Multi-Modal Route Planning in Road and Transit Networks

Master's Thesis

Daniel Tischner

University of Freiburg, Germany,

daniel.tischner.cs@gmail.com

August 5, 2018

Supervisor: Prof. Dr. Hannah Bast
Advisor: Patrick Brosi

Contents

Declaration

I hereby declare, that I am the sole author and composer of my Thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work. I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date

Signature

Zusammenfassung

Wir präsentieren Algorithmen für multi-modale Routenplannung in Straßennetzwerken und Netzwerken des öffentlichen Personennahverkehrs (ÖPNV), so wie in kombinierten Netzwerken.

Dazu stellen wir das Nächste-Nachbar und das Kürzester-Pfad Problem vor und schlagen Lösungen basierend auf Cover Trees, ALT und CSA vor.

Des Weiteren erläutern wir die Theorie hinter den Algorithmen, geben eine kurze Übersicht über andere Techniken, zeigen Versuchsergebnisse auf und vergleichen die Techniken untereinander.

Abstract

We present algorithms for multi-modal route planning in road and public transit networks, as well as in combined networks.

Therefore, we explore the nearest neighbor and shortest path problem and propose solutions based on Cover Trees, ALT and CSA.

Further, we illustrate the theory behind the algorithms, give a short overview of other techniques, present experimental results and compare the techniques with each other.

1 Introduction

Route planning refers to the problem of finding an *optimal* route between given locations in a network. With the ongoing expansion of road and public transit networks all over the world route planner gain more and more importance. This led to a rapid increase in research of relevant topics and development of route planner software.

However, a common problem of most such services is that they are limited to one transportation mode only. That is a route can only be taken by a car or train but not by both at the same time. This is known as uni-modal routing. In contrast to that multi-modal routing allows the alternation of transportation modes. For example a route that first uses a car to drive to a train station, then a train which travels to a another train station and finally using a bicycle from there to reach the destination.

The difficulty with multi-modal routing lies in most algorithms being fitted to networks with specific properties. Unfortunately, road networks differ a lot from public transit networks. As such, a route planning algorithm fitted to a certain type of network will likely yield undesired results, have an impractical running time or not even be able to be used at all on different networks. We will explore this later in **Section 6**.

In this thesis we explore a technique with which we can combine an algorithm fitted for road networks with an algorithm for public transit networks. Effectively obtaining a generic algorithm that is able to compute routes on combined networks. The basic idea is simple, given a source and destination, both in the road network, we select *access nodes* for both. This are nodes where we will switch from the road into the public transit network. A route can then be computed by using the road algorithm for the source to its access nodes, the transit algorithm for the access nodes of the source to the access nodes of the destination and finally the road algorithm again for the destinations access nodes to the destination. Note that this technique might not yield the shortest possible path anymore. Also, it does not allow an arbitrary alternation of transportation modes. However, we accept those limitations since the resulting algorithm is very generic and able to compute routes faster than without limitations. We will cover this technique in detail in **Section 5.3.2**.

Our final technique uses a modified version of ALT [3] as road algorithm and CSA [2] for the transportation network. The algorithms are presented in **Section 5.1.2** and **Section 5.2.1** respectively. We also develop a multi-modal variant of Dijkstra which is able to compute the shortest route in a combined network with the possibility of changing transportation modes arbitrarily. It is presented in **Section 5.3.1** and acts as baseline to our final technique based on access nodes.

We compute access nodes by solving the nearest neighbor problem. For a given node in the road network its access nodes are then all nodes in the transit network which are in the *vicinity* of the road node. We explore a solution to this problem in **Section 4**.

Section 3 starts by defining types of networks. We represent road networks by graphs only. For transit networks we provide a graph representation too. Both graphs can then be combined into a linked graph. The advantage of graph based models is that they are well studied and therefore we are able to use our multi-modal variant of DIJKSTRA [1] to compute routes on them. However, we also propose a non-graph based representation for transit networks, a timetable. The timetable is used by CSA, an efficient algorithm for route planning on public transit networks. With that, our road and transit networks get incompatible and can not easily be combined. Therefore, we use the previously mentioned generic approach based on access nodes for this type of network.

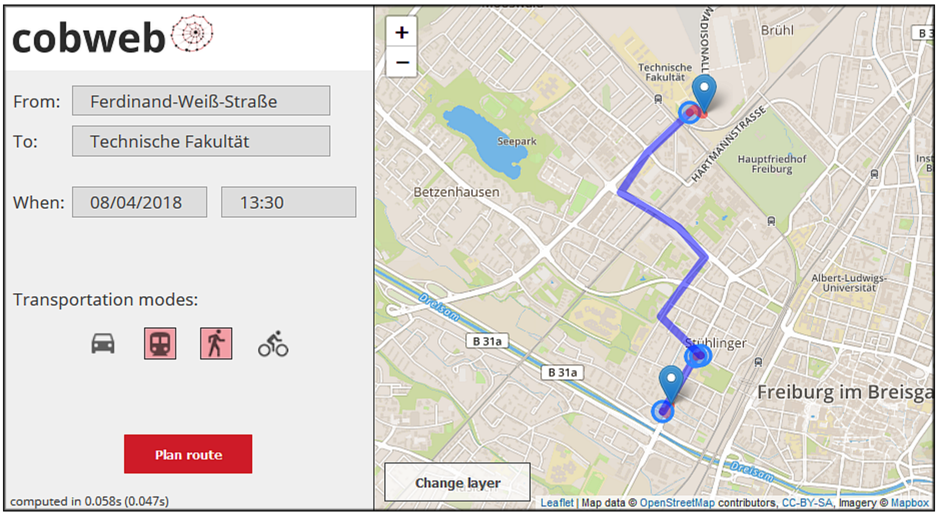

Fig. 1: Screenshot of COBWEBs [6] frontend, an open-source multi-modal route planner. It shows a multi-modal route starting from a given source, using the modes *foot-tram-foot-tram-foot* in that sequence to reach the destination.

Further, we implemented the presented algorithms in the COBWEB [6] project, which is an open-source multi-modal route planner (see **Fig. 1** for an image of its frontend). In **Section 6** we show our experimental results and compare the techniques with each other.

2 Preliminaries

Before we define our specific data models and problems we will introduce and formalize commonly reoccurring terms.

2.1 Graph

Definition 1. *A* graph G *is a tuple* (V, E) *with a set of nodes* V *and a set of edges* $E \subseteq V \times \mathbb{R}_{\geq 0} \times V$. *An* edge $e \in E$ *is an ordered tuple* (u, w, v) *with source node* $u \in V$, *a non-negative weight* $w \in \mathbb{R}_{\geq 0}$ *and a destination node* $v \in V$.

Note that **Definition 1** actually defines a *directed* graph, as opposed to an *undirected* graph where an edge like (u, w, v) would be considered equal to the edge of opposite direction (v, w, u). However, for transportation networks an undirected graph often is not applicable, for example due to one way streets or time dependent connections like trains which depart at different times for different directions.

In the context of route planning we refer to the weight w of an edge (u, w, v) as *cost*. It can be used to encode the length of the represented connection. Or to represent the time it takes to travel the distance in a given transportation mode.

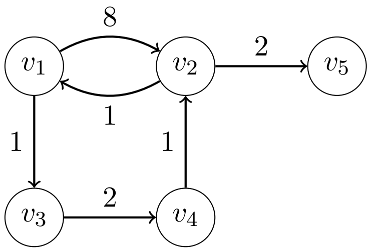

Fig. 2: Illustration of an example graph with five nodes and six edges.

As an example consider the graph $G = (V, E)$ with

$$V = \{v_1, v_2, v_3, v_4, v_5\} \text{ and}$$
$$E = \{(v_1, 8, v_2), (v_1, 1, v_3), (v_2, 1, v_1), (v_2, 2, v_5), (v_3, 2, v_4), (v_4, 1, v_2)\}.$$

which is illustrated by **Fig. 2**.

2.2 Metric

Definition 2. *A function $d : X \times X \to \mathbb{R}$ on a set X is called a* metric *iff for all $x, y, z \in X$*

$$\begin{aligned} d(x, y) &\geq 0, && \textit{non-negativity} \\ d(x, y) = 0 &\Leftrightarrow x = y, && \textit{identity of indiscernibles} \\ d(x, y) &= d(y, x) \ \textit{and} && \textit{symmetry} \\ d(x, z) &\leq d(x, y) + d(y, z) && \textit{triangle inequality} \end{aligned}$$

holds.

A metric is used to measure the distance between given locations. **Section 4** and **Section 5**, in particular **Section 5.1.2**, will make heavy use of this term.

There we measure the distance between geographical locations given as pair of *latitude* and *longitude* coordinates. Latitude and longitude, often denoted by ϕ and λ, are real numbers in the ranges $(-90, 90)$ and $[-180, 180)$ respectively, measured in degrees. However, for convenience we represent them in radians. Both representations are equivalent to each other and can easily be converted using the ratio $360° = 2\pi$ rad.

A commonly used measure is the *as-the-crow-flies* metric, which is equivalent to the euclidean distance in the euclidean space. **Definition 3** defines an approximation of this distance on locations given by latitude and longitude coordinates. The approximation is commonly known as equirectangular projection of the earth [4]. Note that there are more accurate methods for computing the great-circle distance for geographical locations, like the haversine formula [5]. However, they come with a significant computational overhead.

Definition 3. *Given a set of coordinates* $X = \left\{(\phi, \lambda) | \phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \lambda \in [-\pi, \pi)\right\}$ *we define* $\text{asTheCrowFlies} : X \times X \to \mathbb{R}$ *such that*

$$((\phi_1, \lambda_1), (\phi_2, \lambda_2)) \mapsto \sqrt{\left((\lambda_2 - \lambda_1) \cdot \cos\left(\frac{\phi_1 + \phi_2}{2}\right)\right)^2 + (\phi_2 - \phi_1)^2} \cdot 6371000.$$

As a next step we prove that asTheCrowFlies is indeed a metric on the set of coordinates.

Proposition 1. *The function* asTheCrowFlies *is a metric on its domain* X*.*

Proof. We need to prove that all four axioms hold. Let us first set

$$\begin{aligned} x &= (\lambda_2 - \lambda_1) \cdot \cos\left(\frac{\phi_1 + \phi_2}{2}\right) \\ y &= \phi_2 - \phi_1 \end{aligned}$$

then the function simplifies to

$$\sqrt{x^2 + y^2} \cdot 6371000.$$

Obviously this can never resolve to a negative number since

$$\underbrace{\underbrace{\sqrt{\underbrace{x^2}_{\geq 0} + \underbrace{y^2}_{\geq 0}}}_{\geq 0} \cdot 6371000}_{\geq 0}.$$

For the second axiom we assume that $\text{asTheCrowFlies}((\phi_1, \lambda_1), (\phi_2, \lambda_2)) = 0$ for an arbitrary pair of coordinates and follow

$$\begin{aligned} & \sqrt{\left((\lambda_2 - \lambda_1) \cdot \cos\left(\tfrac{\phi_1 + \phi_2}{2}\right)\right)^2 + (\phi_2 - \phi_1)^2} \cdot 6371000 &&= 0 \\ \Leftrightarrow \quad & \sqrt{\left((\lambda_2 - \lambda_1) \cdot \cos\left(\tfrac{\phi_1 + \phi_2}{2}\right)\right)^2 + (\phi_2 - \phi_1)^2} &&= 0 \\ \Leftrightarrow \quad & \left((\lambda_2 - \lambda_1) \cdot \cos\left(\tfrac{\phi_1 + \phi_2}{2}\right)\right)^2 + (\phi_2 - \phi_1)^2 &&= 0 \end{aligned}$$

At this point either both summands are 0 or one is the negative of the other. However, both summands must be positive due to the quadration. Because of that we follow

$$\begin{aligned} (\phi_2 - \phi_1)^2 &= 0 \\ \Leftrightarrow \qquad \phi_2 &= \phi_1 \end{aligned}$$

and with that

$$\begin{aligned} \left((\lambda_2 - \lambda_1) \cdot \cos\left(\tfrac{\phi_1+\phi_2}{2}\right)\right)^2 &= 0 \\ \Leftrightarrow \qquad (\lambda_2 - \lambda_1) \cdot \cos\left(\tfrac{2\phi_1}{2}\right) &= 0 \\ \Leftrightarrow \qquad (\lambda_2 - \lambda_1) \cdot \cos\left(\phi_1\right) &= 0. \end{aligned}$$

Since $\phi_1 \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ it follows that $\cos(\phi_1) \neq 0$. As such

$$\begin{aligned} \lambda_2 - \lambda_1 &= 0 \\ \Leftrightarrow \qquad \lambda_2 &= \lambda_1 \end{aligned}$$

and by that $(\phi_1, \lambda_1) = (\phi_2, \lambda_2)$, so the second axiom holds.

Symmetry follows quickly since

$$\begin{aligned} \phi_1 + \phi_2 &= \phi_2 + \phi_1 \\ (\phi_2 - \phi_1)^2 &= (\phi_1 - \phi_2)^2 \\ \left((\lambda_2 - \lambda_1) \cdot \cos\left(\frac{\phi_1 + \phi_2}{2}\right)\right)^2 &= (\lambda_2 - \lambda_1)^2 \cdot \cos^2\left(\frac{\phi_1 + \phi_2}{2}\right) \\ (\lambda_2 - \lambda_1)^2 &= (\lambda_1 - \lambda_2)^2. \end{aligned}$$

The triangle inequality is a bit trickier, we choose three arbitrary coordinates $c_i =$

(ϕ_i, λ_i) for $i = 1, 2, 3$ and start on the squared left side:

$$\begin{aligned}
\text{asTheCrowFlies}^2(c_1, c_3) &= \left(\left((\lambda_3 - \lambda_1) \cdot \cos\left(\frac{\phi_1 + \phi_3}{2}\right)\right)^2 + (\phi_3 - \phi_1)^2\right) \cdot 6371000^2 \\
&= \left(\left((\lambda_3 - \lambda_2 + \lambda_2 - \lambda_1) \cdot \cos\left(\frac{\phi_1 + \phi_3}{2}\right)\right)^2 + (\phi_3 - \phi_2 + \phi_2 - \phi_1)^2\right) \cdot 6371000 \\
&= \Bigg(\left((\lambda_3 - \lambda_2)^2 + (\lambda_2 - \lambda_1)^2 + 2 \cdot ((\lambda_3 - \lambda_2) \cdot (\lambda_2 - \lambda_1))\right) \cdot \cos^2\left(\frac{\phi_1 + \phi_3}{2}\right) \\
&\qquad + (\phi_3 - \phi_2)^2 + (\phi_2 - \phi_1)^2 + 2 \cdot ((\pi_3 - \phi_2) \cdot (\phi_2 - \phi_1)) \Bigg) \cdot 6371000^2 \\
&= \ldots \text{(TODO: Weitermachen oder ganz entfernen...)} \\
&\leq \left(\left(\left((\lambda_2 - \lambda_1) \cdot \cos\left(\frac{\phi_1 + \phi_2}{2}\right)\right)^2 + (\phi_2 - \phi_1)^2\right) \cdot 6371000^2\right) \\
&\qquad + \left(\left(\left((\lambda_3 - \lambda_2) \cdot \cos\left(\frac{\phi_2 + \phi_3}{2}\right)\right)^2 + (\phi_3 - \phi_2)^2\right) \cdot 6371000^2\right) \\
&\qquad + 2 \cdot \left(\left(\left((\lambda_2 - \lambda_1) \cdot \cos\left(\frac{\phi_1 + \phi_2}{2}\right)\right)^2 + (\phi_2 - \phi_1)^2\right) \cdot 6371000^2\right) \\
&\qquad \cdot \left(\left(\left((\lambda_3 - \lambda_2) \cdot \cos\left(\frac{\phi_2 + \phi_3}{2}\right)\right)^2 + (\phi_3 - \phi_2)^2\right) \cdot 6371000^2\right) \\
&= (\text{asTheCrowFlies}(c_1, c_2) + \text{asTheCrowFlies}(c_2, c_3))^2
\end{aligned}$$

All four axioms hold, asTheCrowFlies is a metric on the set X. □

3 Models

This section defines the models we use for the different network types. We define a graph based representation for road and transit networks. Then both graphs are combined into a linked graph, making it possible to have one graph for the whole network. Afterwards an alternative representation for transit networks is shown.

3.1 Road graph

A road network typically is time independent. It consists of geographical locations and roads connecting them with each other. We assume that a road can be taken at any time, with no time dependent constraints.

Modeling the network as graph is straightforward, **Definition 4** goes into detail.

Definition 4. *A* road graph *is a graph $G = (V, E)$ with a set of geographic coordinates*

$$V = \{(\phi, \lambda) | \phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \lambda \in [-\pi, \pi)\},$$

for example road junctions. There is an edge $(u, w, v) \in E$ iff there is a road connecting the location u with the location v, which can be taken in that direction. The weight w of the edge is the average time needed to take the road from u to v using a car, measured in seconds.

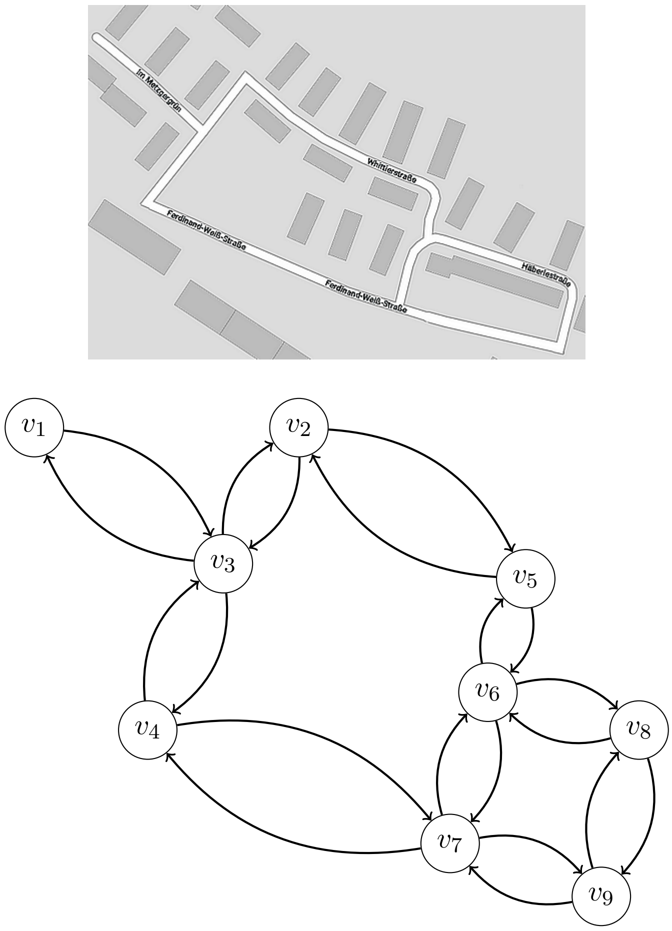

Fig. 3: Example of a road network with its corresponding road graph. White connections indicate roads, dark gray rectangles represent houses or other static objects. Geographical coordinates for each node as well as edge weights are omitted in the graph illustration.

Fig. 3 shows a constructed example road network with the corresponding road graph. Note that two way streets result in two edges, one edge for every direction the road can be taken.

Since edge weights are represented as average time it needs to take the road, it is possible to encode different road types. For example the average speed on a motorway is much higher than on a residential street. As such, the weight of an edge representing a motorway is much smaller than the weight of an edge representing a residential street.

While the example has exactly one node per road junction this must not always be the case. Typical real world data often consists of multiple nodes per road segment. However, **Definition 4** is still valid for such data as long as there are edges between the nodes if and only if there is a road connecting the locations.

3.2 Transit graph

Transit networks can be modeled similar to road graphs. The key difference is that transit networks are time dependent while road networks typically are not. For example an edge connecting *Freiburg main station* with *Karlsruhe main station* can not be taken at any time since trains and other transit vehicles only depart at certain times. The schedule might even change at different days.

The difficulty lies in modeling time dependence in a static graph. There are two common approaches to that problem.

The first approach is called *time-dependent*. There edge weights are not static numbers but functions that take a date with time and compute the cost it needs to take the edge when starting at the given time. This includes waiting time. As an example assume an edge (u, c, v) with the cost function c. The edge represents a train connection and the travel time are 10 minutes. However, the train departs at 10:15 *am* am but the starting time is 10:00 *am*. The cost function thus computes a waiting time of 15 minutes plus the travel time of 10 minutes. Resulting in an edge weight of 25 minutes.

The main problem with this model is that it makes pre-computations for route planning very difficult as the starting time is not known in advance.

The second approach is called *time-expanded*. There idea is to remove any time dependence from the graph by creating additional nodes for every event at a station. A node then also has a time information next to its geographic location.

Definition 5. *A* time expanded transit graph *is a graph $G = (V, E)$ with a set of events at geographic coordinates*

$$V = \{(\phi, \lambda, t, e) | \phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \lambda \in [-\pi, \pi)\,, t \textit{ time}, e \in \{\textit{arrival}, \textit{departure}\}\},$$

for example a train arriving at a train station at a certain time.

A node $(\phi, \lambda, t, e) \in V$ is an arrival node *if $e =$ arrival, analogously it is a* departure node *for $e =$ departure. For a node $v \in V$, v_ϕ and v_λ denote its location, v_t its time and v_e its event type.*

It must be ensured that each connection starts with an arrival node.

There is an edge $(u, w, v) \in E$ iff

1. *$u_e =$ departure $\wedge\, v_e =$ arrival such that there is a vehicle departing from u at time u_t which arrives at v at time v_t without stops in between, or*

2. *$u_e = arrival \wedge v_e = departure$ such that u and v belong to the same connection. For example a train arriving at a station and then departing again, or*

3. *$u_e = arrival \wedge v_e = arrival$ such that v is the node at the same coordinates than u with the smallest time v_t that is still greater than u_t. This edge represents exiting a vehicle and waiting for another connection. That is*

$$\forall v' \in V \setminus \{v\} : v'_\phi = u_\phi \wedge v'_\lambda = u_\lambda \wedge v'_e = arrival \wedge v'_t \geq u_t$$
$$\Rightarrow v'_t - u_t > v_t - u_t.$$

The weight w of an edge (u, w, v) is the difference between both nodes times, that is

$$w = v_t - u_t.$$

Note that weights are still positive since $v_t \geq u_t$ always holds due to construction.

Definition 5 defines such a time expanded transit graph and **Fig. 4** shows an example. For simplicity it is assumed that the trains have no stops other than shown in the schedule. The schedule lists four trains:

1. The ICE 104 which travels from Freiburg Hhbf to Karlsruhe Hbf via Offenburg,
2. the RE 17024 connecting Freiburg Hbf with Offenburg,
3. the RE 17322 driving from Offenburg to Karlsruhe Hbf and
4. ICE 79 which travels in the opposite direction, connecting Karlsruhe Hbf with Freiburg without intermediate stops.

For connections that start at a station, like the $ICE104$ in Freiburg Hbf an additional arrival node is introduced in the transit graph. The arrival node has the same time than its corresponding departure node, the connecting edge has a weight of 0 minutes. The additional node is necessary to fulfil the requirement of **Definition 5** that all connections start with an arrival node. It is also necessary for correctness as it must be possible to transfer from an arriving train to another train that starts at station. For example to transfer from ICE 104 to RE 17322 in Offenburg, which is represented by the waiting arc with cost 7 minutes.

As seen in the example the resulting graph has no time dependency anymore and is static, as well as all edge weights. The disadvantage is that the graph size dramatically increases as a new node is introduced for every single event. In order to limit the growth we assume that a schedule is the same every day and does not change. In fact, most schedules are stable and often change only slightly, for example on weekends or at holidays. In practice hybrid models can be used for those exceptions.

3.3 Link graph

blabla

$\longrightarrow$	Freiburg Hbf	Offenburg		Karlsruhe Hbf
	`departure`	`arrival`	`departure`	`arrival`
ICE 104	3:56 *pm*	4:28 *pm*	4:29 *pm*	4:58 *pm*
RE 17024	4:03 *pm*	4:50 *pm*		
RE 17322			4:35 *pm*	5:19 *pm*
$\longleftarrow$	`arrival`	`departure`	`arrival`	`departure`
ICE 79	8:10 *pm*			7:10 *pm*

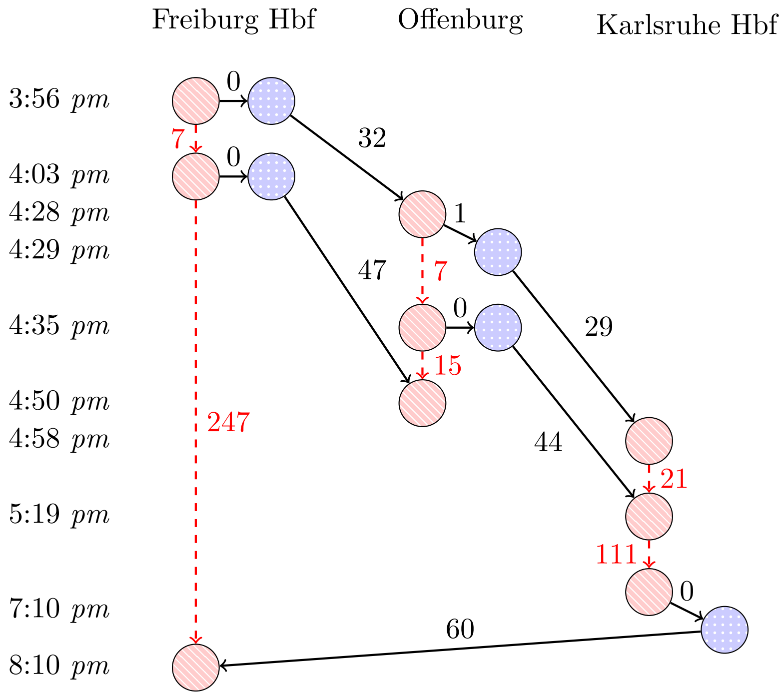

Fig. 4: Example of a transit network with its corresponding time expanded transit graph. The table shows an excerpt of a train schedule. Nodes with stripes are arrival nodes, dotted nodes represent departure nodes. Regular edges indicate a train connection and dashed edges waiting arcs. Edge weights are measured in minutes.

3.4 Timetable

blabla

4 Nearest neighbor problem

Blabla

4.1 Cover tree

Blabla

5 Shortest path problem

Blabla

5.1 Time-independent

Blabla

5.1.1 Dijkstra

Blabla

5.1.2 A* and ALT

Blabla

5.2 Time-dependend

Blabla

5.2.1 Connection scan

Blabla

5.3 Multi-modal

Blabla

5.3.1 Modified Dijkstra

Blabla

5.3.2 Access nodes

Blabla

5.4 Other algorithms

Blabla

6 Evaluation

Blabla

6.1 Input data

Blabla

6.2 Experiments

Blabla

6.2.1 Nearest neighbor computation

Blabla

6.2.2 Uni-modal routing

Blabla

6.2.3 Multi-modal routing

Blabla

6.3 Summary

Blabla

7 Conclusion

Blabla

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[2] Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Connection scan algorithm. *CoRR*, abs/1703.05997, 2017.

[3] Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 156–165, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[4] John P. Snyder. Flattening the earth: Two thousand years of map projections. 10 1994.

[5] C. C. Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, 1957.

[6] Daniel Tischner. Cobweb. https://github.com/ZabuzaW/Cobweb, 2018.