Karlsruher Institut für Technologie

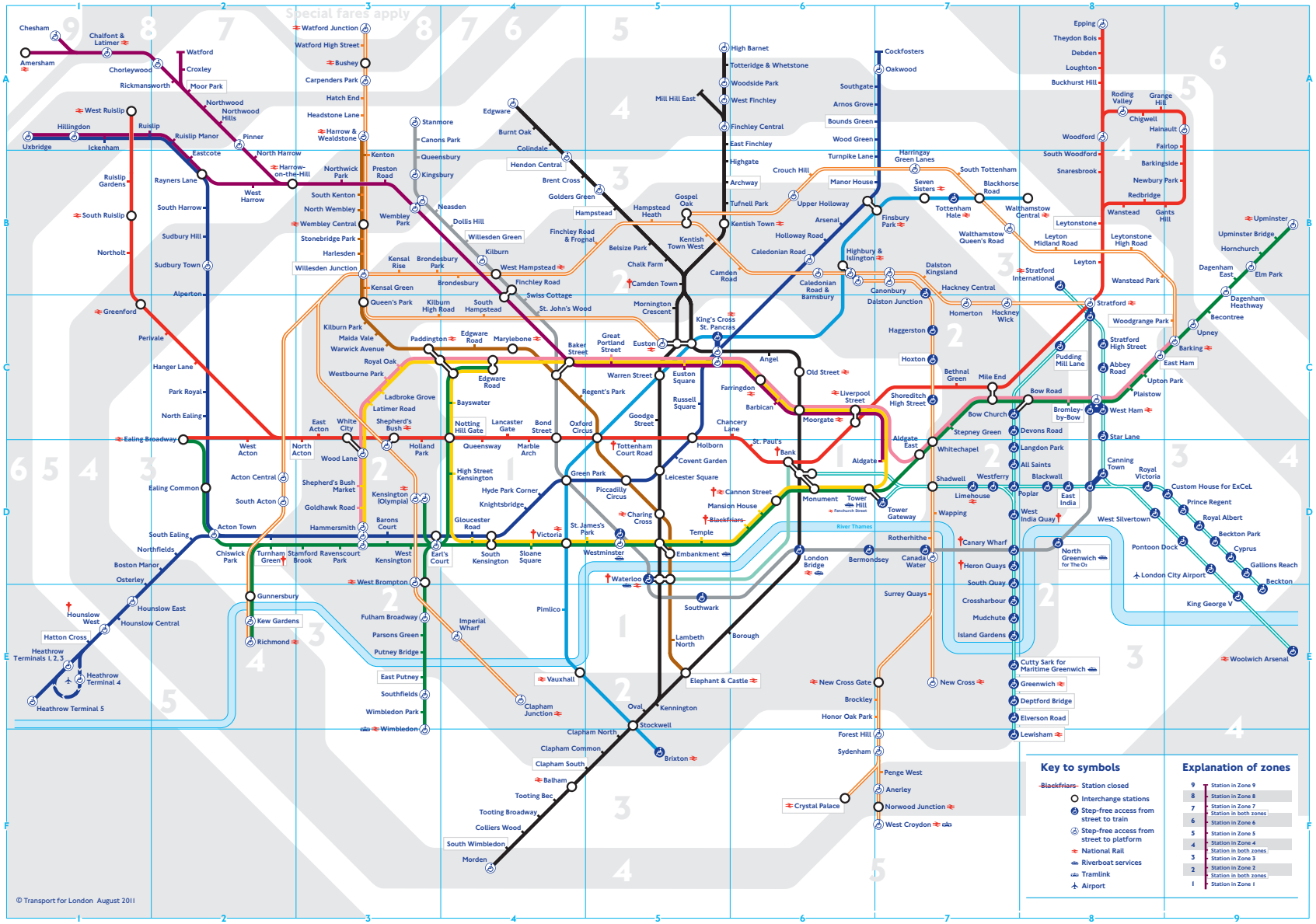
Algorithmen für Routenplanung

14. Sitzung, Sommersemester 2013

Thomas Pajor | 24. Juni 2013

Institut für Theoretische Informatik · Algorithmik · Prof. Dr. Dorothea Wagner

Fahrplanauskunft



Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Abgrenzung zu Straßennetzwerken

Eingabe bei Straßennetzen

- Straßenkarte bestehend aus
- Kreuzungen
- Straßensegmenten
- Verschiedene Metriken (Reisezeit, Distanz, . . .)
- Evtl. historische Reisezeitprofile
- Abbiegekosten, . . .

Abgrenzung zu Straßennetzwerken

Eingabe bei Straßennetzen

- Straßenkarte bestehend aus
- Kreuzungen
- Straßensegmenten
- Verschiedene Metriken (Reisezeit, Distanz, . . .)
- Evtl. historische Reisezeitprofile
- Abbiegekosten, . . .

Was ist Eingabe bei der Fahrplanauskunft?

Fahrpläne

Gegeben (Fahrplan):

- Menge $\mathcal{B}$ von Bahnhöfen (Stops, Bahnsteigen, ...),
- Menge $\mathcal{Z}$ von Zügen (Bussen, Trams, etc)
- Menge $\mathcal{C}$ von elementaren Verbindungen
- Mindestumstiegszeiten transfer : $\mathcal{B} \to \mathbb{N}$.

Fahrpläne

Gegeben (Fahrplan):

- Menge $\mathcal{B}$ von Bahnhöfen (Stops, Bahnsteigen, ...),
- Menge $\mathcal{Z}$ von Zügen (Bussen, Trams, etc)
- Menge $\mathcal{C}$ von elementaren Verbindungen
- Mindestumstiegszeiten transfer : $\mathcal{B} \to \mathbb{N}$.

Elementare Verbindung: Tupel bestehend aus

- Zug $Z \in \mathcal{Z}$
- Abfahrtsbahnhof $S_{\text{dep}} \in \mathcal{B}$
- Zielbahnhof $S_{\text{arr}} \in \mathcal{B}$
- Abfahrtszeit $\tau_{\text{dep}} \in \Pi$
- Ankunftszeit $\tau_{\text{arr}} \in \Pi$

Fahrpläne

Gegeben (Fahrplan):

- Menge $\mathcal{B}$ von Bahnhöfen (Stops, Bahnsteigen, ...),
- Menge $\mathcal{Z}$ von Zügen (Bussen, Trams, etc)
- Menge $\mathcal{C}$ von elementaren Verbindungen
- Mindestumstiegszeiten transfer : $\mathcal{B} \to \mathbb{N}$.

Elementare Verbindung: Tupel bestehend aus

- Zug $Z \in \mathcal{Z}$
- Abfahrtsbahnhof $S_{\text{dep}} \in \mathcal{B}$
- Zielbahnhof $S_{\text{arr}} \in \mathcal{B}$
- Abfahrtszeit $\tau_{\text{dep}} \in \Pi$
- Ankunftszeit $\tau_{\text{arr}} \in \Pi$

Interpretation: Zug Z fährt von S_{dep} nach S_{arr} ohne Zwischenhalt von τ_{dep} bis τ_{arr} Uhr.

Fahrpläne II

Trips

- Fahrt *eines* Zuges *Z*
- Von Endstation zu Endstation
- Abfahrten an den Stops zu bestimmten Zeiten

Fahrpläne II

Trips

- Fahrt *eines* Zuges Z
- Von Endstation zu Endstation
- Abfahrten an den Stops zu bestimmten Zeiten

Routen

- Partitionierung der Trips
- Zwei Trips t_1, t_2 gehören zur gleichen Route, gdw.
- t_1 und t_2 folgen der genau gleichen Sequenz von Stops

Fahrpläne, Beispiel

Beispiel für einen Fahrplan

```
...
(IR 2269, Karlsruhe Hbf,  Pforzheim Hbf,  10:05, 10:23)
(IR 2269, Pforzheim Hbf,  Mühlacker,      10:25, 10:33)
(IR 2269, Mühlacker,      Vaihingen(Enz), 10:34, 10:40)
(IR 2269, Vaihingen(Enz), Stuttgart Hbf,  10:41, 10:57)
...
(ICE 791, Stuttgart Hbf,  Ulm Hbf,        11:12, 12:06)
(ICE 791, Ulm Hbf,        Augsburg Hbf,   12:08, 12:47)
(ICE 791, Augsburg Hbf,   München Hbf,    12:49, 13:21)
...
```

Fahrpläne, Beispiel

Beispiel für einen Fahrplan

```
...
(IR 2269, Karlsruhe Hbf,  Pforzheim Hbf,  10:05, 10:23)
(IR 2269, Pforzheim Hbf,  Mühlacker,      10:25, 10:33)
(IR 2269, Mühlacker,      Vaihingen(Enz), 10:34, 10:40)
(IR 2269, Vaihingen(Enz), Stuttgart Hbf,  10:41, 10:57)
...
(ICE 791, Stuttgart Hbf,  Ulm Hbf,        11:12, 12:06)
(ICE 791, Ulm Hbf,        Augsburg Hbf,   12:08, 12:47)
(ICE 791, Augsburg Hbf,   München Hbf,    12:49, 13:21)
...
```

Frage: Wie Fahrplan modellieren?

Ausgabe

Ausgabe bei der Fahrplanauskunft

- Sequenz von (Teil-)Trips aus dem Fahrplan
- Beginnend bei Startstop
- Ended bei Zielstop
- (Möglicherweise mit Fußwegen dazwischen)

Ausgabe

Ausgabe bei der Fahrplanauskunft

- Sequenz von (Teil-)Trips aus dem Fahrplan
- Beginnend bei Startstop
- Ended bei Zielstop
- (Möglicherweise mit Fußwegen dazwischen)

Reiseroute ist konsistent wenn

- Erster Stop von Trip t_{i+1} entspricht letztem Stop von Trip t_i
- Abfahrt von t_{i+1} ist *nach* Ankunft von t_i

Modellierung von Fahrplänen

Zwei grundlegende Ansätze

1. Modellierung als gerichteter Graph
2. Keine besondere Modellierung (benutze Fahrplan "direkt")

Modellierung von Fahrplänen

Zwei grundlegende Ansätze

1. Modellierung als gerichteter Graph
2. Keine besondere Modellierung (benutze Fahrplan "direkt")

Jetzt ersteres, später zweiteres.

Modellierung von Fahrplänen

Zwei grundlegende Ansätze

1. Modellierung als gerichteter Graph
2. Keine besondere Modellierung (benutze Fahrplan "direkt")

Jetzt ersteres, später zweiteres.

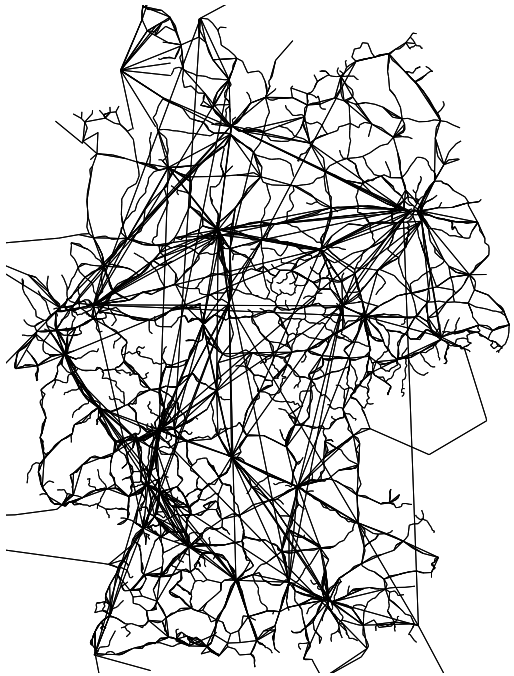
Modellierung als Graph

- Reduziere auf (bekanntes) kürzeste-Wege-Problem
- Optimale Reiserouten entsprechen kürzesten Wegen
- Bekannte Techniken (evtl. leicht) übertragbar

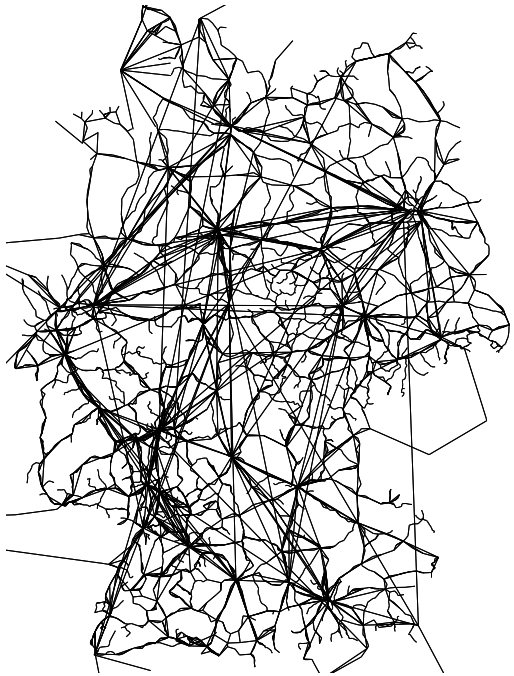
Stationsgraph

Modellierung:

- Für jeden Bahnhof $S \in \mathcal{B}$: Ein Knoten
- Kante (S_i, S_j) gdw. ex. elementare Verbindung von S_i nach S_j
- Kantengewichte: Lower-Bound der Reisezeit zw. Bahnhöfen



Stationsgraph



Modellierung:

- Für jeden Bahnhof $S \in \mathcal{B}$: Ein Knoten
- Kante (S_i, S_j) gdw. ex. elementare Verbindung von S_i nach S_j
- Kantengewichte: Lower-Bound der Reisezeit zw. Bahnhöfen

Nachteile:

- Unrealistische Anfragen (keine Zeitabhängigkeit)
- Mit Zeitabhängigkeit: Unrealistische Umstiege

Modellierung: Zwei Ansätze

1. Zeitexpandiert

- Zeitabhängigkeiten ausrollen
- Knoten entsprechen Ereignissen im Fahrplan
- Kanten verbinden Ereignisse miteinander
  - Zugfahrt eines Zuges,
  - Umstieg zwischen Zügen,
  - Warten

\- Großer Graph (viele Knoten und Kanten)

\+ Einfacher Anfragealgorithmus (Dijkstra)

Modellierung: Zwei Ansätze

1. Zeitexpandiert

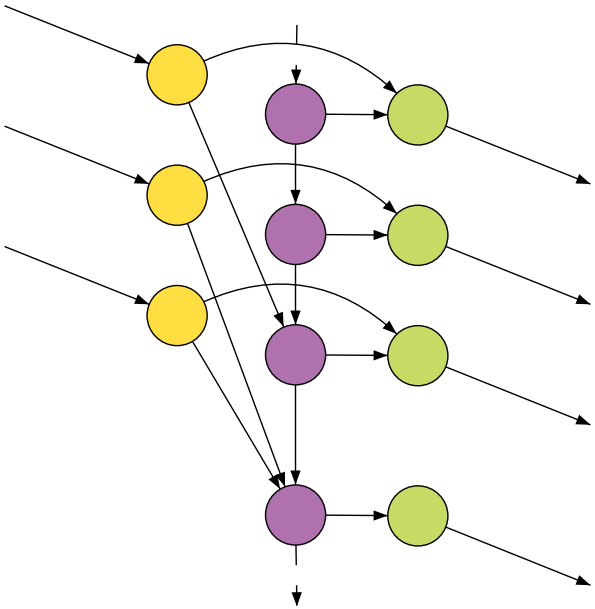
- Zeitabhängigkeiten ausrollen
- Knoten entsprechen Ereignissen im Fahrplan
- Kanten verbinden Ereignisse miteinander
  - Zugfahrt eines Zuges,
  - Umstieg zwischen Zügen,
  - Warten
- \- Großer Graph (viele Knoten und Kanten)
- \+ Einfacher Anfragealgorithmus (Dijkstra)

2. Zeitabhängig

- Zeitabhängigkeit an den Kanten
- Knoten entsprechen Bahnhöfen
- Kante $\Leftrightarrow$ Zug verbindet Bahnhöfe
  - Transferzeiten?
- \+ Kleiner Graph
- \- Zeitabhängige Routenplanung

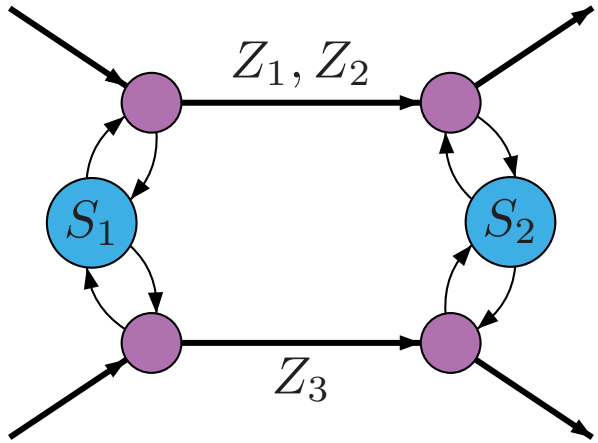
Modellierung: Zwei Ansätze

1. Zeitexpandiert



- Arrival-, Transfer- und Departure-Ereignisse
- Für jeden Zug
- Kantengewicht = Zeitdifferenz

2. Zeitabhängig



- Partitioniere Züge in Routen
- Pro Bahnhof: Stationsknoten
- Pro Route: Route-Knoten
- Routenkanten: Mehrere Züge
- Stationskanten: Transferzeit

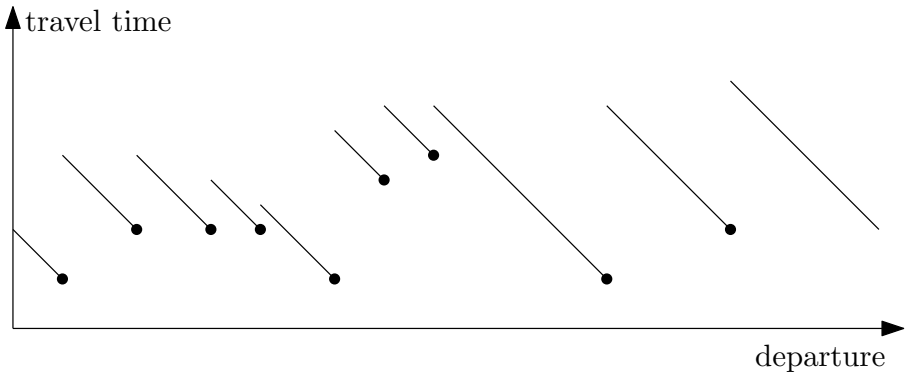
Public Transport Funktionen

Eingabe:

- Reisezeit zu bestimmten Zeitpunkten (Fahrzeiten der Züge)
- Jeden Tag verschieden

Somit:

- Periodische stückweise lineare Funktionen
- Definiert durch Stützpunkte
- Wartezeit zur nächsten Verbindung + Reisezeit



FIFO-Eigenschaft (Wdh.)

Definition

Sei $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ eine Funktion. f erfüllt die *FIFO-Eigenschaft*, wenn für jedes $\varepsilon > 0$ und alle $\tau \in \mathbb{R}_0^+$ gilt, dass

$$f(\tau) \leq \varepsilon + f(\tau + \varepsilon).$$

Diskussion

- Interpretation: “Warten lohnt sich nie”
- Kürzeste Wege auf Graphen mit non-FIFO Funktionen zu finden ist NP-schwer.
  (wenn Warten an Knoten nicht erlaubt ist)

  $\Rightarrow$ Sicherstellen, dass Funktionen FIFO-Eigenschaft erfüllen.

Operationen

Funktion gegeben durch:

- Menge von Interpolationspunkten
- $I^f := \{(t_1^f, w_1^f), \ldots, (t_k^f, w_k^f)\}$

3 Operationen notwendig:

- Auswertung
- Linken $\oplus$
- Minimumsbildung

Beobachtung:

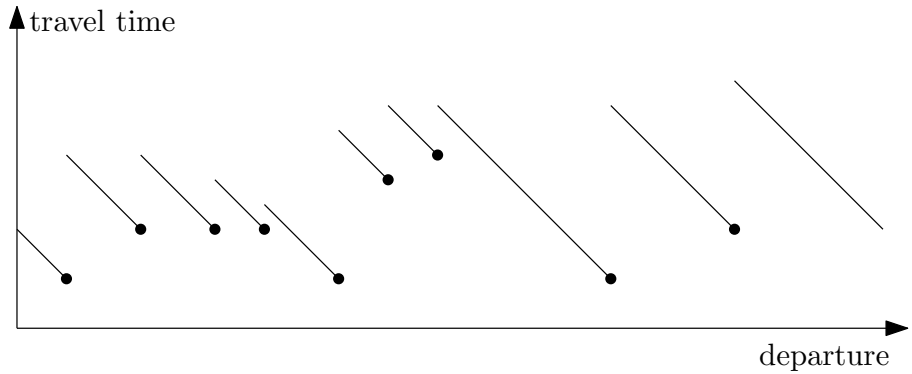
- Unterschiedlich für Straße und Schiene!

Public Transport: Auswertung

Evaluation von $f(\tau)$:

- Suche Punkte mit $t_i \geq \tau$ und $t_i - \tau$ minimal
- dann Evaluation durch

$$f(\tau) = w_i + (t_i - \tau)$$

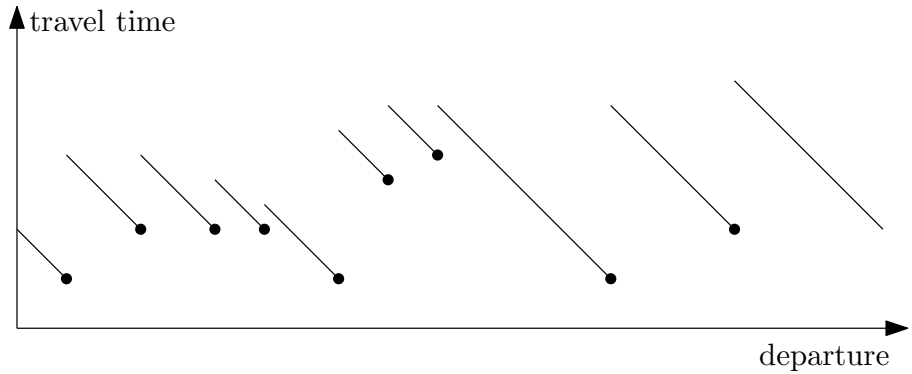


Public Transport: Auswertung

Evaluation von $f(\tau)$:

- Suche Punkte mit $t_i \geq \tau$ und $t_i - \tau$ minimal
- dann Evaluation durch

$$f(\tau) = w_i + (t_i - \tau)$$



Problem:

- Finden von t_i und t_{i+1}
- Theoretisch:
  - Lineare Suche: $\mathcal{O}(|I|)$
  - Binäre Suche: $\mathcal{O}(\log_2 |I|)$
- praktisch:
  - $|I| < 30$: Lineare Suche
  - Sonst: Lineare Suche mit Startpunkt $\frac{\tau}{\Pi} \cdot |I|$

Linken

Definition

Seien $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ und $g : \mathbb{R}_0^+$ zwei Funktionen die die FIFO-Eigenschaft erfüllen. Die Linkoperation $f \oplus g$ ist dann definiert durch

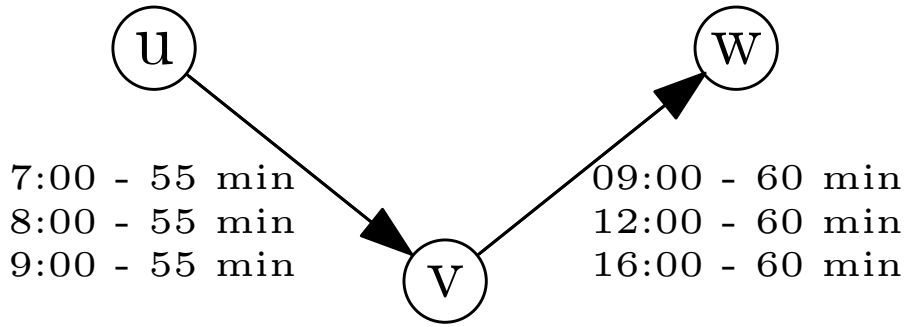
$$f \oplus g := f + g \circ (\mathrm{id} + f)$$

Oder

$$(f \oplus g)(\tau) := f(\tau) + g(\tau + f(\tau))$$

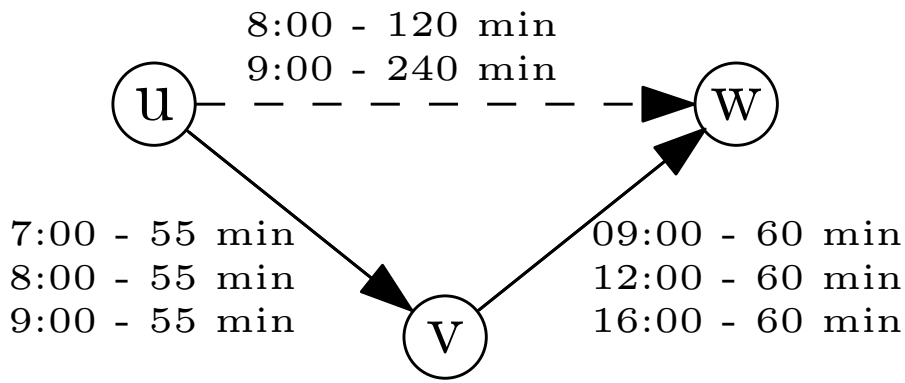
Public Transport: Link

Linken zweier Funktionen f und g



Public Transport: Link

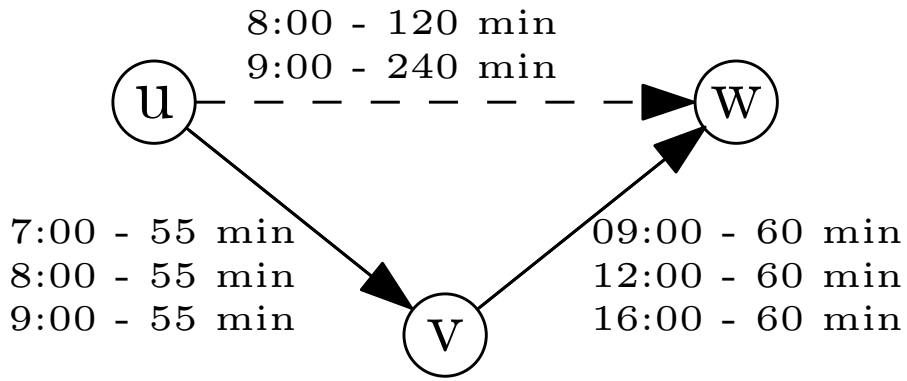
Linken zweier Funktionen f und g



Public Transport: Link

Linken zweier Funktionen f und g

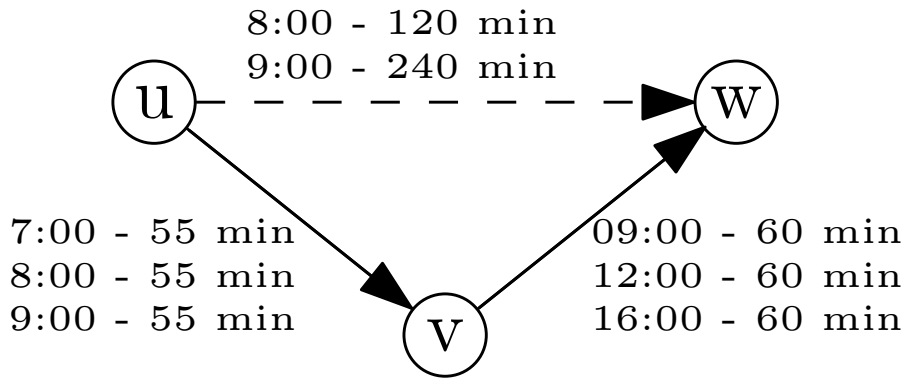
- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann



Public Transport: Link

Linken zweier Funktionen f und g

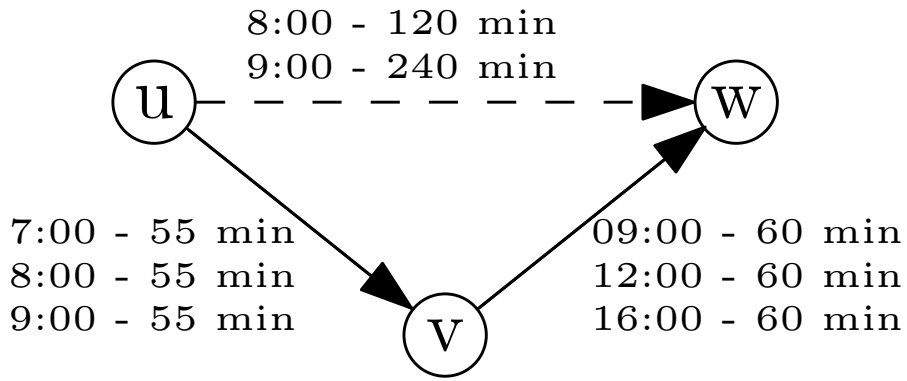
- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann
- Füge $(t_i^f, t_j^g + w_j^g - t_i^f)$ hinzu



Public Transport: Link

Linken zweier Funktionen f und g

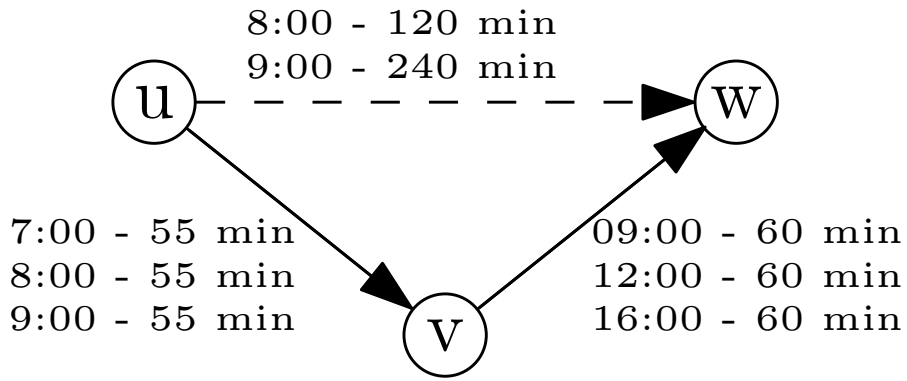
- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann
- Füge $(t_i^f, t_j^g + w_j^g - t_i^f)$ hinzu
- Wenn zwei Punkte den gleichen Verbindungspunkt haben, behalte nur den mit größerem t_i^f



Public Transport: Link

Linken zweier Funktionen f und g

- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann
- Füge $(t_i^f, t_j^g + w_j^g - t_i^f)$ hinzu
- Wenn zwei Punkte den gleichen Verbindungspunkt haben, behalte nur den mit größerem t_i^f
- Wieder Sweep-Algorithmus



Public Transport: Diskussion Link

Karlsruher Institut für Technologie

Laufzeit

- Sweep-Algorithmus
- $\mathcal{O}(|I^f| + |I^g|)$
- Zum Vergleich: Zeitunabhängig: $\mathcal{O}(1)$

Public Transport: Diskussion Link

Laufzeit

- Sweep-Algorithmus
- $\mathcal{O}(|I^f| + |I^g|)$
- Zum Vergleich: Zeitunabhängig: $\mathcal{O}(1)$

Speicherverbrauch

- Gelinkte Funktion hat $\min\{|I^f|, |I^g|\}$ Interpolationspunkte

Public Transport: Diskussion Link

Karlsruher Institut für Technologie

Laufzeit

- Sweep-Algorithmus
- $\mathcal{O}(|I^f| + |I^g|)$
- Zum Vergleich: Zeitunabhängig: $\mathcal{O}(1)$

Speicherverbrauch

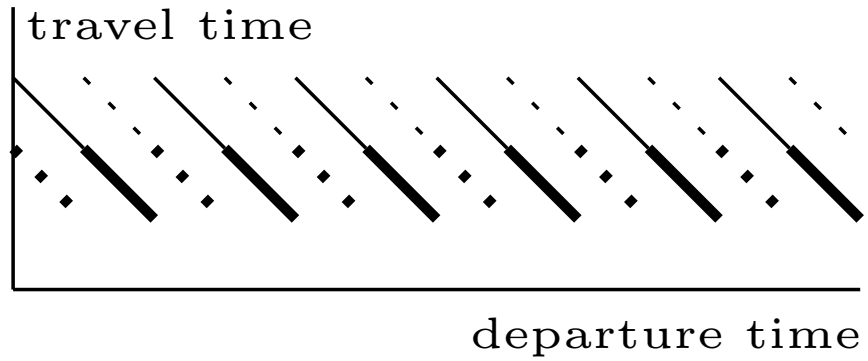
- Gelinkte Funktion hat $\min\{|I^f|, |I^g|\}$ Interpolationspunkte

Somit:

- Deutlich gutmütiger als Straßengraph-Funktionen

Public Transport: Merge

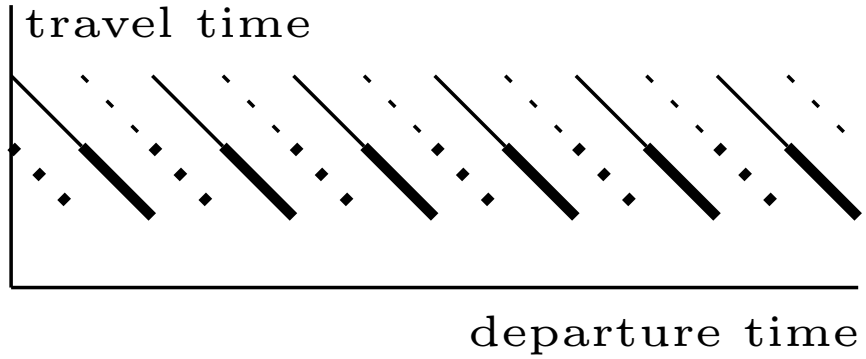
Minimum zweier Funktionen f und g



Public Transport: Merge

Minimum zweier Funktionen f und g

- Für alle (t_i^f, w_i^f): behalte Punkt, wenn $w_i^f < g(t_i^f)$
- Für alle (t_j^g, w_j^g): behalte Punkt, wenn $w_j^g < f(t_j^g)$
- Keine Schnittepunkte möglich(!)



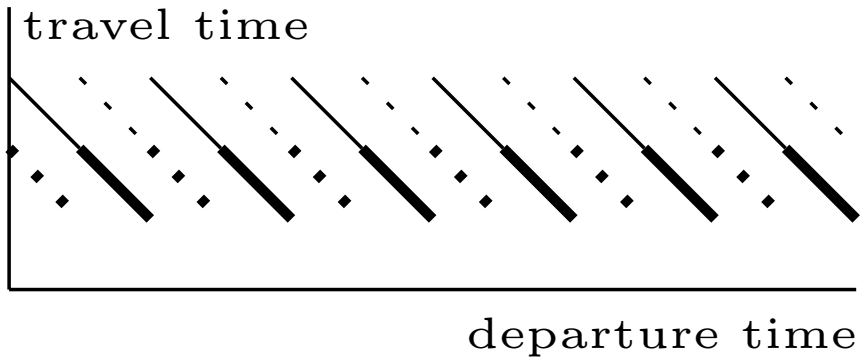
Public Transport: Merge

Minimum zweier Funktionen f und g

- Für alle (t_i^f, w_i^f): behalte Punkt, wenn $w_i^f < g(t_i^f)$
- Für alle (t_j^g, w_j^g): behalte Punkt, wenn $w_j^g < f(t_j^g)$
- Keine Schnittepunkte möglich(!)

Vorgehen:

- Linearer Sweep



Public Transport: Diskussion Merge

Laufzeit

- Sweep-Algorithmus
- $\mathcal{O}(|I^f| + |I^g|)$
- Zum Vergleich: Zeitunabhängig: $\mathcal{O}(1)$

Public Transport: Diskussion Merge

Karlsruher Institut für Technologie

Laufzeit

- Sweep-Algorithmus
- $\mathcal{O}(|I^f| + |I^g|)$
- Zum Vergleich: Zeitunabhängig: $\mathcal{O}(1)$

Speicherverbrauch

- Keine Schnittpunkte

$\Rightarrow$ Minimum-Funktion kann maximal $|I^f| + |I^g|$ Interpolationspunkte enthalten

Schiene vs. Straße

Laufzeit Operationen

- gleich für beide
- $\mathcal{O}(\log |I|)$ für Auswertung
- $\mathcal{O}(|I^f| + |I^g|)$ für Linken und Minimum

Schiene vs. Straße

Laufzeit Operationen

- gleich für beide
- $\mathcal{O}(\log |I|)$ für Auswertung
- $\mathcal{O}(|I^f| + |I^g|)$ für Linken und Minimum

Speicherverbrauch

- Public Transport deutlich geringer
- Link:

$$|I^{f \oplus g}| \leq \min\{|I^f|, |I^g|\} \quad \text{vs.} \quad |I^{f \oplus g}| \approx |I^f| + |I^g|$$

- Merge:

$$|I^{\min\{f,g\}}| \leq |I^f| + |I^g| \quad \text{vs. eventuell} \quad |I^{\min\{f,g\}}| > (|I^f| + |I^g|)$$

Schiene vs. Straße

Laufzeit Operationen

- gleich für beide
- $\mathcal{O}(\log |I|)$ für Auswertung
- $\mathcal{O}(|I^f| + |I^g|)$ für Linken und Minimum

Speicherverbrauch

- Public Transport deutlich geringer
- Link:

$$|I^{f \oplus g}| \leq \min\{|I^f|, |I^g|\} \quad \text{vs.} \quad |I^{f \oplus g}| \approx |I^f| + |I^g|$$

- Merge:

$$|I^{\min\{f,g\}}| \leq |I^f| + |I^g| \quad \text{vs. eventuell} \quad |I^{\min\{f,g\}}| > (|I^f| + |I^g|)$$

Profilsuchen

- Somit in Public Transport Netzen wahrscheinlich schneller

Zeit-Anfragen

Gegeben: Startbahnhof S, Zielbahnhof T und Abfahrtszeit τ_S

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zeit-Anfragen

Gegeben: Startbahnhof S, Zielbahnhof T und Abfahrtszeit τ_S

1. Zeitexpandiert

Startknoten:

- *Erstes* Transferevent in S mit Zeit $\tau \geq \tau_S$.

Zielknoten:

- Im Vorraus unbekannt!
- Stoppkriterium: Erster gesettleter Knoten an T induziert schnellste Verbindung zu T

2. Zeitabhängig

Startknoten:

- Bahnhofsknoten S

Zielknoten:

- Bahnhofsknoten T

Anfrage:

- Time-Dependent Dijkstra mit Zeit τ_S
- Hier: Ankunftszeit im Vorraus unbekannt

Zeit-Anfragen

Algorithm 1: `Time-Dijkstra`$(G = (V, E), s, \tau)$

```
1  d_τ[s] = 0
2  Q.clear(), Q.add(s, 0)
3  while !Q.empty() do
4      u ← Q.deleteMin()
5      for all edges e = (u, v) ∈ E do
6          if d_τ[u] + len(e, τ + d_τ[u]) < d_τ[v] then
7              d_τ[v] ← d_τ[u] + len(e, τ + d_τ[u])
8              p_τ[v] ← u
9              if v ∈ Q then Q.decreaseKey(v, d_τ[v])
10
11             else Q.insert(v, d_τ[v])
```

Diskussion Zeit-Anfragen

Beobachtung:

- Nur ein Unterschied zu Dijkstra
- Auswertung der Kanten

Diskussion Zeit-Anfragen

Beobachtung:

- Nur ein Unterschied zu Dijkstra
- Auswertung der Kanten

non-FIFO Netzwerke:

- Im Kreis fahren kann sich lohnen
- NP-schwer (wenn Warten an Knoten nicht erlaubt ist)
- Transportnetzwerke sind FIFO modellierbar (notfalls Multikanten)

Diskussion Zeit-Anfragen

Beobachtung:

- Nur ein Unterschied zu Dijkstra
- Auswertung der Kanten

non-FIFO Netzwerke:

- Im Kreis fahren kann sich lohnen
- NP-schwer (wenn Warten an Knoten nicht erlaubt ist)
- Transportnetzwerke sind FIFO modellierbar (notfalls Multikanten)

In unserem Szenario:

- Sicherstellen dass alle Routen FIFO sind.
- Für alle Trips t_i, t_j der Route muss gelten:
- t_i fährt an *jeder* Station jeweils vor t_j ab (oder andersherum).

Literatur I

Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis.
Efficient Models for Timetable Information in Public Transportation Systems.
ACM Journal of Experimental Algorithmics, 12(2.4):1–39, 2008.

Nächste Termine

Karlsruher Institut für Technologie

Mittwoch, 26.6.2013

Montag, 8.7.2013
Mittwoch, 10.7.2013