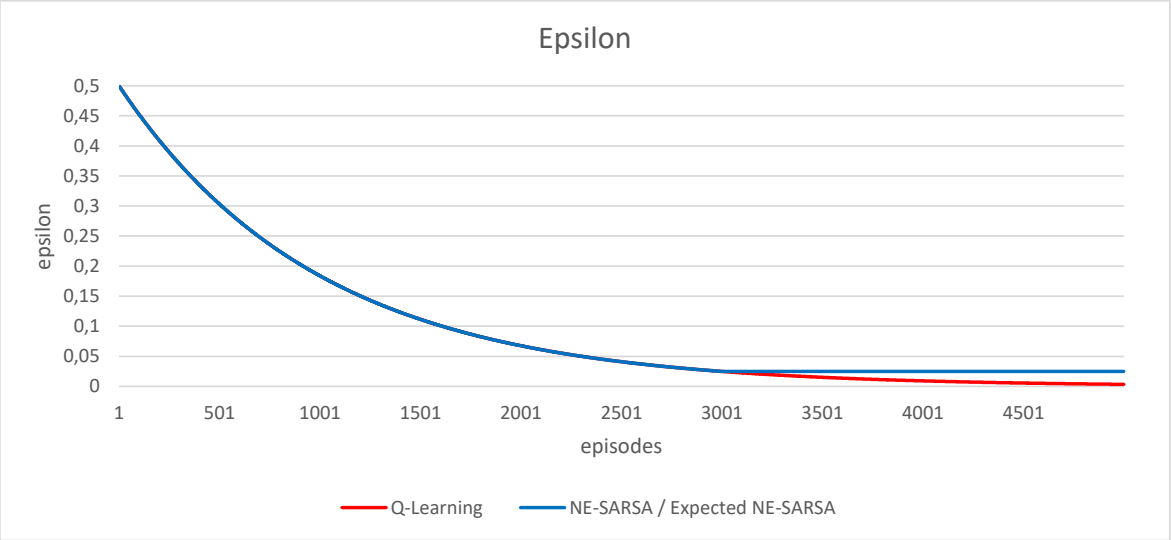


Abbildung 45: Vergleich der ϵ-Werte im Verlauf der On- und Off-Policy Algorithmen

Bei Q-Learning nimmt ϵ mit der Zerfallsrate stetig ab und strebt für unendlich viele Episoden gegen 0. Bei SARSA und Expected SARSA nimmt ϵ zunächst auch mit der Zerfallsrate ab, ist aber nach unten bei einem Wert von 0,025 beschränkt. Ab der 3000. Episode gehen die ϵ-Kurven von Q-Learning und beiden SARSA-Varianten daher auseinander. Damit exploriert Q-Learning ab er 3000. Episode weniger als SARSA und Expected SARSA. Im folgenden Diagramm sind die nicht gemittelten Gewinne zweier SARSA-Implementierungen zu sehen:

Abbildung 46: Vergleich einer ϵ-Greedystrategie und einer $n\epsilon$-Greedystrategie

Gut zu erkennen ist, dass sowohl die ϵ-Greedystrategie als auch die $n\epsilon$-Greedystrategie zu einer etwa gleich starken Exploration führen. Im Gegenzug ist aber auch zu sehen, dass die Gewinneinbrüche bei Verwendung der $n\epsilon$-Greedystrategie deutlich geringer ausfallen. Dies ist auf die eingeschränkte Aktionsauswahl zurückzuführen.

In der Praxis würde das bedeuten, dass das ITCS zwar nicht unbedingt die optimalste Umleitung wählt, allerdings auch keine Umleitung gewählt wird, welche grundsätzlich nicht geeignet ist oder gar keine Wirkung zeigt. Einzige Voraussetzung dazu ist, dass es mindestens so n geeignete Umleitungsfahrwege für jede Linie in jedem Szenario existieren, da ansonsten die Auswahlwahrscheinlichkeit einer gänzlich ungeeigneten Umleitung zwar minimiert, aber niemals ganz ausgeschlossen wird. Dem Hyperparameter n kommt damit bei der Konfiguration des ITCS eine hohe Bedeutung zu.

6 Kritische Diskussion und Zusammenfassung

7 Literaturverzeichnis

Alaliyat, Saleh (2022): Video -based Fall Detection in Elderly's Houses.

Arnold, Jakob (2021): Reinforcement Learning für ein Umplanungsproblem im Scheduling. Masterarbeit. Karl-Franzens-Universität Graz, Graz. Institut für Operations und Information Systems. Online verfügbar unter https://unipub.uni-graz.at/obvugrhs/download/pdf/6690076?originalFilename=true, zuletzt geprüft am 11.12.2022.

Arruda, Carlos E.; Moraes, Pedro F.; Agoulmine, Nazim; Martins, Joberto S. B. (2020): Enhanced Pub/Sub Communications for Massive IoT Traffic with SARSA Reinforcement Learning.

Böhm, Robin (2016): Aktionen autonomer Systeme als Elemente relationaler nebenläufiger Markov-Entscheidungsprozesse. Diplomarbeit. Universität Stuttgart, Stuttgart. Institut für Parallele und Verteilte Systeme. Online verfügbar unter https://elib.uni-stuttgart.de/handle/11682/9652, zuletzt geprüft am 14.11.2022.

Brezina, Tadej; Emberger, Günter; Rollinger, Wolfgang (2012): Vom Beschwerde- zum Anregungsmanagement im Österreichischen öffentlichen Verkehr. Online verfügbar unter https://d1wqtxts1xzle7.cloudfront.net/34231246/brezina-emberger-2012_anregungsmanagement-with-cover-page-v2.pdf?Expires=1668671995&Signature=DBnAREvXAnDgUHEoCOKurfqbkrq7bAri9Iuu3tJrb8wVptyn2a2WV8EhtbrPKyE3ES1YLBy5TUbvl2BvXzbHPT2B42AGGf6dF77CsQ9eJM9vJmE7AzKyH0RIZSuRTNuZacyvi-Jix~X2h8-Qzn0OAa5kk-9Ff66q6HYuiqRAc501ywUSy~gicq7au5rPkl0udvjb4jiAEJTgT69xwoUoLFyQoWI~KU2tw1azHEtkNXEQB0ekoGP0cpeSo9NHmKWGx5cNz72OatR0tYZBo2xGN~WMFWuMvPkywHiKUo2

EqqvLPCggkop8bZrhXMzv4SWybMJv3QDpOhkgX9AcD2v63Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA, zuletzt geprüft am 17.11.2022.

Bundesministerium für Verkehr: Personenbeförderungsgesetz. PBefG. Online verfügbar unter https://www.gesetze-im-internet.de/pbefg/index.html#BJNR002410961BJNE001802305, zuletzt geprüft am 18.11.2022.

Chan, Leong; Hogaboam, Liliya; Cao, Renzhi (2022): Applied Artificial Intelligence in Business. Cham: Springer International Publishing.

Dammann, Patrick (o.D.): Einführung in das Reinforcement Learning. Heidelberg Collaboratory for Image Processing (HCI). Heidelberg. Online verfügbar unter https://hci.iwr.uni-heidelberg.de/system/files/private/downloads/541645681/dammann_reinfocement-learning-report.pdf, zuletzt geprüft am 24.11.2022.

DELFI (2020): Entwicklung von Instrumenten zur Umsetzung der delegierten Verordnung (EU) 2017/1926 (ÖV-Daten für NAP). Schlussbericht zum FE-Projekt: 70.0946. Verein zur Förderung einer durchgängigen elektronischen Fahrgastinformation – DELFI e.V. Frankfurt a.M. Online verfügbar unter https://fops.de/wp-content/uploads/2020/09/70.0946_Schlussber.OeV-Daten-f.d.NAP_.pdf, zuletzt geprüft am 24.11.2022.

Engfer, Andreas (2002): Fuzzy Logik. Ausarbeitung zur Vorlesung Ausarbeitung zur Vorlesung „Methoden der Künstlichen Intelligenz“. Ausarbeitung. Fachhochschule Furtwangen, Furtwangen. Fachbereich Wirtschaftsinformatik.

Exner, Jan-Philipp (2012): SURVEY BEFORE PLAN 2.0. Neue Ansätze von Raumsensorik und Monitaring für die Raumplanung. In: *PLANERIN* (5/12), S. 24–26.

Findl, Renate; Dahmen-Zimmer, Katharina; Kostka, Markus; Zimmer, Alf (2022): Nutzerorientierte Systementwicklung für den ÖPNV. Paper. Universität Regensburg, Regensburg. Lehrstuhl für Experimentelle und Angewandte Psychologie. Online verfügbar unter https://psydok.psycharchives.de/jspui/bitstream/20.500.11780/3433/1/findl.pdf, zuletzt geprüft am 17.11.2022.

Gass, Saul I.; Fu, Michael (Hg.) (2013): Encyclopedia of operations research and management science. Third edition. New York, NY: Springer Science + Business Media (Springer Reference).

Google (2022): GTFS-Static Überblick. Hg. v. Google. Online verfügbar unter https://developers.google.com/transit/gtfs?hl=de, zuletzt aktualisiert am 06.10.2022, zuletzt geprüft am 18.11.2022.

Herzner, Alexander; Schmidpeter, René (Hg.) (2022): CSR in Süddeutschland. Unternehmerischer Erfolg und Nachhaltigkeit im Einklang. Springer-Verlag GmbH. Berlin, Heidelberg: Springer Gabler (Management-Reihe Corporate Social Responsibility). Online verfügbar unter http://www.springer.com/.

Huber, Tobias (2018): Tiefes bestärkendes Lernen: Grundlagen, Approximationseigenschaft und Implementierung multimodaler Erklärungen. Masterarbeit. Universität Augsburg, Augsburg.

Josi, Damian (2020): Qualität von OpenStreetMap-Daten. Einführung und Arten der Qualitätsbeurteilung. Paper. Universität Bern, Bern. Institut für Wirtschaftsinformatik. Online verfügbar unter https://www.digitale-nachhaltigkeit.unibe.ch/unibe/portal/fak_naturwis/a_dept_math/c_iinfamath/abt_digital/content/e90958/e490158/e900462/e977579/e977582/e980453/OpenData2020_DamianJosi_Vertiefungsartikel_ger.pdf, zuletzt geprüft am 18.11.2022.

KARL (2021): Kompetenzzentrum KARL - Künstliche Intelligenz für Arbeit und Lernen in der Region Karlsruhe. Hg. v. Kompetenzzentrum KARL. Online verfügbar unter https://kompetenzzentrum-karl.de/, zuletzt geprüft am 08.01.2023.

Krämer, Andreas (2022): exeo OpinionTRAIN. 9-Euro-Ticket: Blick zurück und nach vorne - Nutzerprofil, Nutzung und Bewertungen. exeo Strategic Consulting AG. Bonn, 08.09.2022. Online verfügbar unter https://www.researchgate.net/publication/363417545_exeo_OpinionTRAIN_IV_9_EUR_Ticket_Post_I_220907, zuletzt geprüft am 10.09.2022.

Larsson, Emil (2018): Evaluation of Pretraining Methods for Deep Reinforcement Learning. Universität Uppsala, Uppsala.

Levy, Sharon; Xiong, Wenhan; Belding, Elizabeth; Wang, William Yang (2018): SafeRoute: Learning to Navigate Streets Safely in an Urban Environment.

Liggieri, Kevin; Müller, Oliver (2019): Mensch-Maschine-Interaktion. Stuttgart: J.B. Metzler.

Lorenz, Uwe (2020): Reinforcement Learning. Berlin, Heidelberg: Springer Berlin Heidelberg.

Lüth, Carsten (2019): A Review of: Human-Level Control through deep Reinforcement Learning. Seminararbeit. Universität Heidelberg, Heidelberg. Department of Computer Science. Online verfügbar unter https://hci.iwr.uni-heidelberg.de/system/files/private/downloads/213797145/report_carsten_lueth_human_level_control.pdf, zuletzt geprüft am 13.11.2022.

Mainzer, Klaus (2019): Künstliche Intelligenz – Wann übernehmen die Maschinen? Berlin, Heidelberg: Springer Berlin Heidelberg.

Meier, Andreas; Portmann, Edy (2016): Smart City. Wiesbaden: Springer Fachmedien Wiesbaden.

Nahrstedt, Harald (2012): Algorithmen für Ingenieure. Wiesbaden: Vieweg+Teubner Verlag.

Niebler, Paul (2018): Datenbasiert Entscheiden. Ein Leitfaden Für Unternehmer und Entscheider. Unter Mitarbeit von Dominic Lindner. Wiesbaden: Gabler (Essentials Ser). Online verfügbar unter https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5566817.

OSM-Highway (2022): Erklärung zum Schlüssel HIGHWAY. Hg. v. OpenStreetMap Community. Online verfügbar unter https://wiki.openstreetmap.org/wiki/Key:highway, zuletzt aktualisiert am 26.10.2022, zuletzt geprüft am 18.11.2022.

OSM-PSV (2022): Erklärung zum Schlüssel PSV. Hg. v. OpenStreetMap Community. Online verfügbar unter https://wiki.openstreetmap.org/wiki/DE:Key:psv?uselang=de, zuletzt aktualisiert am 29.08.2022, zuletzt geprüft am 18.11.2022.

Pong, Vitchyr; Gu, Shixiang; Dalal, Murtaza; Levine, Sergey (2018): Temporal Difference Models: Model-Free Deep RL for Model-Based Control.

Reinhardt, Winfried (2018): Öffentlicher Personennahverkehr. Technik – rechts- und betriebswirtschaftliche Grundlagen. 2., aktualisierte Auflage. Wiesbaden: Springer Vieweg. Online verfügbar unter http://www.springer.com/.

Rellensmann, Johanna (2019): Mathematisches Modellieren. In: Johanna Rellensmann (Hg.): Selbst erstellte Skizzen beim mathematischen Modellieren. Wiesbaden: Springer Fachmedien Wiesbaden (Studien zur theoretischen und empirischen Forschung in der Mathematikdidaktik), S. 5–30.

Rimscha, Markus von (2008): Algorithmen kompakt und verständlich. Lösungsstrategien am Computer. 1. Aufl. Wiesbaden: Vieweg + Teubner (Programmiersprachen, Datenbanken und Softwareentwicklung).

Schaaf, Marc; Wilke, Gwendolin (2015): Ereignisverarbeitung zur Flexiblen Dynamischen Informationsverarbeitung in Smart Cities. In: *HMD* 52 (4), S. 562–571. DOI: 10.1365/s40702-015-0149-x.

Scherm, Jürgen; Hübener, Reinhard; Dobeschinsky, Harry; Kühne, Reinhart D. (2001): Opti*Bus : Optimierungschancen für das Verkehrssystem Bus im ÖPNV ; Ergebnisse des Kongresses im Themenbereich Verkehr und Raumstruktur. Unter Mitarbeit von Universität Stuttgart.

Schmitz, Martin (2017): Ein Vergleich von Reinforcement Learning Algorithmen für dynamische und hochdimensionale Probleme. Bachelorarbeit. Universität Koblenz / Landau, Koblenz. Institute for Web Science and Technology. Online verfügbar unter https://west.uni-

koblenz.de/assets/theses/vergleich-von-reinforcement-algorithmen.pdf, zuletzt geprüft am 13.11.2022.

Schranil, Steffen (2013): Prognose der Dauer von Störungen des Bahnbetriebs. ETH Zurich.

Searle, John R. (1980): Minds, brains, and programs. In: *Behav Brain Sci* 3 (3), S. 417–424. DOI: 10.1017/S0140525X00005756.

Simeonov, Georgi (2017): Ein interaktiver visueller Ansatz für das Map Matching von großen Bewegungsdatensätzen. Bachelorarbeit. Universität Stuttgart, Stuttgart. Institut für Visualisierung und Interaktive Systeme. Online verfügbar unter https://www2.informatik.uni-stuttgart.de/bibliothek/ftp/medoc.ustuttgart_fi/BCLR-2017-62/BCLR-2017-62.pdf, zuletzt geprüft am 23.11.2022.

Soike, Roman; Libbe, Jens (2018): Smart Cities in Deutschland. Eine Bestandsaufnahme. Hg. v. Deutsches Institut für Urbanistik. Berlin (Difu-Papers). Online verfügbar unter https://repository.difu.de/jspui/handle/difu/248050, zuletzt geprüft am 23.11.2022.

Statistisches Bundesamt (2020): Einwohnerzahl der größten Städte in Deutschland am 31. Dezember 2020. Hg. v. Statistisches Bundesamt. Online verfügbar unter https://de.statista.com/statistik/daten/studie/1353/umfrage/einwohnerzahlen-der-grossstaedte-deutschlands/, zuletzt geprüft am 23.11.2022.

Sutton, Richard S.; Barto, Andrew (2018): Reinforcement learning. An introduction. Second edition. Cambridge, Massachusetts, London, England: The MIT Press (Adaptive computation and machine learning).

U-THREAT (2021): U-THREAT. Resilienz unterirdischer ÖPNV-Systeme zur Gewährleistung der Verfügbarkeit. Hg. v. VDI Technologiezentrum GmbH. Düsseldorf.

van Seijen, Harm; van Hasselt, Hado; Whiteson, Shimon; Wiering, Marco (2009): A theoretical and empirical analysis of Expected Sarsa. In: 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning. 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). Nashville, TN, USA, 30.03.2009 - 02.04.2009: IEEE, S. 177–184.

VDV-Schrift 730, 2015: VDV-Schrift 730. Online verfügbar unter https://knowhow.vdv.de/documents/730/, zuletzt geprüft am 07.12.2022.

VDV-Schrift 736-2, 2019: VDV-Schrift 736-2. Online verfügbar unter https://www.vdv.de/736-2-sds.pdfx, zuletzt geprüft am 24.11.2022.

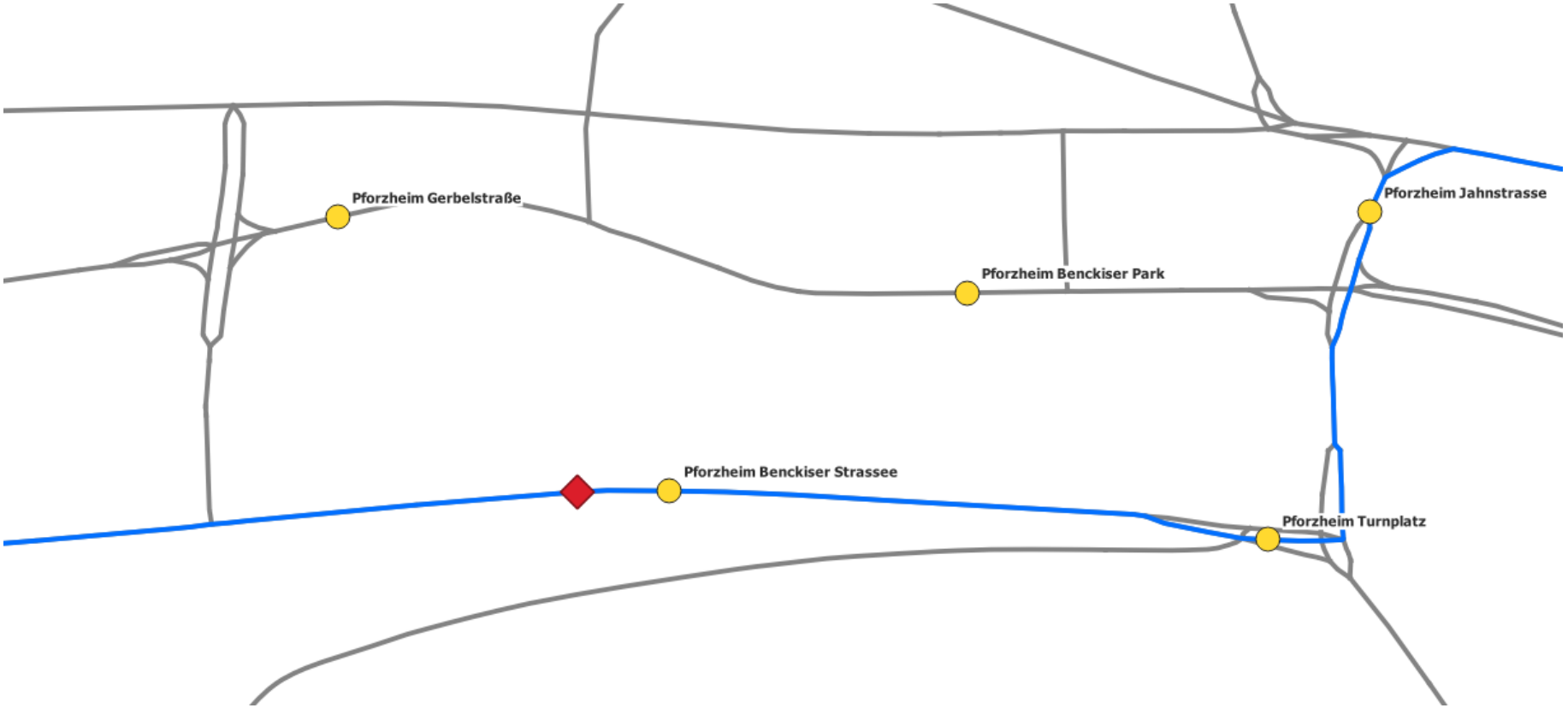
Wagner, Stefan Sylvius (2018): Entwicklung eines Reinforcement Learning basierten Flugzeugautopiloten unter der Verwendung von Deterministic Policy Gradients. Bachelorarbeit. Hochschule für Angewandte Wissenschaften Hamburg, Hamburg. Online

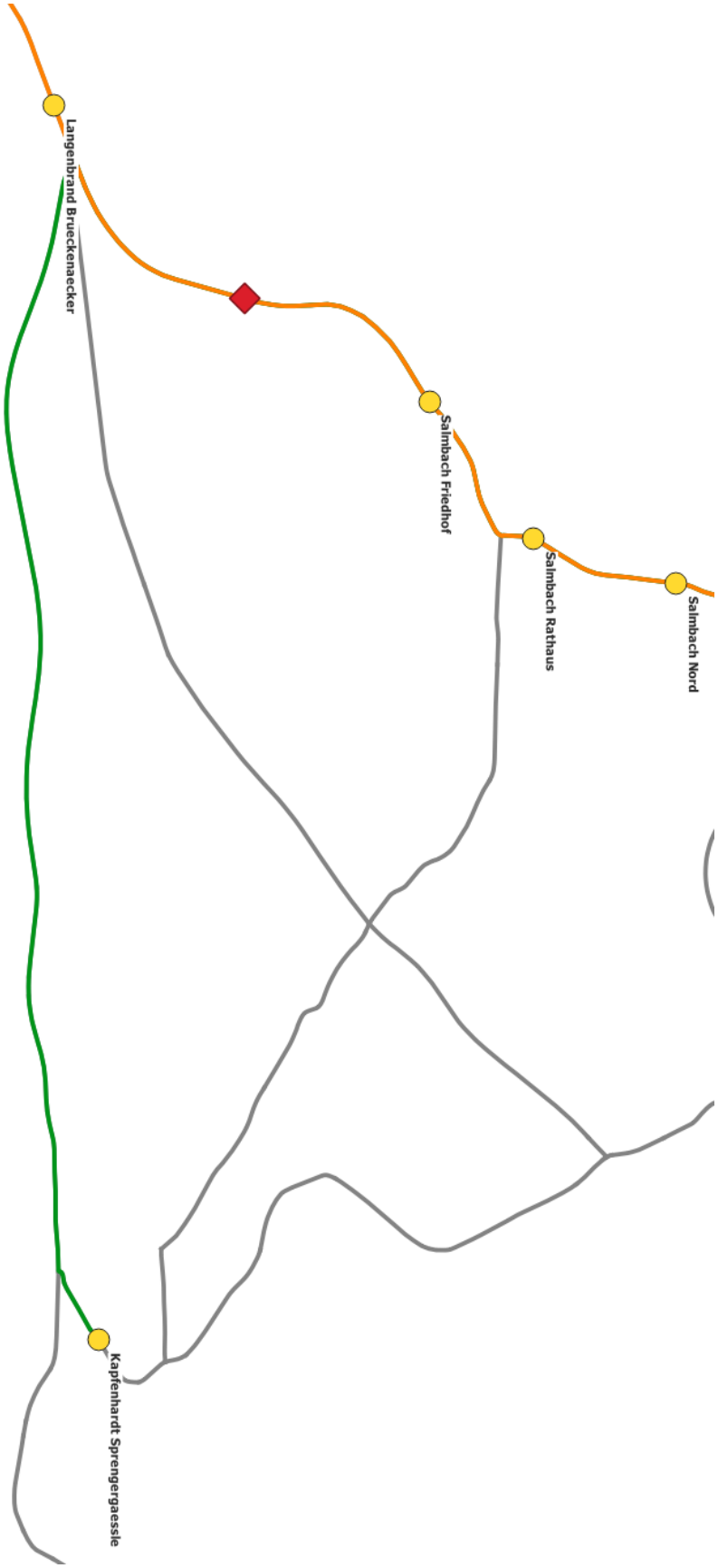
verfügbar unter https://autosys.informatik.haw-hamburg.de/papers/2018Wagner.pdf, zuletzt geprüft am 07.12.2022.

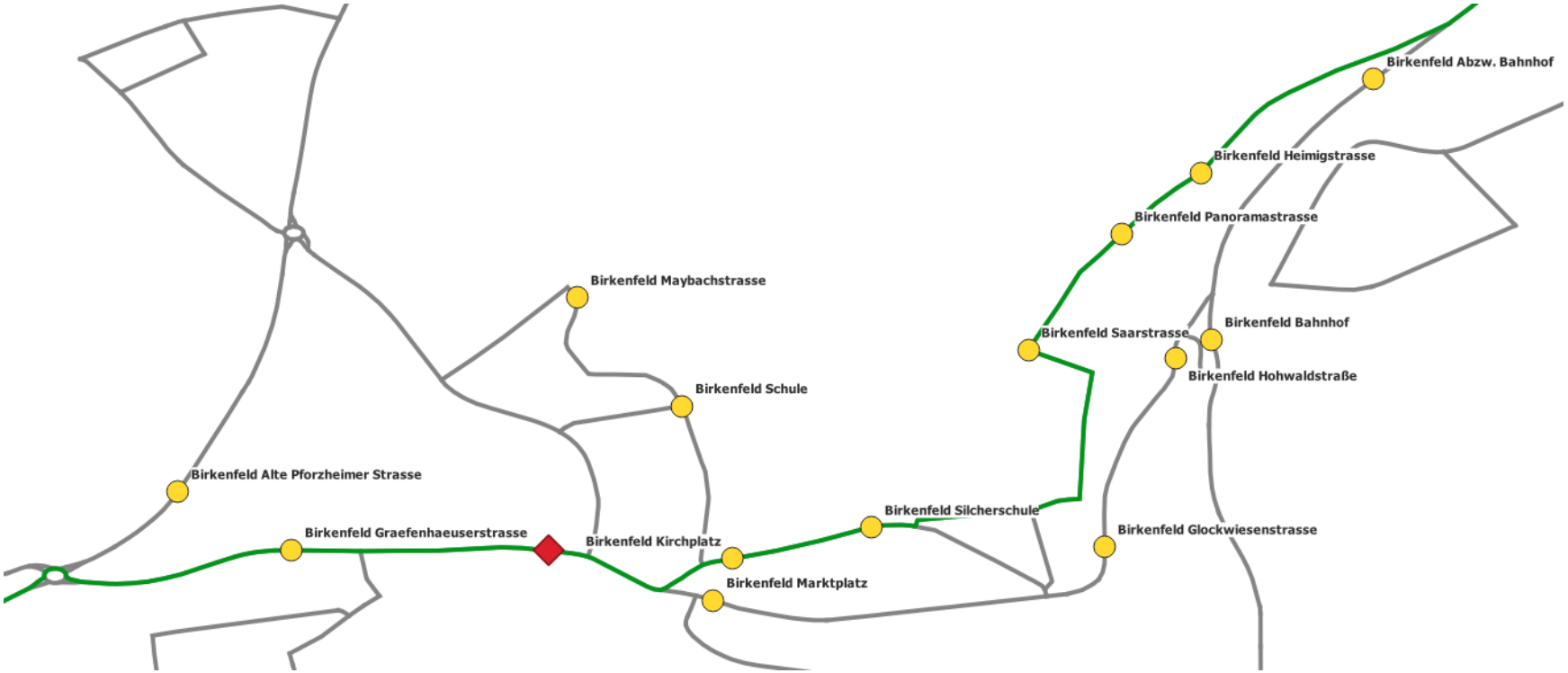
Williams, Ronald J. (1992): Simple statistical gradient-following algorithms for connectionist reinforcement learning. In: *Mach Learn* 8 (3-4), S. 229–256. DOI: 10.1007/BF00992696.

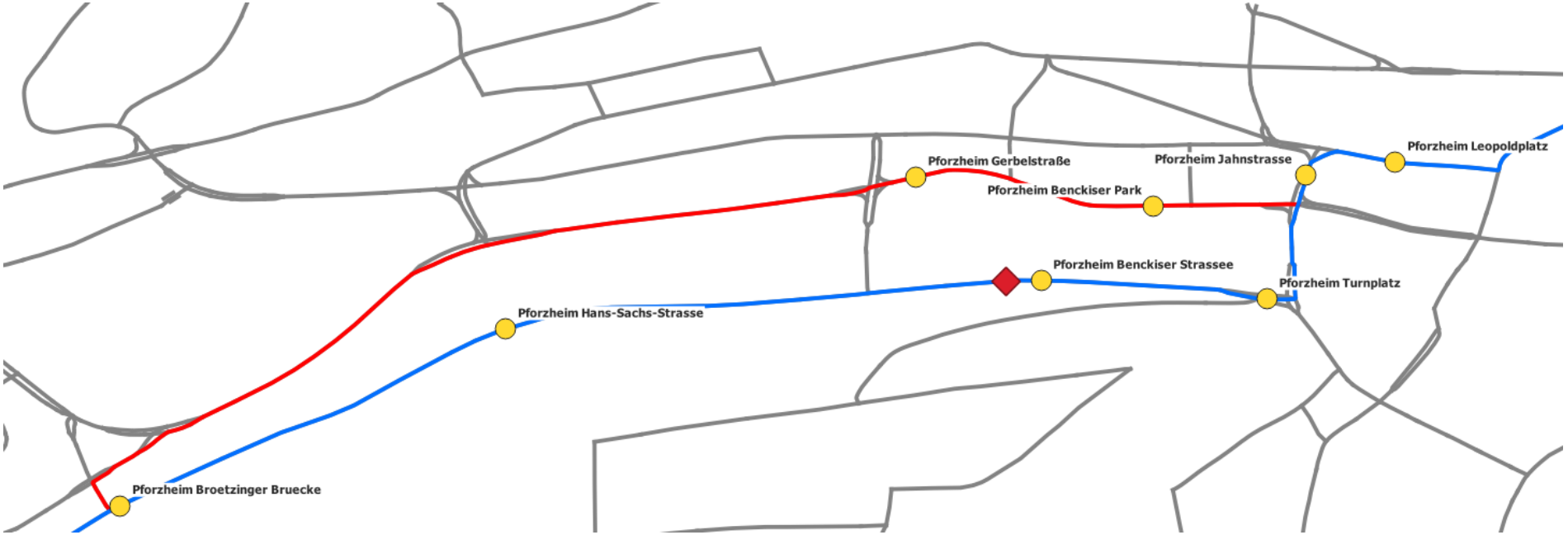
Witt, Laura Jasmin (2019): Evaluierung von Reinforcement Learning Algorithmen zur Erweiterung eines bestehenden Trajektorienfolgeregelungskonzeptes. Masterarbeit. Freie Universität Berlin, Berlin. Online verfügbar unter https://www.mi.fu-berlin.de/inf/groups/ag-ki/Theses/Completed-theses/Master_Diploma-theses/2019/Witt/MA-Witt.pdf, zuletzt geprüft am 14.12.2022.

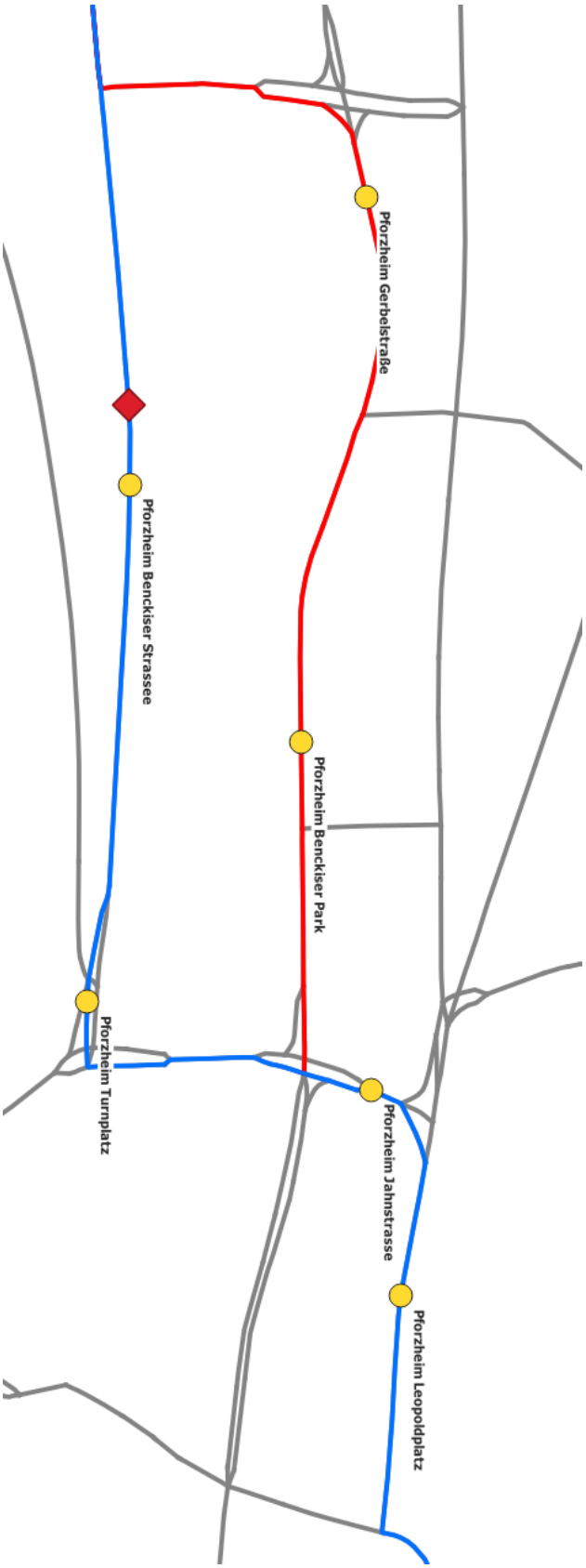
Anhang

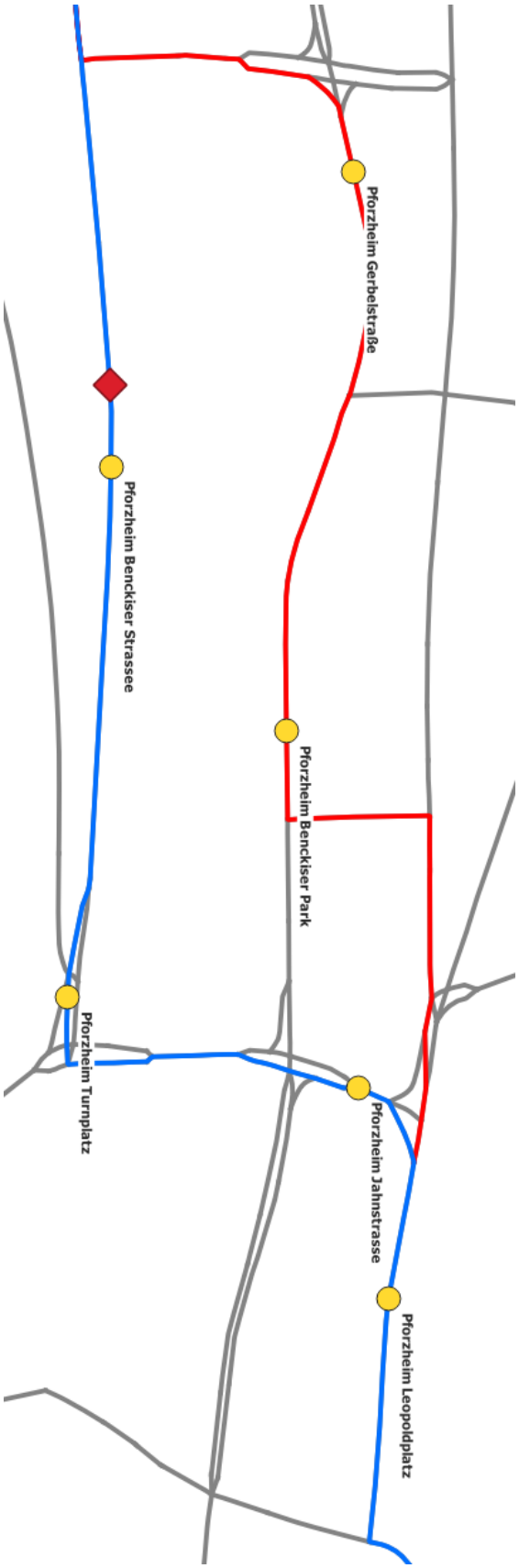

Pforzheim Gerbelstraße
Pforzheim Jahnstrasse
Pforzheim Benckiser Park
Pforzheim Benckiser Strassee
Pforzheim Turnplatz

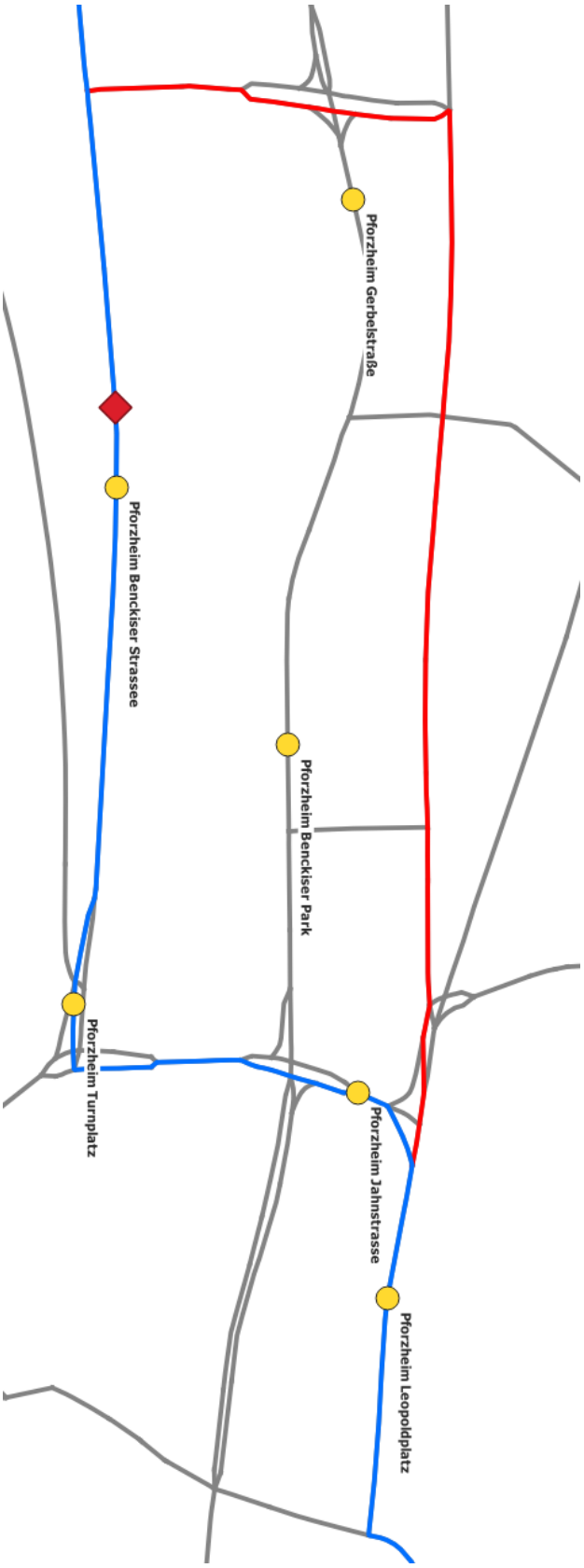

Langenbrand Brueckenaecker
Salmbach Friedhof
Salmbach Rathaus
Salmbach Nord
Kapfenhardt Sprengergaessle


Birkenfeld Abzw. Bahnhof
Birkenfeld Heimigstrasse
Birkenfeld Panoramastrasse
Birkenfeld Maybachstrasse
Birkenfeld Saarstrasse
Birkenfeld Bahnhof
Birkenfeld Hohwaldstraße
Birkenfeld Schule
Birkenfeld Alte Pforzheimer Strasse
Birkenfeld Silcherschule
Birkenfeld Glockwiesenstrasse
Birkenfeld Graefenhaeuserstrasse
Birkenfeld Kirchplatz
Birkenfeld Marktplatz


Pforzheim Gerbelstraße
Pforzheim Jahnstrasse
Pforzheim Leopoldplatz
Pforzheim Benckiser Park
Pforzheim Benckiser Strassee
Pforzheim Turnplatz
Pforzheim Hans-Sachs-Strasse
Pforzheim Broetzinger Bruecke


Pforzheim Gerberstraße
Pforzheim Bencksiser Strassee
Pforzheim Bencksiser Park
Pforzheim Turnplatz
Pforzheim Jahnstrasse
Pforzheim Leopoldplatz


Pforzheim Gerbelstraße
Pforzheim Benckiser Strassee
Pforzheim Benckiser Park
Pforzheim Turnplatz
Pforzheim Jahnstrasse
Pforzheim Leopoldplatz


Pforzheim Gerbelstraße
Pforzheim Benckiser Strassee
Pforzheim Benckiser Park
Pforzheim Turnplatz
Pforzheim Jahnstrasse
Pforzheim Leopoldplatz

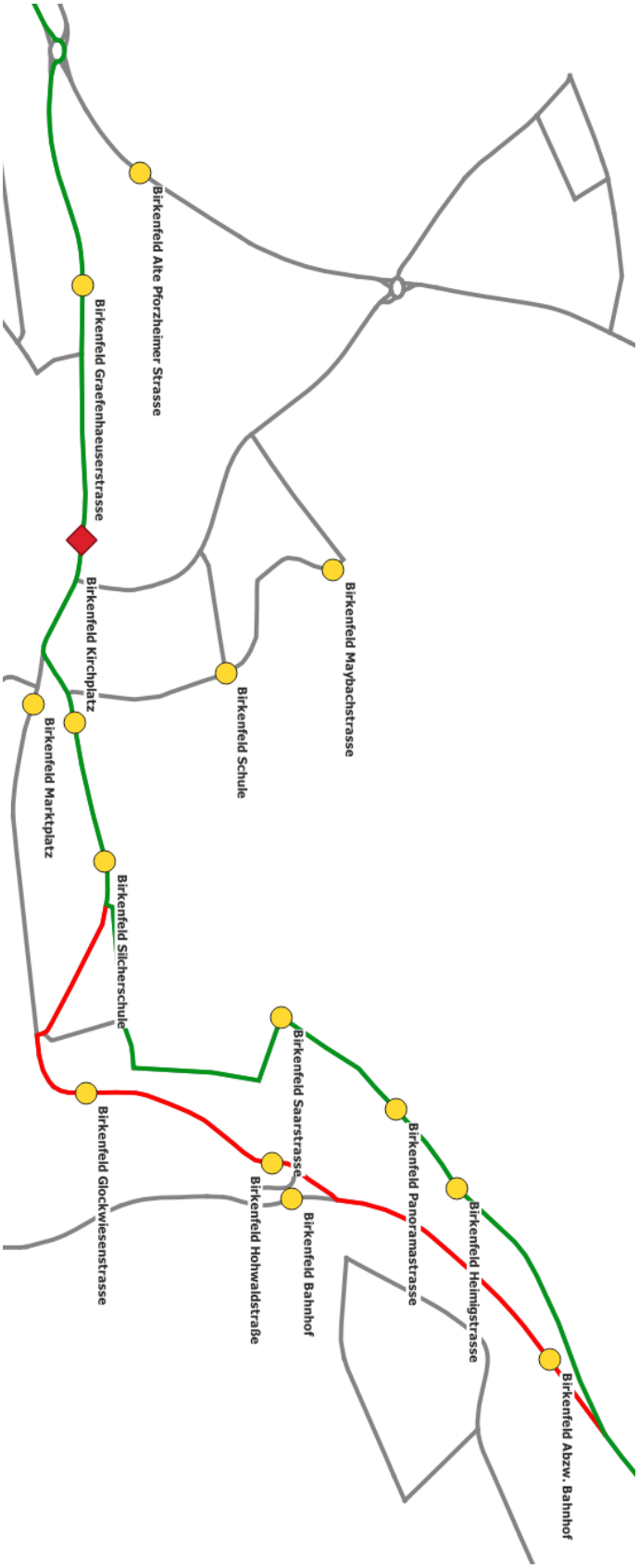

Langenbrand Brueckenaecker
Salmbach Friedhof
Salmbach Rathaus
Salmbach Nord
Kapfenhardt Sprengergaessle

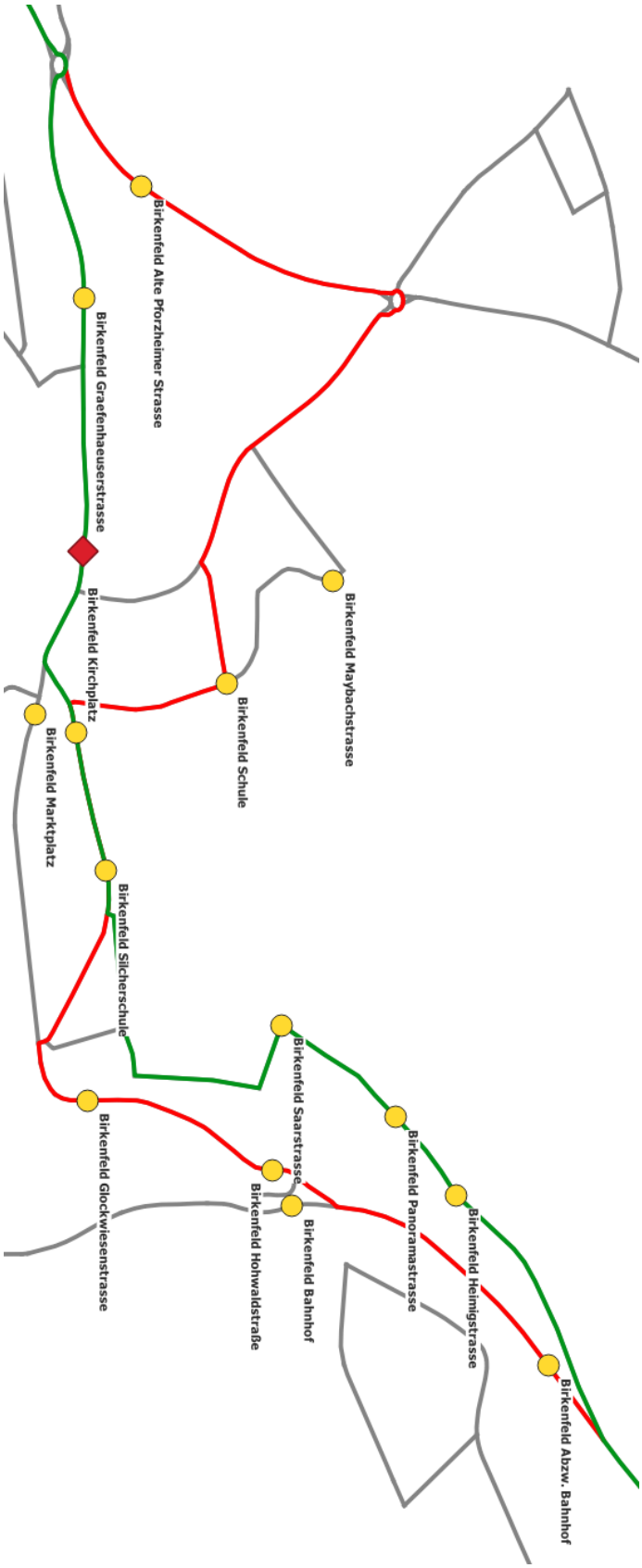

Langenbrand Brueckenaecker
Salmbach Friedhof
Salmbach Rathaus
Salmbach Nord
Kapfenhardt Sprengergaessle

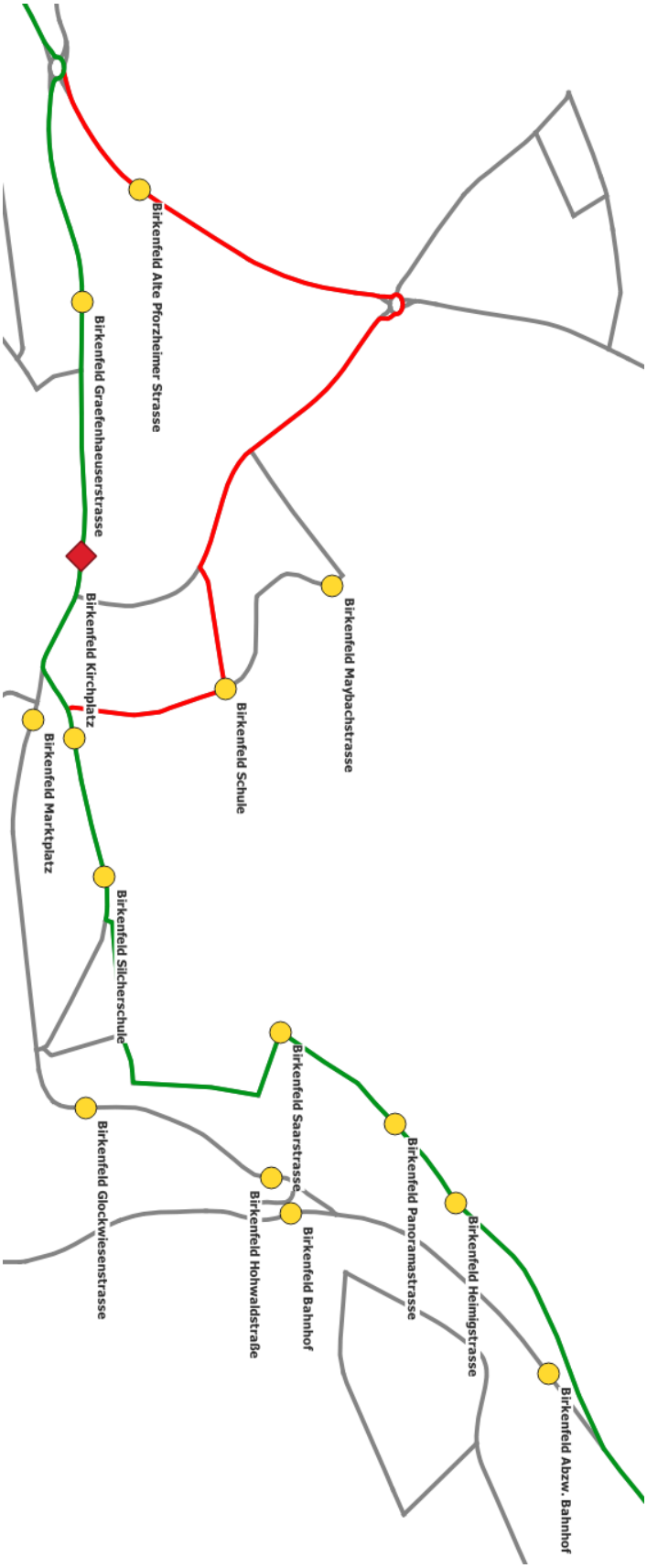

Langenbrand Brueckenaecker
Salmbach Friedhof
Salmbach Rathaus
Salmbach Nord
Kapfenhardt Sprengergaessle

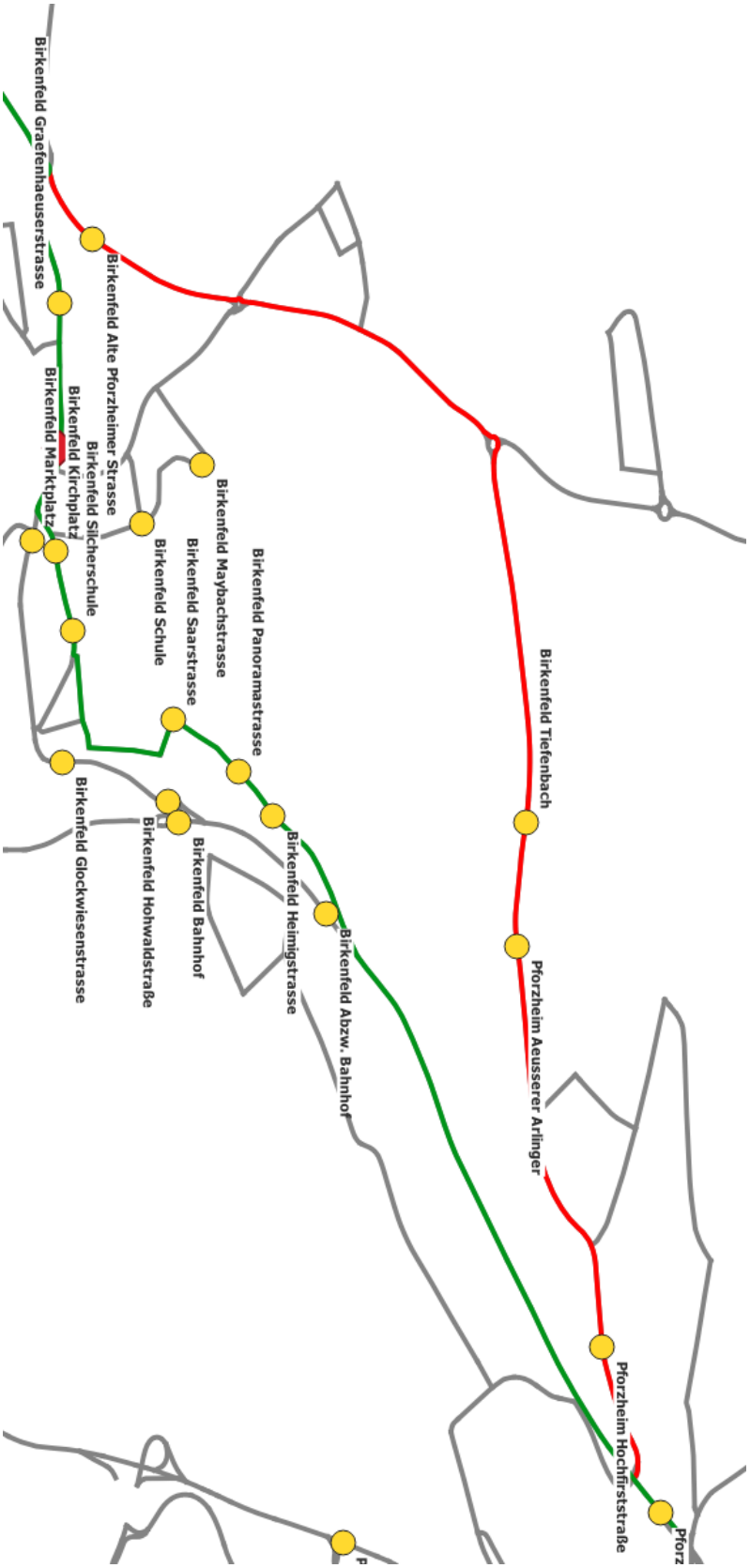

Langenbrand Brueckenaecker
Salmbach Friedhof
Salmbach Rathaus
Salmbach Nord
Kapfenhardt Sprengergaessle


Langenbrand Brueckenaecker
Salmbach Friedhof
Salmbach Rathaus
Salmbach Nord
Kapfenhardt Sprengergaessle


Birkenfeld Alte Pforzheimer Strasse
Birkenfeld Graefenhaeuserstrasse
Birkenfeld Kirchplatz
Birkenfeld Maybachstrasse
Birkenfeld Schule
Birkenfeld Marktplatz
Birkenfeld Silcherschule
Birkenfeld Saarstrasse
Birkenfeld Panoramastrasse
Birkenfeld Heinigstrasse
Birkenfeld Glockwiesenstrasse
Birkenfeld Hohwaldstraße
Birkenfeld Bahnhof
Birkenfeld Abzw. Bahnhof


Birkenfeld Alte Pforzheimer Strasse
Birkenfeld Graefenhaeuserstrasse
Birkenfeld Maybachstrasse
Birkenfeld Kirchplatz
Birkenfeld Schule
Birkenfeld Marktplatz
Birkenfeld Silcherschule
Birkenfeld Saarstrasse
Birkenfeld Glockwiesenstrasse
Birkenfeld Panoramastrasse
Birkenfeld Hohwaldstraße
Birkenfeld Bahnhof
Birkenfeld Heimigstrasse
Birkenfeld Abzw. Bahnhof


Birkenfeld Alte Pforzheimer Strasse
Birkenfeld Graefenhaeuserstrasse
Birkenfeld Kirchplatz
Birkenfeld Marktplatz
Birkenfeld Schule
Birkenfeld Maybachstrasse
Birkenfeld Silcherschule
Birkenfeld Glockwiesenstrasse
Birkenfeld Saarstrasse
Birkenfeld Hohwaldstraße
Birkenfeld Bahnhof
Birkenfeld Panoramastrasse
Birkenfeld Heimigstrasse
Birkenfeld Abzw. Bahnhof


Birkenfeld Graefenhaeuserstrasse
Birkenfeld Alte Pforzheimer Strasse
Birkenfeld Marktplatz
Birkenfeld Kirchplatz
Birkenfeld Silcherschule
Birkenfeld Schule
Birkenfeld Maybachstrasse
Birkenfeld Saarstrasse
Birkenfeld Panoramastrasse
Birkenfeld Glockwiesenstrasse
Birkenfeld Hohwaldstraße
Birkenfeld Bahnhof
Birkenfeld Heimigstrasse
Birkenfeld Abzw. Bahnhof
Birkenfeld Tiefenbach
Pforzheim Aeusserer Arlinger
Pforzheim Hochfirststraße
Pforz