

Inhaltsverzeichnis

1 Einführung	2
1.1 Aufgabenstellung	2
1.2 Vorgehen	2
1.3 Ziel der Bachelorarbeit	2
2 Daten	3
2.1 Fahrplan General Transit Feed Specification (GTFS)	3
2.1.1 Datenstruktur	3
2.1.2 Vor- und Nachteile	5
2.2 GTFS Realtime (GTFS-RT)	5
2.2.1 Datenstruktur	5
2.2.2 Vor- und Nachteile	5
2.3 Fahrplan Hafas Rohdaten Format (HRDF)	5
2.3.1 Datenstruktur	5
2.3.2 Vor- und Nachteile	7
2.4 Fahrplan Überblick (timetable overview)	7
2.5 Ist-Daten (actual data)	7
2.6 Dienststellendokumentation (DiDok)	7
2.7 Geschäftsorganisationen (business organisations)	9
2.7.1 Geschäftsorganisationen mit Echtzeit	9
2.8 GA-HTA-Liste	9
2.9 Bahnhofsliste (station list)	10
2.10 Abfahrts-/Ankunftsanzeiger (departure/arrival display)	10
2.11 Fahrtpрогнose (trip forecast)	11
3 Algorithmen	13
3.1 Graph Basierte Algorithmen	13
3.1.1 Dijkstra	13
3.1.2 A*	13
3.1.3 Bellman-Ford	13
3.2 Non Graph Basierte Algorithmen	14
3.2.1 RAPTOR	14
3.2.2 Connection Scan Algorithm	14
3.3 Transfer Pattern	15
4 Programme	17
4.1 Traintickets.to	17
4.2 Open Trip Planner	17
4.3 R5	18
5 Erkenntnisse	19
5.1 Daten	19

5.2 Algorithmen	19
5.3 Programme	19
6 AufgabeBA	22
6.1 AufgabenstellungBA	22
6.2 Lastenheft	23
6.3 Meilensteine	23
6.4 Zeitplan	24

Glossar

Node testtest. 13

1 Einführung

In Zeiten der Digitalisierung des Öffentlichen Verkehrs wird umso mehr Software benötigt die gewisse Bedürfnisse befriedigt. Seit 2016 stellt die Plattform¹ aktuelle Datensätze für den Öffentlichen Verkehr der Schweiz zur Verfügung. Die Daten umfassen Fahrplandaten, Echtzeitdaten, Statistische Daten und noch mehr. Daraus lassen sich Programme realisieren wie z.B. ein Journey Planner der Verbindungsvorschläge inkl. Umsteigvorgänge erstellt.

1.1 Aufgabenstellung

1.2 Vorgehen

1.3 Ziel der Bachelorarbeit

¹<https://opentransportdata.swiss/>

```

stop_id,stop_name,stop_desc,stop_lat,stop_lon,zone_id,stop_url
FUR_CREEK_RES,Furnace Creek Resort (Demo),,36.425288,-117.133162,,,
BEATTY_AIRPORT,Nye County Airport (Demo),,36.868446,-116.784582,,,
BULLFROG,Bullfrog (Demo),,36.88108,-116.81797,,,
STAGECOACH,Stagecoach Hotel & Casino (Demo),,36.915682,-116.751677,,,
NADAV,North Ave / D Ave N (Demo),,36.914893,-116.76821,,,
NANAA,North Ave / N A Ave (Demo),,36.914944,-116.761472,,,
DADAN,Doing Ave / D Ave N (Demo),,36.909489,-116.768242,,,
EMSI,E Main St / S Irving St (Demo),,36.905697,-116.76218,,,
AMV,Amargosa Valley (Demo),,36.641496,-116.40094,,
```

Abbildung 1: Hier sieht man wie so ein CSV-Format im Texteditor aussieht.

2 Daten

In diesem Abschnitt wird gezeigt was für Daten vom Öffentlichen Verkehr auf der Plattform² zur Verfügung stehen. Anschliessend werden die Datenformate vorgestellt und analysiert. Die Fahrplandaten werden in zwei verschiedenen Formaten bereitgestellt GTFS und HRDF.

2.1 Fahrplan General Transit Feed Specification (GTFS)

General Transit Feed Specification (GTFS) ist ein von Google entwickeltes Dateiformat zum Austausch von Öffentlichen Verkehrsdaten sprich Fahrpläne. Ursprünglich wurde es Google Transit Feed Specification (GTFS) genannt (bis 2010), weil es ausschliesslich für Google Maps genutzt wurde. Dies änderte sich aber mit der Zeit sehr stark da viele neue Applikationen herauskamen die diese Daten verwendeten die nicht von Google waren und somit änderte man den Namen zu General Transit Feed Specification (GTFS). [1] GTFS beinhaltet nicht nur Informationen über Fahrpläne sondern auch über Geographische Orte wie Haltestellen. GTFS ist ein statisches Dateiformat und beinhaltet keine Echtzeitdaten deshalb wird es auch GTFS Static genannt. [2]

2.1.1 Datenstruktur

Die GTFS Datei besteht aus nichts anderen als Textfiles, die durch Datenfelder(Werte) und Kommas getrennt sind, dieses Format nennt man auch Comma-Separated Values (CSV).

Die verschiedenen Textfiles decken viele wichtige Informationen ab, die für ein GTFS benötigt werden.

²<https://opentransportdata.swiss/>

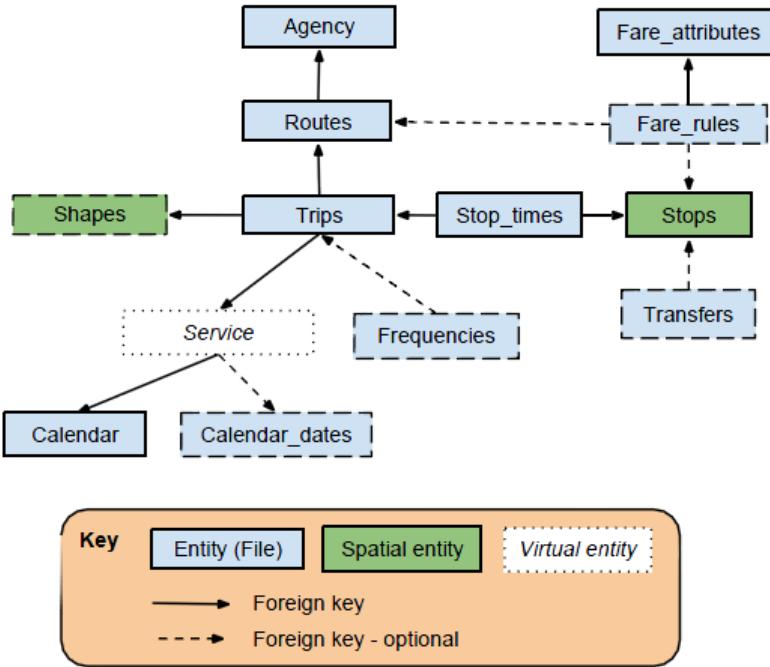


Abbildung 2: Diese Übersicht zeigt die Abhängigkeiten der einzelnen Files.z.B. Braucht das Trips-File die Info der Route-Files(route_id) um zu wissen auf welchem Weg diese Reise stattfindet. [4]

Dateiname	pflicht?	Definition
agency.txt	ja	Geschäftsstellen die Daten zur Verfügung stellen
stops.txt	ja	Haltestellen mit ihrer Position
routes.txt	ja	Verkehrsverbindungen (Linien) mit den Fahrzeugarten
trips.txt	ja	Fahrten
stop_times.txt	ja	Zeiten in der Fahrzeuge Ankommen/Abfahren an Haltestellen
calendar.txt	ja	Fahrplanveränderungen (Jahreszeiten)
calendar_dates	optional	Ausnahmeplan für bestimmtes Datum
fare_attributes.txt	optional	Fahrpreise und die Art der Bezahlung
fare_rules.txt	optional	Fahrpreisregeln verschiedener Zonen
shapes.txt	optional	Beschreibt den Weg eines Fahrzeuges (Darstellung)
frequencies.txt	optional	Fahrpläne ohne fixe stop Zeiten.
transfers.txt	optional	Umsteigepunkte verschiedener Routen (Linien)
feed_info.txt	optional	Zusätzliche Informationen über den Datensatz

[3] Daten die bisher nicht von der Plattform zur Verfügung gestellt werden: fare_attributes.txt, fare_rules.txt, frequencies.txt.

2.1.2 Vor- und Nachteile

Die Daten können einfach von Mensch und Maschine gelesen werden, wegen dem einfachen Aufbau der Textfiles. Zudem stellt Google hierfür eine sehr gute Anleitung zur Verfügung, wie diese Daten verwendet werden und aufgebaut sind.

2.2 GTFS Realtime (GTFS-RT)

GTFS-RT ist eine Erweiterung der GTFS-Static Daten. Wie der Name Realtime schon sagt handelt sich hier um Echtzeitdaten.

2.2.1 Datenstruktur

GTFS-RT stellt folgende Daten zusätzlich in diesem Format zur Verfügung. Die Daten werden geschrieben/gelesen basierend auf sogenannten "Protocol Buffers", die stehen in vielen Programmiersprachen zur Verfügung (C++, C, Go, Java, Python). [5]

- **Trip Updates** -Hier werden Aktuelle Verspätungen, geänderte Routen, Ersatzfahrzeuge oder Ausfälle publiziert.
- **Service Alerts** -Hier werden Informationen über Probleme mit Stationen, Linien, das Ganze Netzwerk etc. übermittelt.
- **Vehicle Positions** -Hier werden Daten geliefert die eine genaue Position des Vehicles mit der dazugehörigen Zeit liefert. [6]

2.2.2 Vor- und Nachteile

Google stellt auch hier eine Gute Übersichtliche Anleitung zur Verwendung von GTFS-RT zur Verfügung.

2.3 Fahrplan Hafas Rohdaten Format (HRDF)

Neben GTFS ist HRDF ein weiteres Dateiformat das die Fahrplandaten zur Verfügung stellt. Dieses Dateiformat kommt von der Firma HaCon. Das Format wird für ihren eigenen Journey Planner (HaCon Fahrplan-Auskunfts-System (HAFAS)) genutzt. Zudem stellt HaCon eine Planungssoftware (Train Planning System TPS) für Infrastrukturbetreiber (wie Eisenbahnverkehrsunternehmen usw.) zur Verfügung. HRDF ist somit ihr eigenes Datenaustauschformat von Fahrplandaten. [7]

2.3.1 Datenstruktur

Ähnlich wie GTFS-Files sind auch HRDF-Files auch Textfiles aber mit dem Unterschied das die Werte im Stil Tab-separated values (TSV) angelegt sind. Die Struktur der Daten ist etwas anders und um einiges komplizierter als bei GTFS. Auch sind die Daten oft schwerer zu lesen von Auge, weil sie manchmal als Bitfeld abgelegt sind. Nebenbei können HRDF Daten auch in GTFS-Daten konvertiert werden. [8]

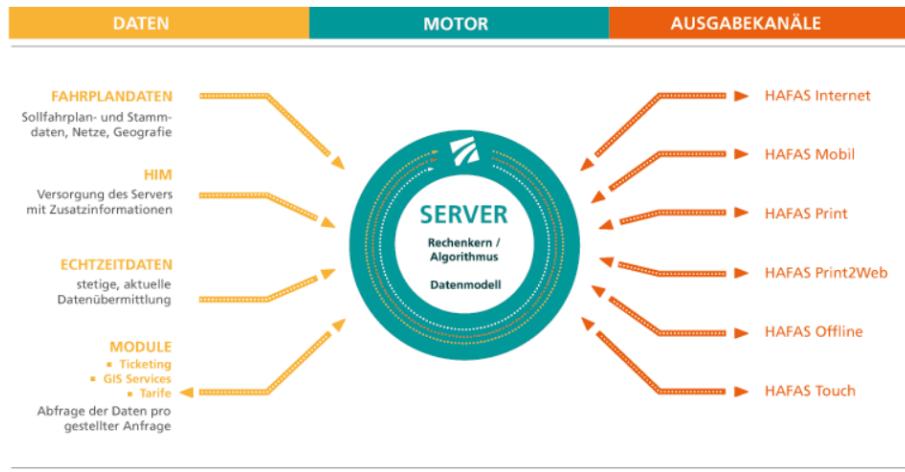


Abbildung 3: Dies ist eine Übersicht wie das HAFAS System aufgebaut ist. Die Fahrplandaten entsprechen hier dem HRDF-Format. [7]

```

00001 K "ASM" L "ASM-bti" V "Aare Seeland mobil (bt)"*
00001 : 000038
00002 K "ASM" L "ASM-ltb" V "Aare Seeland mobil (ltb)"
00002 : 000128
00003 K "ASM" L "ASM-rvo" V "Aare Seeland mobil (rvo)"
00003 : 000056
00004 K "ASM" L "ASM-snb" V "Aare Seeland mobil (snb)"
00004 : 000081
00005 K "LAF" L "LAF" V "Adliswil-Felsenegg"
00005 : 000204
00006 K "AMG" L "AMG" V "Aelplibahn Malans Genossenschaft"
00006 : 003140
00007 K "ARB" L "ARBAG" V "Aletsch Riederalp Bahnen AG"
00007 : 000209
00008 K "ALE" L "ALEX" V "Aletsch-Express Riederalp-Bettmeralp"
00008 : 003002

```

Abbildung 4: Hier sieht man wie so ein TSV-Format im Texteditor aussieht.

Typ	Year	Format	Permalink	Status	Period Start	Period End
Periodic	2016	HRDF	https://opentransportdata.swiss/dataset/timetable-2016-hrdf/permalink	D	13.12.2015	10.12.2016
Periodic	2016	GTFS	https://opentransportdata.swiss/dataset/timetable-2016-gtfs/permalink	D	13.12.2015	10.12.2016
Annual	2016	HRDF	https://opentransportdata.swiss/de/dataset/timetable-2016-hrdf/permalink	D	13.12.2015	10.12.2016
Periodic	2017	HRDF	https://opentransportdata.swiss/dataset/timetable-2017-hrdf/permalink	D	11.12.2016	09.12.2017
Periodic	2017	GTFS	https://opentransportdata.swiss/de/dataset/timetable-2017-gtfs/permalink	D	11.12.2016	09.12.2017
Annual	2017	HRDF	https://opentransportdata.swiss/de/dataset/timetable-2017-hrdf/resource_permalink/fp2017jahresfahrplan.zip	D	11.12.2016	09.12.2017
Periodic	2018	HRDF	https://opentransportdata.swiss/de/dataset/timetable-2018-hrdf/permalink	N	10.12.2017	08.12.2018
Periodic	2018	GTFS	https://opentransportdata.swiss/dataset/timetable-2018-gtfs/permalink	N	10.12.2017	08.12.2018
Annual	2018	HRDF	https://opentransportdata.swiss/de/dataset/timetable-2018-hrdf/permalink	N	10.12.2017	08.12.2018
Periodic	2019	HRDF		N	09.12.2018	15.12.2019
Periodic	2019	GTFS		N	09.12.2018	15.12.2019
Annual	2019	HRDF		N	09.12.2018	15.12.2019

Abbildung 5: Die Fahrplan Überblick Datei wird hier in einer Excel Tabelle zur Verfügung gestellt.

2.3.2 Vor- und Nachteile

Die Plattform warnt vor Verwendung der HRDF-Daten: "Die HRDF-Datei(en) sind relativ komplex. Ohne Not sollte nicht damit gearbeitet werden." [9]

2.4 Fahrplan Überblick (timetable overview)

Die Datei enthält alle Informationen der vorhandenen Fahrplandaten. Zusätzlich wird das Format(HRDF/GTFS) der Status, Gültigkeit und den Permalink der Fahrplandaten zur Verfügung gestellt.

2.5 Ist-Daten (actual data)

Bei den Ist-Daten handelt es sich um eine Ansammlung von Daten, welche die effektive gefahrenen Fahrten des letzten Tages enthalten. Somit sind diese Daten eigentlich in dem Sinne keine wirklichen Ist-Daten. Diese Daten können aber durchaus interessant sein für Statistiken: [10]

- Pünktlichkeit
- Regelmässigkeit.
- Anschlussqualität

Die Daten werden im CSV-Format bereitgestellt.

2.6 Dienststellendokumentation (DiDok)

Bei dieser Dokumentation geht es um die Daten zur Verwaltung der Stammdaten aller Dienststellen(Haltestellen) des öffentlichen Verkehrs der Schweiz. In dem Format werden Daten wie offizieller Name einer Haltestelle und die dazugehörige verantwortliche Geschäftsorganisation. Es werden aber auch die geographischen Koordinaten der Haltestellen mitgeliefert. Die Datei wird im Excel Format zur Verfügung gestellt. Die DiDok Daten werden vom BAV(Bundesamt für Verkehr) veröffentlicht. [11]

Name	Oblig?	Beschreibung	Beispiel
Land_Code	Ja	Der Ländercode	85
Dst-Nr.	Ja	Die eindeutige Nummer der Haltestelle	6013
KZ	Ja	Kontrollziffer. Dies ist ein Wert welcher sich nach einem rechnerischen Schema ergibt. Ursprünglich diente sie dem Zweck zu bemerken ob es allenfalls Zahrendreher in der ID hatte. Heute gibt es noch Systeme welche die Zahl zwingend benötigen.	0
Dst-Bezeichnung-offiziell	Ja	Die offizielle Bezeichnung der Haltestelle	Aarau CT
Dst-Bezeichnung-lang	Nein	Langbezeichnung des Punktes	Annemasse, Amibilly ancien hôpital
Dst-Abk	Nein	Abkürzung der Destination	AE (für Aesch)
GO-Nr	Ja	Geschäftsorganisationsnummer der verantwortlichen Transportunternehmung	11
GO-Abk	Ja	Das Kürzel der Geschäftsorganisation, die für die Haltestelle verantwortlich ist	SBB
Gde-Nr	Ja	Gemeindenummer nach Bundesamt für Statistik (BFS).	4001
BP	Nein	Handelt es sich um einen Betriebspunkt/Haltestelle Ein * entspricht einem True, leer = false	*
VPP	Nein	Verkehrspunkt: Nur Haltestellen mit einem Verkehrspunkt können publiziert werden. Verkehrspunkt für Personenverkehr wird nur nach Antrag beim BAV vergeben. Ein * entspricht einem True, leer = false	*
VPG	Nein	Verkehrspunkt G: Gibt an ob der Punkt für Güterverkehr geöffnet ist. Ein * entspricht einem True, leer = false	*
VD	Nein	Verkaufsdienst: Attribut gibt an ob an diesem Punkt ein Verkaufsdienst (Prisma, Billettautomat mit Abrechnung SBB, Reisebüro, etc.) eröffnet ist. Dies ist in erster Linie ein Attribut welches von den (SBB-)Vertriebssystemen benötigt wird. Achtung: der Rückschluss, dass nur (oder gerade an) Dienststellen mit Verkaufsdienst ein Fahrausweiskauf möglich ist wäre falsch. Ein * entspricht einem True, leer = false	*
KOORDX	Nein	X-Koordinate (Schweiz)	710.377
KOORDY	Nein	Y-Koordinate	260.736
KOORDZ	Nein	Höhe über Mehr	380

Abbildung 6: Hier sieht man ein Beispiel(Haltestelle) für das DiDok-File inklusive der Beschreibung einzelner Attribute. [11]

Feldbezeichnung	Beschreibung	Beispiel
Land_Code	Um unterscheiden zu können, von welcher Nation der Code vergeben wurde, wird der UIC-Ländercode hier verwendet. 85 steht für die Schweiz. Auch wenn die 85 eine Zugehörigkeit zur Schweiz impliziert, kann nicht davon ausgegangen werden, dass diese Geschäftsorganisation in der Schweiz tätig ist. Der Land_Code wird meistens in Kombination mit der GO-Nr verwendet und durch einen Doppelpunkt getrennt, beispielsweise '85:11' für die SBB.	85
GO-Nr	Die Nummer findet sich in anderen Datensätzen wieder.	53
GO-Abk	Die Abkürzung der Geschäftsorganisation	TPF
GO-Bezeichnung	Die Name der entsprechenden Geschäftsorganisation	Transports publics fribourgeois
Typ	Die Geschäftsorganisationen werden noch typisiert. Hier handelt es sich um die ID des Typs	10
GO-Typ-Bezeichnung	Der Name des GO-Typs. Folgende Typen sind aktuell verfügbar, aber nicht näher spezifiziert: <ul style="list-style-type: none">• 10: Bahn• 11: UIC-Bahn• 12: Bahn, ohne Verkehrsabrechnung• 20: Schiff• 22: Schiff, ohne Verkehrsabrechnung• 30: Strasse• 32: Strasse, ohne Verkehrsabrechnung• 45: Luft• 50: Freizeitangebot• 51: Tarifverbund• 52: Messe• 60: Reisebüroorganisation• 80: Tochtergesellschaft (Bahn)• 95: interne Abrechnungszwecke• 99: unbekannt	Bahn

Abbildung 7: Beispiel einer Geschäftsorganisation mit Beschreibung der Attribute. [12]

2.7 Geschäftsorganisationen (business organisations)

Die Daten werden im Excel-Format zur Verfügung gestellt. Hierbei findet man alle in der Schweiz operierenden Geschäftsorganisationen.

2.7.1 Geschäftsorganisationen mit Echtzeit

Hierbei werden alle Geschäftsorganisationen erwähnt, die Echtzeitdaten liefern. Die Daten werden hier auch im Excel-Format zur Verfügung gestellt.

2.8 GA-HTA-Liste

In diesem Datensatz werden die Anzahl der General- (GA) und Halbtax-Abonnemente (HTA) pro Postleitzahl bereitgestellt mit dem dazugehörigen Erfassungsjahr. Diese Daten werden benötigt um Verkehrsmodelle zu verbessern, auf kantonaler und lokaler Basis.

Attribut	Beschreibung
Jahr_An_Anno	Jahr des Stichdatums des Datenauszugs.
PLZ_NPA	Vierstellige Postleitzahl gemäss Ortschaftenverzeichnis
GA_AG	Anzahl Generalabonnemente im Umlauf per Stichdatum.
GA_AG_flag	Mittelwert Generalabonnemente, die weniger als 20Abos.
HTA_ADT_meta-prezzo	Anzahl Halbtaxabonnemente im Umlauf per Stichdatum.
HTA_ADT_meta-prezzo_flag	Mittelwert Halbtaxabonnemente, die weniger als 20Abos

Feld	Beschreibung	Beispiel
_id	ID der Zeile	1
Company-GO-ID	Die Geschäftsorganisation	11
Company name	Der Name der Geschäftsorganisation	Schweizerische Bundesbahnen SBB
Start Interrupt	Im Moment nicht verwendet	-
End Interrupt	Im Moment nicht verwendet	-
Comment	Im Moment nicht verwendet	-

Abbildung 8: Beschreibung der Daten (Geschäftsorganisationen mit Echtzeit) anhand von einem Beispiel. [13]

Jahr_An_Arno	PLZ_NPA	GA_AG	GA_AG_flag	HTA_ADT_meta-prezzo	HTA_ADT_meta-prezzo_flag
2012	1000	72,00		976,00	
2012	1003	744,00		3195,00	
2012	1004	1919,00		8167,00	
2012	1005	860,00		4021,00	
2012	1006	1279,00		5366,00	

Abbildung 9: Ausschnitt der Daten in einer GA-HTA-Liste [14]

2.9 Bahnhofsliste (station list)

Die Bahnhofsliste besteht aus zwei Dateien:

- **Station list** - Hier sind alle Haltestellen der Schweiz enthalten, mit ID und Name.
- **Station geographic** -Hier werden die Koordinaten für die Haltestellen zur Verfügung gestellt.

Die Dateien werden im CSV-Format bereitgestellt. Die Station List entspricht die im HRDF-Format die "BAHNHOF" Datei. [15]

2.10 Abfahrts-/Ankunftsanzeiger (departure/arrival display)

Der Abfahrts- und Ankunftsanzeiger wird als Open Service API (application programming interface) zur Verfügung gestellt. Um die API zu benutzen muss man eine Haltestelle aus der Bahnhofsliste (station list) oder DiDok auswählen. Über diese API können mittels XML (Extensible Markup Language) Anfragen gestellt werden. Zusätzlich wird aber ein API-Key benötigt um Zugriff auf die API zu bekommen. Man verwendet sie für Haltestellenanzeiger. [?]

```

<?xml version="1.0" encoding="UTF-8"?>
<Trias version="1.1" xmlns="http://www.vdv.de/trias" xmlns:siri="http://www.siri.c
  <ServiceRequest>
    <siri:RequestTimestamp>2016-06-27T13:34:00</siri:RequestTimestamp>
    <siri:RequestorRef>EPSa</siri:RequestorRef>
    <RequestPayload>
      <StopEventRequest>
        <Location>
          <LocationRef>
            <StopPointRef>8502113</StopPointRef>
          </LocationRef>
          <DepArrTime>2017-01-03T10:22:00</DepArrTime>
        </Location>
        <Params>
          <NumberOfResults>30</NumberOfResults>
          <StopEventType>departure</StopEventType>
          <IncludePreviousCalls>true</IncludePreviousCalls>
          <IncludeOnwardCalls>true</IncludeOnwardCalls>
          <IncludeRealtimeData>true</IncludeRealtimeData>
        </Params>
      </StopEventRequest>
    </RequestPayload>
  </ServiceRequest>
</Trias>

```

Abbildung 10: Beispielcode einer Abfahrts-/Ankunft-Anfrage. Die Parameter, die übergeben werden sind in schwarzer Schrift dargestellt. [?]

Parameter	Beschreibung
8502113	(StopPointRef) Haltestellencode Didok oder Bahnhofsliste
2017-01-03T10:22:00	(DepArrTime) Ankunfts oder Abfahrtszeit
30	(NumberOfResults) Anzahl Resultate (maximal 40)
departure	(StopEventType) entweder departure(Abfahrt) oder arrival(Ankunft)
true	(IncludePreviousCalls) Haltestellen vor gesuchter Haltestelle mitliefern?
true	(IncludeOnwardCalls) Haltestellen nach gesuchter Haltestelle mitliefern?
true	(IncludeRealtimeData) Sollen Echtzeitdaten mitgeliefert werden?

2.11 Fahrtprognose (trip forecast)

Wie auch beim Abfahrts- und Ankunftsanzeiger wird auch hier die Fahrtprognose als Open Service API zur Verfügung gestellt. Ebenfalls wird auch hier über XML Anfragen gestellt und es wird auch ein API-Key benötigt. Sehr wichtig ist die JourneyRef(Fahrt-ID) diese muss bekannt sein und kann nicht über ein Sollfahrplan abgeleitet werden, stattdessen wird sie über andere API-Request(TripRequest oder Ankunfts- und Abfahrtsanzeiger) abgeleitet. [16]

```

<?xml version="1.0" encoding="utf-16"?>
<Trias version="1.1" xmlns="http://www.vdv.de/trias" xmlns:siri="http://www.siri.
    <ServiceRequest>
        <siri:RequestTimestamp>2016-07-05T18:00:00</siri:RequestTimestamp>
        <siri:RequestorRef>JS</siri:RequestorRef>
        <RequestPayload>
            <TripInfoRequest>
                <JourneyRef>odp:01012::H:j16:30441</JourneyRef>
                <OperatingDayRef>2016-04-02T</OperatingDayRef>
                <Params>
                    <UseTimetabledDataOnly>false</UseTimetabledDataOnly>
                    <IncludeCalls>true</IncludeCalls>
                    <IncludePosition>false</IncludePosition>
                    <IncludeService>true</IncludeService>
                </Params>
            </TripInfoRequest>
        </RequestPayload>
    </ServiceRequest>
</Trias>

```

Abbildung 11: Beispielcode einer Fahrtroute-Anfrage. Die Parameter, die übergeben werden sind in schwarzer Schrift dargestellt. [16]

Parameter	Beschreibung
odp:01012::H:j16:30441	(JourneyRef) wird über andere Quellen bezogen/hergestellt
2016-04-02T	(OperatingDayRef) Betriebstag
false	(UseTimetabledDataOnly) Infos zu Verkehrstagen ausgegeben werden?
true	(IncludeCalls) Sollen Halte der Fahrt ausgegeben werden?
false	(IncludePosition) Aktuelle Position des Fahrzeugs mitliefern?
true	(IncludeService) Verkehrsmittelinformationen ausgeben?

3 Algorithmen

Die für einen Journey Planner zu lösende Aufgabe ist das Shortest-Path Problem. Um dieses Problem zu lösen, gibt es mehrere Herangehensweisen.

3.1 Graph Basierte Algorithmen

Graph-Basierte Algorithmen bilden einen Graphen aus Punkten. Die Verbindungen zwischen den Punkten werden mit einer Gewichtung versehen. Der Algorithmus iteriert nun über den Graphen und findet die Verbindung zwischen dem Start- und Endpunkt mit dem niedrigsten kombinierten Gewicht.

3.1.1 Dijkstra

Der Dijkstra Algorithmus ist ein Single-Source-Shortest-Path Algorithmus. Er berechnet den kürzesten Weg von einem Startnode zu jedem anderen Node im Graphen. Er iteriert über alle möglichen Verbindungen und speichert das Gewicht eines Zielnodes. Wenn eine weitere Verbindung zum gleichen Node gefunden wird, so wird die Verbindung mit dem geringsten Gewicht behalten. Es wird immer die Node mit dem geringsten Gewichtswert zum Startnode als nächstes berechnet. Wenn auch der Zielnode bekannt ist, wird der Algorithmus gleichzeitig vom Start und Zielnode aus gestartet, so dass sie sich in der Mitte treffen.

Der Dijkstra Algorithmus ist der Basisalgorithmus mit dem die anderen Algorithmen verglichen werden. Er ist konzeptionell einfach, jedoch in keiner Weise optimiert. Die Laufzeit erhöht sich exponentiell mit der Anzahl der Nodes und Verbindungen, weshalb er für grosse Netzwerke ungeeignet ist. [17] [18]

3.1.2 A*

Der A* Algorithmus ist eine Erweiterung des Dijkstar Algorithmus. Er sucht nicht wie der Dijkstra Algorithmus linear in alle Richtungen, sondern gezielt in Richtung des Zielnodes. Dazu wird jedem Punkt im Graph eine Entfernung zum Zielnodes zugewiesen. Nun wird dieser Wert mit dem Gewichtswert des Abstandes zum Startnode kombiniert. Der als nächstes zu berechnende Punkt wird nun aufgrund dieses Wertes entschieden. Dadurch werden Verbindungen welche in Richtung des Ziels führen präferiert und es werden viele überflüssige Rechenschritte eingespart.

Obwohl der A* Algorithmus mehr Operationen pro Node durchführen muss, hat er dennoch eine höhere Performance als der Dijkstra Algorithmus, da viel weniger Nodes untersucht werden müssen. Der Nachteil vom A* Algorithmus ist, dass er mehr Memory benötigt. [17]

3.1.3 Bellman-Ford

Der Bellman-Ford Algorithmus verwendet das gleiche Grundkonzept wie der Dijkstra Algorithmus. Er achtet jedoch nicht auf den geringsten Gewichtungsfaktor zum Start-

node, sondern geht der Reihe nach alle Nodes durch. Wenn nun für einen Node nicht alle Zwischennodes zum Starnode schon berechnet wurden, so scheint dieser Node unerreichbar. Um diese Problematik zu lösen wird dieser Prozess $x-1$ mal wiederholt, wobei x die Anzahl der Nodes ist. [18] [19]

Der Bellman-Ford Algorithmus ist langsamer als der Dijkstra Algorithmus, da er mehrere Durchläufe über alle Nodes benötigt. Dies bietet ihm jedoch den Vorteil, dass auch eine negative Gewichtung einer Verbindung möglich ist.

3.2 Non Graph Basierte Algorithmen

In diesem Abschnitt werden Algorithmen erläutert, welche nicht auf das Grundkonzept des Dijkstra Algorithmus aufbauen.

3.2.1 RAPTOR

Der Round-Based Public Transit Routing Algorithmus, auch RAPTOR Algorithmus genannt, ist ein Rundenbasierter, auf öffentliche Verkehrsnetzwerke zugeschnittener Algorithmus, welcher auf die vorgegebenen Zuglinien achtet.

Der RAPTOR Algorithmus durchläuft mehrere Runden um von der Startstation zur Zielstation zu finden. In der ersten Runde werden alle Zuglinien gescannt, welche durch die Startstation verlaufen. Wenn eine andere Zuglinie die gescannte Zuglinie kreuzt, so wird die andere Zuglinie sowie die Kreuzungsstation markiert. In der nächsten Runde werden nun alle markierten Zuglinien gescannt. Dieser Prozess wird so lange weitergeführt, bis sich die Zielstation in einer gescannten Zuglinie befindet. Nun kann anhand der Kreuzungsstationen die Route erstellt werden.

Der RAPTOR Algorithmus besitzt eine höhere Performanz als alle Graph Basierten Algorithmen. [20]

3.2.2 Connection Scan Algorithm

Der Connection Scan Algorithm, kurz CSA, ist schon vom Grundkonzept her auf Zeitplanbasierte Netzwerke zugeschnitten. Er arbeitet mit Stations, Connections, Trips und Footpaths. Eine Connection ist die Verbindung zwischen zwei Stationen während ein Trip den gesamten Weg eines Zuges darstellt.

In einem Ersten Schritt werden die Daten in die benötigte Timetable-Form gebracht und alle Connections nach der Abfahrtszeit sortiert. Diese Schritte werden preprocessed. Danach wird über alle Connections iteriert. Eine Connection wird als erreichbar markiert, wenn sie an bereits als erreichbar markierte Connections anschliesst oder mit der Startstation verbunden ist. Dies wird dann solange durchgeführt, bis die Zielstation erreicht ist. Anschliessend wird von der Zielstation aus der verfolgte Weg zusammengesetzt, so dass ein Journey von der Startstation zur Zielstation entsteht.

Für grosse Netzwerke kann der CSA mit einem Quadtree-Preprocessing Schritt erweitert werden. Dabei wird das Netzwerk in immer kleiner werdende Quadrate unterteilt. Für Quadrate welche nicht die Start- oder Zielstation enthalten, müssen nun nur noch

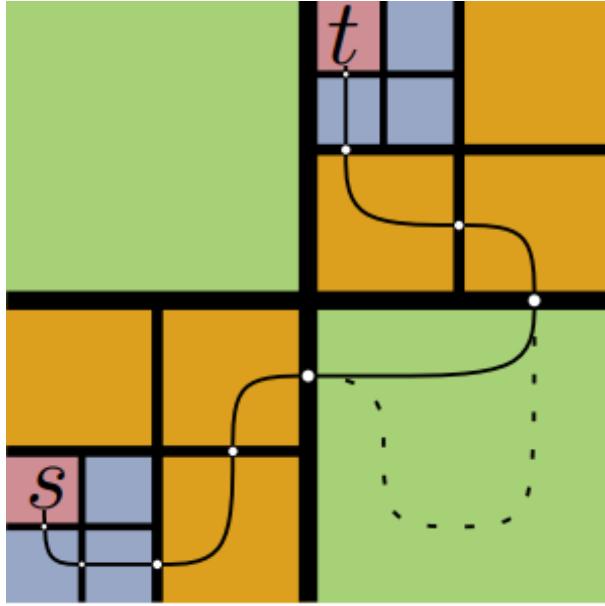


Abbildung 12: Veranschaulichung der Funktion des QuadTree-Preprocessings [21]

die Connections beachtet werden, welche über die Ränder des Quadrates hinaus gehen. Somit können für Fernverbindungen die lokalen Connections ignoriert werden.

Im Gegensatz zum Dijkstra müssen beim CSA die Connections nicht nach dem nächsten Schritt durchsucht werden, da die Abfolge durch die Preprocessing-Schritte schon definiert wurde. Dies führt zu einer Performanceverbesserung im mehrstelligen Bereich. Ein Nachteil ist jedoch, dass Preprocessing-Schritte von Nöten sind und so nicht in echtzeit auf Verspätungen reagiert werden kann. Da diese Schritte jedoch nur wenige Sekunden benötigen ist dies kein schwerwiegender Nachteil. Mit der Erweiterung des Quadtree-Preprocessings können auch Anfragen in landesweite Netzwerke in wenigen Millisekunden berechnet werden.

Im Vergleich zum RAPTOR Algorithmus bietet der CSA eine höhere Performance. [21]

3.3 Transfer Pattern

Transfer Pattern ist eine Algorithmusstrategie welche auf Preprocessing setzt. Alle möglichen Verbindungen werden mithilfe eines Algorithmuses berechnet und in einem Datensatz gespeichert. Die eigentliche Anfrage beschränkt sich dann auf eine Query-Abfrage auf diesen Datensatz.

Für das Preprocessing können alle zuvor genannten Algorithmen verwendet werden. Die Dijkstra basierten Algorithmen haben jedoch den Nachteil, dass Query anfragen für landesweite Netzwerke immer noch mehrere Sekunden benötigen, weshalb der RAPTOR Algorithmus oder der CSA zu bevorzugen sind. [22]

Um das Preprocessing zu beschleunigen können mehrere Erweiterungen hinzugefügt werden. Eine davon ist die Verwendung von Hubs. Dabei werden die grössten Stationen

als Hubs markiert. Zwischen diesen Hubs werden alle Transfer Pattern berechnet. Bei Stationen welche keine Hubs sind werden nur die Verbindungen bis zum nächsten Hub berechnet. Sollte nach drei Zugwechseln kein Hub erreicht sein so wird die Verbindung verworfen. Dies kann zwar zu Fehlern führen, ist aber vernachlässigbar, da die Fehlerrate laut Experimenten bei drei Promille liegt. [23]

Der Preprocessing kann je nach Grösse und Struktur des Netzwerks eine lange Zeit in Anspruch nehmen. Selbst mit allen Erweiterungen braucht das Preprocessing für das CH-Netzwerk vier Stunden. Dafür benötigen die Query-Anfragen nur wenige Millisekunden. Der Nachteil ist jedoch, dass mit Transfer Pattern nicht auf Zugverspätungen und Fahrplanänderungen reagiert werden kann.

4 Programme

Es gibt viele Journey-Planning-Softwares, doch nur wenige davon besitzen eine OpenSource-Lizenierung. Wir werden uns im Rahmen dieses Projektes auf die OpenSource Anwendungen beschränken.

4.1 Traintickets.to

Traintickets.to ist ein von Linus Norton [24] entwickelter Journey Planner für das englische Zugnetzwerk. Linus Norton ist CTO der Firma Assertis [25], welche das Projekt betreibt. Linus Norton erklärt den Programmierprozess in einem Blog [26]

Traintickets.to basiert auf einem modifizierten CSA kombiniert mit TransferPattern. In einem preprocessing Schritt werden aus einem modifizierten GTFS-datasheet die TransferPattern mit dem CSA generiert. [27] Diese werden dann von Hauptprogramm [28] gefiltert und über eine API [29] zur Verfügung gestellt. Assertis ist von der ATOC (Association of Train Operation Companies) lizenziert, so dass sie Ticketpreise anzeigen können sowie Tickets direkt verkaufen können. Es basiert auf PHP für den Algorithmus und Scala für die Transfer Pattern Generierung.

Traintickets.to besitzt zwei verschiedene Lizenzierungen. Die TransferPattern Generierung sowie das Filterprogramm stehen unter einer GNU GPLv3 Lizenz. Die WebAPI ist technisch gesehen ein eigenes Projekt und steht unter exclusive copyright.

Vorteil der Traintickets.to Anwendung ist, dass dieser schon Landesweit implementiert und getestet ist und deshalb in der Schweiz ohne Skalierung anwendbar ist. [30]

Nachteile der Traintickets.to Anwendung sind, dass modifizierte GTFS-Daten verwendet werden und diese so nicht direkt von der OpenData Plattform bezogen werden können, dass der Code schlecht strukturiert ist und dass die Anwendung Aufgrund des preprocessing-Aufbaus nicht auf Verspätungen und Fahrplanänderungen reagieren kann.

4.2 Open Trip Planner

Der OpenTripPlanner, kurz OTP, ist eine auf der Maven-Repository [31] aufbauende Multimodale trip planning Software welche anfangs für Städte ausgelegt war, nun aber auch in ersten landesweiten Netzwerken Anwendung findet. Er wurde von einem OpenSource-Kollektiv aus mehr als 100 Personen in acht Jahren entwickelt. [32]

Der OTP basiert auf dem A*-Algorithmus und verwendet GTFS-Daten und OpenStreetMap Daten in Form einer pbf-Datei. In einem preprocessing Schritt wird der Graph für den Algorithmus erstellt. Dieser kann in einer Datei gespeichert werden oder direkt im RAM des Servers gelagert werden. Selbiges wird für die OpenStreetMap-Daten gemacht. Während dem Betrieb kann der Graph angepasst werden, so dass das Programm auf verspätete Züge reagieren kann. [33]

OTP steht unter einer GNU Lesser General Public License.

Vorteile des OTP sind, dass er in Echtzeit auf Fahrplanänderungen und Verspätungen reagieren kann, dass OTP schon seit mehreren Jahren in verschiedenen Städten im-

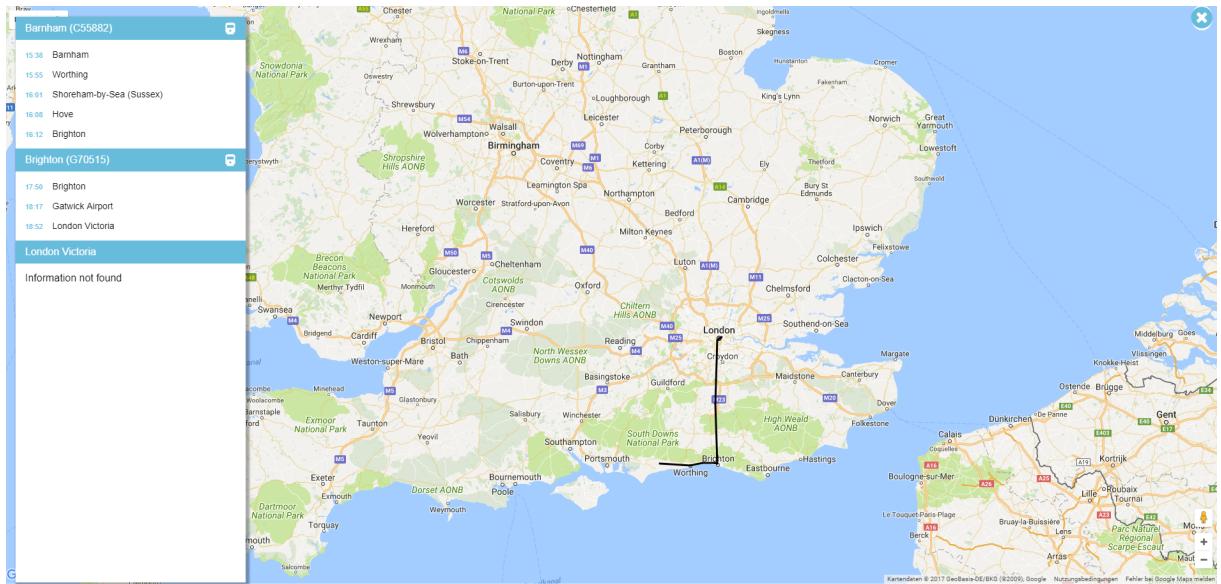


Abbildung 13: Traintickets.To Webseite [30]

plementiert und getestet ist und dass unsere Arbeit vom OpenSource-Kollektiv in den Hauptcode aufgenommen werden könnte.

Nachteile des OTP sind, dass landesweite Implementationen erst in der Beta-Phase sind und somit noch nicht ausreichend getestet sind und dass das Programm unheimlich gross und verzweigt ist, so dass eine lange Einarbeitungszeit von Nötten ist.

4.3 R5

Rapid Realistic Routing on Real-world and Reimagined networks oder kurz R5 ist ein multimodales Trip-Planning-Tool. Er wurde von der Firma Conveyal [34] entwickelt und basiert auf dem OTP.

R5 hat die Grundstruktur des OTP übernommen, jedoch verwendet R5 den RAPTOR-Algorithmus anstelle des A*-Algorithmus. Dadurch wurden einige Programmstrukturen verändert, da der RAPTOR-Algorithmus nicht auf den Dijkstra-Algorithmus aufbaut und auch keinen Graphen verwendet. [35]

R5 steht unter der MIT License.

Vorteile des R5 sind, dass er in Echtzeit auf Fahrplanänderung und Verspätungen reagieren kann und dass der RAPTOR-Algorithmus eine bessere Performance als die Dijkstra-basierten Algorithmen bietet.

Nachteile des R5 sind, dass er unter einer MIT License steht und dass der R5 von einer Firma entwickelt wurde, welche unsere Arbeit nicht übernehmen werden.

5 Erkenntnisse

5.1 Daten

Für unseren Journey Planer benötigen wir Fahrplaninformationen womit wir unseren Algorithmus füttern. Die Fahrplandaten stehen in zwei verschiedenen Dateiformaten zur Verfügung.

Wir haben uns für das GTFS-Format entschieden. Laut der Seite³, die Konvertierung HAFAS-Rohdaten nach GTFS Dokumentation zur Verfügung stellt, erwähnt: "GTFS ist besser strukturiert, einfacher zu benutzen, einfacher zu erweitern und besser dokumentiert als HAFAS-Rohdaten". Zudem wird gesagt das OpenSource Routenfinder wie OTP unterstützen GTFS als einziges Dateiformat für Fahrplandaten. [8]

Zusätzlich laut der Plattform⁴ die Daten zur Verfügung stellt warnt zudem noch: "Die HRDF-Datei(en) sind relativ komplex. Ohne Not sollte nicht damit gearbeitet werden." [9]

Da wir uns für das GTFS-Format entschieden haben ist es auch logisch die dafür zugehörige Erweiterung für Echtzeit zu verwenden (GTFS-RT).

5.2 Algorithmen

Alle von uns genannten Algorithmen wurden schon für Journey Planner verwendet. In den Papers des RAPTORS und des CSA wurden jeweils experimentelle Vergleiche mit Dijkstra basierten Algorithmen gezogen. In allen Testfällen hatten die Dijkstra basierten Algorithmen mindestens doppelt so lange Query-Zeiten. [21] [20]

Der RAPTOR-Algorithmus und der CSA sind sich in Vorgehen und Effizienz sehr ähnlich. Es gibt jedoch mehr Daten aus Experimenten für den CSA. Hannah Bast hat schon einen auf CSA und Transfer Pattern basierten Test mit dem Schweizer Schienennetz durchgeführt. Ihre Implementation hatte Query-Zeiten von wenigen Millisekunden. Dadurch können wir uns sicher sein, dass der CSA mit den Schweizer Daten kompatibel ist und genügend Skalierbarkeit mitbringt. [23]

Der CSA ist für unsere Arbeit am besten geeignet. Sollte jedoch durch andere Faktoren der RAPTOR-Algorithmus verlangt werden ist dieser auch akzeptabel.

5.3 Programme

Die drei Programme Traintickets.to, OTP und R5 wurden in Betracht gezogen.

R5 basiert zwar auf dem modernen RAPTOR Algorithmus, ist jedoch nicht für unser Projekt geeignet. Der Programmcode steht unter einer MIT-Lizenz. R5 wird von der Firma Conveyal [34] entwickelt. Wenn wir mit unserem Projekt auf das R5 Programm

³<http://gtfs.geops.ch/doc/>

⁴<https://opentransportdata.swiss/>

aufbauen, so besteht keine Chance dass unser Code in das Originalprogramm integriert wird.

Die beiden Programme Traintickets.to und OpenTripPlanner wurden genauer betrachtet. Die Programmierer beider Programme wurden von uns angeschrieben. Wir posteten eine Anfrage in die OTP Developer Mailing. Darin fragten wir an, ob sie zurzeit an einer Implementation des CSA in ihrem Programm arbeiten und wie sie zu der Idee unseres Projektes stehen. Nach nur kurzer Zeit bekamen wir eine Antwort. Eine Integration des CSA ist zurzeit nicht geplant. Projekte von neuen Personen sind gerne gesehen, jedoch wird der Code nur in das Hauptprogramm übernommen, wenn das OpenSource-Gremium den Code als gut erachtet. Dazu muss der Code den Programmrichtlinien entsprechen und der Strategie des OTP entsprechen. Die Strategie des OTP liegt darin, dass sie nicht ausschliesslich auf die schnellste Lösung setzen. Wichtiger ist, dass das Programm in Echtzeit auf Verspätungen und Fahrplanänderungen reagieren kann. Wir haben auch Linus Norton von Traintickets.to angeschrieben, haben jedoch keine Antwort erhalten.

Um die Kompatibilität der Programme mit dem Schweizer System überprüfen versuchten wir die Programme mit den Schweizer GTFS-Daten zu betreiben. Traintickets.to lies sich nicht ohne weiteres auf das Schweizer System anwenden. Die GTFS-Daten konnten zwar in die für den CSA benötigte Form umgewandelt werden, jedoch funktionierte der darauffolgende CSA nicht. Wir konnten das Problem im Rahmen des Fachmodules leider nicht lokalisieren. Jedoch ist es von der Skalierbarkeit her möglich das Programm auf einer landesweiten Basis zu implementieren, da es zur Zeit auf dem Zugnetzwerk des United Kingdoms aufbaut. Auch die Implementation des OTP führte zu Problemen. Die Ausführung funktioniert mit GTFS-Daten, welche älter als der 27.09.2017 sind. Mit neueren GTFS-Files schlägt sie jedoch fehl. Wir konnten das Problem finden. Die SBB hat einige der Routen mit dem Routetype 1700 Miscellaneous bezeichnet. Das OTP Programm beachtete diesen Route-Type nicht, obwohl er GTFS konform ist. Wir meldeten das Problem an die Development-Mailing-List, wo man uns mitteilte, dass dieser Typ nicht unterstützt wird, da er nicht eindeutig einem Route-Type zugeordnet werden kann. Wenn wir OTP verwenden müssen wir dies für das Schweizer System selber implementieren. Die älteren GTFS-Daten funktionieren, da die SBB vor dem 27.09.2017 mit den Alten Route-Types arbeiteten und so den Route-Type 1700 nicht verwendeten. Die Implementation an sich funktionierte ohne Probleme und es konnten sogar die Schweizer OpenStreetMap-Daten integriert werden.

Wir verglichen die Programmstrukturen der beiden Programme. Traintickets.to besitzt einige Strukturprobleme. Linus Norton selbst sagte: 'The connection scan algorithm is inherently imperative and does not port well to Scala' (Linus Norton [?]) Es ist in mehrere Programme aufgeteilt, welche jeweils Teilaufgaben übernehmen. OTP besitzt einen sehr gut strukturierten Programmcode. Es ist nach Funktionen in verschiedene Ordner aufgeteilt, welche die jeweiligen Funktionen für die Aufgabe enthalten. Das Problem hierbei ist, dass OTP von mehr als 100 verschiedenen Entwicklern programmiert wurde. Es wurden schon mehr als 10'000 Commits durchgeführt. [32] Dadurch wird für uns das Verstehen des Programmes trotz der guten Struktur viel Zeit in Anspruch nehmen.

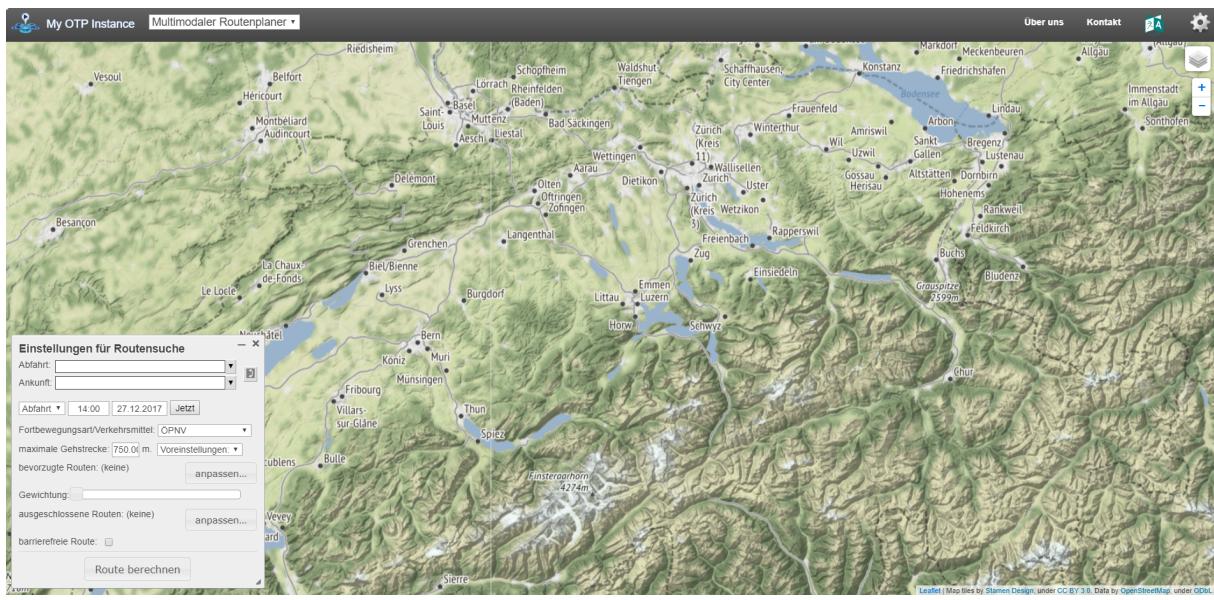


Abbildung 14: Implementation des OTP mit den Schweizer Daten

Aus diesen Erkenntnissen zogen wir, dass das OTP-Programm besser als Grundlage für unsere Bachelorarbeit geeignet ist. Es basiert nicht auf dem CSA. Wir werden den CSA für das OTP implementieren und dem OpenSource Gremium unsere Ergebnisse präsentieren, welche sie dann implementieren oder als Forschungswerte verwenden können.

6 AufgabeBA

6.1 AufgabenstellungBA

- Schreiben sie eine Implementation des CSA für den OpenTripPlanner. Dieser muss OTP-Development-Richtlinien konform sein.
- Implementieren sie die Schweizer GTFS und OpenStreetMap daten für den OTP, so dass Punkt zu Punkt Verbindungen in der Schweiz berechnet werden können.
- Implementieren sie den Schweizer realtime GTFS-Feed, so dass das Programm auf Verspätungen oder Fahrplanänderungen reagieren kann.
- Implementieren sie Skalierungsoptimierungen für das Programm, so dass es mit einem landesweiten System bessere Performanz liefert.
- Führen sie Performance Tests durch und vergleichen sie das Implementierte System mit dem auf Dijkstra basierenden OTP.
- Dokumentieren sie ihre Ergebnisse und schreiben sie einen ausführlichen Bericht.

6.2 Lastenheft

Funktionale Anforderungen		
Anforderung	Anforderungsart	Priorisierung
Der CSA muss als Berechnungsalgorithmus verwendet werden.	Muss	1
Es kann eine Verbindung zwischen 2 Stationen zu einem bestimmten Zeitpunkt errechnet werden.	Muss	2
Fusswege zu Stationen hin können miteinbezogen werden können.	Muss	3
Das Programm kann auf Verspätungen und Fahrplanänderungen reagieren.	Muss	4
Mehrere mögliche Verbindungen können angezeigt werden.	Muss	5
Fusswege zwischen verschiedenen Stationen können miteinbezogen werden.	Soll	6
Die Angezeigte Route soll den Fahrlinien, z.B. Schienen, folgen.	Kann	7

Nicht funktionale Anforderungen

- Der Programmcode muss den OTP-Development-Richtlinien entsprechen.
- Die Query-Zeit muss weniger als 1 Sekunde betragen.
- Die Preprocessing-Zeit muss weniger als 30 Minuten betragen.
- Die Query-Zeit soll schneller als die des originalen OTP sein.

6.3 Meilensteine

1. OTP Analysiert
2. OTP mit dem Schweitzer System implementieren
3. Basis CSA implementiert
4. CSA Spalierungsoptimierung implementiert
5. Performancetests durchgeführt

6. Bericht geschrieben

6.4 Zeitplan

Abbildungsverzeichnis

1	Hier sieht man wie so ein CSV-Format im Texteditor aussieht.	3
2	Diese Übersicht zeigt die Abhängigkeiten der einzelnen Files.z.B. Braucht das Trips-File die Info der Route-Files(route_id) um zu wissen auf welchem Weg diese Reise stattfindet. [4]	4
3	Dies ist eine Übersicht wie das HAFAS System aufgebaut ist. Die Fahrplandaten entsprechen hier dem HRDF-Format. [7]	6
4	Hier sieht man wie so ein TSV-Format im Texteditor aussieht.	6
5	Die Fahrplan Überblick Datei wird hier in einer Excel Tabelle zur Verfügung gestellt.	7
6	Hier sieht man ein Beispiel(Haltestelle) für das DiDok-File inklusive der Beschreibung einzelner Attribute. [11]	8
7	Beispiel einer Geschäftsorganisation mit Beschreibung der Attribute. [12]	9
8	Beschreibung der Daten (Geschäftsorganisationen mit Echtzeit) anhand von einem Beispiel. [13]	10
9	Ausschnitt der Daten in einer GA-HTA-Liste [14]	10
10	Beispielcode einer Abfahrts-/Ankunft-Anfrage. Die Parameter, die Übergeben werden sind in schwarzer Schrift dargestellt. [?]	11
11	Beispielcode einer Fahrtroute-Anfrage. Die Parameter, die Übergeben werden sind in schwarzer Schrift dargestellt. [16]	12
12	Veranschaulichung der Funktion des QuadTree-Preprocessings [21] . . .	15
13	Traintickets.To Webseite [30]	18
14	Implementation des OTP mit den Schweizer Daten	21

Literatur

- [1] Google. [Online]. Available: <http://gtfs.org/gtfs-background/>
- [2] Gtfs-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/gtfs/>
- [3] Google. [Online]. Available: <https://developers.google.com/transit/gtfs/reference/>
- [4] Transitwiki. [Online]. Available: https://www.transitwiki.org/TransitWiki/images/f/ff/GTFS_data_model_diagram.PNG
- [5] Gtfs-rt-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/gtfs-rt/>
- [6] Google. [Online]. Available: <https://developers.google.com/transit/gtfs-realtime/guides/feed-entities>
- [7] Hacon. [Online]. Available: <http://www.hacon.de/hafas/daten>
- [8] Konvertierung hafas-rohdaten nach gtfs dokumentation. [Online]. Available: <http://gtfs.geops.ch/doc/>
- [9] Hrdf-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/hafas-rohdaten-format-hrdf/>
- [10] Ist-daten-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/ist-daten/>
- [11] Didok-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/didok/>
- [12] Geschäftsorganisation-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/geschaeftsorganisation/>
- [13] Geschäftsorganisation-realtime-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/go-realtime/>
- [14] Ga-hta-liste-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/dataset/ga-hta-liste1>
- [15] Bahnhofsliste-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/bahnhofsliste/>
- [16] Fahrtprognose-opentransportdata. [Online]. Available: <https://opentransportdata.swiss/de/cookbook/fahrtprognose/>
- [17] R. Geisberger, “Advanced route planning in transportation networks,” Feb. 2011. [Online]. Available: <http://algo2.iti.kit.edu/1814.php>

- [18] “Dijkstra’s algorithm,” Oct. 2013. [Online]. Available: <http://www.cse.unt.edu/~tarau/teaching/AnAlgo>
- [19] “A survey paper of bellman-ford algorithm and dijkstra algorithm for finding shortest path in gis application,” Feb. 2014. [Online]. Available: <http://www.ijpttjournal.org/volume-5/IJPTT-V5P402.pdf>
- [20] D. Delling, T. Pajor, and R. Werneck, “Round-based public transit routing, raptor,” Jan. 2012. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/round-based-public-transit-routing>
- [21] J. Dibbelt, T. Pajor, B. Strasser, and D. Wagner, “Connection scan algorithm,” *CoRR*, vol. abs/1703.05997, 2017. [Online]. Available: <http://arxiv.org/abs/1703.05997>
- [22] H. Bast, M. Hertel, and S. Storandt, “Scalable transfer patterns,” 2016. [Online]. Available: <http://pubs.siam.org/doi/abs/10.1137/1.9781611974317.2>
- [23] H. Bast, E. Carlsson, A. Eigenwillig, R. Geisberger, C. Harrelson, V. Raychev, and F. Viger, “Fast routing in very larg public transportation networks using transfer patterns,” 2010. [Online]. Available: <https://research.google.com/pubs/pub36672.html>
- [24] “Linus norton personenbeschreibung,” Dec. 2017. [Online]. Available: <https://www.linusnorton.co.uk>
- [25] “Assertis - rail ticketing für online, mobile, digital.” Dec. 2017. [Online]. Available: <https://assertis.co.uk>
- [26] “Entstehungsblog über traintickets.to von linus norton,” Dec. 2017. [Online]. Available: <http://ljn.io/posts/so-you-want-to-build-a-journey-planner>
- [27] “Git-repository der patterngenerierung von traintickets.to,” Dec. 2017. [Online]. Available: <https://github.com/open-track/transfer-pattern-generator-scala>
- [28] “Git-repository des hauptprogrammes von traintickets.to,” Dec. 2017. [Online]. Available: <https://github.com/open-track/journey-planner>
- [29] “Git-repository der weboberfläche von traintickets.to,” Dec. 2017. [Online]. Available: <https://github.com/open-track/traintickets.to>
- [30] “Traintickets.to mit dem uk-schienennetz,” Dec. 2017. [Online]. Available: <http://traintickets.to>
- [31] “Webseite der maven-repository,” Dec. 2017. [Online]. Available: <https://maven.apache.org/guides/introduction/introduction-to-repositories.html>
- [32] “Webseite des otp-projektes,” Dec. 2017. [Online]. Available: <http://www.opentripplanner.org>

- [33] “Git-repository des otp-projektes,” Dec. 2017. [Online]. Available: <https://github.com/opentripplanner/OpenTripPlanner>
- [34] “Conveyal firmenwebseite,” Dec. 2017. [Online]. Available: <https://www.conveyal.com/trip-planning>
- [35] “Git-repository des r5-projektes,” Dec. 2017. [Online]. Available: <https://github.com/conveyal/r5/blob/master>