GraphHopper

ÖV-Routing in GraphHopper

Michael Zilske

Graphhopper

Post-Dijkstra Transit Routing

- RAPTOR (Delling, Pajor, Werneck, 2012)
- Connection Scan (Dibbelt, Pajor, Strasser, Wagner, 2013)

- Verwenden *keinen* Haltestellengraphen, sondern direkt den Fahrplan
- Nutzen Eigenschaften speziell des fahrplanbasierten Verkehrs aus:
  - RAPTOR: Dynamisches Programmieren über Anzahl der Umstiege
  - Connection Scan: Fahrplan ist klein genug, dass ich alle Fahrplanereignisse nach Zeit sortieren und pro Anfrage linear durchlesen kann

- Sind also für *reine* Fahrplanabfragen gedacht
- und sehr schnell

GraphHopper

Multimodales Routing

- Routing auf kombiniertem Graphen aus ÖV und Straße
- Suche auf dem gesamten Möglichkeitsraum aus zu Fuß gehen und ÖV nutzen
- Umstiege müssen nicht extra modelliert sein und auch nicht vorberechnet werden
- Das ist bereits ein *schwierigeres* Problem als das, was Algorithmen wie Connection Scan lösen

TD vs. TE

Graphhopper

- Wie reduziere ich Fahrplanauskunft auf Wegfinden in Graphen?
- Zwei Alternativen:
- 1. Liniengraph, und Kantengewichte (Reisezeiten) sind zeitabhängig
- 2. Kanten werden in die Zeitachse dupliziert (Knoten entspricht “Aufenthalt um Uhrzeit”, Kante entspricht einzelnem Fahrplanereignis)

Früheste Ankunftszeit

Graphhopper

- Gegeben: Start, Ziel, früheste Abfahrtszeit
- Gesucht: Weg mit frühester Ankunftszeit
- Es kann mehrere Wege mit derselben frühesten Ankunftszeit geben. Im Straßennetz passiert das im Allgemeinen nicht. Im Fahrplan ist es der Normalfall.
- Diese Wege unterscheiden sich teilweise drastisch. Insbesondere in der Anzahl der Umstiege.

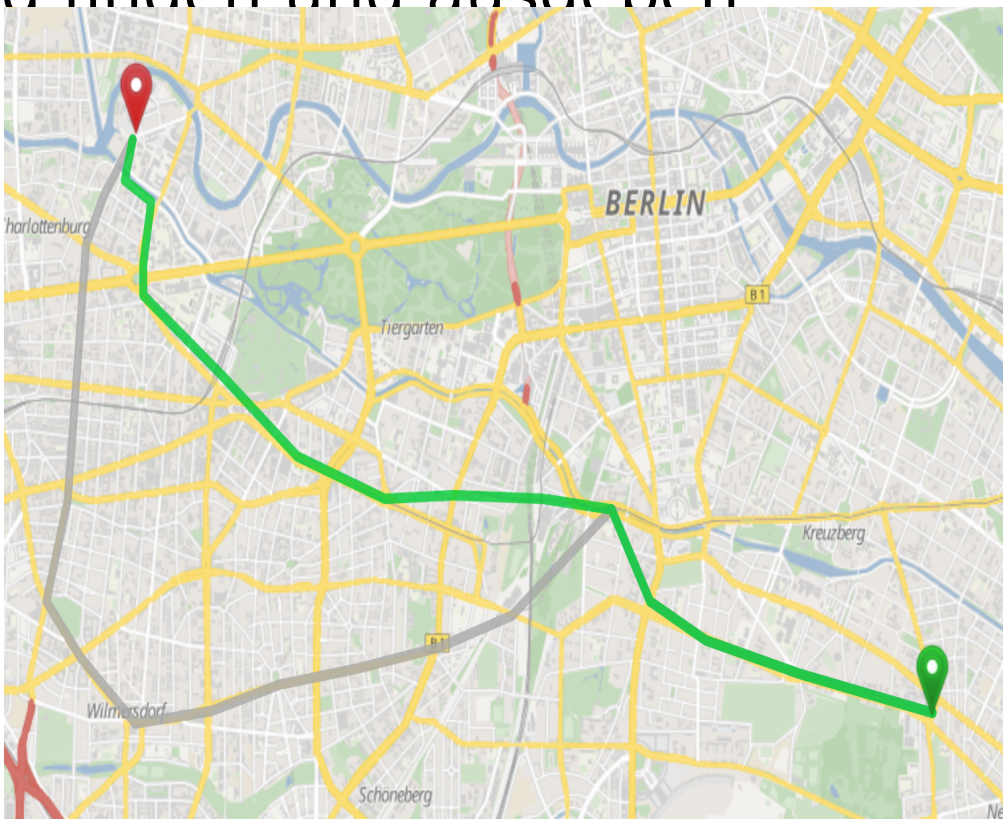
Alternativen

- Theoretisch könnten wir jeden Nutzer sagen lassen, was sein Kriterium ist, und dann den eindeutig besten Weg finden und ausgeben

((unter gewissen Bedingungen für das Kriterium: additiv über den Kanten, nichtnegativ, bei zeitabhängigen Kosten noch eine weitere))

Praktisch weiß keine Person, was ihr allgemeines Kriterium für beste Wege ist.

Deswegen müssen wir “sinnvolle Alternativen” ausgeben.



Umstiege

Graphhopper

- Gegeben: Start, Ziel, früheste Abfahrtszeit
- Gesucht: Wege, für die es keinen in beiden Kriterien (Ankunftszeit, Umstiege) besseren gibt
  - (14:25, 3 Umstiege)
  - (14:34, 2 Umstiege)
  - (14:47, 0 Umstiege)
  - (20:00, nur gehen)

Graphhopper

Multicriteria Label Setting

- Dijkstra
- Mit einer *Menge* von Teillösungen in jedem Knoten, anstatt einer
- Umstiege sind relativ gutartig: Es gibt nur soundso viele sinnvolle Umstiege.
- Bei allgemeinen Kriterien wird *sehr* schnell das Ergebnis *sehr* groß wird. Schlimmstenfalls gibt es 2^n Wege, und keiner ist schlechter als der andere

Weitere Kriterien

Graphhopper

- Spätere Abfahrzeit
  - führt zu Intervallabfrage
- Fahrpreis
  - Hängt sehr vom Preissystem ab, wie einfach das ist
  - Praktisch scheint schon das *Ermitteln* des Fahrpreises für einen *gegebenen* Weg schwer sein

Weitere Kriterien

GraphHopper

- kurze Fußwege
  - schwierig, denn: ich darf keine *versteckten Kriterien* bauen, sonst funktioniert Dijkstra nicht
  - *Länge des Fußwegs* als volles Kriterium ist aber teuer
  - praktisch sehen die Ergebnisse trotzdem nicht schlecht aus, wenn ich einfach nach Fußweg x abbreche

Anderes Problem

Graphhopper

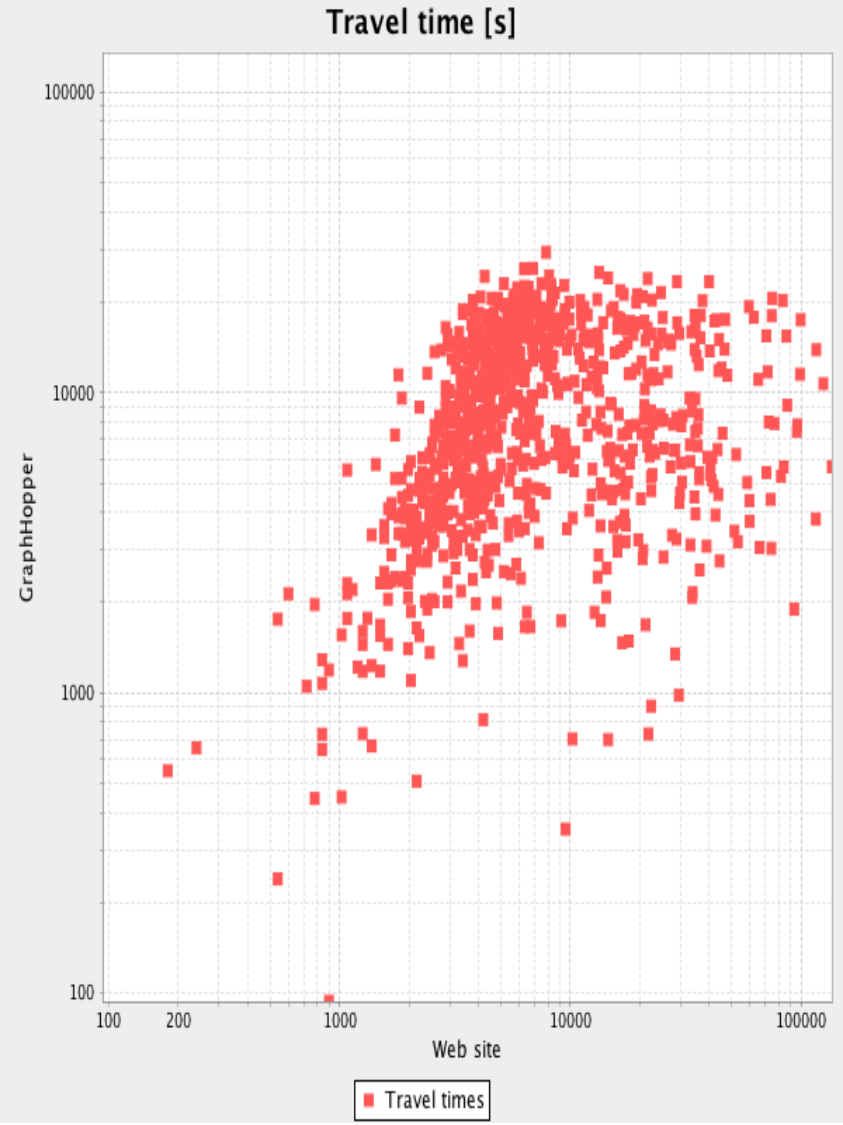
- *Erwartete* Ankunftszeit statt fahrplanmäßiger
  - d.h. Berücksichtigung historischer Verspätungen und Zuverlässigkeit von Anschlüssen

Angebotsdaten

- Stand März 2017: 3 Datensätze in Deutschland
- SWU Verkehr (Ulm)
- Verkehrsverbund Berlin-Brandenburg
- Rhein-Neckar-Verkehr
- ((und ein inoffizieller Fernverkehrsdatensatz))

Test mit bahn.de

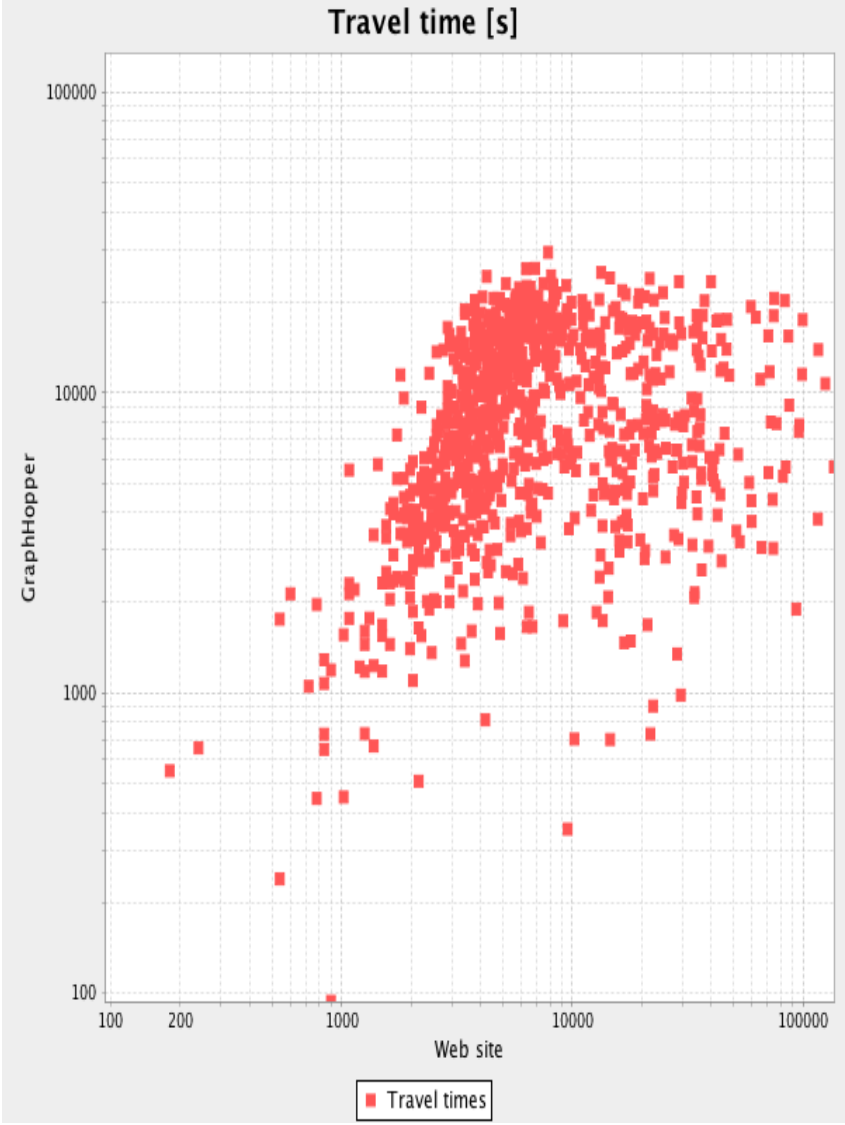
- Testfall-Liste mit zufälligen früheste-Ankunftszeit-Abfragen im Rhein-Neckar-Verkehr.
- Abfrage bahn.de vs. GraphHopper-Prototyp
- Fälle mit gleichen Ergebnissen sollten auf der Diagonalen liegen.



Datenbasiertes Debugging

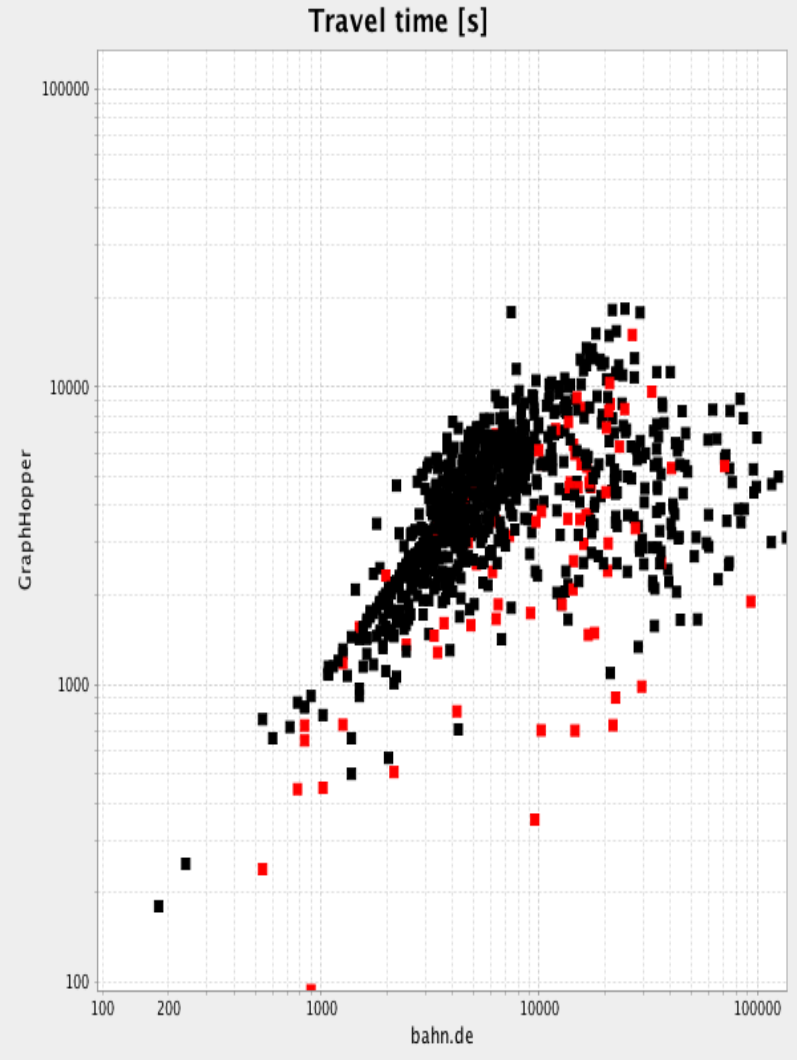
GraphHopper

- Was ist hier passiert?
- Alle gehen zu Fuß
- Falschen Fahrplan genommen.



Datenbasiertes Debugging

- Schon besser.
- Der Ausreißer oben in der Mitte war ein Defekt (repariert).
- Der Rest sind tatsächliche Möglichkeiten, die in bahn.de bzw. dem GTFS-Datensatz nicht abgebildet sind.



Stand März 2017

GraphHopper

- GraphHopper-Graph für Straße
- Zeitexpandiertes Modell für ÖV
- Zeitabhängige Übergangskanten
- Multicriteria-Dijkstra
- GTFS-Features
  - Umsteigebeschränkungen
  - Betriebstage
  - Durchbindungen (im Bus sitzenbleiben)
  - Fahrpreisermittlung
- Gehwegbeschränkung (heuristisch)
- Ansonsten exakt, und noch nicht mit Beschleunigung angefangen

Graphhopper

Beschleunigungsmöglichkeiten

- gute Filterkriterien
- gute Restgewicht-Abschätzung (A*)

- Vorberechnung relevanter Ein- und Ausstiege
  - Access-node routing
  - User-constrained contraction hierarchies

Graphhopper

Wohin soll es gehen?

- Maximale Flexibilität in der Modellierung
- Integration von Echtzeitdaten
- also vermutlich einfache Algorithmen
- eher Neo-Dijkstra als Post-Dijkstra
- Trotzdem erträgliche Antwortzeiten

Danke!

michael.zilske@graphhopper.com