OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

MASTERARBEIT

Helena Huber

Optimierung des ÖPNV am Beispiel einer Passagierroutenplanung in Regensburg

Fakultät:	Informatik und Mathematik
Studiengang:	Mathematik
Abgabefrist:	17.06.2019
Betreuer/Prüfer:	Prof. Dr. Stefan Körkel

Erklärung

1. Mir ist bekannt, dass dieses Exemplar der Masterarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.

2. Ich erkläre hiermit, dass ich diese Masterarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum, Unterschrift

Studentin:	Helena Huber
Matrikelnummer:	3103832
Bearbeitungsdauer:	17.12.2018 - 17.06.2019
Zweitprüfer:	Prof. Dr. Wolfgang Lauf

Zusammenfassung

Die vorliegende Arbeit wurde im Rahmen des Studiengangs Master Mathematik an der Fakultät Informatik und Mathematik der Ostbayerischen Technischen Hochschule Regensburg als Abschlussarbeit zur Erlangung des Titels Master of Science (M. Sc.) angefertigt.

Ziel ist die Optimierung des Öffentlichen Personennahverkehrs (ÖPNV) am Beispiel einer Optimierung der Passagierroutenplanung im Regensburger Stadtbusverkehr. Dabei steht die mathematische Modellierung und Problemformulierung sowie die Anpassung eines Algorithmus zur Lösung des Passagierrouting-Problems im Vordergrund. Die Implementierung einer entsprechenden Fahrplanauskunft wird insofern weniger als praxisrelevantes Abfragetool, sondern viel mehr als Unterstützung der theoretischen Problemanalyse verstanden. Die Arbeit lässt sich grob in drei Teile gliedern:

Im ersten Teil (Kapitel 2) werden verschiedene Optimierungsmöglichkeiten im ÖPNV und die entsprechenden mathematischen Lösungsansätze vorgestellt. Zudem werden grundlegende Begriffe aus dem ÖPNV erklärt und Besonderheiten des städtischen Nahverkehrs in Regensburg analysiert.

Im zweiten Teil (Kapitel 3 und Kapitel 4) folgt die mathematische Aufarbeitung der Passagierroutenoptimierung. Hierfür wird zunächst an relevante graphentheoretische Konzepte und Such-Algorithmen auf Graphen erinnert, das Kürzeste Wege Problem wird formuliert und der Djikstra-Algorithmus als dessen Lösung vorgestellt. Danach werden die Methoden der algorithmischen Graphentheorie auf die Voraussetzungen der Passagierroutenplanung angepasst. Darauf aufbauend wird das mathematische Modell konstruiert. Dabei werden die Ansätze des zeitexpandierten und des zeitabhängigen Graphen vorgestellt und miteinander verglichen. Für die weiterführende Untersuchungen wird das Konzept des zeitabhängigen Graphen mit dynamischer Kantengewichtung betrachtet. Das Kürzeste Wege Problem wird als Earliest Arrival Problem formuliert und der Dijkstra-Algorithmus wird für die Anwendung auf zeitabhängige Graphen angepasst. Zuletzt wird das Modell um Umstiege an Haltestellen erweitert.

Der dritte Abschnitt (Kapitel 5, 6 und 7) befasst sich mit dem praktischen Teil der Arbeit. Die vom Stadtwerk Regensburg (Abteilung Mobilität) zur Verfügung gestellte Datenschnittstelle zu den Linien- und Fahrplandaten wird dokumentiert und die Implementierung der Fahrplanauskunft wird erläutert. Anschließend wird die praktische Problemlösung des Passagierroutings in Regensburg anhand von Beispielen demonstriert. Schließlich werden zusätzliche Anpassungsoptionen für die Fahrplanauskunft vorgeschlagen. Diese werden darüberhinaus mit möglichen Modellerweiterungen wie Fußwege, Multi-Criteria Optimierung und Speed-Up Techniken ergänzt.

Inhaltsverzeichnis

1. Mobilität, Digitalisierung und der Regensburger ÖPNV

Digitalisierung und Umwelt sind wohl die beiden Kernbereiche, die Deutschland und die internationale Politik in der heutigen Zeit bewegen. Extrem kontrovers diskutiert bieten diese Zukunftsfelder neben Problemen vor allem eine Vielzahl an Möglichkeiten für die moderne Gesellschaft. Dabei kann insbesondere eine gelungene Verknüpfung beider Gebiete wegweisend genutzt werden.

Mobilität ist eines der Schlüsselfelder, das Digitalisierung und Umwelt miteinander kombiniert.

> *„Mobil zu sein ist Grundlage für persönliche Freiheit und Teilhabe am gesellschaftlichen Leben. Mit dem digitalen Wandel soll Mobilität für jeden einfacher, sauberer und sicherer werden.“*

heißt es im aktuellen Leitfaden *Digitalisierung gestalten: Umsetzungsstrategie der Bundesregierung*, welche sich unter anderem mit der Gesellschaft im digitalen Wandel beschäftigt. [Bun19, S. 85]
Während die Digitalisierung neue Wege für ein transparenteres, schnelleres und Nutzeroptimiertes Mobilitätskonzept schafft, führt das Thema Umwelt und Nachhaltigkeit zu einem Wandel des Mobilitätsbewusstseins, was eine veränderte Nachfrage und den Wunsch nach alternativen Mobilitätskonzepten zum privaten PKW mit sich bringt.

Auch wenn der motorisierte Individualverkehr immer noch als dominierendes Fortbewegungsmittel in Deutschland gilt, gewinnt der öffentliche Verkehr (ÖV) weiter an Zuwachs. So stieg die Nutzung des ÖV als Verkehrsmittel im Bundesgebiet von 8% im Jahr 2008 bis zum Jahr 2017 um ein Viertel der Prozentpunkte auf 10% an. In den Metropolen liegt der Anteil 2017 sogar bei 20%. Insgesamt ist der ÖV vor allem bei jungen Leuten und in urbanen Gegenden besonders beliebt, wie in der Studie *Mobilität in Deutschland* vom Bundesministerium für Verkehr und digitale Infrastruktur ermittelt wurde. Genau bei dieser Gruppe steigt auch die Nachfrage nach digitaler Mobilitätsdienstleistung immer weiter. Hierzu zählen beispielsweise die Bereiche Fahrkartenerwerb, Auskünfte über Fahrpläne und Verspätungen sowie Routenplanung. Dabei werden Routenplanungs- und Straßennavigationssysteme von 55% der Bevölkerung im ländlichen Raum und sogar von 68% der Metropolbewohner verwendet. [FG19]

Doch wie kommt man tatsächlich am schnellsten von A nach B? Wie funktioniert eine digitale Fahrplanabfrage? Auf welchen Theorien und Modellen basiert das Passagierrouting-Problem? Welches mathematische Optimierungsproblem gilt es bei der Suche nach einer geeigneten Reiseroute zu lösen? Und wie sieht am Ende die Fahrplanauskunft aus?
Die Beantwortung dieser Fragen gilt als Motivation der vorliegenden Arbeit.

Die theoretische Aufarbeitung der Passagierroutenoptimierung wird im praktischen Teil mithilfe der Linien- und Fahrplandaten des städtischen Nahverkehrs von Regensburg ergänzt.

Der ÖV in Regensburg hat eine über 100 Jahre alte Tradition. Das erste öffentliche Verkehrsmittel, eine bis zu acht Personen fassende Pferdekutsche, wurde im Mai 1891 zum ersten Mal eingesetzt. Da sich dies jedoch als unrentabel erwies, wurde die *I. Regensburger Omnibus-Unternehmung* bereits ein halbes Jahr später, im November 1891, wieder eingestellt.
Am 21. April 1903 folgte dann die Einführung einer elektrisch betriebenen Straßenbahn. Anfangs nur zwei Linien von Ost nach West sowie von Nord nach Süd umfassend, entwickelte sich das Regensburger Trambahn-Netz in den nächsten Jahren immer weiter, bis es 1936 mit ca. 17 km Streckenlänge seine maximale Größe erreicht hatte.
Kurz darauf, im Mai 1938, wurde das ÖV-Netz durch die erste Omnibuslinie in Regensburg erweitert.
Doch die Entwicklungen während des Zweiten Weltkriegs führten zum nächsten starken Einschnitt in der Geschichte des ÖV in Regensburg. So wurde durch den Bombenangriff im März 1945 der Strassenbahnverkehr vollkommen außer Betrieb gesetzt. Auch die Omnibusse konnten wegen Treibstoffknappheit nicht mehr verkehren. Während im Oktober 1946 die ersten Busse wieder eingesetzt wurden, konnte die Nord-Süd-Tramlinie vom Bahnhof nach Reinhausen wegen der Sprengung der Steinernen Brücke nicht mehr aufgenommen werden. Die restlichen Straßenbahn-Linien fuhren erst im Sommer 1947 wieder regelmäßig. Da allerdings die Trambahnen in Hinblick auf Flexibilität und Geschwindigkeit nicht mit den Omnibussen mithalten konnten, wurden sie nach und nach vom Markt verdrängt, bis im August 1964 die letzte Trambahn durch die Regensburger Straßen fuhr.
Wegen der Treibstoffknappheit nach dem Zweiten Weltkrieg wurde 1953 der Omnibusverkehr um eine Oberleitungsbuslinie ergänzt. Aufgrund der Mehrbelastung durch den Betrieb von verschiedenen Verkehrsmitteln sowie fehlender Oberleitungen wurde der O-Busbetrieb aber bereits nach zehn Jahren wieder eingestellt.
Seit der Abschaffung der Oberleitungsbusse und Trambahnen verkehren im ÖV der Stadt Regensburg ausschließlich Omnibusse. [Edt13, S. 11-23]

Wie man mithilfe dieser Omnibusse am schnellsten von A nach B kommt, wird im Folgenden untersucht. Die nachstehenden Kapitel setzen sich wie folgt zusammen:

In Kapitel 2 wird der Öffentliche Personennahverkehr (ÖPNV) mit seinen diversen Planungsschritten, Optimierungsmöglichkeiten sowie den jeweiligen mathematischen Lösungsmethoden vorgestellt. Außerdem werden wesentliche Begriffe aus dem ÖPNV erklärt und der städtische Nahverkehr von Regensburg in seiner heutigen Form vorgestellt.

In Kapitel 3 wird anschließend auf mathematische Grundlagen für die Passagierroutenplanung im ÖPNV eingegangen. In diesem Zusammenhang werden neben graphentheoretischen Inhalten auch verschiedene Such-Algorithmen auf Graphen vorgestellt. Die mathematischen Konzepte werden dann für ihre Anwendung im ÖPNV präzisiert.

Die Modellierung der Passagierroutenoptimierung erfolgt in Kapitel 4. Dabei werden zunächst die Modelle des zeitexpandierten und des zeitabhängigen Graphen erläutert und verglichen. Anschließend wird das Optimierungsproblem als Earliest Arrival Problem konkret formuliert und der Dijkstra-Algorithmus zur Lösung des Kürzesten Wege Problems angepasst. Am Ende des Theorie-Teils wird das Modell um Umsteigezeiten an Haltestellen erweitert.

Mit Kapitel 5 beginnt der praktische Teil dieser Abschlussarbeit. Hierbei wird einerseits die Datenschnittstelle zu den Linien- und Fahrplandaten des Regensburger Stadtverkehrs dokumentiert, andererseits wird die Implementierung der Fahrplanabfrage erläutert, die auf dem zuvor konstruierten mathematischen Modell zur Passagierroutenoptimierung basiert.

Beispiele und Besonderheiten der Implementierung werden in Kapitel 6 diskutiert.

Möglichkeiten zur Erweitung der Fahrplanauskunft und des Algorithmus werden als Ausblick in Kapitel 7 vorgestellt. In diesem Zusammenhang werden zum einen Optionen zur Entwicklung einer detaillierteren Fahrplanauskunft gegeben. Zum anderen werden Ansätze zur Ausweitung des Optimierungsmodells aufgezeigt.

Das Fazit in Kapitel 8 rundet die Arbeit ab.

2. Der öffentliche Personennahverkehr

Zunächst werden verschiedene Aspekte des öffentlichen Personennahverkehrs betrachtet. Dabei wird ein Überblick zu den unterschiedlichen Aufgabenbereichen der Prozess- und Betriebsplanung im ÖPNV gegeben und einige Optimierungsansätze skizziert. Anschließend werden für diesen Kontext grundlegende Begriffe erläutert. Außerdem wird der Regensburger Nahverkehr vorgestellt.

2.1. Operative Planung im ÖPNV

Zur Prozessplanung des Öffentlichen Nahverkehrs gehören einerseits die strategische und andererseits die operative Planung. Während in der strategischen Planung unter anderem die Nachfrage an öffentlichen Transportmöglichkeiten sowie die dafür benötigte Infrastruktur ermittelt wird, soll im Zuge der operativen Planung die Disposition der Betriebsmittel und die Erstellung eines konkreten Fahrplans stattfinden. Letzteres stellt einen aufwendigen Prozess dar, welcher in der Praxis in einzelne Teilbereiche gegliedert ist und am Ende den für den Kunden[1] sichtbaren Fahrplan liefert.

Am Beginn der operativen Planungsphase steht die Linienplanung, gefolgt von der Fahrplanerstellung. Im Anschluss daran beginnt die Fahrzeugeinsatz- beziehungsweise Umlaufplanung sowie die Erstellung von – zunächst unpersonalisierten – Dienst- und Dienstreihenfolgeplänen, welche schließlich mithilfe der Personaleinsatzplanung den konkreten Mitarbeitern zugeteilt werden.
Besonders hervorzuheben ist, dass die einzelnen Schritte der operativen Planungsphase von verschiedenen Instanzen bearbeitet werden. Während die Linienplanung und Fahrplanerstellung im Zuständigkeitsbereich der Stadt liegt, werden die anderen Bereiche, deren Hauptziel eine möglichst robuste und reibungslose Umsetzung des Fahrplans ist, von den jeweiligen Verkehrsbetrieben übernommen.

Bei der Linienplanung wird anhand von Informationen über das zu bewältigende Passagieraufkommen der konkrete Verlauf einzelner Linien ermittelt. Das heißt, die in der strategischen Planung ermittelten Haltestellen werden auf dem vorgegebenen Streckennetz derart miteinander verbunden, dass möglichst viele Passagiere auf kurzen Strecken mit wenig Umstiegen von A nach B transportiert werden können.
Grundlage hierfür sind sogenannte OD-Matrizen (vgl. engl. *Origin-Destination-Matrix*), deren Einträge die Transportnachfrage widerspiegeln, indem angegeben wird, wie viele Personen innerhalb einer bestimmten Periode von einem Ort zu einem anderen Ort innerhalb des Netzes befördert werden wollen. Um das Linienplanungsproblem zu lösen, gibt es unterschiedliche Ansätze, beispielsweise können komplette Linien mittels iterativer Prozesse aus kleineren Teilstücken zusammengesetzt werden. Eine andere Idee ist

[1] Aus Gründen der besseren Lesbarkeit wird grundsätzlich das generische Maskulinum verwendet. Es sind damit alle Personen unabhängig von ihrem Geschlecht gemeint.

es, zunächst einen Pool aus potentiellen Linien zu bestimmen und daraus im Anschluss geeignete Linien zu benennen. [BNP08, S. 2-4]
Am Ende der Linienplanung wird die Gesamtheit der ermittelten Linien im Liniennetz beziehungsweise Netzplan zusammengefasst.

Nachdem das Liniennetz erstellt worden ist, wird im nächsten Schritt der Fahrplan ermittelt. Dabei ist besonders die Taktung der einzelnen Linien entscheidend. Um den Passagieren ein attraktives Angebot zu gewährleisten, müssen die Umsteigezeiten so kurz wie möglich gehalten werden. Andererseits führt eine zu knappe Umsteigezeit zu einem störungsanfälligen Fahrplan, was zu vermeiden ist. Diese beiden kontroversen Ziele gilt es zu vereinen.
Eine weitere Schwierigkeit bei der Fahrplanerstellung ist die Einhaltung eines regelmäßigen Taktes, um die Einprägsamkeit des Fahrplans sicherzustellen. So ist eine gleich bleibende Taktung über den ganzen Tag hinweg wünschenswert, die Umsetzung gestaltet sich allerdings aufgrund des schwankenden Verkehrsaufkommens zu den unterschiedlichen Tageszeiten kompliziert.
Für die Optimierung des Taktungsproblems werden neben den bereits oben erläuterten OD-Matrizen und dem Liniennetzplan auch die zur Verfügung stehende Anzahl an Bussen benötigt. Zur Lösung des Problems werden neben nichtlinearen Modellen mit approximierten Ergebnissen auch Gemischt-Ganzzahlige-Ansätze herangezogen. [MMU13]

Wenn die Phase der Fahrplanerstellung abgeschlossen ist, endet auch der Verantwortungsbereich für den Planungsprozess auf kommunaler Ebene. Die nachfolgenden Schritte obliegen dann den jeweiligen Nahverkehrsbetrieben.

Um den gegebenen Fahrplan bedienen zu können, gilt es auf der einen Seite den Fahrzeugeinsatz zu disponieren sowie auf der anderen Seite Dienstpläne für die Fahrer der Fahrzeuge zu bestimmen.

Ziel der Fahrzeugeinsatzplanung ist das Bewerkstelligen des Fahrplans unter Vermeidung unnötiger Leerfahrten. Zusätzlich soll die Menge an eingesetzten Fahrzeugen minimiert werden. Für die Komplexität und die Lösung dieses Minimierungsproblems sind die Anzahl der Betriebshöfe und Fahrzeugtypen entscheidend. Hierbei wird zwischen den sogenannten Eindepot- und Mehrdepot-Umlaufplanungsproblemen differenziert. Es ist zu bemerken, dass zu Singledepot-Problemen nicht alle Fälle mit einem einzigen Depot gehören, sondern nur diejenigen, bei denen zusätzlich nur ein einziger Fahrzeugtyp zum Einsatz kommt. Alle anderen Fälle werden den Multidepot-Problemen zugeordnet, also insbesondere auch jene Probleme, die zwar nur über einen Betriebshof verfügen, aber verschiedene Fahrzeugtypen inkludieren.
Während das Einzeldepot-Umlaufplanungsproblem mit relativ geringem Aufwand in polynomieller Zeit lösbar ist, ist das Mehrdepot-Umlaufplanungsproblem NP-schwer. [Kli05, S. 17]

Zur mathematischen Modellierung werden die Probleme der Fahrzeugeinsatzplanung im Allgemeinen als Mehrgüterflussprobleme aufgefasst. Der Netzplan und die zu bewerkstelligenden Fahrten werden mit den Depots zu einem Netzwerk zusammengefasst, in dem die einzelnen Fahrzeugtypen *("Güter")* auf ihren Fahrtwegen *("Fluss")* verkehren und so den Umlauf bilden. [Bor00, S. 1]

Die Dienst- und Personaleinsatzplanung bildet den letzten großen Baustein im operativen Planungsprozess. Nachdem fast die Hälfte der Gesamtkosten eines städtischen Standardbusses als Personalkosten zu Buche schlagen, besteht ein besonderes Interesse an der Optimierung dieses Planungsschrittes aus Sicht des Verkehrsbetriebes. [Bor00, S. 1] Im Gegensatz zur Umlaufplanung ist dieser Abschnitt an strenge gesetzliche und tarifliche Reglementierungen gekoppelt, was die Komplexität des Problems stark erhöht. Wie oben erwähnt, werden im Zuge dessen zunächst anonymisierte Dienstpläne erstellt, das heißt, die Umläufe werden mit Fahrern ausgestattet, ohne dabei konkrete Personen zu bestimmen. Die daraus resultierenden Dienstarten, wie beispielsweise Früh- oder Spätdienst, werden erst im nächsten Schritt, der sogenannten Personaleinsatzplanung, den einzelnen Fahrern zugeordnet.
Aus mathematischer Sicht handelt es sich hierbei um ein Mengenpartitionierungsproblem, das sich nur näherungsweise lösen lässt.

Die hier dargestellten Teilbereiche der operativen Planung im ÖPNV werden in der Praxis schrittweise nacheinander betrachtet und jede Phase für sich optimiert. Da die Behandlung der einzelnen Probleme separat betrachtet meist schon mit erheblichem Aufwand verbunden ist, findet eine Verknüpfung der einzelnen Phasen in der Regel nur selten statt, obwohl eine wechselseitige Optimierung der einzelnen Prozesse in vielen Fällen zu einem wesentlich besseren Gesamtergebnis führen würde.

Trotz der Vielzahl an Mathematik- und Software-gestützter Techniken zur Optimierung der operativen Planungsbereiche werden auch heutzutage noch die meisten Lösungen nachträglich händisch angepasst, sodass daran erinnert werden sollte, welchen wesentlichen Beitrag nicht quantifizierbare Erfahrungswerte in der ÖPNV-Planung leisten.

2.2. Weitere Optimierungspotentiale im ÖPNV

Neben der Fahrplanung gibt es noch weitere Gebiete im ÖPNV, in denen die mathematische Optimierung einen wesentlichen Beitrag zur Prozessverbesserung leistet.

Ein sowohl für die Verkehrsbetriebe als auch für die Kunden bedeutender Ansatzpunkt ist dabei die Preisplanung. Eine gute Preisgestaltung soll einerseits die Wirtschaftlichkeit des Nahverkehrs garantieren und andererseits die Nachfrage nach ebendiesem steigern.
In Deutschland erfolgt die Preisgestaltung im ÖPNV in den häufigsten Fällen auf Basis eines Zonen- bzw. Wabentarifs, welcher Aspekte aus Einheits- und Entfernungspreissystemen miteinander vereint. Dabei werden Veränderungen mithilfe von Elastizitäten

vorgenommen. Mittels diskreten Logit-Entscheidungsmodellen können darüber hinaus die Zusammenhänge zwischen Kunden-Nachfrage und Höhe des Preises diskutiert werden. [BNP08, S. 12]

Insbesondere auch aus Sicht der Kunden wurden in den letzten Jahren viele hilfreiche Module und Tools entwickelt, um die Benutzung des Nahverkehrs zu erleichtern und damit attraktiver zu gestalten.

So gibt es beispielsweise an Haltestellen und in den Fahrzeugen immer mehr elektronische Anzeigen, welche Minuten-, teilweise sogar Sekunden-genaue Informationen zu den nächsten Haltestellen und Umsteigemöglichkeiten liefern.

Während man früher mühsam einzelne Linienpläne sowie die verschiedenen Abfahrtszeiten an den Haltestellen studieren und kombinieren musste, gibt es heutzutage Apps, die schnell eine optimale Route liefern, um von A nach B zu gelangen.
Ermöglicht wird dies mithilfe von Algorithmen aus der Graphentheorie, einem wichtigen Teilgebiet der kombinatorischen Optimierung. Aus den einzelnen Linien und Haltestellen eines ÖPNV-Gebiets wird dabei eine Art Netz (der sog. *Graph*) konstruiert, in dem man nach bestimmten Vorschriften (z.B. *Tiefensuche* oder *Breitensuche*) einzelne Wege abläuft, um so einen möglichst kurzen Weg vom Startpunkt zum Zielpunkt zu finden (*Kürzeste Wege Suche*).
Die Analyse, wie diese Fahrplanabfragen funktionieren und auf welchen mathematischen Theorien und Grundlagen sie aufbauen, wird Hauptgegenstand dieser Arbeit sein.

2.3. Grundlegende Begriffe im ÖPNV

In den vorherigen Abschnitten wurden diverse Optimierungsansätze im Bereich des ÖPNV angesprochen, von denen später die Fahrplanabfrage als Kürzeste Wege Suche eingehender untersucht wird. Um missverständliche Formulierungen zu vermeiden, eine sinnvolle mathematische Definition zu ermöglichen und die vom Regensburger Stadtwerk zur Verfügung gestellten Daten zu analysieren, werden zunächst die für diesen Zusammenhang grundlegenden Begriffe erklärt. Sofern nicht anders vermerkt, stammen die Informationen aus [HJLH97, S. 610-611] sowie aus Gesprächen mit der Betriebsleiterin und dem stellvertretendem Betriebsleiter vom Stadtwerk-Mobilität Regensburg.

Allgemein unterscheidet man zwei Arten des öffentlichen Personennahverkehrs: *Schienenpersonennahverkehr* und *öffentlicher Straßenpersonennahverkehr (ÖSPV)*. Wie die Namen bereits vermuten lassen, umfasst der Schienenpersonennahverkehr alle ÖPNV-Personentransporte mittels Eisenbahnzügen, während man unter dem öffentlichen Straßenpersonennahverkehr all jene Transportmittel versteht, welche ganz oder zumindest teilweise die öffentliche Straßeninfrastruktur mitbenutzen. Der wesentliche Unterschied liegt darin, dass in Letzterem verkehrsbedingte Störungen in die planerischen Prozesse miteinkalkuliert werden müssen, wohingegen dies beim Schienenverkehr keine Rolle spielt. Auf der anderen Seite muss beim Schienenverkehr die Gleisbelegung berücksichtigt

werden, sodass im Gegensatz zum Straßenverkehr im Allgemeinen kein Überholvorgang möglich ist. [Sch15, S. 1-2]

Die verschiedenen Verkehrsmittel wie Bus, Straßenbahn oder S-Bahn werden jeweils als *Modus* bezeichnet. Die einzelnen Modi können teilweise noch weiter spezifiziert werden (z.B. Solo- oder Gelenkbus beim Modus *Bus*).

Das Liniennetz eines Verkehrsbetriebes setzt sich zusammen aus den einzelnen *Haltestellen*, an denen Passagiere die Fahrzeuge betreten und verlassen können und aus *Linien*, welche bestimmte Haltestellen miteinander verbinden. Für jede Haltestelle werden konkrete *Haltepunkte* angegeben, die *mastenscharf* beschreiben, wo ein Bus, der auf einer bestimmten Linie verkehrt, an dieser Haltestelle hält. Das heißt, zu jeder Haltestelle gibt es ein oder mehrere Haltepunkte, umgekehrt kann jeder Haltepunkt aber eindeutig einer Haltestelle zugeordnet werden.

Neben den Haltestellen gibt es *Betriebshöfe* bzw. *Depots*, welche als Abstellorte für die Fahrzeuge dienen. Die Betriebshöfe sind gleichzeitig auch Start- und Endpunkt für das Ein- und Ausrücken einer Fahrt.

Alle Fahrten, die sich aus dem Fahrplan ergeben und dementsprechend zur Personenbeförderung dienen, werden als *Fahrgastfahrten* oder *Servicefahrten* bezeichnet. Für alle Fahrgastfahrten muss bestimmt werden, welche Fahrzeugart diese durchführen kann und zu welchem Depot diese Fahrzeuge gehören. Die für die jeweilige Fahrgastfahrt geeigneten Depots werden dann zu einer *Depotgruppe* zusammengefasst. Im Zuge der Umlaufplanung wird bestimmt, welches Fahrzeug aus der zugehörigen Depotgruppe die jeweilige Servicefahrt bestreitet.
Neben den Fahrgastfahrten gibt es die sog. *Leer-* bzw. *Betriebsfahrten*, welche alle Fahrten bezeichnen, die nicht zur Fahrgastbeförderung dienen. Dazu zählen:

- *Ausrück-* bzw. *Einsetzfahrten*, die die Fahrten vom Betriebshof zum jeweiligen Startpunkt einer Servicefahrt beschreiben,
- *Einrück-* bzw. *Aussetzfahrten*, welche von der letzten Haltestelle einer Fahrgastfahrt zurück zum Betriebshof führen und
- *Überläuferfahrten*, auch *Kopplungen* oder *Verbindungsfahrten* genannt, welche zwei Fahrgastfahrten miteinander verbinden.

Sowohl Servicefahrten als auch Leerfahrten können mit *Gewichten* versehen werden, um die Längen der Fahrten miteinander in Relation zu setzen. In dieser Arbeit wird als Gewicht für die Fahrtstrecke zwischen zwei Haltepunkten oder zwischen einem Haltepunkt und dem Betriebshof die dafür benötigte Fahrzeit verwendet.

Als *(Fahrzeug-)Umlauf* wird eine Reihe von Fahrten bezeichnet, die mit einer Ausrückfahrt aus dem Betriebshof startet, woraufhin mehrere kompatible Fahrgastfahrten

folgen. Sie endet schließlich mit einer Einrückfahrt zurück zum (nicht notwendigerweise selben) Betriebshof. Dabei heißen zwei Servicefahrten *kompatibel*, wenn sie entweder direkt nacheinander von einem Fahrzeug bedient werden können oder durch eine Kopplung miteinander verbunden sind. [Kli05, S. 16] Start- und Endpunkt eines Depots stellen somit auch Start- und Endpunkt eines Fahrzeugumlaufs dar. Ein solcher Umlauf wird von einem Fahrzeug bedient und bezieht sich auf die Dauer eines Betriebstages.
Der Umlauf wird als *linienrein* bezeichnet, wenn er auf genau einer Linie verkehrt, ohne diese im Laufe des Betriebstages zu wechseln. Alle Umläufe eines Betriebstages werden im *Umlaufplan* festgehalten.

2.4. Der Regensburger ÖPNV

Da die angestrebte Passagierroutenoptimierung bzw. Fahrplan-Abfrage am Beispiel des Regensburger Nahverkehrs erfolgt, wird im folgenden Abschnitt der Regensburger ÖPNV vorgestellt und relevante Aspekte der Passagierroutenplanung durch Anpassung auf diese Daten eingegrenzt. Die Inhalte beziehen sich – wenn nicht anders vermerkt – auf Gespräche mit der Betriebsleitung vom Stadtwerk-Mobilität Regensburg.

Für die Planung des ÖPNV in Regensburg sind einerseits die Stadt Regensburg selbst und andererseits das Regensburger Stadtwerk (Abteilung Mobilität), welches als Nahverkehrsbetrieb agiert, zuständig.

Dabei wird von der Stadt die Verkehrsplanung übernommen. Dazu zählen insbesondere die Bereiche Linienplanung, Taktung- bzw. Frequenzplanung sowie die Fahrplanerstellung. Die Softwares *PTV Visum* und *PTV Vissim* werden als technische Unterstützung verwendet. Während mit PTV Visum eine allgemeine Verkehrsmodellierung möglich ist, kann mithilfe von PTV Vissim der Stadtverkehr mit Schienen- und Straßenverkehrsteilnehmern sowie Fußgänger simuliert werden.[2]

Alle betrieblichen Planungsschritte nach der Fertigstellung des Fahrplans – insbesondere die praktische Abwicklung des Fahrplans mit Verkehrsmitteln und Personal – wird danach vom Stadtwerk übernommen. Ziel ist es, einen robusten und effizienten Fahrplan gewährleisten zu können. Damit fallen beispielsweise die Bereiche Umlauf- und Dienstplanung sowie temporäre Fahrplanänderungen aufgrund von Baustellen in das Tätigkeitsfeld des Stadtwerks. Zur digitalen Planung und Disposition arbeitet das Stadtwerk mit der Software *HASTUS* der Firma *GIRO*.[3]

Als ökonomischer Betrieb unterliegt das Stadtwerk auch gesetzlichen, tariflichen und betrieblichen Rahmenbedingungen wie der Betriebsvereinbarung und der Fahrpersonalverordnung (FPersV), welche z.B durch Vorgaben zu Lenk- und Ruhezeiten den Handlungsspielraum bei der Ausführung des Fahrplans einschränken.

[2] Vgl. http://vision-traffic.ptvgroup.com/de/produkte/ (Stand 06.03.2019).
[3] Vgl. http://www.giro.ca/de/loesungen/bus-u-und-s-bahn-strassenbahn (Stand 06.03.2019).

Momentan verkehren im städtischen Nahverkehr Regensburgs nur Busse.
Im Juni 2018 wurde aber vom Stadtrat die Planung einer neuen Stadtbahn in Regensburg beschlossen. Diese soll auf zwei Linien den Stadtnorden mit dem -süden im fünf-Minuten-Takt verbinden. [Reg18]
Da sich die Einführung der Stadtbahn noch in der Planungsphase befindet, wird im Folgenden nicht näher darauf eingegangen.
Das bedeutet, der in dieser Arbeit betrachtete ÖPNV beschränkt sich auf den ÖSPV mit dem Einzel-Modus *Bus*. Insbesondere findet beim Passagierrouting kein Moduswechsel statt und der multimodale Fall wird nicht weiter untersucht.

Das Stadtwerk verfügt über einen Betriebshof, der in der Markomannenstraße 1, 93053 Regensburg, angesiedelt ist. Die Fahrzeugflotte beläuft sich auf 64 18-Meter-lange Gelenkzüge und 39 zwölf-Meter-lange Standardbusse sowie fünf ca. acht-Meter-lange Elektro-Midi-Busse. Letztere verkehren dabei lediglich auf der Linie A in der Altstadt.

Das Liniennetz im Stadtgebiet besteht aus 46 internen Linien und ein paar weiteren Linien, die von externen Dienstleistern betrieben werden (z.B. Linie 32 zwischen Hauptbahnhof und IKEA). Zu den internen Linien zählen unter Anderem auch die Linien N1-N7, welche an den Wochenenden als Nachtbusse fahren, sowie die Campus-Linien C1, C2, C4 und C6, die während der Semester-Zeiten zwischen Altstadt und Universität bzw. OTH verkehren.

Alle Busse sind mit einem Informationssystem ausgestattet, welches in einem zehn-Sekunden-Takt die Position des Fahrzeugs bestimmt und dadurch die Einhaltung der durch den Fahrplan vorgeschriebenen Fahrzeit misst. Diese Informationen werden sowohl den Busfahrern als auch der Leitstelle übermittelt. Bei zu großen Abweichungen gegenüber dem SOLL-Fahrplan kann die Leitstelle unmittelbar eingreifen bzw. Anweisungen an die Fahrer übermitteln.

Um spezifische Daten über das Fahrverhalten der Passagiere zu sammeln, sind ca. zehn Prozent aller Busse mit Sensoren an den Türen ausgestattet, welche als Fahrgastzählsystem dienen. Damit können nach entsprechender Gewichtung und Validierung der Daten OD-Matrizen zur Linien- und Taktplanung berechnet werden.
Da die Informationen aus dem Fahrgastzählsystem zum Zeitpunkt der Anfertigung dieser Arbeit noch nicht validiert wurden, wird darauf im Folgenden nicht weiter eingegangen.

Detaillierte Informationen zu den Fahrplan- und Liniendaten folgen in Unterkapitel 5.1.

3. Algorithmen auf Graphen

In diesem Kapitel werden grundlegende Inhalte der Graphentheorie und wichtige Algorithmen auf Graphen wiederholt. Dabei wird zuerst auf die theoretischen Konzepte eingegangen, welche im Anschluss auf den ÖPNV angewendet werden.

3.1. Graphentheoretische Konzepte und Algorithmen

Zunächst soll an die Definition eines Graphen erinnert werden (vgl. [Ove08, S.2-3]):

Definition 3.1. Ein *Graph* G besteht aus einem Tupel $G = (V, E)$ mit einer endlichen nicht-leeren Menge V und einer Menge $E \subseteq \{\{u, v\} \mid u, v \in V\}$. Man nennt V die Knoten und E die Kanten des Graphen G.

Weiterhin gilt:

Definition 3.2. Gegeben sei ein Graph $G = (V, E)$.

1. Der Graph heißt *gerichteter Graph* oder *Digraph*, wenn seine Kanten aus geordneten Paaren $(u, v) \in V \times V$, $u \neq v$, bestehen. Damit implizieren die Kanten eine Richtung, d.h. die Kante $(u, v) \in E$ besagt, dass die Verbindungskante zwischen u und v vom *Anfangsknoten* u in den *Endknoten* v verläuft und nicht umgekehrt.

2. Eine Folge $(e_1, \ldots, e_n)$ von Kanten aus G wird *Kantenzug* oder *Pfad* genannt, wenn für die zugehörigen Knoten dieser Kanten gilt: $(v_{i-1}, v_i) = e_i$, $v_0, \ldots, v_n \in V$, $i = 1, \ldots, n$. v_0 wird als *Anfangsknoten* und v_n als *Endknoten* des Kantenzugs bezeichnet. Die natürliche Zahl $n \in \mathbb{N}$ beziffert die *Länge* des Kantenzugs. Für den Fall, dass $v_0 = v_n$ gilt, nennt man den Pfad *geschlossen*.

   Da sich die Kanten $(e_1, \ldots, e_n)$ aus einer eindeutigen Folge an Knoten zusammensetzen, wird ein Kantenzug manchmal auch als Folge von Knoten $(v_0, \ldots v_n)$ mit $(v_{i-1}, v_i) \in E$, $i = 1, \ldots, n$, definiert.

3. Wenn alle Knoten im Pfad paarweise verschieden sind, heißt dieser Kantenzug *Weg*. Ein geschlossener Weg wird als *Kreis* bezeichnet.

4. Der Graph G wird *zusammenhängend* genannt, wenn es zu je zwei Knoten v_i und v_j aus V einen Pfad gibt, der die beiden Knoten verbindet. Damit ist in einem zusammenhängenden Graphen von einem beliebigen Startknoten aus jeder Endknoten erreichbar.

5. Ein *Baum* ist ein zusammenhängender Graph, der keine Kreise enthält.

□

Es gibt eine Vielzahl von Algorithmen auf Graphen. Einige davon arbeiten mit sog. *Traversierungen*, d.h. sie durchlaufen nach und nach alle Knoten des Graphen. Um zu verhindern, in Endlosschleifen zu geraten, wird oft das Prinzip des *label-settings* angewendet, bei dem besuchte Knoten bzw. Kanten markiert werden.
Wie für Traversierungen typisch, werden im Folgenden *nur zusammenhängende Graphen* betrachtet. [GS16, S. 424]

Zwei klassische Methoden, einen Graphen zu durchlaufen, ergeben sich aus den Algorithmen zur Breiten- und Tiefensuche. Die nachfolgenden Informationen hierzu stammen aus [GS16, S. 424-426] und [Sch01, S. 205-207], wobei die Algorithmen 1 (Tiefensuche) und 2 (Breitensuche) aus Letzterem entnommen wurden.

Beide Algorithmen suchen alle Knoten aus einem Graphen $G = (V, E)$, die vom Anfangsknoten v_0 aus erreichbar sind und markieren diese. (Da vorausgesetzt wurde, dass nur zusammenhängende Graphen betrachtet werden, werden alle Knoten von v_0 aus besucht.)

In welcher Reihenfolge die einzelnen Knoten bei der *Tiefensuche*, auch *Preorder-Baumtraversierung* oder *Depth-First Search* genannt, durchlaufen werden, hängt davon ab, wie der Graph abgespeichert ist. In jedem Fall wird ein Ast so lange abgegangen, bis es darauf entweder keinen weiteren Nachbarknoten mehr gibt oder bis alle Nachbarknoten markiert wurden. Danach wird auf den letzten Vorgängerknoten zurückgesprungen, der noch nicht besuchte Nachbarknoten hat und von dort aus das Verfahren wiederholt. Dieses Zurückspringen wird als *Backtracking* bezeichnet. Indem nach und nach jeder Ast bis zum Ende abgelaufen wird, werden alle möglichen Pfade vom Startknoten v_0 aus durchsucht und jeder Knoten im Graphen markiert.

Für die Implementierung wird eine rekursive Prozedur verwendet, die über v_0 vom Hauptprogramm aus aufgerufen wird. Die Prozedur färbt die Knoten des Graphens in drei verschiedene Farben ein:

- noch nicht besuchte Knoten werden *weiß* gefärbt,
- *graue* Knoten entsprechen einer temporären Markierung und
- ein Knoten wird *schwarz* gefärbt, wenn die Markierung permanent ist.

Zunächst sind alle Knoten weiß. Wenn ein Knoten $u \in V$ besucht wird, wird er grau markiert. Schließlich wird u schwarz eingefärbt, wenn die Tiefensuche alle Nachbarknoten v von u, d.h. alle v mit $(u, v) \in E$, abgelaufen hat:
Für die Tiefensuche ergibt sich eine Komplexität von $O(|E|)$.

Anstatt wie die Tiefensuche die Äste komplett entlang zu gehen und damit jeweils zuerst in die Tiefe zu gehen, verläuft die *Breitensuche*, auch *Breadth-First Search* bzw. *Levelorder Baumtraversierung* genannt, „wellenförmig nach außen“. [GS16, S. 426]

Algorithm 1 Tiefensuche

```
procedure Tiefensuche(u : Knoten)
farbe[u] ← grau
for v ∈ V mit (u, v) ∈ E do
  if farbe[v] = weiß then
    Tiefensuche(v)
  end if
end for
farbe[u] ← schwarz
end procedure
```

Dabei werden von einem Knoten u startend zunächst alle Nachbarknoten v mit $(u, v) \in E$, $u, v \in V$, abgelaufen, d.h. alle Knoten mit Abstand 1 zum Knoten u. Im darauffolgenden Schritt werden die Nachbarn der Nachbarn betrachtet, also alle Knoten mit Abstand 2 usw.

Für die Formulierung des Algorithmus wird eine Warteschlange Q nach dem *FIFO*-Prinzip (*first-in, first-out*) realisiert, in der die noch nicht besuchten Nachbarknoten abgespeichert werden.
Gestartet wird am Anfangsknoten $v_0 \in V$. Nach und nach werden immer weitere Nachbarknoten besucht und die minimalen Abstände aller Knoten zum Startknoten v_0 in der Abstands-Tabelle d vermerkt, sodass am Ende jeder Knoten mit dem kleinsten Abstand zum Startknoten v_0 markiert wurde.

Algorithm 2 Breitensuche

```
for v ∈ V do
  d[v] ← ∞
end for
d[v_0] ← 0
Q ← {v_0}
while Q ≠ ∅ do
  u ← erstes Element in Q
  Q ← Q \ {u}
  for v ∈ V mit (u, v) ∈ E do
    if d[v] = ∞ then
      d[v] ← d[v] + 1
      Q ← Q ∪ {v}
    end if
  end for
end while
```

Der Algorithmus zur Breitensuche durchläuft jeden Knoten aus V und jede Kante aus E einmal, sodass sich auch hier eine Komplexität von $O(|E|)$ ergibt, wobei verwendet wurde, dass $G = (V, E)$ zusammenhängend ist, sodass $|E| \geq |V| - 1$ gilt und sich $O(|V| + |E|)$ durch $O(|E|)$ ersetzen lässt.

Die beiden Algorithmen Tiefensuche und Breitensuche berücksichtigen beim Durchlaufen des Graphen G nur, welche Knoten durch Kanten miteinander verbunden sind. Zusätzliche Informationen zu den Kanten werden nicht beachtet. Dabei können Kanten in einem Graphen mit bestimmten Eigenschaften versehen werden, beispielweise können sie mit sogenannten *Kantengewichten* bewertet werden:

Definition 3.3. Gegeben seien ein Graph $G = (V, E)$, zwei Knoten $u, v \in V$ und eine Kante $e = (u, v) \in E$. Wird der Graph mit einer Gewichtsfunktion

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

versehen, nennt man den resultierenden Graphen $G = (V, E, c)$ *(statisch) gewichtet*. Der Funktionswert, der jeder Kante aus E zugeordnet wird, wird *Kantengewicht* genannt. □

Im Allgemeinen können Kantengewichte auch negative Werte annehmen. Für die korrekte Funktionsweise von einigen Algorithmen werden allerdings oft nur nicht-negative Kantengewichte, $c(e) \geq 0$, $e \in E$, vorausgesetzt.

Neben statisch gewichteten Graphen, gibt es auch dynamisch gewichtete Graphen, um eine realitätsnahe Modellierung zu ermöglichen:

Definition 3.4. In einem *zeitabhängigen Graphen (dynamisch gewichteten Graph)* $G = (V, E, c)$ wird jeder Kante $e \in E$ eine Abbildung $f_e : I \to \mathbb{R}$ zugeordnet, welche für jedes Zeitintervall $I \subseteq \mathbb{N}_0$ ein Kantengewicht definiert. Die Gewichtsfunktion

$$\begin{aligned} c : E &\to \mathcal{F} \subseteq \mathit{Abb}(I, \mathbb{R}) \\ e &\mapsto f_e(t) \end{aligned}$$

wird in diesem Fall als *zeitabhängiges Kantengewicht* bezeichnet. □

Um die dynamische Gewichtung hervorzuheben, definiert man den Graph auch als $G = (V, E, f)$ statt $G = (V, E, c)$.

Durch die Einführung von Kantengewichten lässt sich die Suche nach kürzesten Wegen präziser formulieren, da in der Regel nicht die Anzahl an durchlaufenen Kanten bzw. Knoten entscheidend ist, sondern beispielsweise die Länge des Weges oder die für den Weg benötigte Zeit mehr Aussagekraft hat. Beides kann erst durch Kantengewichte sinnvoll definiert und in das Modell eingearbeitet werden.

Mithilfe der Kantengewichte lässt sich nun auch das Kürzeste Wege Problem allgemein definieren [GS16, S. 428]:

Definition 3.5 (Kürzeste Wege Problem)**.** Gegeben sei ein gewichteter Digraph $G = (V, E, c)$. Das *Kürzeste Wege Problem (vgl. engl. Shortest Path, SP)* beschreibt die Suche nach einem Weg von einem Anfangsknoten $s \in V$ zu einem Endknoten $t \in V$ von minimaler Länge, das heißt, die Suche nach einem s-t-Weg, welcher die Summe der Kantengewichte entlang dieses Weges minimiert. □

Aus der Problemformulierung folgt insbesondere, dass ein kürzester s-t-Weg nicht eindeutig sein muss. Es kann auch mehrere Wege mit minimalem Kantengewicht geben. Sofern nicht weitere Bedingungen zu berücksichtigen sind, werden alle Lösungen des SP-Problems als gleichwertig angesehen.

Um mindestens eine Lösung des Problems zu gewährleisten, soll Folgendes vorausgesetzt werden [HRoJ, S. 2]:

1. Der Graph G enthält einen s-t-Weg. – Dies kann beispielsweise durch eine Tiefen- oder Breitensuche sichergestellt werden. (Für $s, t \in V$ beliebig aber fest bedeutet dies insbesondere, dass G zusammenhängend ist.)

2. Der Graph G enthält keine von s aus erreichbare Kreise mit negativer Kantengewichtung. – Dies verhindert, in eine Endlosschleife zu geraten.

3. Es genügt, den kürzesten s-t-Pfad zu finden, da dieser ein Weg ist. – Enthält der kürzeste s-t-Pfad Kreise, so haben diese nach vorherigem Punkt nicht-negative Länge. Nun können alle Kreise aus dem Pfad entfernt werden und der daraus resultierende Pfad ist dann ein s-t-Weg, der maximal genauso lang wie der ursprüngliche Pfad ist. Das wiederum bedeutet, dass der so gefundene s-t-Weg schon optimal sein muss und damit eine Lösung für das SP-Problem liefert.

Das SP-Problem wurde in den letzten Jahrzehnten eingehend untersucht und es gibt verschiedene Algorithmen, mithilfe derer ein kürzester Weg gefunden werden kann. Einer der bekanntesten Algorithmen ist der vom holländischen Mathematiker Edsger Djikstra 1959 entwickelte *Djikstra-Algorithmus*, der von einem gegebenen Startknoten $s \in V$ ausgehend den kürzesten Weg zu allen anderen Knoten $v \in V$ berechnet. [BGL00, S. 13], [HRoJ, S. 6]

Die Idee des Verfahrens ist ein *distance label*-System, welches jedem abgelaufenen Knoten die kürzeste (bisher gefundene) Entfernung vom Startknoten s zuweist. Diese wird als *temporäre Distanzmarke* bezeichnet. Alle Wege entlang der markierten Knoten liefern einen *Kürzeste Wege Baum*. Wenn es keinen kürzeren Weg zu einem temporär markiertem Knoten gibt, wird dieser *permanent* markiert. Das Verfahren endet, wenn alle Knoten permanent markiert sind.

Im vorliegenden Pseudocode verfügen alle Knoten in Q über eine temporäre Distanzmarke. Wird ein Knoten aus Q herausgenommen, gilt er als permanent markiert und seine Distanzmarke wird nicht mehr verändert.

Algorithm 3 Djikstra Algorithmus

```
1:  for all v ∈ V do
2:     d(v) ← ∞
3:     d(s) ← 0
4:     vorgaenger(s) ← s
5:     Q ← {s}
6:     while (Q ≠ ∅) do
7:        wähle u ∈ Q mit d(u) = min_{v∈Q} d(v)
8:        Q ← Q \ {u}
9:        for all v ∈ V mit e = (u, v) ∈ E do
10:          if d(v) > d(u) + c(u, v) then
11:             d(v) ← d(u) + c(u, v)
12:             vorgaenger(v) ← u
13:             Q ← Q ∪ {v}
14:          end if
15:       end for
16:    end while
17: end for
```

Die Distanzfunktion $d : V \to \mathbb{R}$ ordnet jedem Knoten seine Distanzmarke zu. Die Abbildung $vorgaenger : V \to V$ benennt den direkten Vorgängerknoten eines Knotens v auf dem kürzesten s-v-Weg.
Bei der Suche nach dem kürzesten Weg von s nach t liefert der Djikstra-Algorithmus mittels der Distanzfunktion im Zielknoten t die Länge des kürzesten Weges und mithilfe der $vorgaenger$-Funktion können die einzelnen Knoten, die auf diesem Weg liegen, zurückverfolgt werden. Dementsprechend kann der Algorithmus stoppen, sobald der Zielknoten t permanent markiert wurde.

Voraussetzung für die Anwendbarkeit des Djikstra-Algorithmus ist eine nicht-negative Kantengewichtung:

Satz 3.6. *Der Djikstra-Algorithmus liefert für zusammenhängende, gewichtete Graphen $G = (V, E, c)$ mit nicht-negativen Kantengewichten $c(e) \geq 0$, $e \in E$, den kürzesten s-t-Weg für alle $s, t \in V$.*

Für den Beweis des Satzes wird folgendes Lemma betrachtet [HRoJ, S. 7]:

Lemma 3.7. *Betrachtet wird der Djikstra-Algorithmus auf einem zusammenhängenden, gewichteten Graphen $G = (V, E)$ mit nicht-negativer Kantengewichtung und Startknoten $s \in V$.*

1. *In dem Moment, wenn ein Knoten u aus Q ausgewählt und entfernt wird (vgl.*

Algorithmus 3, Zeile 7-8), ist der kürzeste bis jetzt bekannte s-u-Weg schon gleich dem insgesamt kürzesten s-u-Weg.

2. *Wenn u aus Q entfernt wurde, wird der Knoten nicht mehr zur Menge Q hinzugefügt.*

Beweis. (Lemma 3.7)
Zu Lemma 3.7 – 1.
Die Aussage soll mit vollständiger Induktion über die Anzahl an Iterationen der *While*-Schleife gezeigt werden. Um die Funktionsweise des Algorithmus besser nachvollziehen zu können, werden im Induktionsanfang die Fälle $n = 1$ und $n = 2$ betrachtet.

Induktionsanfang.

n = 1. Zu Beginn enthält Q nur den Startknoten s. Dementsprechend wird s in der ersten Wiederholung der *While*-Schleife ausgewählt und entfernt. Somit ist der kürzeste bis jetzt bekannte s-s-Weg, der Weg, der in s startet und endet, ohne den Knoten zu verlassen. Er hat also die Länge Null. Dieser Weg muss dann aber schon gleich dem insgesamt kürzesten s-s-Weg entsprechen, da es nach Voraussetzung keine negativen Kantengewichte gibt.

n = 2. Q beinhaltet nun alle Nachbarn v von s. Unter diesen Knoten wird derjenige Knoten u ausgewählt, der alle bekannten kürzesten s-v-Wege minimiert ($d(u) = \min_{v \in Q} d(v)$). Es gilt $d(u) = d(s) + c(s,u) = c(s,u)$, da $d(s) = 0$. Der Knoten u wird aus Q entfernt und der kürzeste bekannte s-u-Weg entspricht der Kante, die s und u miteinander verbindet.

Angenommen, es gibt einen anderen s-u-Weg $p = (e_1, \ldots, e_k)$ mit kürzerer Distanz. ($e_1, \ldots, e_k \in E$, s ist Anfangsknoten und u ist Endknoten.)

Betrachte nun die erste Kante e_1 auf p. Wegen der Nicht-Negativität der Kantengewichte gilt $c(e_i) \geq 0$ für alle $i = 1, \ldots, k$.

$$\Rightarrow c(e_1) \leq c(p) = \sum_{i=1}^{k} c(e_i) \overset{\text{Ann.}}{<} c(s,u) = d(u)$$

Außerdem ist e_1 der kürzeste bekannte Weg von s zu einem Nachbarn $\tilde{v}$ von s

$$\Rightarrow d(\tilde{v}) = d(s) + c(s,\tilde{v}) \overset{d(s)=0}{=} c(s,\tilde{v}) = c(e_1)$$

Wegen $c(s,\tilde{v}) < c(s,u)$ gilt also auch $d(\tilde{v}) < d(u)$ und damit hätte $\tilde{v}$ aus Q anstelle von u ausgewählt werden müssen. Widerspruch!

Damit ist die Kante (s,u) schon der insgesamt kürzeste s-u-Weg.

Induktionsvoraussetzung.

Die Behauptung gelte für die n-te Iteration.

Induktionsschritt.

$n \curvearrowright n+1$. Q enthält alle Nachbarn von Knoten, für die der kürzeste bekannte Abstand bereits der insgesamt kürzeste Abstand ist (nach Induktionsvoraussetzung).

Jetzt wird der Knoten u_{n+1} gewählt mit $d(u_{n+1}) = \min_{v \in Q} d(v)$ und aus Q entfernt. Dabei ist $d(u_{n+1})$ die Länge des kürzesten bekannten s-u_{n+1}-Weges.

Zu Zeigen: Dieser Weg ist auch der insgesamt kürzeste s-u_{n+1}-Weg.

Annahme: Es gibt einen kürzeren s-u_{n+1}-Weg p^*.

Weil die Kantengewichte nicht-negativ sind, muss dann aber jedes Teilstück von p^* und jede Kante in p^* kleiner sein als $d(u_{n+1})$. Die Knoten, die auf p^* liegen, haben somit auch kleinere Distanzmarken als u_{n+1}. Weil der Graph zusammenhängend ist und der Algorithmus jeweils den Knoten mit der aktuell kleinsten Distanzmarke sucht, hätten nach Induktionsvoraussetzung schon vor dem Induktionsschritt alle Vorgängerknoten von u_{n+1} auf p^* besucht werden müssen.

Dann wiederum wäre $d(u_{n+1})$ schon früher upgedatet worden und der ausgewählte s-u_{n+1}-Weg wäre nicht einmal unter den bekannten s-u_{n+1}-Wegen minimal. Widerspruch!

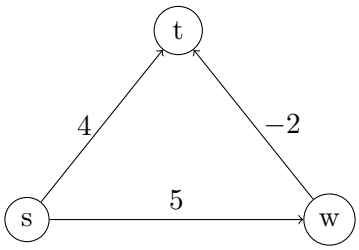
Es gibt also keinen kürzeren s-u_{n+1}-Weg, sodass die Behauptung folgt. □

Zu Lemma 3.7 – 2.
Die zweite Aussage folgt unmittelbar aus der ersten, denn in Zeile 13 des Algorithmus wird ein Knoten v nur dann zu Q hinzugefügt, wenn für ihn ein Weg mit kürzerer Entfernung zu s gefunden wurde, d.h. wenn seine Distanzmarke $d(v)$ verringert werden kann. Da aber der momentan gefundene kürzeste s-v-Weg dem insgesamt kürzesten s-v-Weg entspricht, kann auch kein kürzerer Weg und damit keine kleinere Distanzmarke mehr gefunden werden, sodass v nicht noch einmal zu Q hinzugefügt wird. □

Beweis. (Satz 3.6) Aus Lemma 3.7 folgt, dass der Dijkstra-Algorithmus zu jedem Knoten $v \in V$ aus dem Graphen $G = (V, E)$ den kürzesten Weg findet und dessen Länge in der permanenten Distanzmarke $d(v)$ speichert. Insbesondere liefert das Verfahren bei nicht-negativer Kantenbewertung also auch den kürzesten s-t-Weg und löst dadurch das SP-Problem. ⇒ Behauptung.
Außerdem folgt daraus, dass der Algorithmus stoppen kann, sobald t in Zeile 7 ausgewählt wird. □

Folgendes Beispiel zeigt, warum der Dijkstra-Algorithmus bei negativen Kantengewichten versagt:

Beispiel 3.8. Betrachtet wird folgender Graph mit negativer Kantengewichtung auf der Kante $e = (w, t)$:



Gesucht: der kürzeste Weg von s nach t

Lösung: Offensichtlich liefert der Weg von s über w nach t einen kürzeren Weg als der direkte Pfad von s nach t (denn $5 + (-2) < 4$). Da aber die Teilstrecke von s nach w länger ist als der direkte s-t-Weg, wird im Djikstra-Algorithmus der Knoten t schon vorzeitig permanent mit der Marke 4 gekennzeichnet und der Pfad über den Knoten w wird nicht weiter untersucht. Dadurch findet das Verfahren den kürzeren Weg nicht und liefert das falsche Ergebnis. □

Die Laufzeit des Djikstra-Algorithmus lässt sich durch die zweite Aussage in Lemma 3.7 leicht berechnen: Weil jeder Knoten nur einmal zu Q hinzugefügt und wieder entfernt wird, wird die *While*-Schleife nicht öfter als $|V|$ mal wiederholt. In jeder Runde werden höchstens $|V|-1$ Kanten zu benachbarten Knoten untersucht, sodass sich insgesamt eine Komplexität von $O(|V|^2)$ ergibt. [HRoJ, S. 8]

3.2. Algorithmen auf Graphen am Beispiel des ÖPNV

In diesem Abschnitt soll kurz skizziert werden, wie die theoretischen Konzepte der algorithmischen Graphentheorie auf den ÖPNV angewendet werden können. Diese Modellierungen werden in Kapitel 4 weiter vertieft.

Um Fragestellungen aus dem ÖPNV mathematisch modellieren zu können, bietet es sich an, die Liniennetz- und Fahrpläne eines Verkehrsverbundes als Graphen darzustellen. Dabei können Haltestellen den Knoten und Linien den Kanten im Graphen entsprechen. Für eine detailliertere Modellierung von Liniennetzen und Fahrplänen als Graphen sei an Kapitel 4 verwiesen.

Die Algorithmen zur Breiten- und Tiefensuche können beispielsweise dafür verwendet werden, eine Übersicht von allen Haltestellen zu erzeugen, die man auf dem Weg von Haltestelle A zu Haltestelle B passiert oder ebendiese Zwischenhalte zu zählen.

Während Kantengewichte im Allgemeinen auch negative Werte annehmen können, wird in der ÖPNV-Modellierung als Gewicht oftmals die (durchschnittlich) benötigte Fahr-

zeit angegeben, um vom Knoten u zum Knoten v zu gelangen. In diesem Zusammenhang werden nur nicht-negative Kantengewichte $c(e) \geq 0$, $e \in E$, angenommen. Dementsprechend liefert der Dijkstra-Alogrithmus für ein solches Modell eine korrekte Lösung für das SP-Problem.

Werden in der Modellierung von Fahrplänen die durchschnittlich benötigten Fahrzeiten betrachtet, so genügt ein statisch gewichteter Graph als Modellgrundlage.
In der Realität schwanken diese benötigten Fahrzeiten allerdings im Laufe eines Tages oder einer Woche mehrfach. So benötigt ein Bus zu den Hauptverkehrszeiten in der Regel mehr Zeit als mittags oder am späten Abend. Ebenso ist an Werktagen häufig mehr Verkehr auf den Straßen als an Wochenenden. Um diesen ungleichen Fahrzeiten gerecht zu werden, wird das Modell der Kantengewichtung oftmals vom statisch gewichteten zum zeitabhängigen Graphen erweitert.
In der vorliegenden Arbeit werden die Zeitintervalle I im zeitabhängigen Fall über bestimmte zeitliche Abschnitte aus den ÖPNV-Fahrplänen definiert, die in Sekunden angegeben sind. Ein Intervall kann beispielsweise eine Woche, einen Tag oder auch nur einige Stunden umfassen. Oftmals wird ein Intervall auf Basis des Fahrplans definiert und für die darauffolgenden Intervalle wird das ursprüngliche Intervall periodisch wiederholt.

4. Mathematische Modellierung der Passagierroutenoptimierung

Aus den Fahr- und Liniennetzplänen eines ÖPNV-Verbunds können Passagiere eine Route bestimmen, die angibt, mit welchem Bus sie an welcher Haltestelle zu welcher Uhrzeit fahren müssen, um von A nach B zu gelangen. In vielen Fällen wird ein Umstieg von einer Linie zu einer anderen unvermeidbar sein. Je nach Größe des Verkehrsnetzes kann die händische Suche nach der optimalen Reiseroute zur Sisyphusarbeit werden. Doch mittlerweile gibt es eine Vielzahl an Software-gestützter Verfahren, welche schnell die gewünschten Lösungen liefern.
In diesem Kapitel wird eine mögliche mathematische Modellierung für die Frage nach dem schnellsten Weg von A nach B erläutert. Hierfür muss zuerst ein Graph definiert werden, der die Informationen aus dem ÖPNV-Fahrplan abstrahiert. Auf diesem Graphen kann im Anschluss die Passagierroutenplanung als Kürzeste Wege Problem formuliert werden. Eine Anpassung des Djikstra-Algorithmus liefert schließlich einen Lösungsweg.

4.1. Modellierung des Graphen

Ziel dieses Abschnitts ist es, alle notwendigen Informationen aus den Liniennetz- und Fahrplänen eines Verkehrsverbundes in einem Graphen derart einzubetten, dass auf diesem mithilfe der bereits skizzierten Such-Verfahren das Passagierrouting optimiert werden kann.

Eine naheliegende Idee für die Modellierung der einzelnen Haltestellen ist, jede Haltestelle als einen Knoten im Graphen darzustellen. In einigen Fällen reicht dies aus, gelegentlich ist aber eine spezifischere Darstellung notwendig.
Beispielsweise ist bei der Untersuchung von Umstiegen die Betrachtung der einzelnen Haltepunkte notwendig, sodass in diesem Fall die Haltestelle nicht mit einem Knoten modelliert werden sollte, sondern für jeden Haltepunkt ein Knoten im Graphen erzeugt werden sollte.
Die genaue Bezeichnung der Knoten hängt also vom Modellierungsansatz ab und muss jeweils separat definiert werden. Apriori sollen im folgenden die einzelnen Haltestellen als Knoten genügen. Eine detaillierte Differenzierung findet entsprechend bei der jeweiligen Modellbeschreibung statt.

Obwohl in vielen Fällen eine Linie zwischen den Endhaltestellen in beide Richtungen gleichermaßen verkehrt, macht es Sinn, einen Digraphen als Modellgrundlage zu verwenden, dessen gerichtete Kanten die Verbindungen von aufeinanderfolgenden Haltestellen in Fahrtrichtung der einzelnen Linien wiedergeben. Für die Rückrichtungen werden entsprechend gerichtete Kanten in die entgegengesetzten Richtungen erzeugt. (vgl. Abbildung 4.1)

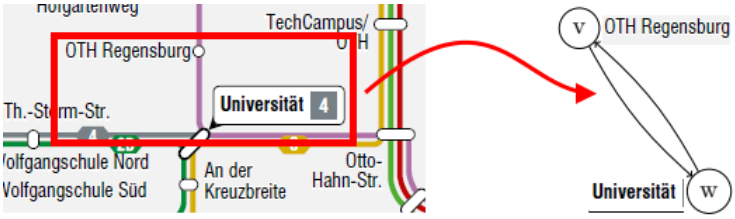


Abbildung 4.1: Linie 11 (lila) wird zwischen den beiden benachbarten Haltestellen OTH und Universität in zwei entgegengesetzt gerichtete Kanten aufgeteilt.

Dadurch können insbesondere auch Unterschiede der beiden Fahrtrichtungen einer Linie modelliert werden. Beispielsweise verkehrt die Linie A des Regensburger Busnetzes durch die Altstadt zwischen Altem Rathaus und Neupfarrplatz nur in eine Richtung wie in Abbildung 4.2 zu sehen ist. Eine korrekte Modellierung des Fahrtverlaufs ist dadurch nur mit einem gerichteten Graphen möglich.

Typischerweise wird der Dijkstra-Algorithmus zudem meist auf gerichteten Graphen definiert, obwohl er auch bei ungerichteten Graphen anwendbar ist.

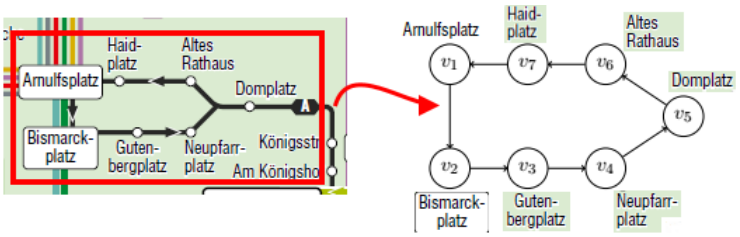


Abbildung 4.2: Linie A verkehrt zwischen Arnulfsplatz und Domplatz nicht auf derselben Strecke bei der Hin- und Rückfahrt.

Eine wichtige Komponente des Verkehrsnetzes aus Kundensicht sind die darin enthaltenen Modi, d.h. die verschiedenen Verkehrsmittel, die innerhalb des Netzes fahren. Bei Verkehrssystemen mit mehreren Modi erhöht sich die Komplexität des Passagierroutingproblems erheblich, da insbesondere die Modellierung eines Moduswechsels (z.B. der Umstieg von Bus zu U-Bahn) das Hinzuziehen mehrerer neuer Variablen erfordert.
Der Regensburger Stadtverkehr umfasst nur den Modus Bus, sodass sich diese Arbeit auf die Modellierung eines einzelnen Modus beschränkt. Zwar gibt es innerhalb des Bus-Modus verschiedene Fahrzeugtypen wie Single- oder Gelenkbusse, diese spielen aber für die Passagierrouting-Modellierung keine Rolle. Der multimodale Fall soll daher im Nachfolgenden nicht weiter untersucht werden.

Wie bereits erwähnt, sollen die benötigten Fahrzeiten als Kantengewichte dienen. Zwar wäre beispielsweise auch die zurückgelegte Fahrtstrecke ein möglicher Gewichtungsfaktor. Da aber bei der Benutzung öffentlicher Verkehrsmittel die benötigte Fahrzeit relevanter ist als die Länge der Fahrtstrecke, bietet sich die Betrachtung der Fahrzeiten in der Kantenbewertung besonders an.
In der Literatur gibt es für die Modellierung der Fahrzeiten und damit für die Definition der Kantengewichte zwei beliebte Modelle: [PSWZ04, S. 86]

- der *zeitexpandierte Graph* und
- der *zeitabhängige Graph*.

Modell des zeitexpandierten Graphen

Zunächst wird der Begriff *Ereignis* eingeführt:

Definition 4.1. Ein *Ereignis* ist ein Tupel (t, s) bestehend aus einem Zeitpunkt $t \in \mathbb{N}$ und einer Haltestelle $s \in S$, wobei S die endliche Menge aller Haltestellen im Liniennetz

beschreibt. Das *Ankunftsereignis* (t_a, s_a) definiert den Ankunftszeitpunkt eines Verkehrsmittels an der Station s_a, das *Abfahrtsereignis* (t_d, s_d) benennt den Abfahrtszeitpunkt eines Verkehrsmittels an der Station s_d. Die beiden Stationen s_a und s_d müssen nicht notwendigerweise verschieden sein. Falls in beiden Ereignissen dieselbe Haltestelle angesprochen wird (d.h. $s_a = s_d$) und zusätzlich dasselbe Verkehrsmittel involviert ist, oder das Erreichen des Anschlusses an dieser Haltestelle garantiert werden soll, muss zusätzlich $t_a \leq t_d$ vorausgesetzt werden. □

Der Zeitpunkt $t \in \mathbb{N}$ wird hierbei in Sekunden nach 00.00 Uhr angegeben (z.B. der Zeitpunkt 08.00 Uhr entspricht $t = 28.800$).

Im zeitexpandierten Graphen entspricht jeder Knoten einem Ereignis. Kanten einer Linie werden einerseits vom Abfahrtsereignis einer Station zum Ankunftsereignis der darauffolgenden Station gezogen (offensichtlich muss dabei $t_d \leq t_a$ gelten). In diesem Fall stellt eine Kante die Fahrt von einer Haltestelle zur nächsten dar. Andererseits verlaufen Kanten von den Ankunftsereignissen zu den Abfahrtsereignissen derselben Station, wenn dabei $t_a \leq t_d$ gilt. Dieser Fall beschreibt also das Warten auf die Fortsetzung der Fahrt an einer Haltestelle. Die Ereignisse müssen dabei nicht linienrein sein, sondern können auch als Umstiegskanten von einer Linie zu einer anderen dienen (vgl. Abbildung (4.3) als Beispiel[4]).

HBF/Albertstraße	**Linie 11**	**Linie 6**
Ankunftszeit	8:03 Uhr	08:07 Uhr
Abfahrtszeit	8:04 Uhr	08:08 Uhr

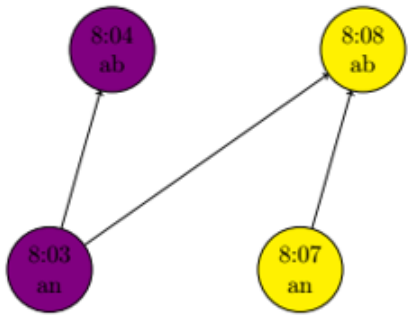


Abbildung 4.3: Zeitexpandierter Graph für die Ankunfts- und Abfahrtsereignisse an der Station HBF/Albertstraße für die Linien 6 und 11 zwischen 08.00 Uhr und 08.10 Uhr.

Die Kantengewichtung ergibt sich aus der Differenz der beiden Ereignisse, die durch die Kante verbunden werden. Es handelt sich also um einen statisch gewichteten Graphen. Da der Zeitpunkt im Endknoten nie kleiner als der im Startknoten ist, werden nur nichtnegative Kantengewichte angenommen.

[4]Die Fahrplanauszüge in Abbildung 4.3 und Abbildung 4.4 stammen aus den Fahrplandaten des RVV (siehe Anhang).

Um einen Linienwechsel an einer Station, d.h. einen Umstieg, präzise zu modellieren, ist es nötig, neben den Ereignissen Ankunft und Abfahrt ein weiteres Ereignis *Umstieg* zu ergänzen. Dabei wird die benötigte Umsteigezeit auf die Ankunftszeit addiert, woraus das Umsteigeereignis (t_u, s) an der Station s resultiert. Aus dieser Definition ergibt sich die Bedingung $t_a \leq t_u$. Wenn darüberhinaus $t_u \leq t_d$ gilt, kann der Anschluss erreicht werden. Statt die nicht-linienreinen Ankunfts- und Abfahrtsereignisse an einer Station s direkt durch eine Kante zu verbinden, werden neben den linenreinen Kanten nur Kanten von Ankunftsereignissen zu Umsteigeereignissen und weiter zu Abfahrtsereignissen gelegt (sofern die Bedingung $t_a \leq t_u \leq t_d$ erfüllt ist). Beispielhaft wird in Abbildung (4.4) das Modell aus Abbildung (4.3) erweitert.

HBF/Albertstraße	**Linie 11**	**Linie 6**
Ankunftszeit	8:03 Uhr	08:07 Uhr
Abfahrtszeit	8:04 Uhr	08:08 Uhr

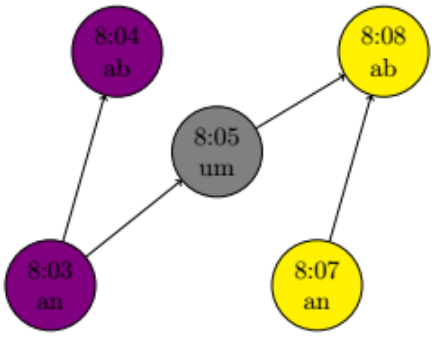


Abbildung 4.4: Zeitexpandierter Graph mit Umsteigeereignis und einer Umsteigezeit von zwei Minuten.

Für eine detaillierte Modellbeschreibung sei auf [Paj09, S. 19-21] verwiesen.

Modell des zeitabhängigen Graphen

Wie aus Definition 3.4 bekannt ist, zeichnet sich ein zeitabhängiger Graph durch dynamisch gewichtete Kanten aus.
Zur Modellierung (siehe [BJ04]) muss zuerst eine Zeitkomponente in einen Graphen $G = (V, E)$ integriert werden.

Definition 4.2. Ein *zeitlich festgelegter Weg* in einem Graphen G setzt sich zusammen aus einer Folge von Knoten $v_1, \ldots, v_k \in V$ in G und einer Folge von Zeitpunkten $t_1, \ldots, t_k \in I$, für die $e = (v_i, v_{i+1}) \in E$ eine Kante in G ist und für die $t_{i+1} = f_e(t_i) + t_i$ gilt.
Der Knoten v_1 wird als *Startpunkt* zur *Abfahrtszeit* t_1 bezeichnet, v_k bildet den *Zielpunkt* mit *Ankunftszeit* t_k. □

Durch diese zeitliche Komponente wird aus der Kürzesten Wege Suche die Suche nach dem *frühest möglichen Ankunftszeitpunkt* (vgl. engl. *Earliest Arrival Problem, EAP*) bei gegebenen Start- und Zielpunkten s und d sowie einer festgelegten Abfahrtszeit t im Startpunkt. Das EAP liefert als Output einen zeitlich festgelegten Weg p von s nach d, bei dem die Ankunftszeit in d minimal ist.

Auf Grundlage dessen kann nun ein zeitabhängiger Graph $G = (V, E, f)$ konstruiert werden.

Definition 4.3. 1. Ein *Fahrplan* T ist eine Menge von ÖPNV-Verbindungen, die innerhalb eines bestimmten Zeitintervalls I gültig sind.

2. Unter einer *(ÖPNV-)Verbindung* versteht man ein Tupel aus Ereignissen (vgl. Definition 4.1) (e_1, e_2) mit einem Abfahrtsereignis $e_1 = (t_d, s_d)$ und einem Ankunftsereignis $e_2 = (t_a, s_a)$, wobei die Ankunftszeit t_a nicht früher sein darf als die Abfahrtszeit t_d (d.h. $t_a \geq t_d$), außer die Verbindung startet vor und endet nach Mitternacht.

3. Eine *Route* in T ist eine Folge von Ereignissen $e_1, \ldots, e_k$, welche abwechselnd aus einem Abfahrts- und einem Ankunftsereignis besteht. Dabei bilden zwei Ereignisse e_{2i-1}, e_{2i} aufeinanderfolgende Abfahrts- und Ankunftsereignisse eines Verkehrsmittels in T an zwei aufeinanderfolgenden Stationen und zwei Ereignisse e_{2i} und e_{2i+1} entsprechen einer Ankunft und einer Abfahrt an derselben Station (d.h. $s_{2i} = s_{2i+1}$). □

Definition 4.3 liefert einige Modelleinschränkungen. So kann durch die Konstruktion der Anschlussverbindungen an einer Station nicht unterschieden werden, ob die Passagiere im selben Verkehrsmittel bleiben oder das Fahrzeug wechseln. Außerdem bleibt die Möglichkeit unberücksichtigt, zu Fuß von einer Haltestelle zu einer anderen zu laufen. Weiterhin wird auch die für einen Linienwechsel benötigte Umsteigezeit an einer Haltestelle vernachlässigt.
In Unterkapitel 4.3 soll das Modell aus diesem Abschnitt um letzteren Aspekt erweitert werden.

Um auch Fahrplan-Verbindungen über den Zeithorizont I hinaus modellieren zu können, wird das Intervall periodisch wiederholt.
Als Zeitintervall $I = [0, 86.400]$ wird die Dauer eines Tages in Sekunden angegeben, wobei $i = 0$ der Uhrzeit 00.00 Uhr und $j = 86.400$ der Uhrzeit 24.00 Uhr entspricht. Überschreitet ein Fahrplan dieses Intervall, d.h. die Reise startet oder endet erst am darauffolgenden Tag, werden die Sekunden weitergezählt und mittels modulo-Rechnung $mod\, 86.400$ auf das periodisch wiederkehrende Tageszeitintervall I übertragen.
Die Betrachtung von mehrtägigen Reiserouten ist analog umsetzbar, wird aber in dieser Arbeit nicht weiter untersucht, da eine Route dieser Länge im relativ kleinen ÖPNV-

Gebiet von Regensburg unplausibel ist. Deshalb soll folgende Ungleichung gelten:

$$t_a < t_d + 86.400 \tag{4.1}$$

Genauer gesagt soll die Gewichtsfunktion f_e also periodisch sein, d.h. für alle $t \in \mathbb{N}_0$ gilt $f_e(t) = f_e(t \; mod \; 86.400)$.

Darüberhinaus wird für eine effiziente Implementierung des Algorithmus zusätzlich die stückweise Periodizität der Gewichtsfunktion gefordert:

Definition 4.4. Eine periodische Funktion $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ heißt *stückweise linear*, wenn sie aus einer endlichen Menge an Abschnitten besteht, auf denen die Funktion jeweils linear ist. Wenn f stückweise linear ist, kann die Funktion mit endlich vielen Interpolationspunkten $\mathfrak{p}_i \in \mathfrak{P}$ beschrieben werden. Diese enthalten die Abfahrtszeiten t_i zusammen mit den entsprechenden Funktionswerten $f(t_i)$ (Fahrzeiten). □

Für einen beliebigen Zeitpunkt t, der nicht unbedingt einer Abfahrtszeit aus dem Fahrplan entspricht, wird der Funktionswert von f mittels Interpolation berechnet.
Diese Interpolation ist allerdings nicht linear, sondern abhängig vom nächst größeren Interpolationspunkt t_i mit $t \leq t_i$. Bildlich gesprochen bedeutet dies, dass man an der Haltestelle warten muss, bis der nächste Bus abfährt. Ein Bus, der die Haltestelle bereits verlassen hat, spielt keine Rolle mehr. Der Funktionswert $f(t)$ entspricht dann der Summe aus Warte- und Fahrzeit. Die Interpolation ergibt sich daraus mit der Formel:

$$f(t) = -\gamma \cdot (t_i - t) + f(t_i)$$

Dabei beschreibt γ den fixierten Gradienten von f. Um die nachfolgende *FIFO*-Bedingung (Definition 4.6) zu gewährleisten, soll $\gamma \in [-1, 0]$ sein.
Weil f periodisch ist, kann es passieren, dass kein t_i mit $t_i \geq t$ existiert. Dies bedeutet, dass am selben Tag kein Bus mehr abfährt. In diesem Fall wird $t_i = t_0$ gesetzt, also dem ersten Abfahrtszeitpunkt des (Folge-)Tages. [Paj09, S. 14-15]

Aufgrund der Periodizität des Fahrplans kann es außerdem vorkommen, dass die Ankunftszeit vor der Abfahrtszeit liegt ($t_a < t_d$), nämlich genau dann, wenn eine Fahrt erst nach Mitternacht endet. In diesem Fall kann als Dauer bzw. Reisezeit Δ der ÖPNV-Verbindung nicht die Differenz $t_a - t_d$ betrachtet werden, sondern ergibt sich als Summe der beiden Zeitspannen zwischen t_d und Mitternacht sowie Mitternacht und t_a. Damit folgt insgesamt [Paj09, S. 18]:

$$\Delta(t_d, t_a) := \begin{cases} t_a - t_d & t_a \geq t_d \\ 86.400 - t_d + t_a & \text{sonst} \end{cases}$$

Mithilfe der Funktion Δ ergeben sich die Interpolationspunkte $\mathfrak{p} := (t_d, \Delta(t_d, t_a))$, das

bedeutet, dass die Funktion f_e im i-ten Interpolationspunkt t_i die Fahrzeit des i-ten Busses zwischen den entsprechenden Haltestellen s_d und s_a liefert. Zusammen mit einer eventuellen Wartezeit ergibt sich die Interpolationsformel (für $\gamma = -1$) [Paj09, S. 22]:

$$f_e(t) = \underbrace{(t_i - t)}_{\text{Wartezeit}} + \underbrace{f_e(t_i)}_{\text{Fahrzeit}}$$

Beispiel 4.5. In Abbildung 4.5 ist ein typischer Funktionsgraph (vgl. [Paj09, S. 22]) einer stückweise linearen, periodischen Funktion mit acht Interpolationspunkten zu sehen. Es ist zu erkennen, dass die Kante $e \in E$ von „schnellen“ und „langsamen“ Bussen befahren wird. Die schnellen Busse benötigen eine Fahrzeit von fünf Minuten und haben die Abfahrtszeiten 5:00, 9:00, 12:00 sowie 19:00 Uhr. Die langsamen brauchen für die Strecke entlang der Kante e zehn Minuten und fahren jeweils um 7:00, 14:00, 17:00 und 21:00 Uhr ab. Zwischen den Abfahrtszeiten der Busse müssen die Passagiere die zusätzliche Wartezeit miteinkalkulieren.

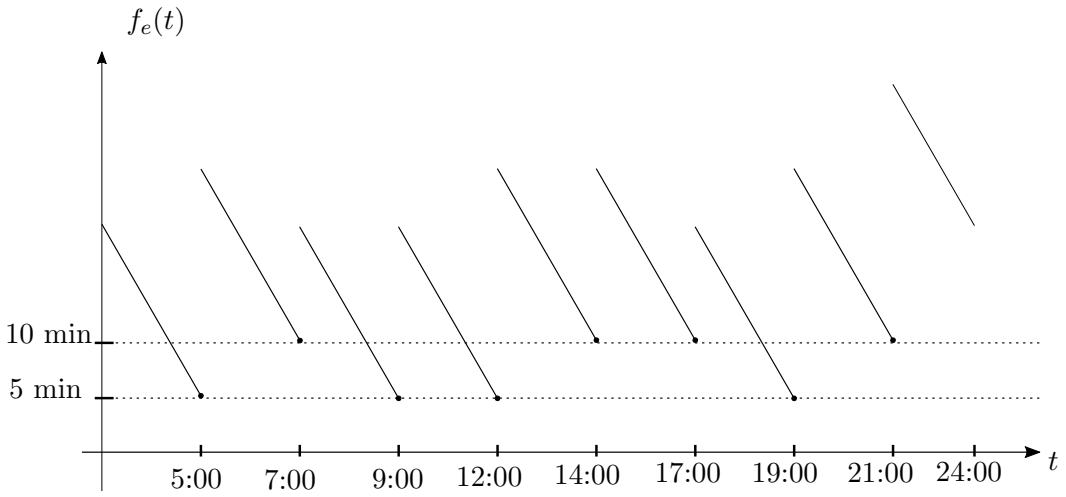


Abbildung 4.5: Stückweise lineare Funktion mit einer Periode von 24h.

Damit im zeitabhängigen Fall das EAP nicht NP-schwer wird, ist es nötig, die *FIFO*-Bedingung für die Gewichtsfunktionen vorauszusetzen. Diese besagt:

Definition 4.6 (FIFO-Bedingung)**.** Wenn für zwei Zeitpunkte $t_1, t_2 \in \mathbb{N}_0$ mit $t_1 < t_2$ stets $f_e(t_1) + t_1 < f_e(t_2) + t_2$ gilt, erfüllt die Funktion f_e die *FIFO*-Bedingung. □

Die *FIFO*-Bedingung stellt sicher, dass es entlang einer Kante keine Überholvorgänge gibt (*„first in first out“*). Es kann also nicht passieren, dass ein Fahrzeug eine Haltestelle

später verlässt, aber früher an der nächsten Station ankommt als ein anderes Fahrzeug. [Paj09, S. 9-10, 25]

Im Folgenden sei die Knotenmenge V die Menge aller Haltestellen, die im Fahrplan T enthalten sind. Die Kanten $e \in E$ von G setzen sich zusammen aus allen Haltestellentupeln $e = (u, v)$, die sich aus Verbindungen (t, u) sowie (t', v) in T ergeben.

Seien u und v zwei Haltestellen aus V. Dann definiert die Menge

$$C_{uv} = \{(t, t') : ((t, u), (t', v)) \in T\}$$

für jede Verbindung der beiden Haltestellen u und v aus dem Fahrplan T ein *Fahrzeit-Paar*, sofern eine solche Verbindung existiert. Andernfalls ist die Menge C_{uv} leer.
Daraus ergibt sich die Menge aller Reisezeiten $\Delta(C_{uv})$ der ÖPNV-Verbindungen zwischen u und v.

Mithilfe der Menge C_{uv} kann nun die Gewichtsfunktion definiert werden:

Definition 4.7. Gegeben seien ein Zeitpunkt $t \in I$ und zwei Knoten $u, v \in V$ in G mit einer Verbindung in T, die über die Kante $e = (u, v)$ dargestellt wird. Dann wird im zeitabhängigen Graphen das dynamische Kantengewicht über folgende Definitionsvorschrift[5] bestimmt:

$$f_e(t) = f_{uv}(t) = \min\{\Delta(C_{uv})\}$$

Genauer:

$$f_e(t) = f_{uv}(t) = \begin{cases} \min\{t'' : (t', t'') \in C_{uv})\} - t & t' \leq t'' \text{ und } t \leq t' \\ \min\{86.400 + t'' : (t', t'') \in C_{uv}\} - t & \text{sonst} \end{cases} \tag{4.2}$$

□

Anders als in [BJ04] liefert in dieser Arbeit die Gewichtsfunktion nicht den frühest möglichen Ankunftszeitpunkt an der Station v (dieser ist durch die Minimumsfunktion gegeben). Stattdessen beschreibt das dynamische Gewicht die (mindestens) benötigte Fahrzeit, um von Haltestelle u nach v zu gelangen, wie es auch im statischen Modell der Fall ist. Dies wird erreicht, indem vom frühest möglichen Ankunftszeitpunkt die Startzeit t in v abgezogen wird.

Proposition 4.8. *Die Gewichtsfunktion $f_e : I \to \mathbb{R}$ ist nicht-negativ und $f_e + t$ ist monoton steigend.*

Beweis. Zunächst ist zu bemerken, dass die Funktion f_e *wohldefiniert* ist. Da $t \in I$ liegt, gibt es tatsächlich – wie nach Voraussetzung gefordert – eine Verbindung von u

[5] Die hier definierte Fallunterscheidung gilt für alle weiteren Ergebnisse und Folgerungen. Um die Formulierung von Definitionen und Sätzen übersichtlicher zu gestalten, wird im Folgenden meist nur der erste Fall betrachtet.

nach v in T mit Ereignissen (t', u) und (t'', v), für die $(t', t'') \in C_{uv}$ und $t \leq t'$ oder $t' < t \leq 86.400 + t'$ gilt. (Es ist zu beachten, dass bei einem sehr späten Startzeitpunkt t die Reise eventuell erst nach Mitternacht startet, sodass in diesem Fall $t' < t \leq 86.400 + t'$ gilt.)
Es wird, wie in der Ungleichung (4.1) erwähnt, angenommen, dass es aufgrund der Größe des ÖPNV-Gebiets in Regensburg keine Reisezeiten gibt, die 24 Stunden übersteigen ($t_a < t_d + 86.400$). Daher wurde im zweiten Fall der Gleichung (4.2) – welche den Fall beschreibt, dass die Reise erst am Folgetag startet und / oder endet – auf die Möglichkeit verzichtet, dass die Route in t', mit $t' < t \leq 86.400 + t'$ startet und dann erst an einem späteren Tag endet.
Damit ist die Menge, über die die Minimumsfunktion gebildet wird, nie leer, sodass die *min*-Funktion in beiden Fällen eine natürliche Zahl $t'' \in \mathbb{N}_0$ als frühest möglichen Ankunftszeitpunkt liefert. Somit folgt die Wohldefiniertheit von f_e.

Für den Beweis der Nicht-Negativität wird zuerst der Fall betrachtet, dass Abfahrt und Ankunft noch am selben Tag stattfinden, sodass der erste Fall der Gleichung (4.2) ($t' \leq t''$ und $t \leq t'$) greift.
Zusammen ergibt sich $t \leq t' \leq t''$, also insbesondere $t \leq t''$, woraus die Nicht-Negativität von f_e folgt.
Für den zweiten Fall gilt ebenfalls die Nicht-Negativität, da nach Voraussetzung t in I liegt, d.h. $t \in [0, 86.400]$ und die Minimumsfunktion wegen $t'' \in I$ einen Funktionswert > 86.400 liefert.

Eine Funktion f heißt monoton steigend, wenn für alle $t \leq \tilde{t}$ gilt: $f(t) \leq f(\tilde{t})$. Warum ist $f_e + t$ monoton steigend?
Es sei $t \leq \tilde{t}$. Zu zeigen: $f_e(t) + t \leq f_e(\tilde{t}) + \tilde{t}$.
Diese Ungleichung ist erfüllt, weil die *FIFO*-Bedingung (Definition 4.6) für die Gewichtsfunktion vorausgesetzt wurde.
Zum Verständis wird die Ungleichung zusätzlich für den ersten Fall von (4.2) explizit überprüft. Der zweite Fall folgt analog.
Es gilt:

$$\begin{aligned} f_e(t) + t &= \min\{t'' : (t', t'') \in C_{uv} \textit{ und } t \leq t'\} - t + t \\ &= \min\{t'' : (t', t'') \in C_{uv} \textit{ und } t \leq t'\} \end{aligned} \tag{4.3}$$

$$\begin{aligned} f_e(\tilde{t}) + \tilde{t} &= \min\{\tilde{t''} : (\tilde{t'}, \tilde{t''}) \in C_{uv} \textit{ und } \tilde{t} \leq \tilde{t'}\} - \tilde{t} + \tilde{t} \\ &= \min\{\tilde{t''} : (\tilde{t'}, \tilde{t''}) \in C_{uv} \textit{ und } \tilde{t} \leq \tilde{t'}\} \end{aligned} \tag{4.4}$$

Um zu zeigen, dass (4.3) $\leq$ (4.4) gilt, wird folgende Fallunterscheidung betrachtet:

1. Fall:
Das Minimum t''^* von (4.3) wird für (t'^*, t''^*) mit $t \leq \tilde{t} \leq t'^*$ angenommen.
In diesem Fall liefern (4.3) und (4.4) dasselbe Ergebnis. D.h. es gilt (4.3) = (4.4), also insbesondere (4.3) $\leq$ (4.4).

2. Fall:
Das Minimum t''^* von (4.3) wird für (t'^*, t''^*) mit $t \leq t'^* < \tilde{t}$ angenommen.
Wenn das Minimum an dieser Stelle angenommen wird, gilt $\forall t' \geq \tilde{t}$: t'' mit (t', t'') ist nicht kleiner als t''^*, da sonst t''^* nicht minimal wäre. Damit ist aber auch das Minimum in (4.4) nicht kleiner als t''^*, sodass auch in diesem Fall (4.3) $\leq$ (4.4) gilt, womit die Monotonie von $f_e + t$ folgt.

□

Mit Proposition 4.8 folgen zwei wichtige Beobachtungen:

Lemma 4.9 (Zeitlich festgelegte Wege sind tatsächlich Wege.)**.** *Sei $G = (V, E, f_e)$ ein endlicher zeitabhängiger Graph mit einer nicht-negativen Gewichtsfunktion f_e, für die $f_e(t) + t$ monoton steigend für alle $t \in T$ ist. Weiter gebe es einen zeitlich festgelegten Kantenzug p in G, der in s zum Zeitpunkt t startet und d zur Zeit t' erreicht. Dann gibt es auch einen zeitlich festgelegten Weg $\tilde{p}$ in G, der in s zur Zeit t startet und d zum Zeitpunkt t'' erreicht mit $t'' \leq t'$.*

Beweis. Um die Behauptung zu zeigen, wird folgende Fallunterscheidung betrachtet:

1. Fall:
Der zeitlich festgelegte Kantenzug p ist selbst ein Weg. Dann folgt die Behauptung unmittelbar, indem man $p = \tilde{p}$ setzt.

2. Fall:
Der zeitlich festgelegte Kantenzug p ist selbst kein Weg. In diesem Fall existieren zwei Knoten v_i und v_j in der Folge von Knoten $v_1, \ldots, v_k \in V$ ($v_1 = s$, $v_k = d$) aus p mit $v_i = v_j$. O.B.d.A. sei $i < j \in \mathbb{N}$. Da v_i und v_j dieselbe Haltestelle beschreiben, können aus p die Knoten $v_{i+1}, \ldots, v_{j-1}$ herausgenommen werden und p liefert weiterhin einen zeitlich festgelegten Kantenzug von s nach d. Dieses Wegnehmen von überflüssigen Knoten wiederholt man so lange, bis alle Haltestellen paarweise verschieden sind und nennt den resultierenden Weg $\tilde{p}$. Weil f_e nicht-negativ ist, wird beim Weglassen von Kanten die Gesamtfahrzeit nicht erhöht, sodass die Abfahrtszeit t_j in v_j sicher erreicht wird. Insbesondere kann dadurch aber auch der Anschluss von v_j nach v_{j+1} erreicht werden, sodass sich durch Streichen der Folgeglieder die Ankunftszeit in d nicht erhöht. Genauer:

Fall 2a:

Durch das Streichen der Folgeglieder ergeben sich keine Verbindungen, die eine frührere Ankunftszeit am Zielort d liefern. Dann erreicht $\tilde{p}$ den Zielort d zum Zeitpunkt $t'' = t'$.

Fall 2b:

Durch das Streichen der Folgeglieder wird es möglich, eine frühere Verbindung zu einer Haltestelle v_m mit $j < m \leq k$ zu nehmen (d.h. $\tilde{t}_m \leq t_m$). Wegen der Monotonie von $f_e(t) + t$ für alle $t \in T$ folgt damit auch im Zielpunkt d: $t'' \leq t'$.

□

Korollar 4.10 (Die Lösung des EAP ist wohldefiniert.)**.** *Sei $G = (V, E, f_e)$ ein endlicher zeitabhängiger Graph mit einer nicht-negativen Gewichtsfunktion f_e, für die $f_e(t) + t$ monoton steigend für alle $t \in T$ ist. Betrachte zwei Knoten $s, d \in V$ und eine Abfahrtszeit $t \in T$. Dann existiert ein Zeitpunkt $t' \in T$, der der minimalen Ankunftszeit an der Haltestelle d aller zeitlich festgelegten Wege von s nach d entspricht.*

Beweis. Folgt unmittelbar aus Lemma 4.9. □

Mithilfe der Gewichtsfunktion f_e (und der darin enthaltenen Minimums-Abbildung) lassen sich jetzt möglichst schnelle Reiserouten konstruieren:

Proposition 4.11. *Man findet zu einer beliebigen Route R in T (vgl. Definition 4.3) einen zeitlich festgelegten Weg p in G (vgl. Definition 4.2), auf dem man nie später an einer bestimmten Haltestelle ankommt als über die Route R. Dadurch erreicht man auch einen vorgegebenen Zielpunkt d nie später über den zeitlich festgelegten Weg p als über irgendeine andere Route R.*

Beweis. Sei R eine Route in T. D.h. R ist eine Folge von Abfahrts- und Ankunftsereignissen $e_1, \ldots, e_k$, $k \in \mathbb{N}$. OBdA ist $k = 2l$ gerade, da die Route sonst nicht mit einem Ankunftsereignis enden würde. Weil Abfahrts- und Ankunftsereignisse alternieren, gibt es jeweils l Abfahrts- und Ankunftsereignisse. Damit durchläuft die Route R $l+1$ Haltestellen $v_1, \ldots, v_{l+1}$. Denn e_1 sowie e_k enthalten Haltestellen, die auf der Route nur einmal enthalten sind, e_{2i} und e_{2i+1}, $i = 1, \ldots, l-1$ beschreiben jeweils dieselben Haltestellen. Nun wird mit den Knoten $v_1, \ldots, v_{l+1}$ ein zeitlich festgelegter Weg p konstruiert, wobei anders als auf der Route R nicht beliebige Verbindungen mit Abfahrtsort v_j und Ankunftsort v_{j+1}, $j = 1, \ldots, l$, gewählt werden dürfen, sondern nur diejenigen, die die Bedingung $t_{j+1} = f_e(t_j) + t_j$ für die Verbindungskante (v_j, v_{j+1}) erfüllen.
Dadurch werden für p immer genau die Verbindungen bestimmt, die die frühest mögliche Ankunftszeit an der nachfolgenden Haltestelle liefern. Also folgt, dass man mit p nie später an einer Haltestelle ankommen kann als mit R. Insbesondere kann man also auch einen vorher festgelegten Zielort $d \in V$ nie später erreichen. □

Umgekehrt ist hervorzuheben, dass zu einem zeitlich festgelgten Weg p in G die Gewichtsfunktion f_e zu jeder Kante $e = (v_i, v_{i+1})$ von p im ersten Teil (1) der Gleichung (4.2) einen Eintrag t_{i+1} aus dem Fahrplan T liefert. Werden diese Einträge mit den zugehörigen Haltestellen zu Ereignissen verknüpft, erhält man eine gültige Route in T.

Insgesamt findet man demzufolge eine Route im Fahrplan T, die am Startpunkt s zur Zeit t starten und zu d führen soll, indem man das EAP auf G mit ebendiesen Input-Daten löst.

Vergleich der zeitabhängigen und zeitexpandierten Modellansätze

Die Modellierungsansätze für den zeitexpandierten und den zeitabhängigen Graphen haben jeweils Vor- und Nachteile. Einige davon sollen im Folgenden kurz skizziert werden, bevor sich dann für einen der beiden Ansätze entschieden wird.

Der wohl offensichtlichste Vorteil des zeitexpandierten Modells gegenüber des zeitabhängigen Ansatzes ist die weniger aufwendige Konstruktion des Graphen. Besonders die Bestimmung und Einbettung der dynamischen Kantengewichte intensivieren den Aufwand im zeitabhängigen Graphen im Gegensatz zu den statischen Gewichten im zeitexpandierten Fall.

Durch diese statischen Gewichte ist der Djikstra-Algorithmus auf den zeitexpandierten Graphen direkt anwendbar. Dagegen muss das Verfahren im zeitabhängigen Fall etwas abgewandelt werden, damit der Djikstra-Algorithmus auch dort eine Lösung für das Earliest Arrival Problem liefert.

Mit der Verwendung der dynamischen Kantengewichte als stückweise lineare Funktionen geht zudem ein erhöhter Speicherbedarf einher. [Paj09, S. 26]

Der Hauptnachteil des zeitexpandierten Graphen ist dessen Größe. Indem für jedes Ereignis aus dem Fahrplan ein separater Knoten erzeugt wird, liefert dieses Modell eine sehr große Anzahl an Knoten und Kanten, was wiederum starken negativen Einfluss auf die Laufzeit des Djikstra-Algorithmus $O(|V|^2)$ (vgl. Unterkapitel 3.1, S. 23) hat. [PSWZ04, S. 86]

Da das Ziel dieser Arbeit ist, ein Verfahren für Passagierrouting-Optimierung im ÖPNV zu liefern, spielt die Algorithmus-Laufzeit eine übergeordnete Rolle. Deshalb wird im Folgenden der zeitabhängige Modellansatz weiter verfolgt, auch wenn dies mit einer etwas aufwendigeren Vorarbeit bei der Konstruktion des Graphen, einem erhöhten Speicherbedarf und der Anpassung des Algorithmus verbunden ist. In Experimenten von Pyrga et al. hat sich zudem gezeigt, dass der Laufzeit-Vorteil im Allgemeinen alle Nachteile aufwiegt. [PSWZ07, S. 38]

Ein weiterer Pluspunkt des zeitabhängigen Graphen ist außerdem, dass er sehr nah am Fahrplan selbst konstruiert wird und dadurch im Algorithmus nur ein kleiner Teil des

Fahrplans betrachtet werden muss. [BJ04, S. 4]

4.2. Problemformulierung und Anpassung des Algorithmus

Im Abschnitt über das Modell des zeitabhängigen Graphen wurde das EAP bereits informell eingeführt. An dieser Stelle soll das Problem noch einmal genau definiert werden:

Definition 4.12 (Earliest Arrival Problem – EAP)**.** Gegeben seien

- ein zeitabhängiger Digraph $G = (V, E, f_e)$,
- ein Startpunkt $s \in V$,
- ein Zielpunkt $d \in V$ und
- eine Startzeit $t \in I$ (frühest mögliche Abfahrtszeit in s).

Das *Earliest Arrival Problem (EAP, Suche nach dem frühest möglichen Ankunftszeitpunkt)* zu diesen Input-Daten beschreibt die Suche nach einem zeitlich festgelegten Weg $p = (v_1 \ldots, v_k)$ von $s = v_1$ nach $d = v_k$, der

- nicht vor dem Zeitpunkt t startet sowie
- die Ankunftszeit t' in d – und damit auch die mindestens benötigte Reisezeit von s nach d – minimiert.

Hierbei ergibt sich die mindestens benötigte Reisezeit als Differenz von der Ankunftszeit $t' = t_d$ am Zielpunkt d und der tatsächlichen Startzeit $t_s \geq t$ am Startpunkt s. Das heißt, die Summe der Kantengewichte f_e wird minimiert:

$$\min_{p} \sum_{\substack{t_i \\ s.d. \\ v_i \in p,\, (t_i, v_i) \in T \\ i=1,\ldots,k-1}} f_e(t_i)$$

□

Mit dieser Definition ist das EAP ein Spezialfall des Kürzeste Wege Problems aus Definition 3.5 und kann mithilfe des Djikstra-Algorithmus gelöst werden. Allerdings muss das Verfahren auf die dynamischen Kantengewichte angepasst werden [BJ04, S. 7-8]:

Die Kernidee ist die Verwendung einer *Vorrangwarteschlange* Q über alle Ereignisse (t, v) aus der Menge $I \times V$, welche mit der $extractMin$-Operation (Zeile 8) ein Tupel mit minimalem Zeitwert $t \in I$ auswählt. Genauer gesagt ist Q eine Vorrangwarteschlange von Zeitpunkten in I, die mit Knoten aus dem Graphen G zu Ereignissen im Fahrplan T kombiniert werden.

Algorithm 4 Djikstra-Algorithmus für zeitabhängige Graphen

1: **Input:** Graph $G = (V, E)$, für jede Kante $e \in E$ eine nicht-negative Gewichtsfunktion $f_e : I \to \mathbb{R}$ mit $f_e(t) + t$ monoton steigend $\forall t \in I$, Startpunkt $s \in V$, Zielpunkt $d \in V$ und Startzeit $t \in I$.
2: **for all** $v \in V$ **do**
3: $a(v) \leftarrow \infty$
4: **end for**
5: $a(s) \leftarrow t$
6: $S \leftarrow \emptyset$
7: $Q \leftarrow \{(t, s)\}$
8: **while** $(Q \neq \emptyset)$ **do**
9: $(t'', u) \leftarrow extractMin(Q)$
10: $S \leftarrow S \cup \{u\}$
11: stoppe *while*-Schleife, wenn $u = d$
12: **for all** $e = (u, v) \in E$ mit $v \notin S$ **do**
13: $t' \leftarrow f_e(a(u)) + a(u)$
14: **if** $a(v) > t'$ **then**
15: **if** $a(v) = \infty$ **then**
16: $insert(Q, t', v)$
17: **else**
18: $update(Q, t', v)$
19: **end if**
20: $a(v) \leftarrow t'$
21: $vorgaenger(v) \leftarrow u$
22: **end if**
23: **end for**
24: **end while**
25: $v \leftarrow d$
26: $p \leftarrow (a(v), v)$
27: **while** $v \neq s$ **do**
28: $v \leftarrow vorgaenger(v)$
29: $p \leftarrow (a(v), v).p$
30: **end while**
31: **Output:** zeitlich festgelegter Weg p

Damit ergibt sich die nachfolgende Abwandlung des Djikstra-Algorithmus aus Algorithmus 3.

Ähnlich wie im Algorithmus 3 (Djikstra-Algorithmus) fügen wir zu Beginn unser Startereignis der Vorrangwarteschlange Q (Zeile 6) hinzu und erweitern diese Menge nach und nach mit den weiteren Ereignissen. Das Label-Setting im Algorithmus 4 funktioniert folgendermaßen: Permanent markierte Knoten werden über $extractMin$ aus der Menge Q entfernt und anschließend in eine neue Menge S aufgenommen. Diese Menge S ist (vgl. nachfolgenden Beweis zu Satz 4.13) genau so konstruiert, dass keine optimierenden Updates mehr möglich sind. Zu Beginn ist S leer.
Die Distanzfunktion $d : V \to \mathbb{R}$ wird durch die Abbildung $a : V \to I$ ersetzt und ordnet nun den einzelnen Knoten ihre Zeitmarken zu.
In Zeile 10 wurde ein Abbruchkriterium für das Erreichen des Zielknotens d eingefügt. Die *insert*-Operation (Zeile 15) erweitert die Vorrangwarteschlange um ein Ereignis (t', v) mit temporärer Marke, welches mittels *update* (Zeile 17) solange verbessert wird, bis der frühest mögliche Ankunftszeitpunkt in v erreicht ist.

Warum liefert der Algorithmus 4 einen korrekten Output?
Wie beim originalen Djikstra-Algorithmus die Nicht-Negativität der Kantengewichte Voraussetzung für die richtige Funktionsweise des Verfahren ist, funktioniert auch dieser Algorithmus nur unter bestimmten Bedingungen:

- f_e ist eine nicht-negative Funktion und
- $f_e(t) + t$ ist monoton steigend für alle $t \in I$.

Satz 4.13. *Die Funktionsweise des Djikstra-Algorithmus für zeitabhängige Graphen (Algorithmus 4) ist unter den oben erwähnten Bedingungen korrekt.*

Beweis. Vgl. [BJ02, S. 4].
Definiere $\delta(s, t, u)$ als frühest möglichen Ankunftszeitpunkt am Knoten u bei gegebener Startzeit t im Startknoten s, d.h. $\delta(s, t, u)$ ist die minimale Ankunftszeit aller zeitlich festgelegten Wege p von s nach u, die zum Zeitpunkt t starten.
Die Korrektheit des Algorithmus folgt aus der Tatsache, dass die Markierung

$$a(u) \leftarrow \delta(s, t, u)$$

für alle Knoten $u \in S$ vor und nach der Ausführung des Schleifenkörpers in der *while*-Schleife gleich bleibt, d.h. die Menge S enthält alle Knoten mit permanenter Markierung. Durch die Funktionsweise der Label-Updates im Algorithmus, werden die permanenten Markierungen nicht verändert, sodass für die Abbildungen im Verfahren gilt:

- Die Funktion *vorgaenger*() definiert einen gerichteten Baum R in G mit Kanten in Richtung der Wurzel s.

- Sei p' ein Weg, der nur Kanten von R in umgekehrter Richtung durchläuft. Dann liefert $a(\cdot)$ eine Zeitmarke, die zusammen mit p' einen zeitlich festgelegten Weg p bildet, welcher zur Zeit t in s startet. Es existiert dann für jeden Knoten $u \in R$ ein solcher Weg p.
- $a(u) \geq \delta(s, t, u)\ \forall u \in R$, d.h. die Zeitmarke kann nie kleiner als die frühest mögliche Ankunftszeit sein.
- Für alle permanent markierten Knoten $u \in S$ gilt sogar Gleichheit: $a(u) = \delta(s, t, u)$, d.h. die Zeitmarke entspricht der minimalen Ankunftszeit.

Denn angenommen, es gibt einen Knoten $u \in S$ mit

$$a(u) > \delta(s, t, u) = \delta. \tag{4.5}$$

OBdA sei u der erste Knoten aus dem Algorithmus, für den die Ungleichung (4.5) gilt. Insbesondere gilt also für alle anderen Knoten $v \in S \setminus \{u\}$:

$$a(v) = \delta(s, t, v). \tag{4.6}$$

Weiter sei p ein Weg, der das EAP von s nach u zur Startzeit t löst.
Für die Knoten v in p definiere $p(v)$ eine Abbildung, die angibt, zu welcher Zeit der Weg p den Knoten v passiert.
Nun sei y der erste Knoten in p, der nicht in S enthalten ist und x sei der Vorgänger von y in p (insbesondere gilt also: $x \in S$).
Wegen (4.6) gilt $a(x) = \delta(s, t, x)$. Da p einen Teilweg von s nach x zur Startzeit t induziert, der zum Zeitpunkt $p(x)$ den Knoten x erreicht, gilt außerdem:

$$\delta(s, t, x) \leq p(x) \tag{4.7}$$

Als x zu S hinzugefügt wurde, wurde im Algorithmus ein Update von $a(y)$ gemacht. Da aber kein Update eine bestehende Markierung erhöht, gilt:

$$a(y) \leq f_{xy}(\delta(s, t, x)) + \delta(s, t, x) \tag{4.8}$$

Mit der Monotonie von $f_{xy}(\tilde{t}) + \tilde{t}$ für alle $\tilde{t} \in I$ ergibt sich zusammen mit (4.7):

$$f_{xy}(\delta(s, t, x)) + \delta(s, t, x) \leq f_{xy}(p(x)) + p(x) = p(y). \tag{4.9}$$

Aufgrund der Nicht-Negativität der Gewichtsfunktionen können die Ankunftszeiten auf dem restlichen Weg von y nach u nicht kleiner werden, sodass

$$p(y) \leq p(u) = \delta \tag{4.10}$$

folgt.
Insgesamt erhält man damit:

$$\begin{aligned} a(y) &\overset{(4.8)}{\leq} f_{xy}(\delta(s,t,x)) + \delta(s,t,x) \\ &\overset{(4.9)}{\leq} p(y) \\ &\overset{(4.10)}{\leq} \delta \\ &\overset{(4.5)}{<} a(u). \end{aligned}$$

Für $u = y$ steht dies im Widerspruch zur Wohldefiniertheit der Funktion a.
Für den Fall $u \neq y$ hätte der Algorithmus wegen der $extractMin$-Operation den Knoten y statt u als Nächstes zur Menge S hinzugefügt, was wegen der Definition von u und y ($u \in S$, $y \notin S$) auch zum Widerspruch führt.
Es gibt also keinen Knoten in S, der die Ungleichung (4.5) erfüllt, sodass die permanenten Marken tatsächlich die minimalen Ankunftszeiten liefern und damit das EAP gelöst werden kann.

Dies ist auch der Grund, warum nicht überprüft werden muss, ob beim Label-Vergleich $a(v) > t'$ (Zeile 13) der Knoten $v \in S$ enthalten ist. Da in S nur permanent markierte Knoten enthalten sind, findet man auch keinen Knoten, bei dem sich die Label noch verbessern lassen. □

Bei der Ermittlung des Speicherbedarfs ist vor allem die Kantendichte des Graphen relevant. So werden bei einem vollständigen Graphen bereits im Laufe der ersten While-Schleife alle Knoten aus V gespeichert, da s in S hinterlegt wird und alle Nachbarn von s – also im vollständigen Fall alle $v \in V \setminus \{s\}$ – zu Q hinzugefügt werden. Diese Knoten werden auch in späteren Iterationen nicht mehr aus dem Speicher entfernt, sie wandern höchstens von der Menge der temporär markierten Knoten Q in die Menge der permanent markierten Knoten S.
In der Realität wird der betrachtete Graph, welcher das Liniennetz des ÖPNV-Gebiets abbildet, nicht vollständig sondern dünn sein, d.h. eine geringe Kantendichte aufweisen. In diesem Fall wird der maximale Speicherbedarf in den letzten beiden Iterationen der While-Schleife erreicht. Dann befinden sich nämlich im Speicher alle Knoten u, deren Distanz zu s nicht größer ist als der Abstand von d zu s, sowie deren Nachfolgerknoten v mit $(u, v) \in E$.
Es ist unschwer zu erkennen, dass diese Speichernutzung nicht sehr effizient ist, denn zum einen befinden sich auch Knoten im Speicher, die von s ausgehend in einer ganz anderen Richtung als d liegen und damit sicher kein Teil des kürzesten s-d-Weges sein können. Zum anderen werden auch jene Knoten $\tilde{u}$ nicht mehr aus dem Speicher entfernt, die für kein $u \in V$ mit $u \neq \tilde{u}$ auf einem kürzesten s-u-Weg liegen. Inwieweit die Speichernutzung

optimiert werden kann, wird bei der Vorstellung der Speed-Up Techniken in Kapitel 7 skizziert.

4.3. Umstiege als Modellerweiterung

Das bisherige Modell des zeitabhängigen Graphen macht die nicht sehr realistische Annahme, dass beim Wechsel von einem Fahrzeug zum nächsten keine zusätzliche Umsteigezeit benötigt wird. Für eine bessere Modellierung soll deshalb im Folgenden die Umsteigezeiten als Fußwege an Haltestellen eingeführt werden. Hierfür werden zwei verschiedene Konstruktionsansätze, die in [BJ04, S. 13] und [Paj09, S. 23-24] skizziert werden, erläutert. Nach einem Vergleich der beiden Ansätze wird der Algorithmus entsprechend angepasst.

Zunächst werden als Knoten nicht mehr nur die einzelnen Haltestellen betrachtet. Stattdessen wird eine mastenscharfe Modellierung über Haltepunkte eingeführt. Entsprechend verkehren die Busse nicht mehr auf den Kanten zwischen den Haltestellen, sondern auf Kanten zwischen ihren Haltepunkten an den verschiedenen Haltestellen, sodass ein detaillierteres Modell entsteht. Zwar kommt es dadurch unmittelbar zu einer Laufzeitsteigerung (jede Haltestelle besteht aus mindestens einem Haltepunkt), diese soll aber für eine realistischere Modellierung in Kauf genommen werden.
Es ist zu beachten, dass trotz der Verwendung von Haltepunkten anstelle von Haltestellen die Größe des Graphen im zeitabhängigen Fall immer noch deutlich kleiner ausfällt als im zeitexpandierten Ansatz, in dem für jedes Ereignis ein eigener Knoten konstruiert wird.

Die einzelnen Haltepunkte einer Haltestelle werden nun mit Kanten verbunden, welche den Fußwegen zwischen verschiedenen Steigen entsprechen sollen. Dabei können die Haltepunkte entweder direkt miteinander zu einem vollständigen Teilgraphen verbunden werden oder die Kanten laufen sternförmig zu einem zentralen Haltestellenknoten S – sozusagen eine Wartehalle – und von dort weiter zu den Haltepunkten (vgl. Abbildung 4.6).

Obwohl sich durch die Einführung des zentralen Haltestellenknotens S die Anzahl der Haltestellen zusätzlich erhöht, lohnt sich dieses Konzept vor allem bei Haltestellen mit vier oder mehr Haltepunkten, da sich dadurch die Anzahl der Kanten deutlich reduzieren lässt. Denn für einen vollständigen, ungerichteten Graphen mit n Knoten $v_1, \ldots, v_n \in V$ setzt sich die Kantenmenge aus den zwei-elementigen Teilmengen von V zusammen [Ove08, S. 2], d.h. der vollständige Umsteigegraph hat als Digraph eine Kantenmenge der Mächtigkeit $|E| = 2 \cdot \binom{n}{2} = 2 \cdot \frac{n(n-1)}{2} = n(n-1)$. Dagegen hat der sternförmige Umsteigegraph jeweils zwischen jedem Haltepunkt-Knoten $v_1, \ldots, v_n \in V$ und dem zentralen Haltestellenknoten S zwei entgegengesetzt gerichtete Kanten, also insgesamt $|E| = 2n$.

Abbildung 4.6: Sternförmiger und vollständiger Umsteigegraph ohne Kantengewichtung.

Dass $2n < n(n-1)$ für $n \geq 4$, $n \in \mathbb{N}$ gilt, lässt sich mit vollständiger Induktion leicht nachrechnen.

Die neu erzeugten Kanten müssen nun noch gewichtet werden.
Beim sternförmigen Umsteigegraphen ist die Idee, dass das Verlassen eines Fahrzeugs keine Zeit in Anspruch nimmt, sondern nur der Umsteige-Weg gewichtet werden muss. Um dies graphentheoretisch umzusetzen, wird jede Kante, die von einem Haltepunktknoten zu S verläuft, mit dem konstanten Kantengewicht 0 versehen. Die entgegen gesetzt gerichteten Kanten von S zu den Haltepunktknoten werden mit der Zeit versehen, die benötigt wird, um zum jeweiligen Steig zu gelangen (vgl. beispielhaft die linke Seite von Abbildung 4.7).

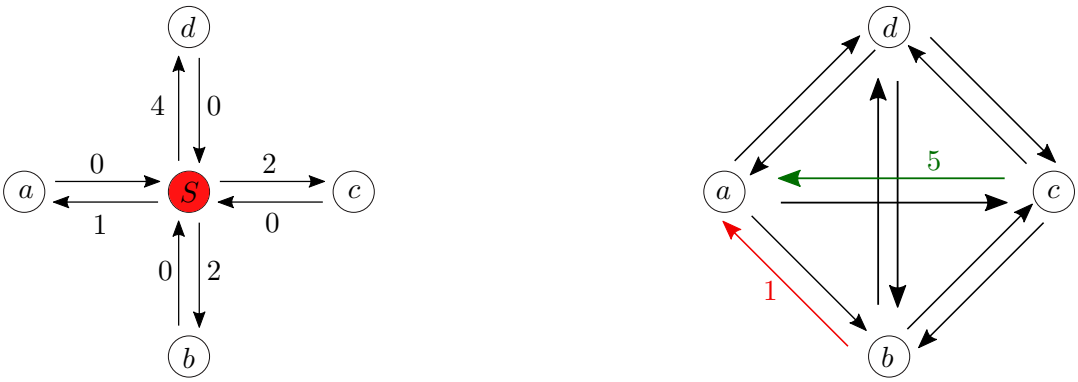


Abbildung 4.7: Sternförmiger und vollständiger Umsteigegraph mit Kantengewichtung.

Es ist hervorzuheben, dass dadurch jedem Haltepunkt eine *konstante* Umsteigezeit zugeordnet wird, sodass die relative Lage zweier Haltepunkte unberücksichtigt bleibt. Dass

dies nicht sehr realistisch ist, sieht man in Abbildung 4.8, die die Haltestelle *HBF/Albertstraße* aus dem Regensburger ÖPNV-Gebiet zeigt. Während man vom benachbarten Haltepunkt B nur eine Minute braucht, um zu A zu kommen, benötigt man vom Steig D drei Minuten. Diese relativen Umsteigezeiten können im konstanten Modell des sternförmigen Umsteigegraphen nicht realisiert werden.
Obwohl es durchaus möglich ist (siehe Abbildung 4.7), jedem Haltepunkt einer Haltestelle eine eigene konstante Umsteigezeit zuzuordnen, ist dieses Verfahren relativ ungünstig. Denn es hat zur Folge, dass oftmals die Dauer zweier entgegengesetzt gerichteter Fußwege stark voneinander abweichen kann. Beispielsweise benötigt man in Abbildung 4.7 von Steig a zu Steig d vier Minuten, während man für den umgekehrten Weg von d nach a nur eine Minute braucht. Dass man aber für den gleichen Weg unterschiedlich lange braucht, nur weil man ihn anders herum geht, ist nur selten realistisch.
Aus diesem Grund wird das Modell mit sternförmigen Umsteigegraphen dahingehend vereinfacht, dass nicht jeder Haltepunkt, sondern jede *Haltestelle* eine *konstante Umsteigezeit* zugewiesen bekommt, sodass alle Kanten, die aus S hinausführen, dieselbe Gewichtung haben. Zwar ist damit keine Lösung für das Problem der relativen Umsteigezeiten gefunden, aber es wird auf eine zu detaillierte Modellierung ohne Mehrwert verzichtet.

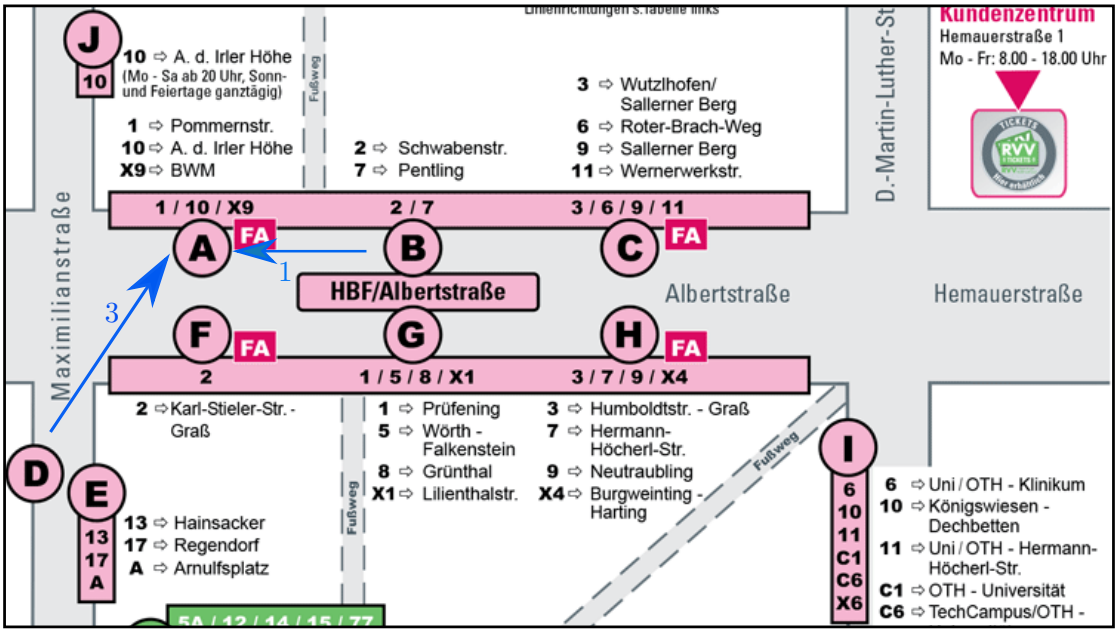


Abbildung 4.8: Haltepunkte an der Haltestelle HBF/Albertstraße.

Das Problem der relativen Umsteigezeiten ist beim vollständigen Umsteigegraphen nicht gegeben. Jede Kante wird dort mit der tatsächlich benötigten Umsteigezeit vom einen zum anderen Haltepunkt versehen. Das liefert eine *variable* Umsteigezeit für jeden Haltepunkt. Als Beispiel kann man auf der rechten Seite von Abbildung 4.7 leicht erkennen, dass man von b nach a nur eine Minute benötigt, während man von c nach a fünf Minuten einplanen muss.

Insgesamt ist zu erkennen, dass sowohl das Modell des sternförmigen Umsteigegraphen wie auch das Modell des vollständigen Umsteigegraphen Vor- und Nachteile mit sich bringen. Weil letzteres Konzept die Komplexität des Graphen deutlich steigert und zudem die Berechnung der einzelnen Umsteigezeiten zwischen den Haltepunkten jeder Station mit einem erheblichen Aufwand verbunden wäre, wohingegen eine konstante Umsteigezeit für jeden Graphen wesentlich leichter abschätzbar ist, soll im Folgenden das Modell des sternförmigen Umsteigegraphen mit konstanter Umsteigezeit für jede Haltestelle weiter untersucht werden.

Dies hat zusätzlich den Vorteil, dass jeder Haltestelle der eindeutige Bezugsknoten S zugeordnet werden kann, was für eine Fahrplan-Abfrage mit gewünschten Start- und Zielhaltestellen nützlich sein wird. Mit einem vollständigen Umsteigegraphen würde aus dem EAP ein *Many-To-Many-Shortest-Path-Problem* werden, einer Verallgemeinerung des SP Problems. Anstatt nur je einen Start- und Zielknoten zu betrachten, werden Teilmengen $S, T \subset E$ als mögliche Start- und Zielknoten deklariert und der optimale Weg aller Paare $(s, t) \in S \times T$ gesucht. [Paj09, S. 25,44]

Das Modell des zeitabhängigen Graphen mit sternförmigen Umsteigegraphen wird nun als realistisches zeitabhängiges Modell bezeichnet.

Um eine Lösung des EAP mithilfe des Dijkstra-Algorithmus zu finden, bei dem nach dem schnellstmöglichen Weg von der Anfangshaltestelle S zur Zielhaltestelle T bei gegebener Startzeit t gefragt wird, muss der Input-Graph geeignet definiert sein.

Die Konstruktion des für den Dijkstra-Algorithmus notwendigen Digraphen $G = (V, E)$ im realistischen zeitabhängigen Modell soll noch einmal zusammenfassend erläutert werden. (Vgl. [PSWZ07, S. 10-13])

Es ist zu beachten, dass in diesem Modell nicht mehr jeder Knoten einer Haltestelle entspricht. Stattdessen gibt es nun Haltepunkte, die Teil der einzelnen Bus-Routen sind und zentrale Haltestellenknoten, die den ursprünglichen, in Unterkapitel 4.1 definierten Knoten entsprechen.
Im Folgenden werden alle zentralen Haltestellenknoten des ÖPNV-Gebiets mit Kleinbuchstaben $u, v, \ldots$ bezeichnet. Alle Haltestellen werden in der Menge $\mathcal{S}$ zusammengefasst. Damit gibt es für jede Haltestelle im Liniennetz einen Knoten in $\mathcal{S}$ und jeder zentrale Haltestellenknoten $u \in \mathcal{S}$ kann eindeutig einer Haltestelle zugeordnet werden. Mit der Funktion $station(u)$, $u \in \mathcal{S}$, kann die Station aufgerufen werden, zu der u gehört,

d.h. die aktuelle Haltestelle wird abgefragt.
Für jede Route r sei $k_r > 0$ die Anzahl an Haltestellen, an der der Bus, der auf r verkehrt, hält. Die angefahrenen Haltestellen müssen nicht unbedingt paarweise verschieden sein und die Haltestelle, an der die Route beginnt, wird mitgezählt. Werden die einzelnen Haltestellen durchnummeriert zu $S_1, \ldots, S_{k_r}$, ergeben sich daraus die zugehörigen zentralen Haltestellenknoten $v_1, \ldots, v_{k_r} \in \mathcal{S}$ mit $station(v_i) = S_i$, $i = 1, \ldots, k_r$. Für jedes dieser i definiert man nun eine Menge $\mathcal{P}_{v_i}$, indem man jeweils den Haltepunktknoten, an dem der Bus hält, zu $\mathcal{P}_{v_i}$ hinzufügt, wenn die Route r über die Haltestelle $station(v_i)$ läuft. Dieses Verfahren wiederholt man für alle Bus-Routen und erhält daraus für alle $u \in \mathcal{S}$ eine Knotenmenge $\mathcal{P}_u$, in der alle Haltepunkte einer Haltestelle enthalten sind.
Die Mächtigkeit der Menge $\mathcal{P}_u$ bezeichnet man als $|\mathcal{P}_u| = P_u$ und die Vereinigung über alle $\mathcal{P}_u$ sei $\mathcal{P} = \bigcup_{u \in \mathcal{S}} \mathcal{P}_u$. Hieraus ergibt sich die Knotenmenge V von G als Vereinigung $V = \mathcal{S} \cup \mathcal{P}$, also alle zentralen Haltestellenknoten vereint mit den Haltepunktknoten jeder Station. Für jeden zentralen Haltestellenknoten $u \in \mathcal{S}$ definiere p_i^u, $i = 0, \ldots, P_u - 1$, die in $\mathcal{P}_u$ enthaltenen Haltepunktknoten, wobei die Reihenfolge beliebig ist.

Nun muss noch die Kantenmenge E des Digraphen G bestimmt werden. Diese setzt sich aus drei disjunkten Teilmengen zusammen: die *Umsteigekanten* A von einem Haltepunktknoten zum zentralen Haltestellenknoten, die *Umsteigekanten* D vom zentralen Haltestellenknoten zu den Haltepunktknoten sowie die *Routen-* bzw. *Fahrplankanten* R zwischen den einzelnen Haltepunkten einer Route. Damit ergibt sich die Kantenmenge

$$E = A \cup D \cup R$$

mit

- $A = \bigcup_{u \in \mathcal{S}} A_u$ mit $A_u = \bigcup_{0 \leq i < P_u} \{(p_i^u, u)\}$,
- $D = \bigcup_{u \in \mathcal{S}} D_u$ mit $D_u = \bigcup_{0 \leq i < P_u} \{(u, p_i^u)\}$ sowie
- $R = \bigcup_{\substack{u,v \in \mathcal{S} \\ 0 \leq i < P_u, 0 \leq j < P_v}} \{(p_i^u, p_j^v) : (*) \text{ gilt}\}$

  Dabei besagt $(*)$: $station(u)$ und $station(v)$ werden nacheinander in derselben Busroute angefahren und p_i^u, p_j^v sind die Haltepunktknoten von $\mathcal{P}_u$ und $\mathcal{P}_v$.

Für die Korrektheit des Modells ist es wichtig, die *FIFO*-Bedingung vorauszusetzen. Angewendet auf die oben definierte Notation bedeutet dies:
Seien $u, v \in \mathcal{S}$, $p_i^u \in \mathcal{P}_u$ und $p_j^v \in \mathcal{P}_v$, sodass $(p_i^u, p_j^v) \in R$. Wenn t_1^d, t_2^d Abfahrtszeiten von p_i^u und t_1^a, t_2^a Ankunftszeiten in p_j^v sind, so muss aus $t_1^d \leq t_2^d$ stets $t_1^a \leq t_2^a$ folgen.
Anschaulich heißt dies, dass kein Überholvorgang von Bussen auf derselben Route erlaubt ist. Sollte diese Voraussetzung verletzt sein, können die Busse einer Route in unterschied-

liche Geschwindigkeitsklassen (z.B. normale und Expressbusse) eingeteilt werden. Jede Klasse induziert dann eine eigene Busroute, obwohl sie dieselben Haltestellen anfahren.

Seien $u, v \in \mathcal{S}$, $0 \leq i < P_u$ und $0 \leq j < P_v$. Dann werden die dynamischen Kantengewichte folgendermaßen definiert:

- Jede Kante $(p_i^u, u) \in A$ hat die Gewichtung Null.
- Das Gewicht einer Umsteigekante $(u, p_i^u) \in D_u$ ist durch die konstante Funktion $g_u : I \to I$ gegeben, wobei $g_u(t) \equiv t^u_{umst.}$ die Umsteigezeit ist, die benötigt wird, um an der Haltestelle $station(u)$ von einem Haltepunkt zum anderen zu gelangen.
- Das Gewicht der Fahrplankanten $(p_i^u, p_j^v) \in R$ wird mithilfe der Gewichtsfunktion $f_{p_i^u p_j^v} : I \to \mathbb{N}_0$ aus Gleichung 4.2 bestimmt.

Gemäß der Konstruktion sind diese Gewichtsfunktionen für alle $t \in I$ nicht-negativ. Außerdem ist $f_e(t) + t$ monoton wachsend für alle Kanten $e \in E$ und alle Zeiten $t \in I$.

Insgesamt erhalten wir damit als Input des Djikstra-Algorithmus für zeitabhängige Graphen (Algorithmus 4) im realistischen zeitabhängigen Modell den oben konstruierten Graphen $G = (V, E)$, einen Startpunkt $s \in \mathcal{S}$, einen Zielpunkt $t \in \mathcal{S}$ und eine Startzeit $t \in I$. Dabei ist $station(s) = S$, $station(t) = T$ und die Kantengewichte aller Kanten $e \in D_s$ müssen verschwinden ($g_s(t) \equiv 0$).

Satz 4.14. *Algorithmus 4 liefert eine Lösung des EAP im realistischen zeitabhängigen Modell, wenn die* FIFO*-Bedingung gilt.*

Beweis. Durch die Definitionen der Umsteige- und Routengewichte werden nicht-negative Gewichtsfunktionen f_e und g_u mit monoton steigenden $f_e(t) + t$ bzw. $g_u(t) + t$ für alle Knoten $u \in \mathcal{S} \subseteq V$ und Kanten $e \in R$ des Graphen G sowie für alle Zeiten $t \in I$ sichergestellt.
Damit folgt der Beweis dieses Satzes analog zum Beweis von Satz 4.13. □

Zur Übersicht sind in Abbildung 4.9 beispielhaft Knoten, Kanten und Gewichte im realistischen zeitabhängigen Modell graphisch dargestellt. (Vgl. [PSWZ07, S. 12])

Die Laufzeit des Djikstra-Algorithmus im realistischen zeitabhängigen Fall beläuft sich auf $O(m \log W + n \log n)$, wobei $|V| = n$, $|E| = m$ und W die maximale Anzahl an ÖPNV-Verbindungen ist, welche in den Routen auftauchen, mit denen der Graph G konstruiert wurde.

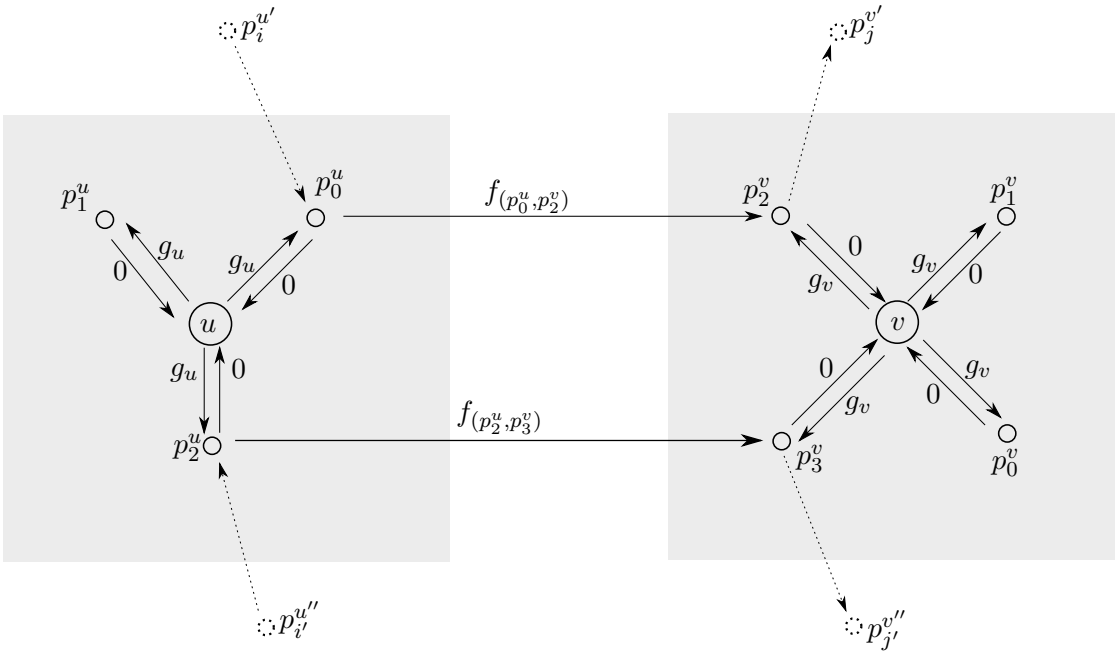


Abbildung 4.9: Realistisches zeitabhängiges Modell.

5. Dokumentation der Schnittstelle und Implementierung des Algorithmus

Nachdem das Modell für den realistischen zeitabhängigen Graphen konstruiert worden ist und der Dijkstra-Algorithmus entsprechend angepasst wurde, wird im Folgenden die Implementierung des Algorithmus erörtert. Hierfür wird vorab die Schnittstelle dokumentiert, welche als Datengrundlage der Passagierrouten-Abfrage dient.

5.1. VDV-Schnittstelle zu Liniennetz und Fahrplan des Regensburger Stadtverkehrs

Um den Informationsfluss der Verkehrsbetriebe zwischen Software-Modulen unterschiedlicher Hersteller oder zwischen Verkehrsbetrieb und Öffentlichkeit realisieren zu können, ist eine Standardisierung der Datenschnittstellen unumgänglich.

Die *VDV-Standard-Schnittstelle Liniennetz/Fahrplan* ist Teil des *ÖPNV-Datenmodells*, auch *VDV-Datenmodell* genannt, eine vom *Verband Deutscher Verkehrsunternehmen (VDV)* herausgegebene Basisschrift zur Datenmodellierung und Realisierung von standardisierten Datenschnittstellen im ÖPNV. Sofern nicht anders vermerkt, stammen die Informationen in diesem Abschnitt aus der Schnittstellendokumentation zur VDV-Standardschnittstelle Liniennnetz/Fahrplan (VDV-Schrift 452, Version 7/13) [VDV13].

Die Schnittstellendefinition zu Liniennetz- und Fahrplan-Daten dient zur Übertragung von Informationen zum Liniennetz und zu den Fahrplänen. Die Schnittstelle definiert das Datenmodell und gewährt den Zugriff via SQL oder dem ÖPNV-Dateiformat, einem ASCII-Datenformat zur Offline-Übertragung der Daten.

Die Standardisierung der ÖPNV-Datenschnittstelle erfolgt unter anderem zur Einhaltung der nachfolgenden Ziele:

- Reduktion von individuellen Schnittstellen,
- Einheitliche Dokumentation,
- Verwendung von Standardprodukten unabhängig der verwendeten Systeme bei allen Verkehrsbetrieben,
- Transparenter und System-übergreifender Zugang zu den Daten sowie
- Einheitliche Bezeichnungen von numerischen und alphanumerischen Datenfeldern als Schlüsselattribute.

Im Rahmen der Übertragung von Liniennetzdefinitionen und Fahrplänen zwischen Quell- und Zielsystemen findet ein Informationsaustausch zwischen dem Planungsprogramm der

Verkehrsplanung (Fahr- und Dienstplanung) und den Konsumentensystemen (z.B. zur Betriebsüberwachung und Steuerung) statt.

In der Dokumentation zur VDV-Standard-Schnittstelle Liniennetz/Fahrplan werden die Grundlagen zum Datenmodell und der Rahmen für die Datenschnittstelle formuliert. Dabei werden insbesondere formale sowie logische und inhaltliche Voraussetzungen festgelegt, die die Schnittstelle erfüllen muss.
Beispielsweise ist es notwendig, Haltestellen und Fahrzeiten meter- bzw. sekundengenau anzugeben und neben produktiven Fahrten, wie Servicefahrten, auch Betriebshof- oder Leerfahrten in die Datenbank mitaufzunehmen, um eine exakte Modellierung zu ermöglichen.

Für die Einarbeitung und Pflege der Daten in der Schnittstelle sind die jeweiligen Betriebshöfe des ÖPNV unter Einhaltung der Rahmenbedingungen selbst zuständig. Im Regensburger Nahverkehr wird diese Aufgabe vom Stadtwerk.Mobilität übernommen.

Für die vorliegende Arbeit wurde der Autorin die aktuelle Version der Schnittstelle (Stand 28.01.2019) vom Regensburger Stadtwerk zur Verfügung gestellt. Eine Zusammenfassung der Inhalte soll im Folgenden gegeben werden. Die zur Lösung des EAP benötigten Daten werden danach noch eingehender untersucht.

Die Daten der Schnittstelle können in verschiedene Bereiche eingeteilt werden:

- Kalenderdaten,
- Ortsdaten,
- Betriebsdaten,
- Netzdaten,
- Liniendaten und
- Fahrplandaten.

Die einzelnen Datengruppen und Datensätze werden in den folgenden Unterabschnitten spezifiziert. Für jeden Datensatz werden Beziehungen zu anderen Relationen definiert. Diese Primär- und Fremdschlüssel werden in den Abschnitten nicht explizit angegeben. Für konkrete Schlüssel sei an [VDV13] verwiesen.

Kalenderdaten

Die Kalenderdaten bilden den zeitlichen Rahmen der Schnittstelle. Sie umfassen die Gültigkeitszeiträume der Basisversionen, die Versionshaltung der Stamm-, Fahrplan- und Umlaufdaten, die Zuordnung der Tagesarten zum Kalenderdatum sowie die Auflistung der verschiedenen Tagesarten.

Mithilfe dieser Daten lässt sich die aktuell gültige Version des Fahrplans/Liniennetzes bestimmen. Zudem ist angegeben, für welchen Zeitraum die Version tatsächlich gültig ist. Dieser Zeitraum wird im Datensatz *FIRMENKALENDER* mit Kalenderdaten und Wochentagen versehen. Dabei werden die Wochentage numerisch mit den Zahlen von 1 (Montag) bis 7 (Sonntag) gekennzeichnet.

Ortsdaten

Zu den Ortsdaten zählen alle Datensätze über Haltestellen, Haltepunkte und Betriebshöfe eines Verkehrsnetzes.
Diese Ortstypen werden entsprechend ihrer Funktion gegliedert in Haltepunkte, Betriebshofpunkte, Ortsmarken, LSA-Punkte (Lichtsignalanlagen/Ampeln), Routenzwischenpunkte, Betriebspunkte und Grenzpunkte (Gebietskörperschaftsgrenzen). Anstelle der genauen funktionalen Zuordnung genügt oft eine Einteilung durch Gruppierungsmerkmale für Orte. Im Datensatz *MENGE_ORT_TYP* werden die Merkmale Haltestelle (Nr. 1) und Betriebshof (Nr. 2) definiert.
In *REC_HP* werden alle Haltepunkte im ÖPNV-Gebiet definiert und abgespeichert. Die Haltepunkte können sowohl Haltestellen als auch Betriebshöfen zugeordnet werden und bilden die kleinste örtliche Einheit der Fahrplanung. Jeder Haltestelle bzw. jedem Betriebshof können nicht mehr als 100 Haltepunkte zugeordnet werden. Die Haltepunkte werden mit eindeutigen Haltepunktnummern und konkreten Orts-Bezeichnern versehen. Der Orts-Bezeichner *ORT_NR* markiert jeden Haltepunkt mit einer eindeutigen numerischen ID, z.B. „101001“. Dieser Bezeichner enthält am Ende die Haltepunktnummer (hier: „01“). Der vordere Teil der Nummer gibt Auskunft über die zugehörige Haltestelle bzw. den zugehörigen Betriebshof (hier: „1010“ für die Haltestelle *Arnulfsplatz*). In der Spalte *ZUSATZ_INFO* erfolgt eine genaue Lage-Beschreibung des Haltepunktes innerhalb der Haltestelle (hier: „Theater (alle Li Ri FISH)“, d.h. der Haltepunkt befindet sich vor dem Theater und ist allen Linien Richtung Fischmarkt zugeordnet).
Eine Gruppierung der Haltepunkte findet in der Relation *REC_ORT* statt. Neben der Zuordnung eines Haltepunktes (z.B. „101001“) zu seinem funktionalem Ortstypen (Haltestelle $\widehat{=}$ 1 oder Betriebshof $\widehat{=}$ 2, hier: 1) wird die zugehörige Haltestelle (der zugehörige Betriebshof) mit Namen (*Arnulfsplatz*), Kennziffer („1010“) sowie einer Abkürzung („APLZ“) versehen. Darüberhinaus sind Informationen zu den entsprechenden Tarifzonen eingepflegt, alle Haltepunkte mit Längen- und Breitengraden geocodiert und mit ihren internationalen Haltestellen-IDs gekennzeichnet.

Betriebsdaten

In den Betriebsdaten werden Informationen zum Fahrzeugbestand und zu den Fahrzeugarten gespeichert. Außerdem können Daten zu Ansage- und Zieltexten integriert

werden, also Daten, die primär für das Verkehrsunternehmen selbst gedacht sind. Letztere werden in der vorliegenden Schnittstelle nicht verwendet.
Neben einer Relation zu den im Verkehrsgeschehen beteiligten Verkehrsbetriebe, welche hier nur die „RVB Regensburger Verkehrsbetriebe“ umfasst, werden vor allem die verschiedenen Fahrzeugtypen genauer beschrieben.
Die zugehörige Relation trägt den Namen *MENGE_FZG_TYP* und listet sowohl RVB-interne Bustypen (Standardlinien-, Standardgelenk- und E-Midibus) als auch Fahrzeugtypen von externen Firmen auf. Jeder Typ wird mit einer ID und einem Kürzel ausgestattet. Zusätzliche Variablen zu Gesamtlänge, Anzahl der Sitz-, Steh- und Sonderplätze (Behindertenplätze) im Fahrzeug sind ohne Ausprägungen in der Relation enthalten.

Netzdaten

Die Netzdaten umfassen alle Informationen, die die relative Lage der Punkte im Netzplan beschreiben. Dazu gehören Strecken, Entfernungen und Fahrzeiten zwischen Ortsdaten, Fahrzeitgruppen sowie Haltezeiten.
In der Relation *REC_SEL* werden Linien durch Angabe von aufeinanderfolgenden Haltepunkten als Start- und Zielpunkte definiert. Zwischen beiden Punkten verlaufen in der Regel zwei Strecken in entgegengesetzter Richtung, sodass die Punkte dementsprechend zweimal in vertauschten Rollen aufgeführt werden. Sowohl Start- als auch Zielpunkte werden als Haltestellen- oder Betriebshofpunkte gekennzeichnet, sodass nicht nur Strecken von Servicefahrten abgebildet werden, sondern auch Strecken von Leerfahrten. Die jeweiligen Streckenlängen sind in Metern vermerkt.
Für jede Linie gibt es verschiedene Fahrzeitgruppen, welche für ein bestimmtes Tageszeitintervall die Fahr- und Haltezeiten angibt. Die Fahrzeitgruppen werden mit numerischer Kennzeichnung in der Relation *MENGE_FGR* definiert. Da für diese Relation die Schnittstellenbeschreibung nicht mit der Schnittstellen-Datei übereinstimmt, ist eine manuelle Zuordnung der Fahrzeitgruppen erforderlich, auf welche später genauer eingegangen wird.
In *ORT_HZTF* werden zu den Fahrzeitgruppen ortspezifische Haltezeiten angegeben. Diese Relation wird nur für die Haltestelle HBF/Albertstraße verwendet.
Die Relation *SEL_FZT_FELD* gibt Auskunft über die planmäßige Fahrzeit für vordefinierte Streckenabschnitte. Diese Zeit kann je nach Tageszeit variieren, weshalb die Fahrzeit an die Fahrzeitgruppen gekoppelt ist. Alle Zeiten werden in Sekunden angegeben. Die Streckenstart- und -zielpunkte entsprechen jeweils zwei aufeinanderfolgenden Haltepunkten. Sie werden zusätzlich mit ihrem funktionalem Ortstypen (Haltestelle/Betriebshof) gekennzeichnet.
REC_UEB definiert Überläuferfahrten zwischen zwei Haltepunkten im Netz. Es werden Start- und Zielorte der Fahrten angegeben und mit ihrem Ortstypen versehen. In der vorliegenden Schnittstelle sind nur Überläuferfahrten zwischen Haltepunkten von Haltestellen gespeichert, keine Fahrten, die an Betriebshofpunkten starten oder enden. Damit

wird die Endhaltestelle einer Linie mit der Starthaltestelle einer weiteren Linie verknüpft, sodass die beiden Linien nacheinander in einem Umlauf angefahren werden können. Die Streckenlänge des Überläuferfahrweges wird in Metern angegeben. Obwohl in der Schnittstellen-Dokumentation anders vermerkt, werden in der vorliegenden Schnittstelle auch Überläuferfahrten mit identischem Start- und Zielpunkt gespeichert. In diesem Fall ergibt sich eine Weglänge von Null Metern.
Die planmäßig vorgesehenen Fahrzeiten der Überläuferfahrten werden in *UEB_FZT* erfasst. Diese Zeiten sind wieder je nach Tageszeit unterschiedlich, sodass es für jede Tageszeitgruppe eine eigene Fahrzeit der Überläuferfahrten gibt. Die Fahrzeit wird in Sekunden angegeben.
In der Relation *MENGE_FAHRTART* werden die verschiedenen Fahrtarten definiert und mit Nummern gekennzeichnet. Die Kennzahl 1 entspricht einer Normalfahrt (N), also einer Servicefahrt, 2 steht für Betriebshofausfahrt (Depot-Ausrückfahrt, DA), 3 für eine Betriebshofeinfahrt (Depot-Einrückfahrt, DE) und 4 steht für eine Zufahrt (Z), also einer Route für Überläuferfahrten.
Räumlich zusammenhängende Gebiete werden als Flächenzonen in der Relation *FLAECHEN_ZONE* abgespeichert. Diese einzelnen Zonen müssen nicht disjunkt sein und werden mit ihrem Zonentyp versehen. In dieser Schnittstelle gibt es nur den Flächenzonentyp *Gebietskörperschaften* (Typ-Nummer 1, vgl. Relation *MENGE_FLAECHEN_ZONE_TYP*).
In *FLAECHEN_ZONE* werden Regensburg und weitere umliegende Ortschaften (z.B. Pentling, Neutraubling, Cham) mit einer (internen) Flächennzonennummer und der amtlichen Gemeindekennziffer aufgeführt.
Die Netzpunkte (Haltepunkte an Haltestellen und im Depot) werden in *FL_ZONE_ORT* den verschiedenen Flächenzonen zugeordnet.

Liniendaten

Mithilfe der Liniendaten werden die Orts- und Netzdaten zu Linien im ÖPNV-Gebiet verknüpft, indem Linienverläufe für die unterschiedlichen Routen definiert werden.
Zu den Liniendaten gehören die beiden Relationen *LID_VERLAUF* und *REC_LID*.
In *LID_VERLAUF* werden die nacheinander abgefahrenen Haltepunkte einer Linie durchnummeriert, sodass ein Linienverlauf definiert wird. Jeder Linie wird eine Liniennummer zugeordnet und eventuelle alternative Varianten für diese Linie werden gespeichert. Eine Voraussetzung in der Dokumentationsschrift ist das Verbot zirkulärer Verläufe, d.h. ein Bus darf zwar eine Haltestelle mehrmals innerhalb einer Linienfahrt anfahren, nicht aber denselben Haltepunkt. Die Gesamt-Fahrzeit und -Distanz in einem Linienverlauf muss größer Null sein. Bei den vorliegenden Daten bleiben die Optionen zur Angabe von Zielanzeigen, Ansagen sowie Beförderungs-, Einstiegs- und Ausstiegsverbote ungenutzt. Auch die Variable *EINFANGBEREICH*, die den Bereich angibt, in dem ein Haltepunkt oder eine Haltestelle vom Bordcomputer erkannt wird, wird nicht gepflegt. Es kann zwischen Zeit-relevanten und nicht-Zeit-relevanten Orten, zwischen

normalen Stopps und Bedarfshalten sowie zwischen produktiven Servicefahrten oder nicht-produktiven Leerfahren unterschieden werden. In diesen Daten werden nur Zeitrelevante Orte berücksichtigt. Produktive und nicht-produktive Fahrtwege dürfen nicht vermischt werden.
REC_LID ordnet jeder Linie einem Betriebszweig zu. Eine eindeutige Liniennummer innerhalb eines Netzes wird vorausgesetzt. Die Routennummer verlangt Eindeutigkeit bezüglich der Linie und dem Linienverlauf. Es können zwei verschiedene Linienrichtungen selektiert werden. Jeder Linie wird eine Kurzbezeichnung (z.B. „C1“) und eine Beschreibung durch Angabe der beiden Endhaltestellen (z.B. „Albertstraße - Universität“) zugeordnet. Außerdem wird die Routenart (N, DE, DA oder Z) vermerkt.

Fahrplandaten

Schließlich werden in den Fahrplandaten konkrete Fahrten, fahrtabhängige Haltezeiten und Fahrzeugumläufe festgehalten.
REC_FRT bildet Fahrten und Umläufe ab. Hierfür wurden im Vorhinein zusammenpassende Routen und zulässige Linienwechsel analysiert und zu vollständigen Fahrtrelationen gruppiert. Die Kursnummer ordnet jedes Fahrzeug auf einer Linie einem eindeutigen Fahrplan zu. Dadurch lässt sich aus den Kursen auch die Anzahl an Fahrzeugen ermitteln, die zu einem konkreten Zeitpunkt im Einsatz sind. Unter Berücksichtigung der Linie und des Zeitintervalls, in dem sich ein gewisses Fahrzeug der Linie gerade befindet, ist die Eindeutigkeit der Kursnummer zu gewährleisten. In der Relation werden alle Fahrten durchnummeriert, die Abfahrtszeit in Sekunden nach Mitternacht sowie die Tagesart angegeben und jedem linienreinen Umlaufstück eine Kursnummer zugeordnet. Die Fahrtart und Fahrzeitgruppe wird definiert, ebenso wie die Umlauf-ID und eventuelle Linien-Varianten. Zusätzlich kann vermerkt werden, ob sich am Fahrtende bzw. Fahrtanfang Passagiere im Fahrzeug befinden dürfen. Eine solche *Durchbindung* wird häufig auf Ringlinien oder im Schülerverkehr eingesetzt. Im Regensburger Modell sind keine Durchbindungen erlaubt.
Bei der Bildung der Fahrzeugumläufe sind generell zwei Varianten möglich:

1. In der Relation *REC_FRT* werden alle Fahrten der Fahrzeugumläufe, auch Überläuferfahrten, gespeichert. Dadurch werden *REC_UEB* und *UEB_FZT* überflüssig. Diese Variante hat den Vorteil, dass eine gültige Fahrzeitengruppe und eine eindeutige Kursnummer für alle Fahrten inklusive Überläuferfahrten besteht.

2. In *REC_FRT* werden nur die Fahrten gespeichert, die keine Überläuferfahrten sind. Wenn im Laufe eines Fahrzeugumlaufs in *REC_FRT* festgestellt wird, dass die Orts-ID der n-ten Fahrt ungleich der Orts-ID des Startpunkts der $(n+1)$-ten Fahrt ist, wird eine geeignete Überläuferfahrt in *REC_UEB* gesucht. Entsprechend muss dann die gültige Fahrzeitengruppe für die Überläuferfahrt aus der n-ten Fahrt (oder $(n+1)$-ten, falls es keine n-te Fahrt gibt) berechnet werden.

In der vorliegenden Schnittstelle wird die zweite Variante verwendet.
In *REC_UMLAUF* werden die konkreten Fahrzeugumläufe beschrieben. Diese müssen jeweils mit einer Depot-Ausrückfahrt starten und mit einer Depot-Einrückfahrt enden. Jedem Umlauf wird eine Kennziffer, eine Tagesart, Anfang- und Endhaltepunkt (an den Betriebshöfen) sowie ein Fahrzeugtyp zugeordnet.

Nicht-verwendete Datenmodelle

Im zugrundeliegenden Datenmodell des VDV sind noch weitere Datenstrukturen definiert, die nicht in der betrachteten Schnittstelle des Regensburger Stadtwerks verwendet werden. Ergänzend sollen auch diese kurz skizziert werden.

Die Ortsdaten können um die Relation *REC_OM* erweitert werden, welche Orte mit Ortsmarken unter Angabe ihrer Codierung versieht. Diese Ortsmarken können dann dazu verwendet werden, eine Standortabfrage von Fahrzeugen zu generieren, indem beim Vorbeifahren an den Orten ein Signal gesendet wird.

In den Betriebsdaten können zudem verschiedene Informationen zu den Fahrzeugen in die Schnittstelle aufgenommen werden. Die Relation *FAHRZEUG* bietet eine Möglichkeit zur Auflistung aller Fahrzeuge samt Fahrzeugtyp, amtlichem Kennzeichen und Verkehrsunternehmen.
Außerdem können Fahrzeug-Anzeigetexte formuliert werden, die durch numerische Kennzeichnung leicht abrufbar werden. Analog können auch die am Fahrzeug angezeigten Fahrtziele in einer Relation abgespeichert werden.

Im Bereich der Netzdaten können neben Haltepunkten an Haltestellen und Betriebshöfen auch zusätzliche Zwischenpunkte entlang der Strecken vermerkt werden, welche den geographischen Verlauf der Linienroute genauer definieren, z.B. Ortsmarken oder Routen-Zwischenpunkte.
Verläuft ein Streckenabschnitt über einen Grenzpunkt zweier Gebietskörperschaften, kann dieser als zusätzlicher Netzpunkt angesehen werden und die planmäßige Fahrzeit zwischen dem letzten angefahrenen Haltepunkt und dem Grenzpunkt in der Relation *SEL_FZT_FELD_ZP* abgespeichert werden. Diese Zeiten werden unter Verwendung der zugehörigen Fahrzeitgruppe in Sekunden angegeben.

In den Fahrplandaten können fahrtabhängige Wartezeiten an den Haltepunkten definiert werden. Diese setzen sich zusammen aus den Fahrgastwechselzeiten und den eigentlichen Wartezeiten an den Haltestellen und sind nur für solche Haltestellen erlaubt, die keine Anfangs- oder Endhaltestellen einer Linie sind.

Die verschiedenen Daten-Bereiche können um die Menge der *Anschlussdaten* erweitert werden. Dort werden alle Informationen, die zur Anschlusssicherung benötigt werden, hinterlegt. Dabei werden Einzelanschlüsse dokumentiert und zur Anschlussüberwachung können Tageszeit-spezifische Gültigkeiten definiert werden.

Darüberhinaus können auch Angaben zu Flächenzonen und Gebietskörperschaften, wie Grenzpunkte, detaillierter eingepflegt werden.

5.2. Implementierung des Algorithmus

Um den Dijkstra-Algorithmus für zeitabhängige Graphen im realistischen zeitabhängigen Modell zu implementieren und auf die Fahrplandaten des Regensburger ÖPNVs anzuwenden, wird im praktischen Teil dieser Masterarbeit die Programmiersprache R (Version 3.3.2 (2016-10-31)) verwendet. Da die Daten aus der VDV-Schnittstelle in tabellarischer Form vorliegen, wird in R ein einfacher Zugriff auf die verschiedenen Relationen, Variablen und Ausprägungen gewährleistet.

In den folgenden Abschnitten soll die Implementierung erörtert werden. Hierfür werden zunächst die für die Lösung des EAP benötigten Daten extrahiert und an das Modell angepasst.
Im Anschluss daran wird der Graph in Anlehnung an Unterkapitel 4.3 konstruiert, welcher als Input für den problemlösenden Dijkstra-Algorithmus dient.

5.2.1. Auswahl und Anpassung relevanter Daten

Die Namen der Dateien aus der VDV-Schnittstelle setzen sich zusammen aus

- dem Präfix „i“ (Abkürzung für **I**nterface-Datei),
- der dreistelligen Relations-ID,
- dem dreistelligen Tag des Jahres,
- „0“ sowie
- dem Suffix „.x10“ (ASCII e**X**change format version **1.0**). [VDV99, S. 9]

Beispielsweise hat die Relation *REC_ORT* den Dateinamen `i2533530.x10`, d.h. die zugehörige Tabellennummer ist 253 und die Datei wurde am 353. Tag des Jahres (19. Dezember 2018) zum letzten Mal aktualisiert.

Für weitere Informationen zum Dateiformat der Schnittstelle sei an dieser Stelle auf das ÖPNV-Datenmodell 5.0 [VDV99] verwiesen.

Die Original-Dateien aus der Schnittstelle wurden für den Import in R-Studio in CSV-Dateien konvertiert. Unnötige Inhalte wie Header, Format-Beschreibungen und Tabellenenden wurden entfernt.
Die importierten Dateien umfassen die Relationen

- `REC_ORT` (Ortsinformationen),

- LID_VERLAUF (durchnummerierte Linienverläufe),
- REC_FRT (Fahrplandaten) sowie
- SEL_FZT_FELD (Fahrzeiten).

Nach dem Einlesen der Daten werden nicht benötigte Variablen entfernt.
Darüberhinaus werden die Ortsdaten auf relevante Haltestellenpunkte reduziert, indem Betriebshofpunkte, Teststeige und E-Ladestationen entfernt werden. Bei den Linienverläufen werden alle Zeilen, die Teststeige und unproduktive Fahrten enthalten, gelöscht. In den Fahrplandaten werden nur mehr Normalfahrten betrachtet. Überläuferfahrten sowie Ein- und Ausrückfahrten werden aus den Tabellen entfernt ebenso wie alle Fahrzeitinformationen zu Fahrten, deren Start oder Ziel Betriebshofpunkte sind.

Die relevanten Ortsdaten REC_ORT setzen sich zusammen aus den Variablen

- ONR_TYP_NR (Bezeichner für den Ortstyp des Haltepunktes),
- ORT_NR (Haltepunkt-Nummer),
- ORT_NAME (Name der Haltestelle),
- ORT_REF_ORT (Haltestellen-Nummer),
- ORT_REF_ORT_KUERZEL (Abkürzung der Haltestelle) und
- ORT_REF_ORT_NAME (detaillierter Name der Haltestelle).

Die relevanten Daten zu den Linienverläufen (LID_VERLAUF) umfassen die Variablen

- LI_LFD_NR (fortlaufende Nummerierung der Haltepunkte, die nacheinander von einer Linie/Linienvariante angefahren werden),
- LI_NR (Linien-Nummer),
- STR_LI_VAR (Linien-Variante),
- ORT_NR (Haltepunkt-Nummer) sowie
- PRODUKTIV (gibt an, ob es sich um eine Servicefahrt (= 1) handelt oder nicht(= 0)).

Zu den relevanten Fahrplandaten (REC_FRT) zählen die Variablen

- FRT_START (Startzeitpunkt der Linie in Sekunden nach Mitternacht),
- LI_NR (Linien-Nummer),
- FGR_NR (Fahrzeitgruppe) und

- `STR_LI_VAR` (Linien-Variante).

Um die Fahrplandaten aus `REC_FRT` zu vervollständigen, liefert `SEL_FZT_FELD` konkrete Fahrzeiten zwischen zwei benachbarten Haltepunkten. Diese werden charakterisiert durch die Variablen

- `FGR_NR` (Fahrzeitgruppe),
- `ORT_NR` (Haltepunkt-Nummer an der Vorgängerhaltestelle),
- `SEL_ZIEL` (Haltepunkt-Nummer an der Nachfolgerhaltestelle) und
- `SEL_FZT` (Fahrzeit zwischen Vorgänger- und Nachfolgerhaltepunkt in Sekunden).

Neben dem Exzerpieren relevanter Daten werden durch den nachstehenden Befehl die zu untersuchenden Daten an eine starke Modellannahme geknüpft:

```
1 REC_FRT <- sqldf("select * from REC_FRT where [TAGESART_NR]=1") # nur Montage
```

Um die große Menge an Fahrplandaten zu komprimieren, wird in der folgenden Implementierung nur der Montagsfahrplan betrachtet. Es wird angenommen, dass der Busverkehr erst mit Betriebsbeginn um 05.00 Uhr morgens startet, obwohl in Realität einzelne Stationen kurz nach 00.00 Uhr von den letzten Fahrten des Sonntagsfahrplans bedient werden. Darüberhinaus wird festgesetzt, dass sich die Abfahrts- und Ankunftszeiten des Montagsfahrplans periodisch wiederholen, sodass eine Fahrt, welche nach Mitternacht (von Montag auf Dienstag) beginnt oder endet, abgebildet werden kann. Die Modellierung einer Reisezeit, die die Dauer eines Tages überschreitet, ist auf Grundlage des Regensburger Verkehrsnetzes nicht vonnöten.
Kommt es dazu, dass die Fahrt über Mitternacht hinausgeht, wird dies in der späteren Fahrplanauskunft mit $+1d$ vermerkt.
Bis auf wenige Ausnahmen entspricht der Montagsfahrplan auch den von Dienstag bis Freitag gültigen Fahrplänen, sodass die periodische Fahrplan-Wiederholung die Wirklichkeit angemessen modelliert. Für eine Fahrplanabfrage am Wochenende müssten die betrachteten Daten entsprechend angepasst werden.
Zur weiteren Modellvereinfachung wird angenommen, dass es sich um einen Montag handelt, der kein Feiertag ist, nicht in den bayerischen Schulferien und nicht in der vorlesungsfreien Zeit liegt, sodass keine Ausnahmeregelungen zu bestimmten Abfahrtszeiten und -orten betrachtet werden müssen und jede Linie mit allen Montags-Varianten im Fahrplan auftaucht.

Um bei der Fahrplan-Abfrage die konkreten Start- und Zielhaltestellen auswählen zu können, muss sichergestellt werden, dass die Haltestellen-Bezeichnungen nicht mehrfach verwendet werden.

```
# Haltestellen mit nicht-eindeutigen REC_ORT$ORT_NAME
for (k in REC_ORT$ORT_NAME) {
  if (length(unique(REC_ORT[which(REC_ORT$ORT_NAME ==k), 4])) > 1)
  {print(unique(REC_ORT[which(REC_ORT$ORT_NAME == k), c(3,4,6)]))}
}

# ORT_NAME ORT_REF_ORT                              ORT_REF_ORT_NAME
# 64  Keplerstra_e        1181  "Keplerstra_e"
# 245 Keplerstra_e        3060  "Neutraubling Keplerstra_e"

# ORT_NAME ORT_REF_ORT                              ORT_REF_ORT_NAME
# 82 AussigerStra_e        2005  "Aussiger Stra_e"
# 88 AussigerStra_e        2010  "Aussiger Stra_e"

# ORT_NAME ORT_REF_ORT                              ORT_REF_ORT_NAME
# 140 Pommernstra_e        2046  "Pommernstra_e"
# 238 Pommernstra_e        3044  "Neutraubling Pommernstra_e"

# ORT_NAME ORT_REF_ORT                              ORT_REF_ORT_NAME
# 204 Friedhof        3010  "Harting Friedhof"
# 248 Friedhof         320  "Neutraubling Friedhof"
```

Hierfür werden die mehrfach belegten Ortsnamen durch eindeutige Bezeichnungen ersetzt.

In den Ortsdaten werden alle Haltepunkte (`REC_ORT$ORT_NR`) eindeutigen Haltestellen (`REC_ORT$ORT_REF_ORT`) zugeordnet. Letztere eignen sich daher als zentrale Haltestellenknoten, mithilfe derer Umstiege modelliert werden.
Während das Kantengewicht von einem Haltepunkt zum zentralen Haltestellenknoten verschwindet, wird als Kantengewicht für die entgegengesetzte Richtung die Umsteigezeit verwendet, die benötigt wird, um an einer Haltestelle von einem Haltepunkt zum anderen zu laufen. (vgl. Unterkapitel 4.3)
Da die erforderlichen Umsteigezeiten an den einzelnen Haltestellen nicht in der Datenschnittstelle vermerkt sind, wurden diese von der Autorin untersucht. Anstatt eine pauschale Umsteigezeit für alle Haltestellen vorauszusetzen, wurde dabei für jede einzelne Haltestelle die konkrete Umsteigezeit durch Messungen vor Ort und via Google Maps berechnet.
Hierfür wurden zunächst die Anzahl an Haltepunkte für jede Haltestelle ermittelt. Danach mussten für alle Haltestellen mit mehr als einem Haltepunkt die Umsteigezeiten definiert werden. Bei der Analyse der benötigten Zeiten zeichnete sich ab, dass die Umsteigezeiten in Abhängigkeit der Anzahl an Haltepunkten je Haltestelle dargestellt werden kann. Es gilt in der Regel:

$$\text{Umsteigezeit in Minuten} = \#\text{Haltepunkte} - 1.$$

Alle Umsteigezeiten, die nicht dieser Formel entsprechen, wurden separat ergänzt.

Um aus den gegebenen Daten die tatsächlichen Fahrpläne zu erzeugen, müssen die einzelnen Dataframes geeignet verknüpft werden. Als Schlüssel bietet sich hierzu die Variable Fahrzeitgruppe (`FGR_NR`) an. Wie in der Schnittstellenbeschreibung erwähnt, liefert die Relation *MENGE_FGR* keine Charakterisierung der verschiedenen Fahrzeitgruppen. Aus diesem Grund wurden die einzelnen Startzeiten jeder Linie mit Abzügen der aktuellen Fahrplan-Tabellen des RVV[6] verglichen, um manuell zu überprüfen, welche Fahrzeitgruppe den jeweiligen Linien, Linienvarianten und Startzeiten am Beginn der Fahrten zugeordnet werden müssen.
Unter der Annahme, dass der zu erzeugende Fahrplan für einen Montag außerhalb der Schulferien und der vorlesungsfreien Zeit gelten soll, ergab sich, dass die Variable `FGR_NR` mit allen sich momentan im Speicher befindenden Ausprägungen als Schlüssel für die Verknüpfungen von `REC_FRT` und `SEL_FZT_FELD` genutzt werden kann.
Mit `FGR_NR` wird entsprechend eine innere Verknüpfung zwischen `SEL_FZT_FELD` und `REC_ORT` erstellt, welche in `fahrplan` gespeichert wird. Dieser Dataframe wird um die Variablen `Abfahrt` und `Ankunft` ergänzt und nach Fahrzeitgruppe und Abfahrtszeit am Startpunkt der Servicefahrten sortiert.
Im Zwischenspeicher `startpunkte` werden diese Startpunkte für jede Fahrzeitgruppe hinterlegt. Darauf aufbauend werden die Abfahrts- und Ankunftszeiten befüllt, indem zuerst für alle Linien(-varianten) die Abfahrtszeiten am ersten Haltepunkt der Route bestimmt werden und danach die Zeiten für die weiteren Haltepunkte ergänzt werden.

Während der Dataframe `fahrplan` Informationen über alle angefahrenen Haltepunkte, Abfahrts- und Ankunftszeiten an den Haltepunkten, Fahrzeiten, Linien und Linienvarianten enthält, werden in `fahrtereignisse` nur die jeweiligen Ereignisse Abfahrtsort (`ORT_NR`) und -zeit (`Abfahrt`) sowie Ankunftsort (`SEL_ZIEL`) und -zeit (`Ankunft`) gespeichert.

Mit den auf diese Weise eingepflegten Daten kann nun der Graph analog zu seiner Konstruktion in Unterkapitel 4.3 erstellt werden.

5.2.2. Implementierung des Graphen

Bei der Konstruktion des Graphen werden zuerst die Knoten erzeugt. Im Anschluss werden benachbarte Knoten zu Kanten verbunden. Schließlich werden die Kanten mit den jeweiligen Gewichten versehen.

Die Knoten des Graphen setzen sich einerseits aus den zentralen Haltestellenknoten und andererseits aus den einzelnen Haltepunktknoten zusammen.
Für die zentralen Haltestellenknoten `zhs` werden die verschiedenen Ausprägungen der Variable `REC_ORT$ORT_REF_ORT` verwendet. Mit der Funktion `station()` kann der zugehörige Haltestellen-Name abgefragt werden.

[6] Vgl. http://www.rvv.de/Fahrplan.n6.html (Stand 15.05.2019).

Als Haltepunktknoten `hpk` werden die in der Variable `REC_ORT$ORT_NR` gespeicherten Haltepunkt-Nummern gewählt.
Die Vereinigung beider Knotenmengen liefert die Knotenmenge V des Graphen $G = (V, E)$.

Zur Implementierung der Kanten werden Dataframes erzeugt, in denen die Quell- und Zielknoten der einzelnen Kanten in den Variablen `VON` und `NACH` abgespeichert werden. Die Kanten werden je nach Kantenart in die verschiedenen Kantenmengen `A` (Umsteigekanten von den `hpk` zum `zhs`), `D` (Umsteigekanten vom `zhs` zu den `hpk`) sowie `R_allg` bzw. `R` (Routenkanten) gegliedert. Während in `R_allg` alle Routenkanten des Regensburger ÖPNV enthalten sind – beispielsweise auch die Kanten der Routen von Nachtbussen, welche nur an Freitagen und Samstagen verkehren – werden in `R` nur diejenigen Routenkanten gespeichert, die tatsächlich im Montagsfahrplan abgefahren werden. Für die Fahrplanabfrage ist der Dataframe `R_allg` zwar nicht relevant, wird aber zur Vollständigkeit ergänzt.
Zur leichteren Zuordnung der Routenkanten wird im Dataframe `R_allg` neben den Start- und Endknoten auch die zugehörige Linie und Linienvariante vermerkt.
Die für den Graphen relevanten Kanten `A`, `D` und `R` werden im Dataframe `E1` vereinigt.

Die Gewichtung der Kanten aus `E1` erfolgt durch eine separate Betrachtung der einzelnen Kantenmengen.

Weil das Gewicht aller Kanten aus `A` verschwindet, wird der Dataframe um eine Gewichts-Variable mit konstanter Ausprägung 0 erweitert.

Die Kanten aus `D` werden mit der für jede Haltestelle ermittelte Umsteigezeit versehen. Hierfür wird kurzzeitig ein weiterer Dataframe `HSmitHP` erzeugt, in welchem die jeweiligen Haltestellen mit ihrer Anzahl an Haltepunkten und den zugehörigen Umsteigezeiten festgehalten werden. Nachdem die Gewichte zu `D` hinzugefügt und in Sekunden umgerechnet wurden, wird `HSmitHP` nicht mehr benötigt und kann wieder aus dem Speicher entfernt werden.

Um die Kantengewichte für die Routenkanten zu bestimmen, wird analog zur Gleichung (4.2) die Gewichtsfunktion `Kantengew()` definiert, welche die Fahrzeit in Abhängigkeit der Startzeit liefert.
Der Funktion werden die Knoten u und v der betrachteten Routenkante $e = (u, v) \in R$ sowie die gewünschte Startzeit t übergeben. Hieraus werden zunächst alle Verbindungen entlang dieser Routenkante ausgewählt und daraus diejenige Verbindung ermittelt, welche die früheste Ankunft am Knoten v liefert. Eine eventuelle Abfahrt oder Ankunft nach Mitternacht wird berücksichtigt.
Um die Funktion im Dijkstra-Algorithmus auf alle Knoten des Graphen anwenden zu können, wurden auch die Kantengewichte für Knoten aus `A` und `D` ergänzt.

Bevor der Algorithmus implementiert werden kann, muss noch überprüft werden, ob die

FIFO-Bedingung (vgl. Definition 4.6) erfüllt ist.
Hierfür werden alle Fahrtereignisse nach gemeinsamen Start- und Zielhaltestellen sortiert und in chronologischer Reihenfolge der Abfahrtszeiten gegliedert. Im Anschluss werden je zwei Fahrzeitpaare mit demselben Start- und Zielort verglichen. Dabei wird überprüft, ob bei einer Verbindung mit späterer Abfahrtszeit am Abfahrtsort auch die Ankunftszeit am Ankunftsort nicht vor der Ankunftszeit des früher gestarteten Busses liegt, d.h. ob es während eines Streckenabschnitts zu Überholungen kommt oder nicht. Um die Anzahl an Vergleichen zu reduzieren werden keine Verbindungen betrachtet, bei dem die Abfahrtszeit des zweiten Busses am Startpunkt später ist als die Ankunftszeit des ersten Busses am Zielort.
In `fifox` werden alle Verbindungspaare gelistet, bei denen die FIFO-Bedingung verletzt ist.
Insgesamt ergeben sich elf Verbindungspaare in `fifox`. Die ersten vier Verbindungspaare betreffen die Strecke zwischen Dachauplatz und Gabelsbergerstraße. Während die Linie 10 dort nur eine Minute Fahrzeit benötigt, sind es bei der Linie 5 laut Fahrplan zwei Minuten, sodass die Busse gleichzeitig an der Gabelsbergerstraße ankommen, obwohl sie mit einer Minute Zeitdifferenz am Dachauplatz starten. Analog verhält es sich zwischen Weissenburgstraße und Gabelsbergerstraße, wie in Zeile 9-18 zu sehen ist.
Auch zwischen Haydnstraße und Arcaden gibt es ein Verbindungspaar aus den Linien C1 und 10, bei denen die FIFO-Bedingung verletzt ist. Während die Linie C1 um 19.52 Uhr an der Haydnstraße startet und um 19.54 Uhr an der Haltestelle HBF Süd Arcaden ankommt, startet die Linie 10 erst um 19.53 Uhr an der Haydnstraße, kommt aber zur selben Zeit wie C1 an den Arcaden an.
Das letzte Verbindungspaar beschreibt die Linien C6 und 6, welche sich zwischen Otto-Hahn-Straße und Universität analog zu den vorigen Paaren verhalten.
Da aber die auf diese Weise ermittelten Verbindungspaare jeweils aus unterschiedlichen Linien stammen, können sie auch in verschiedene Routen eingeordnet werden. Damit ist die FIFO-Bedingung nach Unterkapitel 4.3 trotzdem erfüllt, sodass der Dijkstra-Algorithmus anwendbar ist.

5.2.3. Implementierung des Dijkstra-Algorithmus

Um den Dijkstra-Algorithmus für zeitabhängige Graphen im realistischen zeitabhängigen Modell mit Umsteigegraphen zu implementieren, wird eine Vorrangwarteschlange benötigt. Hierfür wurde die vorimplementierte Rosetta-Code Priority Queue für R[7] verwendet, mit der man eine Vorrangwarteschleife erzeugen, neue Elemente mit Prioritäten einfügen und das Element mit höchster Priorität herausnehmen kann. Updates für Elemente und ihre Prioritäten, die bereits in der Queue enthalten sind, werden im Algorithmus direkt eingebaut, weil diese nicht im Rosetta-Code implementiert sind.

[7] Vgl. https://rosettacode.org/wiki/Priority_queue (Stand 15.05.2019).

Da dem Algorithmus die Start- und Zielhaltestellen `s` bzw. `d` als String übergeben werden müssen, können diese zuvor über eine Auswahlliste selektiert und abgespeichert werden. Eine manuelle Eingabe im Nachhinein ist ebenso möglich.

Der Dijkstra-Algorithmus selbst wird über die Funktion `dijkstra(V,E,s,d,t)` aufgerufen. Die Eingabewerte setzen sich zusammen aus

- der Knotenmenge `V` = `V`,
- der Kantenmenge `E` = `E1`,
- der Starthaltestelle `s`, welche aus der Variable `REC_ORT$ORT_NAME` stammen muss,
- der Zielhaltestelle `d`, welche ebenfalls aus `REC_ORT$ORT_NAME` stammen muss sowie
- der Startzeit `t`, welche im Format „`HH:MM:SS`“ zu übergeben ist.

Zunächst wird ein leerer Vektor `a` sowie leere Dataframes `Weg` und `fahrplanauskunft` erzeugt. In `a` werden die Zeiten der Fahrplan-Ereignisse abgespeichert. `Weg` sammelt alle abgelaufenen Vorgänger- und Nachfolgerknoten sowie die zugehörige Kanteninformation, ob es sich um einen Umstieg oder eine Routenkante einer bestimmten Linie handelt. Der Dataframe `fahrplanauskunft` liefert am Ende eine Tabelle mit allen Reiseinformationen zu Zeiten, angefahrenen Haltestellen, Umstiegen und Linien.
Die als Zeichenkette übergebenen Eingabewerte zu Abfahrtszeit, Start- und Zielhaltestelle werden anschließend zu Sekunden bzw. Haltepunkt-Nummern konvertiert.
Allen Knoten aus `V` wird die Zeitmarke ∞ zugeordnet. `s` wird mit der Zeitmarke `a[[s]] <- t` versehen.
Im anfangs leeren Vektor `S` werden später die permanent markierten Knoten gesammelt. Nachdem die Vorrangwarteschlange `Q` erzeugt worden ist, wird das Startereignis `(t,s)` hinzugefügt. Die Zeitkomponente `t` der Ereignisse bilden jeweils die Prioritätsschlüssel der Warteschlangenelemente.
In der While-Schleife werden nach und nach die Ereignisse mit höchster Priorität, d.h. diejenigen mit der kleinsten Zeitkomponente ausgewählt. Die Ortskomponente wird zum aktuell untersuchten Knoten `u` und die Zeitkomponente wird als Zeitmarke von `u` gesetzt; `u` wird permanent markiert. Falls es sich bei `u` um den Zielknoten `d` handelt, bricht die Schleife ab. Ansonsten werden die Nachfolgerknoten `v` von `u` betrachtet. Sofern die Knoten nicht permanent markiert sind, werden die Ankunftszeiten `t1` in `v` über die Gewichtsfunktion `Kantengew()` berechnet. Für den „Umstieg“ am Startbahnhof vom zentralen Haltestellenknoten zum Haltepunktknoten verschwindet die Umsteigezeit. Im Anschluss wird überprüft, ob `v` schon in `Q` enthalten ist und je nachdem zu `Q` hinzugefügt oder upgedatet; `t1` wird als aktuelle Zeitmarke gesetzt.
Danach wird der Dataframe `Weg` befüllt. Wenn `v` schon einmal über einen anderen Pfad abgelaufen wurde, so wird der frühere Vorgänger mit `u` überschrieben. Falls es sich um eine Kante aus `D` handelt, wird die Reiseinformation „`Weiterfahrt um`“ gespeichert.

Bei Kanten aus `A` wird die Information „`Umstieg`“ übergeben. Für Routenkanten aus `R` wird die Linie, aus der die Routenkante stammt, hinterlegt.
Nach Abbruch der While-Schleife wird die `fahrplanauskunft` befüllt. Diese wird beim Zielknoten beginnend von hinten nach vorne erstellt. Zeitmarken werden über die Funktion `seconds_to_period()` in das Format `HH:MM:SS` umgerechnet. Für die Zeitmarken der Zwischenhalte wird angenommen, dass die Ankunftszeit der Abfahrtszeit entspricht. Den Knoten werden die jeweiligen Haltestellen-Namen zugeordnet und die entsprechende Reiseinformation wird von `Weg` übergeben. Ziel- und Startknoten werden mit speziellen Reisezeitinformationen versehen. Darüberhinaus werden die Abfahrtszeiten der Weiterfahrten korrigiert und die Abfahrtszeit an der Starthaltestelle bestimmt.
Bevor die Fahrplanauskunft in chronologische Reihenfolge gebracht wird, wird eine Bedingung eingefügt, welche eine bevorzugte Weiterfahrt mit derselben Linie ermöglicht, sodass der Linienwechsel zum spätest möglichen Zeitpunkt erfolgt.

Am Ende liefert die Funktion `dijkstra()` als Ausgabe eine vollständige Fahrplanauskunft in tabellarischer Form. Zusätzlich werden die Reisezeiten in der Konsole angezeigt.

6. Beispiele zur Lösung des Earliest Arrival Problem

In diesem Abschnitt wird die Lösung des EAP auf Basis der Regensburger Fahrplandaten vorgestellt. Hierfür werden einzelne Fahrplanauskunft-Beispiele des zuvor implementierten Dijkstra-Algorithmus diskutiert.

Dabei ist anzumerken, dass die folgenden Beispiele keinen Anspruch darauf zu erheben, eine für die Passagiere des Regensburger Busverkehrs entwickelte und optimierte Reiseauskunft zu sein, wie es beispielsweise die RVV-App oder der DB Navigator anbieten. Vielmehr sollen die Beispiele aus dem Regensburger Stadtverkehr als praktische Ergänzung zur theoretischen Modellausarbeitung des EAP verstanden werden.

Im Kapitel 7 werden dann mögliche Aspekte und Ansatzpunkte für umfangreichere sowie detailliertere Fahrplanabfragen erörtert.

Beispiel 6.1. Im ersten Beispiel wird eine Fahrplanauskunft ohne Umstiege erzeugt. Gesucht wird eine Busverbindung vom Fischmarkt zum Arnulfsplatz um 08.00 Uhr morgens. Über die Funktion `dijkstra()` erhält man eine Fahrplanauskunft, welche den frühest möglichen Ankunftszeitpunkt an der Zielhaltestelle Arnulfsplatz liefert. Bei der Abfrage der Funktion ist auf die richtigen Eingabewerte zu achten. Insbesondere darf der zugrundeliegende Graph $G = (V, E1)$ in den Input-Daten nicht vergessen werden:

```
1 dijkstra(V, E1, "Fischmarkt", "Arnulfsplatz", "08:00:00")
```

Als Ergebnis wird folgende Fahrplanauskunft wiedergegeben:[8]

	Zeit	Haltestelle	Linie_Umstieg
5	8H 0M 0S	Fischmarkt	gewuenschter Start
4	8H 3M 0S	Fischmarkt	Abfahrt am Start
3	8H 4M 0S	Keplerstra_e	1
2	8H 6M 0S	Arnulfsplatz	1
1	8H 6M 0S	Arnulfsplatz	Ankunft am Ziel

Abbildung 6.1: Fahrplanauskunft vom Fischmarkt zum Arnulfsplatz um 08.00 Uhr.

Die linke Spalte (`Zeit`) liefert alle relevanten Zeiten, die mittlere Spalte (`Haltestelle`) enthält die angefahrenen Haltestellen bzw. Zwischenhaltestellen und die rechte Spalte (`Linie_Umstieg`) umfasst weitere Fahrplaninformationen.
In der ersten Zeile ist die Starthaltestelle mit der gewünschten Startzeit abzulesen. Die tatsächliche Abfahrtszeit an der ersten Haltestelle findet man in der zweiten Zeile der Tabelle. Danach folgen alle Zwischenhaltestellen bis zur Zielhaltestelle. Diese Zwischenhalte werden mit den Zeiten versehen, an denen sie fahrplanmäßig passiert werden. An den Zwischenhalten wird angenommen, dass die Ankunftszeit der Abfahrtszeit entspricht. In der rechten Spalte ist in diesen Zeilen die entsprechende Buslinie abzulesen, auf der man verkehrt. In der letzten Zeile ist das Ziel mit Ankunftszeit vermerkt.
Aufgrund der Funktionsweise des Algorithmus ist die Ankunftszeit in der letzten Zeile der Fahrplanauskunft die frühest mögliche Ankunftszeit zur gegebenen Startzeit auf Basis der Fahrplandaten.

Auf das vorliegende Beispiel bezogen, liefert die Fahrplanauskunft folgende konkrete Informationen:

- Die vom Benutzer übergebene Starthaltestelle ist die Haltestelle Fischmarkt. Die gewünschte Abfahrtszeit am Fischmarkt ist 08.00 Uhr.
- Die tatsächliche Abfahrt am Fischmarkt ist 08.03 Uhr.
- Mit der Linie 1 verkehrt man über die Haltestelle Keplerstraße um 08.04 Uhr bis zum Arnulfsplatz um 08.06 Uhr.

[8] Die inkorrekte Rechtschreibung innerhalb der Fahrplanauskunft-Tabellen stammen aus der Datenschnittstelle und werden von der Autorin unverändert übernommen.

- Die sich aus dieser Route ergebende Ankunftszeit an der Zielhaltestelle Arnulfsplatz ist 08.06 Uhr.

□

Nachdem das erste Beispiel den grundlegenden Aufbau der Fahrplanauskunft aufgezeigt hat, soll im folgenden Beispiel eine Route mit Umstieg betrachtet werden.

Beispiel 6.2. In diesem Beispiel ist ein potentielles Routing-Problem eines Studierenden der OTH Regensburg zu sehen. Es wird angenommen, dass der Studierende im Süden der Regensburger Altstadt wohnt und morgens um 07.30 Uhr den Bus zur OTH nehmen möchte. Seine Reise soll am zentralen Busbahnhof HBF-Albertstraße starten und an der Haltestelle Tech-Campus OTH enden.

```
1 dijkstra(V,E1, "HBF/Albertstra_e", "TechCampus/OTH", "07:30:00")
```

Folgende Reiseroute liefert hieraus die frühest mögliche Ankunft am Ziel:

	Zeit	Haltestelle	Linie_Umstieg
8	7H 30M 0S	HBF/Albertstra_e	gewuenschter Start
7	7H 30M 0S	HBF/Albertstra_e	Abfahrt am Start
6	7H 32M 0S	HBFSöd/Arcaden	3
5	7H 32M 0S	HBFSöd/Arcaden	Umstieg
4	7H 38M 0S	HBFSöd/Arcaden	Weiterfahrt um
3	7H 40M 0S	Haydnstra_e	6
2	7H 41M 0S	TechCampus/OTH	6
1	7H 41M 0S	TechCampus/OTH	Ankunft am Ziel

Abbildung 6.2: Fahrplanauskunft von HBF/Albertstr. zu Tech Campus/OTH um 07.30 Uhr.

Die gewünschte Abfahrtszeit 07.30 Uhr an der Alberstraße entspricht hier der tatsächlichen Abfahrtszeit. Mit der Linie 3 erreicht der Studierende um 07.32 Uhr die Haltestelle HBF Süd Arcaden. Da die Linie 3 Richtung Graß allerdings nicht über die Hochschule verkehrt, wird ein Umstieg nötig. Dies ist mit der Reiseinformation `Umstieg` in der rechten Spalte der Zeile[9] 5 vermerkt, denn der Umstieg ist in diesem Fall mit einem Haltepunktwechsel verbunden. Aus Zeile 4 ergibt sich anschließend eine Weiterfahrt um 07.38 Uhr. Diese Weiterfahrt erfolgt mit der Linie 6 über die Haltestelle Haydnstraße

[9]Es ist zu beachten, dass die Zeilennummern aufgrund der Implementierung des Algorithmus von oben nach unten absteigen.

um 07.40 Uhr, bis zur Zielhaltestelle Tech Campus. Die Ankunftszeit am Ziel beträgt 07.41 Uhr. □

Ein weiterer Spezialfall der Fahrplanauskunft wird im dritten Beispiel deutlich, das den Unterschied beim Umsteigen mit und ohne Haltepunktwechsel aufzeigen soll.

Beispiel 6.3. Gegeben seien die Starthaltestelle Universität, die Zielhaltestelle Prüfening sowie die frühest mögliche Abfahrtszeit 17.45 Uhr.
Daraus wird folgende Reiseroute berechnet:

	Zeit	Haltestelle	Linie_Umstieg
21	17H 45M 0S	Universitdt	gewuenschter Start
20	17H 52M 0S	Universitdt	Abfahrt am Start
19	17H 55M 0S	WolfgangschuleNord	2
18	17H 56M 0S	Theodor-Storm-Stra_e	2
17	17H 57M 0S	Nibelungenstra_e	4
16	17H 58M 0S	Kaulbachweg	4
15	17H 59M 0S	Klenzestra_e	4
14	18H 1M 0S	Dr.-Gessler-Stra_e	4
13	18H 2M 0S	Dörerstra_e	4
12	18H 3M 0S	Schwalbenneststra_e	4
11	18H 5M 0S	Dechbetten-TELISFINANZ	4
10	18H 7M 0S	Lilienthalstra_e	4
9	18H 7M 0S	Lilienthalstra_e	Umstieg
8	18H 15M 0S	Lilienthalstra_e	Weiterfahrt um
7	18H 16M 0S	Rennplatz	1
6	18H 17M 0S	Killermannstra_e	1
5	18H 18M 0S	Deiningerstra_e	1
4	18H 19M 0S	Annahofstra_e	1
3	18H 20M 0S	AndenKlostergrönden	1
2	18H 23M 0S	Pröfening	1
1	18H 23M 0S	Pröfening	Ankunft am Ziel

Abbildung 6.3: Fahrplanauskunft von Universität nach Prüfening um 17.45 Uhr.

Auf den ersten Blick ähnelt diese Fahrplanauskunft vom Aufbau her der aus Beispiel 2. Nach der Abfahrt um 17.52 Uhr an der Universität folgen einige Zwischenhalte bis an der Haltelinie Lilienthalstraße ein Umstieg von der Linie 4 zur Linie 1 stattfindet. Dabei ändert sich allerdings nicht nur an dieser Haltestelle die Buslinie sondern bereits zuvor, wie zwischen Zeile 18 und Zeile 17 einzusehen ist. Während der Streckenabschnitt

von der Universität über die Wolfgangschule (Nord) bis zur Theodor-Storm-Straße mit der Linie 2 zurückgelegt wird, verläuft die Weiterfahrt ab Theodor-Storm-Straße bis zur Lilienthalstraße mit der Linie 4.

Warum wird der Umstieg an der Theodor-Storm-Straße von Linie 2 zu Linie 4 nicht genauso deutlich vermerkt wie der von Linie 4 zu Linie 1 an der Lilienthalstraße?
Der Grund hierfür ist, dass an der Lilienthalstraße ein Umstieg mit Haltepunktwechsel vollzogen wird, während an der Theodor-Storm-Straße der Bus der Linie 4 am gleichen Haltepunkt hält wie der der Linie 2, sodass dort zwar der Bus gewechselt werden muss, aber keine Strecke zu einem anderen Steig zurückgelegt werden muss.
Im graphentheoretischen Modell ergibt sich daraus im ersten Fall ein Pfad vom Ankunftshaltepunkt über den zentralen Haltestellenknoten zum – vom Ankunftshaltepunkt verschiedenen – Abfahrtshaltepunkt an derselben Haltestelle Lilienthalstraße. Dieser modellierte Umstieg über den zentralen Haltestellenknoten liefert in der Fahrplanauskunft die Reiseinformation `Umstieg`.
Dagegen wird der Umstieg bei einer Weiterfahrt am selben Steig nicht im graphentheoretischen Modell erfasst. Das liegt daran, dass jeder Haltepunkt im Modell genau einem Knoten entspricht und alle mit dem Haltepunkt verbundenen Ankunfts- und Abfahrtsereignisse (Kanten) in diesem Knoten enden bzw. von ihm starten. Dadurch wird beim Umstieg kein zusätzlicher Pfad innerhalb der Station über den zentralen Haltestellenknoten zurückgelegt, sondern direkt am Knoten auf den Anschluss „gewartet“. Dementsprechend entfällt dann aber die Zeile mit der Umstiegs-Information in der Fahrplanauskunft. □

Wie in Beispiel 6.3 ersichtlich können also zwei verschiedene „Umsteige-Arten“ aus der Fahrplanauskunft gelesen werden.

- Wenn beim Umstieg der Steig gewechselt werden muss, taucht in der Fahrplanauskunft eine separate Zeile auf, welche in der rechten Spalte die Reiseinformation `Umstieg` enthält.
- Wenn beim Umstieg der Steig nicht gewechselt werden muss, taucht in der Fahrplanauskunft keine zusätzliche Zeile auf, die den Umstieg beschreibt. Stattdessen muss man auf einen eventuellen Wechsel der Liniennummern in der dritten Spalte achten.

Im vierten Beispiel wird aufgezeigt, wie eine Fahrplanabfrage um Mitternacht berechnet wird. Dies soll auf die Probleme hinweisen, welche sich durch die Einschränkung auf den Montagsfahrplan ergeben.

Beispiel 6.4. Ziel ist es, einen Reiseplan von der Goethestraße zum Rennplatz für die Zeit um 00.00 Uhr zu berechnen. Hierfür werden zwei verschiedene Startzeitpunkte gewählt und die Ergebnisse miteinander verglichen.

```
1 dijkstra(V,E1, "Goethestra_e", "Rennplatz", "23:55:00")
```

Wenn als gewünschter Startzeitpunkt die Zeit 23.55 Uhr gewählt wird, ergibt sich die Fahrplanauskunft:

	Zeit	Haltestelle	Linie_Umstieg
8	23H 55M 0S	Goethestra_e	gewuenschter Start
7	1d 0H 8M 0S	Goethestra_e	Abfahrt am Start
6	1d 0H 9M 0S	Lessingstra_e	1
5	1d 0H 10M 0S	Margaretenau/KHBarmh.Bröder	1
4	1d 0H 11M 0S	Michael-Burgau-Stra_e	1
3	1d 0H 12M 0S	Lilienthalstra_e	1
2	1d 0H 13M 0S	Rennplatz	1
1	1d 0H 13M 0S	Rennplatz	Ankunft am Ziel

Abbildung 6.4: Fahrplanauskunft von Goethestraße zum Rennplatz um 23.55 Uhr.

Bei einer gewünschten Abfahrt vor Mitternacht liefert der Dijkstra-Algorithmus als tatsächliche Abfahrtszeit an der Goethestraße 00.08 Uhr des Folgetages (`1d 0H 8M 0S`). Mit der Linie 1 verkehrt man von dort aus über vier Zwischenhalte zur Zielhaltestelle Rennplatz, welche man um 00.13 Uhr erreicht.

In der zweiten Abfrage wählen wir Mitternacht als Abfahrtszeit:

```
1 dijkstra(V,E1, "Goethestra_e", "Rennplatz", "00:00:00")
```

Bei einer Startzeit um 00.00 Uhr ergibt sich nun diese Fahrplanauskunft:

	Zeit	Haltestelle	Linie_Umstieg
8	0S	Goethestra_e	gewuenschter Start
7	5H 46M 0S	Goethestra_e	Abfahrt am Start
6	5H 47M 0S	Lessingstra_e	1
5	5H 48M 0S	Margaretenau/KHBarmh.Bröder	1
4	5H 49M 0S	Michael-Burgau-Stra_e	1
3	5H 50M 0S	Lilienthalstra_e	1
2	5H 51M 0S	Rennplatz	1
1	5H 51M 0S	Rennplatz	Ankunft am Ziel

Abbildung 6.5: Fahrplanauskunft von Goethestraße zum Rennplatz um 00.00 Uhr.

Die Tabelle liefert als tatsächliche Abfahrtszeit an der Goethestraße 05.46 Uhr. Über dieselben Zwischenhalte wie bei der obigen Fahrplanauskunft erreicht man mit der Linie 1 die Zielhaltestelle Rennplatz um 05.51 Uhr.

Nach Konstruktion des Algorithmus liefern beide Tabellen die frühest mögliche Ankunftszeit zu gegebenen Startwerten auf Basis des Fahrplans. Trotzdem wird in der zweiten Abfrage nicht die Verbindung von 00.08-00.13 Uhr angezeigt (welche bei einer frühest möglichen Startzeit von 00.00 Uhr erreichbar wäre), sondern erst die fünfeinhalb Stunden spätere Verbindung von 05.46-05.51 Uhr. Wieso liefern die beiden Abfragen unterschiedliche Ergebnisse?
Der Grund für die inkonsistente Fahrplanauskunft liegt darin, dass bei der Implementierung des Algorithmus nur Daten aus dem Montagsfahrplan verwendet wurden. Jede potentielle Startzeit, die dem Algorithmus als Input übergeben wird, bezeichnet also nicht die Tageszeit an einem beliebigen Wochentag, sondern ist nur montags gültig.
Dementsprechend liegen zwischen den beiden betrachteten Zeiten 23.55 Uhr und 00.00 Uhr nicht fünf Minuten, sondern fast 24 Stunden. Bei der zweiten Abfrage wird nach der ersten überhaupt möglichen Abfahrt ab 00.00 Uhr in der Nacht von Sonntag auf Montag gesucht. Da keine Fahrplandaten der letzten Busfahrten des Sonntagsfahrplans eingepflegt sind, ergibt sich die erste Reisemöglichkeit erst nach Betriebsbeginn um 05.00 Uhr morgens.
Mit der ersten Abfrage (Startzeit 23.55 Uhr) wird dagegen auf die letzten Fahrten des Montags zugegriffen, welche den Fahrzeitraum von 00.08-00.13 Uhr aus der letzten Tagesfahrt der Linie 1 liefert. Diese Fahrt findet allerdings, wie in der Fahrplanauskunft mit `1d` vermerkt, am Folgetag (dem Dienstag) statt.

Unter Berücksichtigung dieses Sachverhalts lassen sich die Reisepläne zu beliebigen Abfahrtszeiten korrekt berechnen. Sie sind dann auch auf andere Wochentage übertragbar, da sich der Montagsfahrplan nicht wesentlich von den anderen Tagesfahrplänen (außer an Wochenenden und Feiertagen) unterscheidet. □

Das letzte Beispiel soll verdeutlichen, dass die Routenabfrage allein auf den Fahrplandaten beruht. Ein potentieller Fußweg von einer Haltestelle zur anderen wird nicht berücksichtigt. Zudem zeigt es, wie sich der Montagsfahrplan für den Folgetag wiederholt.

Beispiel 6.5. Der Dijkstra-Funktion werden Inputdaten für eine Fahrt übergeben, welche vom Fischmarkt zum Dachauplatz führen soll und welche als frühest möglichen Startzeitpunkt 23.55 Uhr festsetzt.

```
1 dijkstra(V, E1, "Fischmarkt", "Dachauplatz","23:55:00")
```

Hieraus ergibt sich die nachstehende Fahrplanauskunft:

	Zeit	Haltestelle	Linie_Umstieg
9	23H 55M 0S	Fischmarkt	gewuenschter Start
8	1d 0H 8M 0S	Fischmarkt	Abfahrt am Start
7	1d 0H 10M 0S	Thundorferstra_e	4
6	1d 0H 11M 0S	EiserneBröcke	4
5	1d 0H 11M 0S	EiserneBröcke	Umstieg
4	1d 5H 18M 0S	EiserneBröcke	Weiterfahrt um
3	1d 5H 19M 0S	MuseumderBayerischenGeschichte	3
2	1d 5H 21M 0S	Dachauplatz	3
1	1d 5H 21M 0S	Dachauplatz	Ankunft am Ziel

Abbildung 6.6: Fahrplanauskunft von Fischmarkt zum Dachauplatz um 23.55 Uhr.

Die Fahrt beginnt am Fischmarkt um 00.08 Uhr des Folgetages und führt mit der Linie 4 über die Thundorferstraße zur Eisernen Brücke um 00.11 Uhr. Dort erfolgt ein Umstieg zur Linie 3. Hervorzuheben ist dabei, dass die Weiterfahrt mit der Linie 3 erst um 05.18 Uhr beginnt, was eine Wartezeit von über fünf Stunden ergibt. Die Fahrt wird über die Haltestelle Museum der Bayerischen Geschichte bis zum Dachauplatz fortgesetzt und erreicht das Ziel um 05.21 Uhr.

Hieraus ergeben sich zwei Erkenntnisse:

- Zum einen wird der Montagsfahrplan nach Beendigung des Betriebstages wiederholt, sodass eine Verbindungssuche darüberhinaus möglich ist. Die gewünschte Abfahrtszeit darf sich jedoch nicht auf den Folgetag beziehen.
- Zum anderen werden bei der Fahrplanabfrage in dieser Form lediglich konkrete Fahrplan-Verbindungen und keine Fußwege von einer Haltestelle zu einer anderen

berücksichtigt. Ansonsten hätte man als Fahrplanauskunft wohl kaum den obigen erhalten, welcher eine Wartezeit von über fünf Stunden an der Haltestelle Eiserne Brücke vorschlägt. Schließlich bedarf der Fußmarsch von dieser Haltestelle zum Dachauplatz nur ca. zehn Minuten.

□

Die Liste an Beispielen lässt sich noch weiter fortsetzen. Da dieses Kapitel jedoch hauptsächlich einen Einblick in den praktischen Teil der Lösung des EAP geben soll, sei an dieser Stelle auf weitere Beispiele verzichtet.

Stattdessen sollen im nächsten Abschnitt einige Erweiterungsmöglichkeiten des hier vorgestellten Lösungsansatzes gegeben werden.

7. Ausblick

In diesem Kapitel werden verschiedene Ansätze erläutert, wie die Lösung des Earliest Arrival Problem und die damit einhergehende Fahrplanauskunft noch weiter verfeinert und optimiert werden kann. Diese sollen als Vorschläge für eine weiterführende Beschäftigung mit der Fragestellung verstanden werden und so die Optimierung des ÖPNV am Beispiel der Passagierroutenplanung im Regensburger Stadtverkehr abrunden.

Das Kapitel ist in zwei Teile gegliedert. Im ersten Abschnitt sollen Möglichkeiten erörtert werden, wie die Fahrplanabfrage auf Basis des Dijkstra-Algorithmus für den zeitabhängigen Graphen im realistischen zeitabhängigen Modell erweitert werden kann. Im zweiten Abschnitt sollen Ansätze diskutiert werden, die das Modell präzisieren und ausweiten.

7.1. Optimierung der Fahrplanauskunft auf Basis des Modells

Betrachtet man die Beispiele aus Kapitel 6, werden einige Optimierungspotentiale deutlich. Diese betreffen zum Beispiel die Bereiche Umstiege und Fahrpläne über mehrere Betriebstage hinweg.

Umstiege

Wie in Beispiel 6.3 zu sehen ist, werden die Umstiege je nachdem, ob an der Haltestelle ein Haltepunktwechsel vollzogen werden muss oder nicht, in der Fahrplanauskunft unterschiedlich dargestellt. Während ein Umstieg mit Wechsel des Steiges durch einen entsprechenden Hinweis deutlich zu erkennen ist sowie die Abfahrtszeit der Weiterfahrt

separat vermerkt ist, wird ein Umstieg ohne Plattformwechsel nur durch die veränderte Buslinie ersichtlich.

Wie im vorherigen Kapitel erklärt wurde, liegt dies an der Konstruktion des Graphen. Aufgrund der Funktionsweise des Djikstra-Algorithmus kann der Umsteigeprozess ohne Haltepunktwechsel auch nicht ohne Weiteres im graphentheoretischen Modell eingebaut werden. Allerdings kann der Umstieg bei der Implementierung des Algorithmus eingepflegt werden. Hierfür kann beispielsweise beim Wechsel der Liniennummern in der Fahrplanauskunft die Tabelle um zusätzliche Zeilen für den Umstieg und die Weiterfahrt ergänzt werden. Damit wäre auch ein Umstieg ohne Steigwechsel einfacher aus der Fahrplanauskunft herauszulesen. Problematisch ist allerdings, dass wegen des zugrundeliegenden Modells für einen solchen Umstieg keine Umsteigezeit eingearbeitet werden kann. Dieses Problem lässt sich auch mit zusätzlichen Zeilen in der Fahrplanauskunft nicht beheben.

Ein weiteres Defizit der Fahrplanauskunft sind die fehlenden Verweise auf die konkreten Haltepunkte, an denen die Busse halten. Schließlich ist aus der Tabelle nur die jeweilige Haltestelle ohne Kennzeichnungen des Steiges vermerkt.
Grund hierfür ist, dass in der Datenschnittstelle keine konkrete Aufschlüsselung zwischen den Haltepunkt-Nummern und den alphabetisch sortierten Masten der Haltestellen existiert. Diese Identifizierung müsste entweder bei den Stadtwerken erfragt werden oder manuell in die Daten eingearbeitet werden.
Alternativ kann die Orientierung an den einzelnen Haltestellen erleichtert werden, indem neben den Liniennummern zusätzlich die Fahrtrichtungen durch Angabe der Endhaltestellen vermerkt werden. Aus diesen Informationen kann der Fahrgast in der Regel den angefahrenen Steig bestimmen.

Darüberhinaus wären auch Informationen zu den benötigten und tatsächlich zur Verfügung stehenden Umsteigezeiten an den einzelnen Haltestellen sinnvoll.

Erweiterung der Fahrplandaten

Aus Beispiel 6.4 ist erkennbar, dass besonders Fahrplanabfragen für Fahrten um Mitternacht Probleme bereiten können. Grund hierfür ist die Reduktion der Fahrplandaten auf den Montagfahrplan, welche bei der Implementierung vorgenommen wurde. Diese bewirkt, dass keine Daten zu den Fahrzeiten der Busse an anderen Tagen abgefragt werden. Um Reisen über den Betriebsschluss hinaus darzustellen, werden die vorhandenen Fahrplandaten zwar wiederholt, trotzdem kann keine vollkommen realistische Abfrage mit dieser Datengrundlage erzeugt werden.
Um dem Problem zu entgehen, können zusätzlich die Fahrplandaten der restlichen Wochentage betrachtet werden und eventuell weitere Kalenderdaten wie Semester- und Schulferien eingearbeitet werden. Die Spezialfälle, die sich daraus ergeben, können über die Fahrzeitgruppen unterschieden werden. Allerdings liefert die Relation *MENGE_FGR*

nicht die korrekten Charakterisierungen der Fahrzeitgruppen, sodass diese manuell eingepflegt werden müssen. Alle Fahrzeitgruppen des Montagsfahrplans wurden von der Autorin bereits spezifiziert.
Die Erweiterung der Fahrplandaten auf den kompletten Wochen- oder sogar Jahresfahrplan macht allerdings auch eine Konkretisierung der Input-Daten unausweichlich. Schließlich ist die Berechnung der Route dann nicht mehr nur von der Uhrzeit sondern auch vom jeweiligen Wochentag bzw. Datum abhängig. Also muss hier zudem der Algorithmus entsprechend angepasst werden.

Eine weitere Möglichkeit zur Präzisierung der Fahrplanauskunft ist die Verwendung von sogenannten Echtzeit-Daten, welche die aktuellen Fahrzeiten der Busse beinhalten, sodass auch alternative Routen bei Bus-Verspätungen vorgeschlagen werden können.

Routen-Priorisierung

Da das EAP als Optimierungsproblem nur die Suche nach dem frühest möglichen Ankunftszeitpunkt als Ziel verfolgt, wird die Fahrplanauskunft auch nur darauf basierend berechnet. Dementsprechend sind alle Reisewege, die dieselbe Ankunftszeit an der Zielstation liefern, gleichwertig. Welche dieser Wege ausgewählt wird, hängt hauptsächlich davon ab, in welcher Reihenfolge die Kanten im Graphen gespeichert sind und in welcher Reihenfolge sie während des Algorithmus aufgerufen werden.

Auch wenn der Fahrgast die frühest mögliche Ankunft als Hauptentscheidungskriterium wählt, wird er nicht alle Reisewege, die zur selben Zeit am Ziel ankommen, als gleichwertig ansehen. Beispielsweise möchte er Umstiege soweit wie möglich vermeiden oder durch eine möglichst späte Abfahrt an der Starthaltestelle die Reisezeit verkürzen.

Ändert man in Beispiel 6.2 die gewünschte Abfahrtszeit von 07.30 Uhr auf 07.37 Uhr, so wird folgende Fahrplanauskunft erzeugt:

	Zeit	Haltestelle	Linie_Umstieg
6	7H 37M 0S	HBF/Albertstra_e	gewuenschter Start
5	7H 37M 0S	HBF/Albertstra_e	Abfahrt am Start
4	7H 38M 0S	HBFSöd/Arcaden	6
3	7H 40M 0S	Haydnstra_e	6
2	7H 41M 0S	TechCampus/OTH	6
1	7H 41M 0S	TechCampus/OTH	Ankunft am Ziel

Abbildung 7.1: Fahrplanauskunft von HBF/Albertstr. zu Tech Campus/OTH um 07.37 Uhr.

Wie man sieht liefert diese Abfrage dieselbe Ankunftszeit am Ziel wie die Abfrage aus

Beispiel 6.2. Trotzdem ist die hier erzeugte Verbindung auch bei einer gewünschten Startzeit um 07.30 Uhr wesentlich attraktiver, da sie einerseits die Reisezeit verkürzt und andererseits eine Route ohne Umstieg liefert.

Eine Überarbeitung des Algorithmus, sodass Routen mit kürzeren Reisezeiten und einer möglichst geringen Anzahl an Umstiegen priorisiert werden, liefert einen guten Ansatz zur Optimierung der Fahrplanauskunft. Die Reduktion der Reisezeit kann z.B. über eine zusätzliche Backward-Suche erreicht werden, welche die spätestmögliche Abfahrtszeit bei gegebener Ankunftszeit berechnet.

7.2. Zusätzliche Modellerweiterungen

Um das Passagierrouting-Problem zu verfeinern, kann nicht nur die Fahrplanauskunft durch Anpassung der Implementierung oder durch Erweiterung der Fahrplandaten spezifiziert werden. Stattdessen können Optimierungspotentiale auf einer tiefer gehenden Ebene genutzt werden, indem das graphentheoretische Modell sowie der zur Lösung führende Algorithmus adaptiert werden. In diesem Abschnitt sollen hierzu diverse Möglichkeiten aufgezeigt werden.

Fußwege

Die Fahrplanabfrage zur Lösung des EAP basiert auf der Annahme, dass außerhalb einer Haltestelle die Fortbewegung nur mittels Bussen möglich ist. Es werden keine Fußwege von einer Station zur nächsten berücksichtigt, obwohl es in vielen Fällen sinnvoller wäre, zu einer anderen Haltestelle zu laufen, als auf den nächsten Bus zu warten. Beispiel 6.5 aus Kapitel 6 schildert diesbezüglich einen Extremfall.

Aber Fußwege anstelle von Busverbindungen liefern nicht nur nach Abfahrt des letzten Busses eines Betriebstages eine Alternative. Ein Fußmarsch bietet sich häufig auch dann an, wenn zwei Haltestellen nah beieinander liegen, aber nicht durch eine Buslinie miteinander verbunden sind. Zum Beispiel ist die Haltestelle Gumpelzhaimerstraße von der Haltestelle Hans-Sachs-Straße über die Schenkendorfstraße zu Fuß schnell erreichbar. Würde man dieselbe Strecke aber mit dem Bus zurücklegen, müsste man zuerst mit der Linie 11 zur Boessnerstraße oder Ostdeutschen Galerie fahren und dort in die Linie 6 umsteigen.

Damit diese möglichen Fußwege als alternative Routen berücksichtigt werden, müssen neue Kanten im Graphen erzeugt werden, die naheliegende Stationen (z.B. über deren zentrale Haltestellenknoten) miteinander verbinden. Diese Kanten können mit der benötigten Gehzeit gewichtet werden.

Sicherlich ist es dabei nicht sinnvoll, alle Stationen eines Liniennetzes miteinander über Fußwege zu verbinden, da die meisten Wege in diesem Fall nur unnötigen Speicher kosten würden, ohne einen Mehrnutzen zu erzeugen. Stattdessen könnten beispielsweise alle

Wege als Kanten modelliert werden, für die eine Gehzeit von maximal zehn Minuten zu veranschlagen ist.

Passagierrouten, die Fußwege enthalten, sollten in jedem Fall nur als Alternative zu reinen Busrouten betrachtet werden und diese nicht ersetzen. Schließlich gibt es einige Fahrgäste, die physisch nicht dazu in der Lage sind, selbst kleine Strecken zu Fuß zurückzulegen und gerade deshalb den ÖPNV nutzen. Zudem empfinden viele Passagiere das Einbinden von Fußwegen als Komfort-Verlust, auch wenn dies eine deutlich schnellere Ankunft ermöglicht, wie die Betriebsleiterin vom Stadtwerk-Mobilität Regensburg aus langjähriger Erfahrung im ÖPNV berichtet.

Früheste Ankunft zu allen Zeiten

Die durch den Dijkstra-Algorithmus gefundene Lösung liefert das Optimum für genau einen Zeitpunkt, nämlich der gewünschten Abfahrtszeit t. Entspricht aber diese *gewünschte* Abfahrtszeit im eigentlichen Sinne der *frühest möglichen* Abfahrtszeit, ist nicht nur die frühest mögliche Ankunftszeit zur Zeit t sondern zu allen Abfahrtszeiten τ eines Tages mit $t \leq \tau$ relevant.

Betrachtet man zum Beispiel die Strecke zwischen Dachauplatz und Weißenburgstraße, dann liefert die Fahrplanauskunft auf Basis des EAP für eine Abfahrtszeit t um 09.28 Uhr diese Verbindung mit Ankunftszeit 9.36 Uhr:

	Zeit	Haltestelle	Linie_Umstieg
6	9H 28M 0S	Dachauplatz	gewuenschter Start
5	9H 28M 0S	Dachauplatz	Abfahrt am Start
4	9H 31M 0S	HBF/Albertstra_e	1
3	9H 34M 0S	Stobdusplatz	1
2	9H 36M 0S	Wei_enburgstra_e	1
1	9H 36M 0S	Wei_enburgstra_e	Ankunft am Ziel

Abbildung 7.2: Fahrplanauskunft von Dachauplatz zu Weißenburgstraße um 09.28 Uhr.

Startet man dagegen erst um 09.33 Uhr, so liefert der Dijkstra-Algorithmus eine Verbindung, die nur eine Minute später am Ziel ankommt:

	Zeit	Haltestelle	Linie_Umstieg
5	9H 33M 0S	Dachauplatz	gewuenschter Start
4	9H 33M 0S	Dachauplatz	Abfahrt am Start
3	9H 34M 0S	Gabelsbergerstra_e	10
2	9H 37M 0S	Wei_enburgstra_e	10
1	9H 37M 0S	Wei_enburgstra_e	Ankunft am Ziel

Abbildung 7.3: Fahrplanauskunft von Dachauplatz zu Weißenburgstraße um 09.33 Uhr.

Da die zweite Verbindung später an der Zielhaltestelle ankommt, stellt sie nur eine suboptimale Lösung des EAP zum Startzeitpunkt 9.28 Uhr dar und wird dementsprechend nicht angezeigt. Trotzdem ist die Verbindung in den meisten Fällen attraktiver, da sie die Reisezeit um die Hälfte (von acht auf vier Minuten) verkürzt.

Wie man sieht, liefert die optimale Lösung des EAP nicht immer die optimale Reiseroute für den Fahrgast, sodass statt einer eindeutigen Lösung eine Auswahl aller optimalen Reiserouten zu jeder Zeit größer t als Fahrplanauskunft geeigneter wäre.
Mathematisch entspricht das Ergebnis der Fahrplanabfrage in diesem Fall einer stückweise linearen Funktion, bei der jeder Interpolationspunkt die optimale Route für eine bestimmte Zeit repräsentiert. Aus dem aktuellen Dijkstra-Algorithmus wird ein *Multi-Label-Correcting-Algorithmus* mit mehreren Zeitmarken für jeden Knoten, was eine Erhöhung der Komplexität und der Rechenzeit mit sich bringt. Für weitere Informationen sei an dieser Stelle auf [Paj09, S. 48-51] verwiesen.

Multi-Criteria Optimierung

Wie im Unterabschnitt zur Routen-Priorisierung bereits vermerkt, gibt es neben dem frühest möglichen Ankunftszeitpunkt noch weitere Kriterien, die eine Reiseroute besonders attraktiv machen. Beispielsweise soll die Reisezeit möglichst kurz sein, Umstiege ebenso wie überfüllte Busse sollen vermieden werden, Reisekosten reduziert werden oder die Verbindung soll möglichst robust gewählt werden. Letzteres bedeutet, die Route soll nicht anfällig für Verspätungen oder Busausfälle sein bzw. geeignete Alternativen für solche Situationen bereithalten.

Während die oben genannte Routen-Priorisierung das EAP optimal löst und aus der Lösungsmenge diejenige Route wählt, welche die weiteren Kriterien bestmöglich erfüllt, wird bei der Multi-Criteria Optimierung eine mögliche Lösungsmenge und eine mehrdimensionale Zielfunktion bestimmt, welche ein Pareto-Optimum für alle relevanten Kriterien über Trade-Offs berechnet.
Diese Optimierungsprobleme sind wegen der meist exponentiell wachsenden Größe der Pareto-Menge in der Regel NP-schwer, weswegen neben exakten Lösungswegen häufig

auch heuristische Lösungsverfahren zum Einsatz kommen. In [MHSWZ07, S. 79-85] werden verschiedene Ansätze zur Multi-Criteria Shortest Path Optimierung genauer diskutiert.

Speed-Up Techniken

Obwohl der aktuelle Dijkstra-Algorithmus für relativ kleine Verkehrsnetze wie Regensburg eine akzeptable Geschwindigkeit bei der Erstellung der Fahrplanauskunft liefert, werden vor allem bei großen Liniennetzen, beispielsweise Netze des deutschland- oder europaweiten Fernverkehrs, sogenannte Speed-Up Techniken unerlässlich. Die Informationen in diesem Abschnitt stammen, soweit nicht anders vermerkt, aus [DWoJ].

Obwohl eigentlich nur der das EAP lösende Weg relevant ist, berechnet der Dijkstra-Algorithmus die Wege zu allen möglichen Stationen im Liniennetz, welche zum gegebenen Startpunkt näher liegen als die gewünschte Zielhaltestelle. Um diesen unnötigen Rechenaufwand zu reduzieren, werden die Liniennetz- und Fahrplandaten während eines *Preprocessings* vorsortiert und gewichtet, sodass daraus eine beschleunigte Abfrage resultiert.

Das Preprocessing ist typischerweise Hauptbestandteil der Speed-Up Techniken. Auch wenn durch diese Vorsortierung die für den eigentlichen Algorithmus relevante Knoten- bzw. Kantenzahl teilweise erheblich reduziert werden kann, muss stets die dafür benötigte Rechenzeit und Speicherkapazität berücksichtigt werden.

Abschließend sollen die beiden Speed-Up Techniken ALT und Arcflag exemplarisch skizziert werden.

Ein mögliches Verfahren zur beschleunigten Kürzeste Wege Suche bietet der sogenannte *A^*-Algorithmus*. Dieser ist eine Erweiterung des Djikstra-Algorithmus und versucht, die Suche über eine Potential-Funktion in Richtung Zielknoten zu leiten. Das Potential nimmt Einfluss auf die Prioritäten der einzelnen Elemente in der Vorrangwarteschlange Q und verändert dadurch ihre Reihenfolge, sodass nicht zum Ziel führende Wege seltener abgelaufen werden.
Der A^*-Algorithmus liefert in Verbindung mit sogenannten *Landmarks* den *ALT-Algorithmus*. Dieser verwendet im Preprocessing die Landmarks – eine Teilmenge der Knotenmenge V – zusammen mit der Dreiecksungleichung, um geeignete Potentiale zu berechnen. Die Potentiale bilden Untergrenzen für die Distanz zum Zielknoten d. Um eine signifikante Beschleunigung zu erreichen, ist eine geeignete Wahl der Landmarks (z.B. über die *Avoid-* oder *Max-Cover-Technik*) entscheidend.
Besonders in Kombination mit einer bidirektionalen Suche liefert der ALT-Algorithmus gute Ergebnisse in der Laufzeitoptimierung.

Bei der *Arcflag*-Beschleunigungstechnik werden die Kanten E des Graphen mit Labels (den Arcflags) versehen. Hierfür wird der Graph zunächst möglichst gleichmäßig parti-

tioniert. Anschließend werden für jede Partition alle Kanten des Graphen auf Arcflag = `TRUE` gesetzt, wenn sie Teil eines kürzesten Weges zu einem Knoten in der gerade betrachteten Partition sind.
Im Arcflag-Dijkstra-Algorithmus werden dann nur diejenigen Kanten betrachtet, deren Arcflag für die Partition, in der d enthalten ist, `TRUE` anzeigt.
Zusammen mit einer geeigneten Kantenkontraktion, d.h. einer systematischen Reduktion der Kantenmenge im Graphen, wird die Arcflag-Technik z.B. beim *SHARC-Routing* angewendet.

8. Fazit

Ziel dieser Arbeit war es, ein mathematisches Modell für die Suche nach dem schnellstmöglichen Weg von A nach B aufzustellen und dadurch zu analysieren, wie eine digitale Fahrplanauskunft als Lösung der Passagierroutenplanung entsteht bzw. auf welchen theoretischen Grundlagen sie beruht.
Wie in den vorgehenden Kapiteln gezeigt wurde, bieten Methoden aus der algorithmischen Graphentheorie wirkungsvolle Möglichkeiten für die Analyse und Lösungsfindung von Optimierungsproblemen aus dem ÖPNV, wie der Passagierroutenplanung. Indem die Frage, wie man am schnellsten von A nach B kommt, als EAP formuliert wird, kann der Dijkstra-Algorithmus zur Suche nach einer Fahrplanauskunft, die die frühest mögliche Ankunft garantiert, verwendet werden. Durch die Analyse, welche Optionen es dabei für die Konstruktion des Graphen gibt sowie welche Besonderheiten, Vor- und Nachteile zeitexpandierte bzw. zeitabhängige Graphen aufweisen, entsteht ein aussagekräftiges graphentheoretisches Modell. Dieses wird durch die Einführung von Umsteigewegen an Haltestellen noch realistischer und bietet letztendlich eine Konstruktionsanleitung für die Erstellung eines zeitabhängigen Graphen im realistischen zeitabhängigen Modell auf Basis der Fahrplandaten.
Die Implementierung der Fahrplanabfrage und die Anwendung auf die Regensburger Liniennetz- und Fahrplandaten liefern unter Berücksichtigung der vorausgesetzten Optimierungskriterien gute Ergebnisse. Dennoch legt diese Fahrplanauskunft viel mehr den Grundstein für die Entwicklung eines für die Praxis relevanten Routenplanungstools, welches die ideale Reiseroute auf Basis verschiedener Variablen ermitteln kann, als dass es selbst alle Anforderungen hierfür erfüllt. Beispielsweise wären neben der Suche nach der frühest möglichen Ankunftszeit weitere Optimierungsbedingungen wie die Reduktion von Umstiegen oder die Minimierung der Reisedauer notwendig.
Darüberhinaus ist darauf hinzuweisen, dass die entwickelte Variante des Dijkstra-Algorithmus für Datenmengen in der vorliegenden Größenrelation zwar hinreichend ist, in der Realität jedoch eine Abfrage von sehr großen Fahrplandaten zu lange dauern würde. Die in Kapitel 7 vorgestellten Speed-Up Techniken bieten diesbezüglich interessante Entwicklungsspielräume.

Insgesamt sind die bedeutenden Optimierungspotentiale zu akzentuieren, welche die Digitalisierung im Bezug auf den Öffentlichen Personennahverkehr ermöglicht. Durch die Einführung neuer Technologien wie Fahrgastinformationssysteme, mobiler Fahrkartenerwerb und insbesondere auch Abfragetools zur Routenplanung steigt die Attraktivität des ÖPNV. Zudem wird so eine nachhaltige Mobilitätsinfrastruktur gefördert.
Die gemeinsame Nutzung des ÖPNV zusammen mit digitalen Mobilitätsdienstleistungen ermöglicht dabei jetzt schon die Umsetzung von Mobilitätsbedürfnissen der Zukunft, wie sie in der Studie des MÜNCHNER KREISES *Innovationsfelder der digitalen Welt - Bedürfnisse von übermorgen* analysiert wurden.

In dieser Untersuchung wurden mehr als 7000 Privatpersonen aus sechs Ländern unter anderem zu ihren Erwartungen an die durch die Digitalisierung beeinflusste Mobilität der Zukunft befragt. Die daraus entstandenen Bedürfnismuster umfassen beispielsweise Zeit für andere Beschäftigungen während der Fahrt, eine unabhängige, spontane und flexible Reisemöglichkeit, vielseitiges Serviceangebot sowie ein umweltfreundliches Fortbewegen durch die Stadt. [Neu18]

Es bleibt abzuwarten, inwieweit mathematische Optimierung und Digitalisierung den ÖPNV in Zukunft weiterentwickeln und neu gestalten werden.

A. Anhang

Pläne des RVV

Die nachfolgenden Auszüge aus den Liniennetz-, Haltestellen- und Fahrplänen wurden von der Regensburger Verkehrsverbund GmbH (RVV) zur Verfügung gestellt. Die aktuellen Versionen können auf der Webseite http://www.rvv.de/ eingesehen werden.

Fahrpläne der Linien 11 und 6 zu Abbildung 4.3 und 4.4

Linie 11 Wernerwerkstraße - Arnulfsplatz - HBF/Albertstraße - OTH - Universität - Burgweinting

gültig ab 13. April 2018

Montag - Freitag		5	6	7 - 11	12	13	14 - 17	18	19	20	21 - 22	23	0
VERKEHRSHINWEIS					s 11A								
Wernerwerkstraße		16 36 59	19 38 58	18 38 58	18 38 — 58	18 38 58	18 38 58	18 49	19 50	20 50	20 50	20	
Westbad		17 37 00	20 39 59	19 39 59	19 39 — 59	19 39 59	19 39 59	19 50	20 51	21 51	21 51	21	
Westheim		18 38 01	21 40 00	20 40 00	20 40 — 00	20 40 00	20 40 00	20 51	21 52	22 52	22 52	22	
Weinweg		19 39 02	22 41 01	21 41 01	21 41 — 01	21 41 01	21 41 01	21 52	22 53	23 53	23 53	23	
Boessnerstraße (4)		20 40 03	23 42 02	22 42 02	22 42 — 02	22 42 02	22 42 02	22 53	23 54	24 54	24 54	24	
Dornierstraße		22 42 05	25 44 04	24 44 04	24 44 — 04	24 44 04	24 44 04	24 55	25 56	26 56	26 56	26	
Obermaierstraße		23 43 07	27 46 06	26 46 06	26 46 — 06	26 46 06	26 46 06	26 57	27 58	28 58	28 58	28	
Goethe-Gymnasium		≪	≪	≪	05	≪	≪	≪	≪	≪	≪	≪	
Hans-Sachs-Straße		24 44 08	28 47 07	27 47 07	27 47 — 06	07 27 47	07 27 47	07 27 58	28 59	29 59	29 59	29	
Ostdeutsche Galerie		25 45 09	29 49 09	29 49 09	29 49 — 08	09 29 49	09 29 49	09 29 00	30 00	30 00	30 00	30	
Arnulfsplatz (1)		28 48 13	33 53 13	33 53 13	33 53 — ≪	13 33 53	13 33 53	13 33 04	34 03	33 03	33 03	33	
Keplerstraße		29 49 14	34 54 14	34 54 14	34 54 — ≪	14 34 54	14 34 54	14 34 05	35 04	34 04	34 04	34	
Bismarckplatz		≪	≪	≪	12	≪	≪	≪	≪	≪	≪	≪	
Justizgebäude (1)		≪	≪	≪	14	≪	≪	≪	≪	≪	≪	≪	
Fischmarkt		30 50 15	35 55 15	35 55 15	35 55 — ≪	15 35 55	15 35 55	15 35 06	36 05	35 05	35 05	35	
Thundorferstraße		31 51 17	37 57 17	37 57 17	37 57 — ≪	17 37 57	17 37 57	17 37 08	38 07	37 07	37 07	37	
Dachauplatz (4)		34 54 20	40 00 20	40 00 20	40 00 — ≪	20 40 00	20 40 00	20 40 11	41 10	40 10	40 10	40	
HBF/Albertstraße (I)	an	37 57 23	43 03 23	43 03 23	43 03 — ≪	23 43 03	23 43 03	23 43 14	44 12	42 12	42 12	42	
HBF/Albertstraße (I)	ab	38 58 24	44 04 24	44 04 24	44 04 — ≪	24 44 04	24 44 04	24 44 15	45 15	45 15	45 15	45	
HBF Süd/Arcaden		39 59 25	45 05 25	45 05 25	45 05 — ≪	25 45 05	25 45 05	25 45 16	46 16	46 16	46 16	46	
Evang. Zentralfriedhof		40 00 26	46 06 26	46 06 26	46 06 — ≪	26 46 06	26 46 06	26 46 17	47 17	47 17	47 17	47	
Oberer kath. Friedhof		42 02 28	48 08 28	48 08 28	48 08 — ≪	28 48 08	28 48 08	28 48 19	49 18	48 18	48 18	48	
OTH Regensburg		43 03 29	49 09 29	49 09 29	49 09 — ≪	29 49 09	29 49 09	29 49 20	50 19	49 19	49 19	49	
Universität (D)		45 05 31	51 11 31	51 11 31	51 11 — ≪	31 51 11	31 51 11	31 51 22	52 21	51 21	51 21	51	
Otto-Hahn-Straße		47 07 33	53 13 33	53 13 33	53 13 — ≪	33 53 13	33 53 13	33 53 24	54 23	53 23	53 23	53	
Am BioPark		48 08 35	55 15 35	55 15 35	55 15 — ≪	35 55 15	35 55 15	35 55 26	56 24	54 24	54 24	54	
Franz-Hartl-Straße		49 09 36	56 16 36	56 16 36	56 16 — ≪	36 56 16	36 56 16	36 56 27	57 25	55 25	55 25	55	
Johann-Hösl-Straße (6)		50 10 37	57 17 37	57 17 37	57 17 — ≪	37 57 17	37 57 17	37 57 28	58 26	56 26	56 26	56	
Schwabenstraße		51 11 38	58 18 38	58 18 38	58 18 — ≪	38 58 18	38 58 18	38 58 29	59 27	57 27	57 27	57	
Schöneberger Straße		52 12 39	59 19 39	59 19 39	59 19 — ≪	39 59 19	39 59 19	39 59 30	00 28	58 28	58 28	58	
Benzstraße		53 13 40	00 20 40	00 20 40	00 20 — ≪	40 00 20	40 00 20	40 00 31	01 29	59 29	59 29	59	
Safferlingstraße		≪	≪	≪	21	≪	≪	≪	≪	≪	≪	≪	
Bajuwarenstraße		≪	≪	≪	24	≪	≪	≪	≪	≪	≪	≪	
Burgweinting Kirche		55 15 42	02 22 42	02 22 42	02 22 25 42	02 22 42	02 22 42	02 33	03 31	01 31	01 31	01	
Friedrich-Viehbacher-Allee		56 16 43	03 23 43	03 23 43	03 23 26 43	03 23 43	03 23 43	03 34	04 32	02 32	02 32	02	
Kurzer Weg		57 17 45	05 25 45	05 25 45	05 25 28 45	05 25 45	05 25 45	05 36	06 33	03 33	03 33	03	
Droste-Hülshoff-Weg		58 18 46	06 26 46	06 26 46	06 26 29 46	06 26 46	06 26 46	06 37	07 34	04 34	04 34	04	
Langer Weg		59 19 47	07 27 47	07 27 47	07 27 30 47	07 27 47	07 27 47	07 38	08 35	05 35	05 35	05	
Sophie-Scholl-Straße		00 20 48	08 28 48	08 28 48	08 28 31 48	08 28 48	08 28 48	08 39	09 36	06 36	06 36	06	
Hermann-Höcherl-Straße		02 22 50	10 30 50	10 30 50	10 30 33 50	10 30 50	10 30 50	10 41	11 38	08 38	08 38	08	

s = nur an Schultagen 11A = fährt als Linie 11A

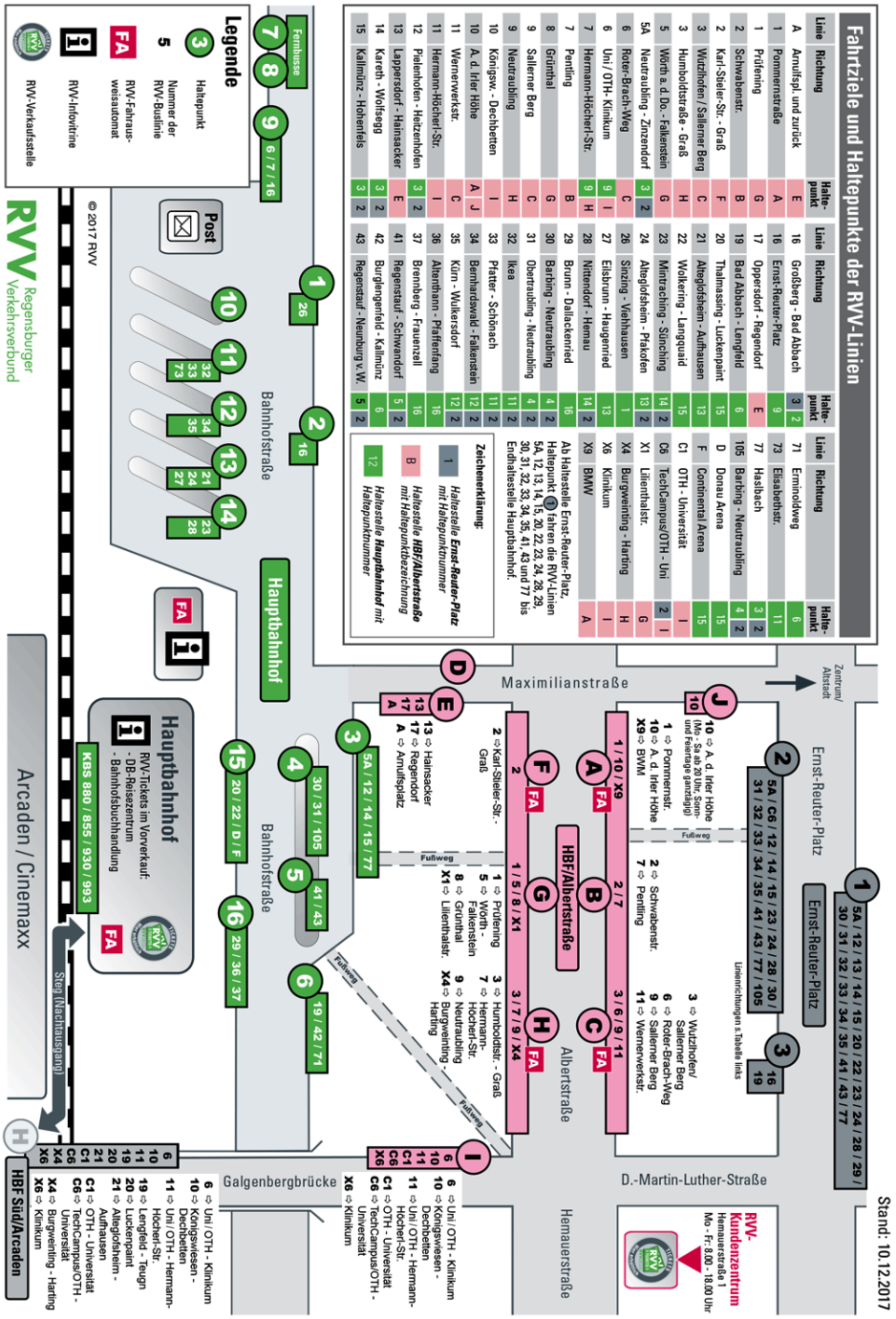
Linie 6

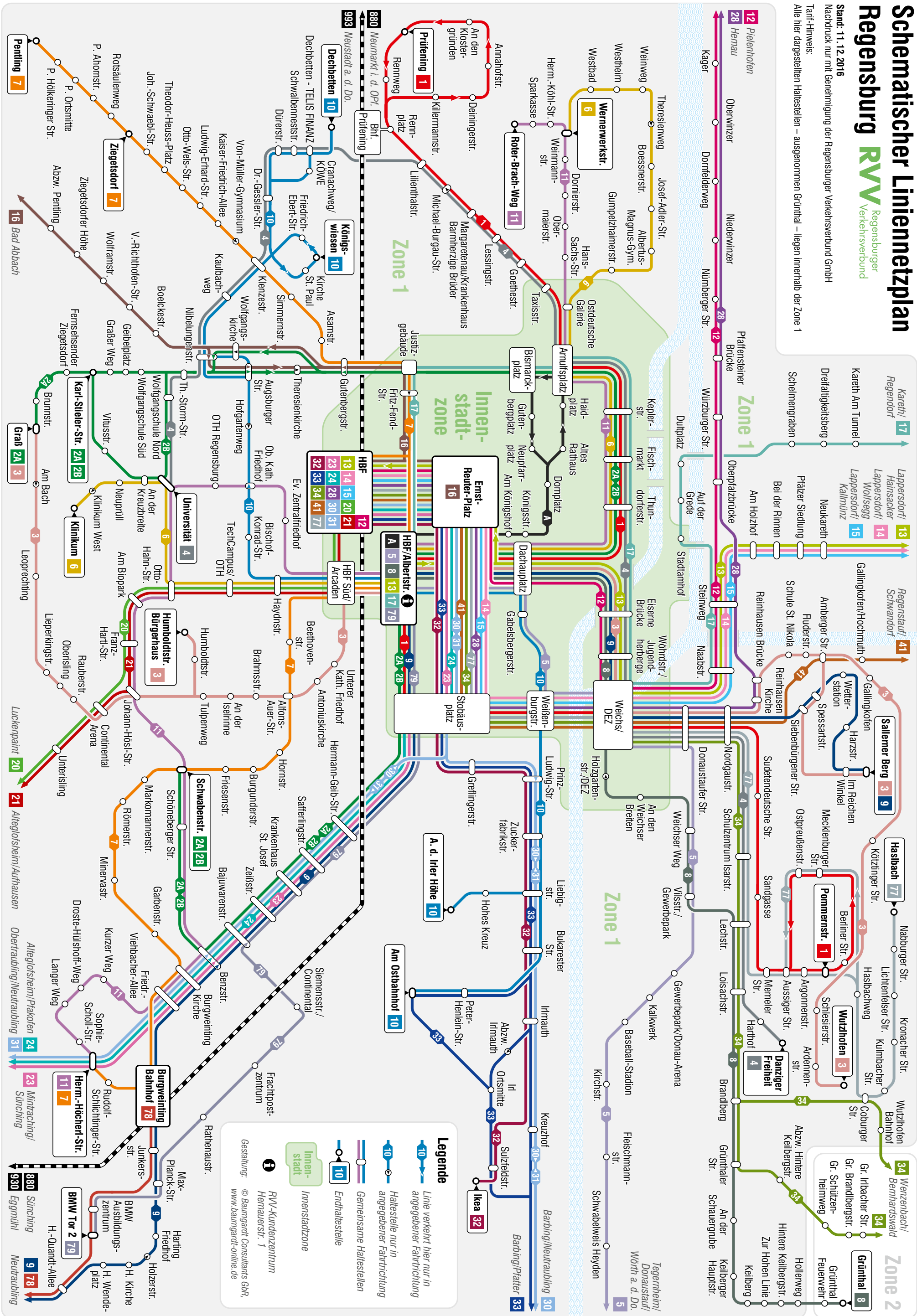
Roter-Brach-Weg - Clermont-Ferrand-Allee - Bismarckplatz - HBF/Albertstraße - Universität - Klinikum

gültig ab 10.12.2017

Montag - Freitag		5	6	7 - 8	9	10 - 14	15	16 - 17	18
Roter-Brach-Weg		18 38 58	08 18 28 38 46 56	06 16 26 36 46 56	06 16 26 36 46	– 06 – 26 – 46	– 06 16 26 36 46 56	06 16 26 36 46 56	16
Sparkasse		18 38 58	08 18 28 38 46 56	06 16 26 36 46 56	06 16 26 36 46	– 06 – 26 – 46	– 06 16 26 36 46 56	06 16 26 36 46 56	16
Hermann-Köhl-Straße		18 38 59	09 19 29 39 47 57	07 17 27 37 47 57	07 17 27 37 47	– 07 – 27 – 47	– 07 17 27 37 47 57	07 17 27 37 47 57	17
Wernerwerkstraße		19 39 00	10 20 30 40 48 58	08 18 28 38 48 58	08 18 28 38 48	– 08 – 28 – 48	– 08 18 28 38 48 58	08 18 28 38 48 58	18
Clermont-Ferrand-Allee		20 40 01	11 21 31 41 49 59	09 19 29 39 49 59	09 19 29 39 49	– 09 – 29 – 49	– 09 19 29 39 49 59	09 19 29 39 49 59	19
Boessnerstraße (2)		21 41 02	12 22 32 42 50 00	10 20 30 40 50 00	10 20 30 40 50	– 10 – 30 – 50	– 10 20 30 40 50 00	10 20 30 40 50 00	20
Josef-Adler-Straße		22 42 03	13 23 33 43 51 01	11 21 31 41 51 01	11 21 31 41 51	– 11 – 31 – 51	– 11 21 31 41 51 01	11 21 31 41 51 01	21
Albertus-Magnus-Gymnasium		23 43 04	14 24 34 44 52 02	12 22 32 42 52 02	12 22 32 42 52	– 12 – 32 – 52	– 12 22 32 42 52 02	12 22 32 42 52 02	22
Gumpelzhaimerstraße		24 44 05	15 25 35 45 53 03	13 23 33 43 53 03	13 23 33 43 53	– 13 – 33 – 53	– 13 23 33 43 53 03	13 23 33 43 53 03	23
Ostdeutsche Galerie		26 46 07	17 27 37 47 55 05	15 25 35 45 55 05	15 25 35 45 55	– 15 – 35 – 55	– 15 25 35 45 55 05	15 25 35 45 55 05	25
Bismarckplatz		29 49 09	19 29 39 49 59 09	19 29 39 49 59 09	19 29 39 49 59	– 19 – 39 – 59	– 19 29 39 49 59 09	19 29 39 49 59 09	29
Justizgebäude (3)		31 51 11	21 31 41 51 01 11	21 31 41 51 01 11	21 31 41 51 01	– 21 – 41 – 01	– 21 31 41 51 01 11	21 31 41 51 01 11	31
Fritz-Fend-Straße		32 52 12	22 32 42 52 02 12	22 32 42 52 02 12	22 32 42 52 02	– 22 – 42 – 02	– 22 32 42 52 02 12	22 32 42 52 02 12	32
Hauptbahnhof (9)		34 54 14	24 34 44 54 04 14	24 34 44 54 04 14	24 34 44 54 04	– 24 – 44 – 04	– 24 34 44 54 04 14	24 34 44 54 04 14	34
HBF/Albertstraße (I)	an	37 57 17	27 37 47 57 07 17	27 37 47 57 07 17	27 37 47 57 07	– 27 – 47 – 07	– 27 37 47 57 07 17	27 37 47 57 07 17	37
HBF/Albertstraße (I)	ab	38 58 18	28 38 48 58 08 18	28 38 48 58 08 18	28 38 48 58 08	18 28 38 48 58 08	18 28 38 48 58 08 18	28 38 48 58 08 18	38
HBF Süd/Arcaden		39 59 19	29 39 49 59 09 19	29 39 49 59 09 19	29 39 49 59 09	19 29 39 49 59 09	19 29 39 49 59 09 19	29 39 49 59 09 19	39
Haydnstraße		40 00 21	31 41 51 01 11 21	31 41 51 01 11 21	31 41 51 01 11	21 31 41 51 01 11	21 31 41 51 01 11 21	31 41 51 01 11 21	41
TechCampus/OTH		41 01 22	32 42 52 02 12 22	32 42 52 02 12 22	32 42 52 02 12	22 32 42 52 02 12	22 32 42 52 02 12 22	32 42 52 02 12 22	42
Otto-Hahn-Straße		42 02 23	33 43 53 03 13 23	33 43 53 03 13 23	33 43 53 03 13	23 33 43 53 03 13	23 33 43 53 03 13 23	33 43 53 03 13 23	43
Universität (A)		44 04 26	36 46 56 06 16 26	36 46 56 06 16 26	36 46 56 06 16	26 36 46 56 06 16	26 36 46 56 06 16 26	36 46 56 06 16 26	46
An der Kreuzbreite		45 05 27	37 47 57 07 17 27	37 47 57 07 17 27	37 47 57 07 17	27 37 47 57 07 17	27 37 47 57 07 17 27	37 47 57 07 17 27	47
Neuprüll		46 06 28	38 48 58 08 18 28	38 48 58 08 18 28	38 48 58 08 18	28 38 48 58 08 18	28 38 48 58 08 18 28	38 48 58 08 18 28	48
Klinikum West		48 08 30	40 50 00 10 20 30	40 50 00 10 20 30	40 50 00 10 20	30 40 50 00 10 20	30 40 50 00 10 20 30	40 50 00 10 20 30	50
Klinikum		50 10 32	42 52 02 12 22 32	42 52 02 12 22 32	42 52 02 12 22	32 42 52 02 12 22	32 42 52 02 12 22 32	42 52 02 12 22 32	52

Liniennetz- und Haltestellenplan




Schematischer Liniennetzplan
Regensburg
RVV Regensburger Verkehrsverbund
Stand: 11.12.2016
Nachdruck nur mit Genehmigung der Regensburger Verkehrsverbund GmbH
Tarif-Hinweis:
Alle hier dargestellten Haltestellen – ausgenommen Grünthal – liegen innerhalb der Zone 1
Legende
Linie verkehrt hier nur in angegebener Fahrtrichtung
Haltestelle nur in angegebener Fahrtrichtung
Gemeinsame Haltestellen
Endhaltestelle
Innenstadtzone
RVV-Kundenzentrum Hemauerstr. 1
Gestaltung: © Baumgardt Consultants GbR, www.baumgardt-online.de

Abbildungsverzeichnis

Abkürzungsverzeichnis

EAP Earliest Arrival Problem

FIFO First in First out

FPersV Fahrpersonalverordnung

OD Origin-Destination

ÖPNV Öffentlicher Personennahverkehr

ÖSPV Öffentlicher Straßenpersonennahverkehr

ÖV Öffentlicher Verkehr

RVB Regensburger Verkehrsbetriebe

RVV Regensburger Verkehrsverbund

SP Shortest Path

VDV Verband Deutscher Verkehrsunternehmen

Literatur

[BGL00] Borndörfer, Ralf ; Grötschel, Martin ; Löbel, Andreas: *Der Schnellste Weg zum Ziel*. Berlin : Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000 (Zitiert auf Seite 19.)

[BJ02] Brodal, Gerth S. ; Jacob, Riko: *Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries*. Aarhus-Dänemark : BRICS, 2002 (Zitiert auf Seite 39.)

[BJ04] Brodal, Gerth S. ; Jacob, Riko: *Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries*. Aarhus-Dänemark, Zürich-Schweiz : ELSEVIER, 2004 (Zitiert auf den Seiten 28, 32, 37 und 42.)

[BNP08] Borndörfer, Ralf ; Neumann, Marika ; Pfetsch, Marc: *Angebotsplanung im öffentlichen Nahverkehr*. Berlin : Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2008 (Zitiert auf den Seiten 9 und 11.)

[Bor00] Borndörfer, Ralf: *Optimierung im Nahverkehr*. Berlin : Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000 (Zitiert auf Seite 10.)

[Bun19] Bundesregierung: *Digitalisierung gestalten: Umsetzungsstrategie der Bundesregierung*. Berlin : Presse- und Informationsamt der Bundesregierung, 2019 https://www.bundesregierung.de/breg-de/themen/digital-made-in-de (Zitiert auf Seite 5.)

[DWoJ] Delling, Daniel ; Wagner, Dorothea: *Time-Dependent Route Planning*. Karlsruhe : Universität Karlsruhe, o.J. (Zitiert auf Seite 78.)

[Edt13] Edtmaier, Bernd: *San Sie der Achter? - Naa, i bin der Meier: Eine Fahrt durch die Geschichte der Regensburger Stadtbuslinien*. Regensburg : Thomas Braun Offsetdruck, 2013 (Zitiert auf Seite 6.)

[FG19] Follmer, Robert ; Gruschwitz, Dana: *Mobilität in Deutschland: Ergebnisbericht*. Bonn, Berlin : infas, DLR, IVT und infas 360 im Auftrag des Bundesministers für Verkehr und digitale Infrastruktur (FE-Nr. 70.904/15), 2019 (Zitiert auf Seite 5.)

[GS16] Gumm, Heinz-Peter ; Sommer, Manfred: *Informatik - Programmierung, Algorithmen und Datenstrukturen*. Marburg : De Gruyter, 2016 (Zitiert auf den Seiten 16 und 18.)

[HJLH97] Hoffmann, Karl-Heinz ; Jäger, Willi ; Lohmann, Thomas ; (Hrsg.), Hermann S.: *Mathematik - Schlüsseltechnologie für die Zukunft*. Berlin Heidelberg : Springer Verlag, 1997 (Zitiert auf Seite 11.)

[HRoJ] Herzog, Melanie ; Riedl, Wolfgang F.: *Kürzeste Wege*. München : TUM, o.J. (Zitiert auf den Seiten 19, 20 und 23.)

[Kli05] Kliewer, Natalia: *Optimierung des Fahrzeugeinsatzes im öffentlichen Personennahverkehr*. Paderborn : Dissertation an der Fakultät für Wirtschaftswissenschaften, Universität Paderborn, 2005 (Zitiert auf den Seiten 9 und 13.)

[MHSWZ07] Müller-Hannemann, Matthias ; Schulz, Frank ; Wagner, Dorothea ; Zaroliagis, Christos: *Timetable Information: Models and Algorithms*. Berlin Heidelberg : Springer Verlag, 2007 (Zitiert auf Seite 78.)

[MMU13] Martinez, Hector ; Mauttone, Antonio ; Urquhart, Maria E.: *Frequency optimization in public transportation systems: Formulation and metaheuristic approach*. Montevideo/Uruguay : European Journal of Operational Research, 2013 (Zitiert auf Seite 9.)

[Neu18] Neuburger, Rahild: *Digitalisierung und Mobilität*. München : Bayerische Eisenbahngesellschaft mbH, 2018 https://beg.bahnland-bayern.de/de/die-beg-20-jahre/digitalisierung (Zitiert auf Seite 81.)

[Ove08] Overhagen, Theo: *Optimierung: Vorlesungsskript*. Siegen : Universität Siegen, 2008 https://www.uni-siegen.de/fb6/analysis/overhagen/vorlesungsbeschreibungen/skripte/linopt1.pdf (Zitiert auf den Seiten 15 und 42.)

[Paj09] Pajor, Thomas: *Multi-Modal Route Planning*. Karlsruhe : Diplomarbeit am Institut für Theoretische Informatik der Universität Karlsruhe, 2009 (Zitiert auf den Seiten 28, 30, 31, 32, 36, 42, 45 und 77.)

[PSWZ04] Pyrga, Evangelia ; Schulz, Frank ; Wagner, Dorothea ; Zaroliagis, Christos: *Towards Realistic Modeling of Time-Table Information through the Time-Dependent Approach*. Patras-Griechenland, Karlsruhe-Deutschland : ELSEVIER, 2004 (Zitiert auf den Seiten 26 und 36.)

[PSWZ07] Pyrga, Evangelia ; Schulz, Frank ; Wagner, Dorothea ; Zaroliagis, Christos: *Efficient Models for Timetable Information in Public Transportation Systems*. New York : ACM Journal of Experimental Algorithmics, 2007 (Zitiert auf den Seiten 36, 45 und 47.)

[Reg18] Regensburg, Stadt: *Planung zur Einführung einer Stadtbahn*. Regensburg : Webseite der Stadt Regensburg, 2018 https://www.regensburg.de/leben/verkehr-u-mobilitaet/bus-und-bahn/planung-zur-einfuehrung-einer-stadtbahn (Zitiert auf Seite 14.)

[Sch01] Schöning, Uwe: *Algorithmik*. Heidelberg/Berlin : Spektrum Verlag, 2001 (Zitiert auf Seite 16.)

[Sch15] Schnieder, Lars: *Betriebsplanung im öffentlichen Personennahverkehr*. Berlin Heidelberg : Springer Viewag, 2015 (Zitiert auf Seite 12.)

[VDV99] VDV, Verband-Deutscher-Verkehrsunternehmen: *ÖPNV-Datenmodell 5.0 - Schnittstellen-Initiative*. Köln : VDV-Schrift 451, 1999 (Zitiert auf Seite 56.)

[VDV13] VDV, Verband-Deutscher-Verkehrsunternehmen: *VDV-Standardschnittstelle Lininennetz/Fahrplan*. Köln : Arbeitsgruppe ÖPNV-Datenmodell, 2013 (Zitiert auf den Seiten 49 und 50.)