Algorithmen für Routenplanung

13. Sitzung, Sommersemester 2011

Reinhard Bauer, Thomas Pajor | 9. Juni 2011

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

Zeitabhängige Routenplanung

Zeitabhängiges Szenario

Szenario:

- Historische Daten für Verkehrssituation verfügbar
- Verkehrssituation vorhersagbar
- Berechne schnellsten Weg bezüglich der erwarteten Verkehrssituation (zu einem gegebenen Startzeitpunkt)

Beobachtung:

- Kein konzeptioneller Unterschied zu Public Transport
- Somit (eventuell) Techniken übertragbar

Herausforderung

Hauptproblem:

- Kürzester Weg hängt von Abfahrtszeitpunkt ab
- Eingabegröße steigt massiv an

Vorgehen:

- Modellierung
- Anpassung Dijkstra
- Anpassung der Basismodule für Beschleunigungstechniken

Heute:

- Modellierung und Dijkstra

Herausforderung

Hauptproblem:

- Kürzester Weg hängt von Abfahrtszeitpunkt ab
- Eingabegröße steigt massiv an

Vorgehen:

- Modellierung
- Anpassung Dijkstra
- Anpassung der Basismodule für Beschleunigungstechniken

Heute:

- Modellierung und Dijkstra

Herausforderung

Hauptproblem:

- Kürzester Weg hängt von Abfahrtszeitpunkt ab
- Eingabegröße steigt massiv an

Vorgehen:

- Modellierung
- Anpassung Dijkstra
- Anpassung der Basismodule für Beschleunigungstechniken

Heute:

- Modellierung und Dijkstra

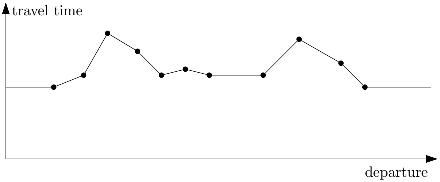
Zeitabhängige Straßennetzwerke

Eingabe:

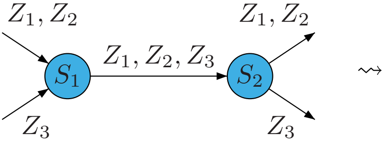
- Durchschnittliche Reisezeit zu bestimmten Zeitpunkten
- Jeden Wochentag verschieden
- Sonderfälle: Urlaubszeit

Somit an jeder Kante:

- Periodische stückweise lineare Funktion
- Definiert durch Stützpunkte
- Interpoliere linear zwischen Stützpunkten

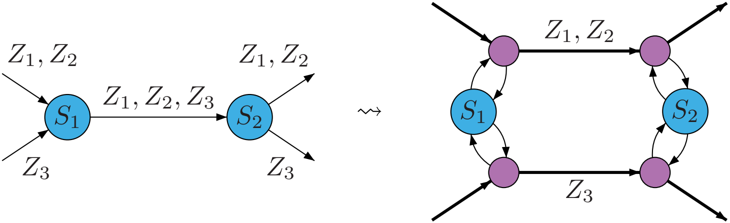


Eisenbahn-Netzwerke



- Teile Züge Z_i in Routen ein
- Für alle Routen, die eine Station bedienen: Ein extra Knoten
  - Modelliert Umstiege zwischen Zügen
  - Mindesttransferzeit für jede Station: an Stations-Kanten

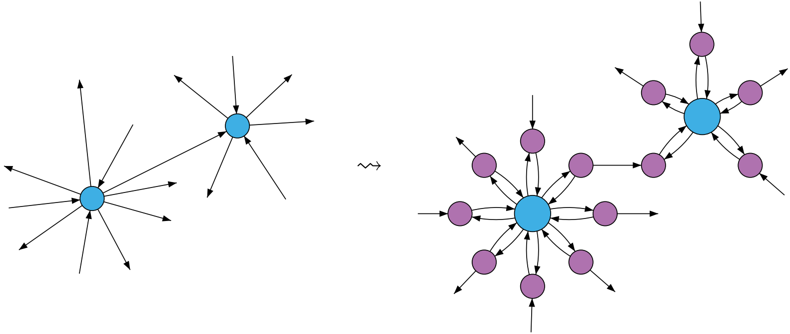
Eisenbahn-Netzwerke



- Teile Züge Z_i in Routen ein
- Für alle Routen, die eine Station bedienen: Ein extra Knoten
  - Modelliert Umstiege zwischen Zügen
  - Mindesttransferzeit für jede Station: an Stations-Kanten

Flugnetzwerke

Nutzung des Eisenbahnansatzes nicht sinnvoll

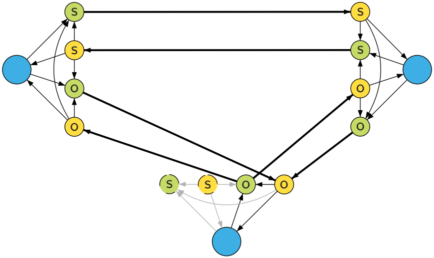


Graphen werden zu groß.

Flugnetzwerke

Modellierung:

- Check-in,
- Check-out,
- Transfer,
- zwischen Allianzen.

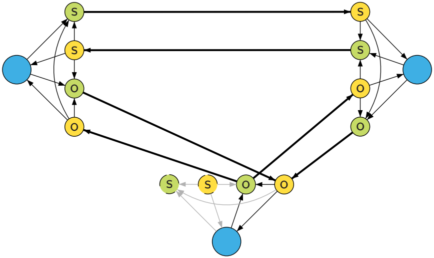


⇒ Zwei Knoten pro Allianz (Abflug und Ankunftsknoten)

Flugnetzwerke

Modellierung:

- Check-in,
- Check-out,
- Transfer,
- zwischen Allianzen.



⇒ Zwei Knoten pro Allianz (Abflug und Ankunftsknoten)

Public Transport Funktionen

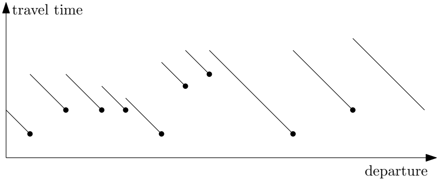
Karlsruher Institut für Technologie

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

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.

Diskussion

Eigenschaften:

- Topologie ändert sich nicht
- Kanten gemischt zeitabhängig und konstant
- variable (!) Anzahl Interpolationspunkte pro Kante

Beobachtungen:

- FIFO gilt auf allen Kanten
- später wichtig

Diskussion

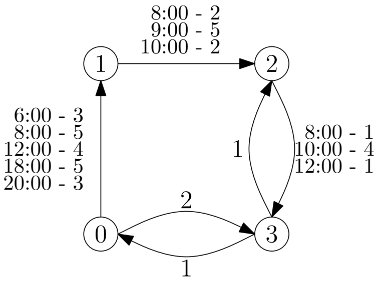
Eigenschaften:

- Topologie ändert sich nicht
- Kanten gemischt zeitabhängig und konstant
- variable (!) Anzahl Interpolationspunkte pro Kante

Beobachtungen:

- FIFO gilt auf allen Kanten
- später wichtig

Datenstruktur

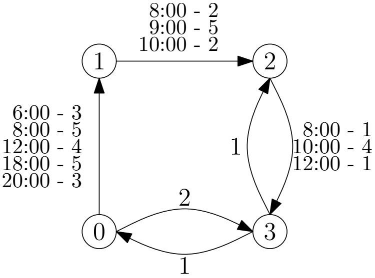


Datenstruktur

firstEdge					
firstEdge	0	2	3	4	6

targetNode	1	3	2	3	2	0	–
firstPoint	0	5	6	9	12	13	14

time	6	8	12	18	20	0	8	9	10	8	10	12	0	0
weight	3	5	4	5	3	2	2	5	2	1	4	1	1	1



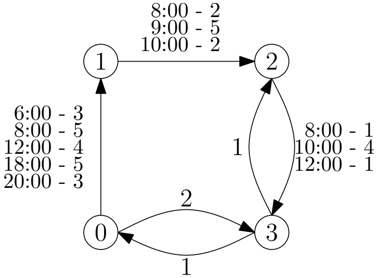
Datenstruktur

sourceNode	3	0	1	3	0	2
lowerBound	1	3	2	1	2	1

firstInEdge	0	1	2	4	6
firstOutEdge	0	2	3	4	6

targetNode	1	3	2	3	2	0	–
firstPoint	0	5	6	9	12	13	14

time	6	8	12	18	20	0	8	9	10	8	10	12	0	0
weight	3	5	4	5	3	2	2	5	2	1	4	1	1	1



Anfrageszenarien

Zeit-Anfrage:

- finde kürzesten Weg für Abfahrtszeit τ
- analog zu Dijkstra?

Profil-Anfrage:

- finde kürzesten Weg für alle Abfahrtszeitpunkte
- analog zu Dijkstra?

Anfrageszenarien

Zeit-Anfrage:

- finde kürzesten Weg für Abfahrtszeit τ
- analog zu Dijkstra?

Profil-Anfrage:

- finde kürzesten Weg für alle Abfahrtszeitpunkte
- analog zu Dijkstra?

Zeit-Anfragen

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

$d_\tau[s] = 0$
$Q.clear()$, $Q.add(s, 0)$

while $!Q.empty()$ **do**
    $u \leftarrow Q.deleteMin()$
    for *all edges* $e = (u, v) \in E$ **do**
        if $d_\tau[u] + \mathrm{len}(e, \tau + d_\tau[u]) < d_\tau[v]$ **then**
            $d_\tau[v] \leftarrow d_\tau[u] + \mathrm{len}(e, \tau + d_\tau[u])$
            $p_\tau[v] \leftarrow u$
            if $v \in Q$ **then** $Q.decreaseKey(v, d_\tau[v])$
            else $Q.insert(v, d_\tau[v])$

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)

Profil-Anfragen

`Profile-Search`$(G = (V, E), s)$

$d_*[s] = 0$
$Q.clear()$, $Q.add(s, 0)$
while $!Q.empty()$ **do**
    $u \leftarrow Q.deleteMin()$
    for *all edges* $e = (u, v) \in E$ **do**
        if $d_*[u] \oplus \text{len}(e) \not\geq d_*[v]$ **then**
            $d_*[v] \leftarrow \min(d_*[u] \oplus \text{len}(e), d_*[v])$
            if $v \in Q$ **then** $Q.decreaseKey(v, \underline{d}[v])$
            else $Q.insert(v, \underline{d}[v])$

Diskussion Profile-Anfragen

Karlsruher Institut für Technologie

Beobachtungen:

- Operationen auf Funktionen
- Priorität im Prinzip frei wählbar
  ($\underline{d}[u]$ ist das Minimum der Funktion $d_*[u]$)
- Knoten können mehrfach besucht werden $\Rightarrow$ label-correcting

Herausforderungen:

- Wie effizient $\oplus$ berechnen (Linken)?
- Wie effizient Minimum bilden?

Operationen

Karlsruher Institut für Technologie

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:

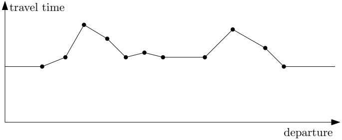
- Unterschiedlich für Straße und Schiene

Straße: Auswertung

Evaluation von $f(\tau)$:

- Suche Punkte mit $t_i \leq \tau$ und $t_{i+1} \geq \tau$
- dann Evaluation durch

$$f(\tau) = w_i + (\tau - t_i) \cdot \frac{w_{i+1} - w_i}{t_{i+1} - t_i}$$



Problem:

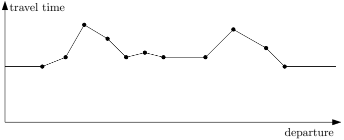
- Finden von t_i und t_{i+1}
- Theoretisch:
  - Lineare Suche: $\mathcal{O}(|I|)$
  - Binäre Suche: $\mathcal{O}(\log_2 |I|)$
- Praktisch:
  - $|I| < 30 \Rightarrow$ lineare Suche
  - Sonst: Lineare Suche mit Startpunkt $\frac{\tau}{\Pi} \cdot |I|$
    wobei Π die Periodendauer ist

Straße: Auswertung

Evaluation von $f(\tau)$:

- Suche Punkte mit $t_i \leq \tau$ und $t_{i+1} \geq \tau$
- dann Evaluation durch

$$f(\tau) = w_i + (\tau - t_i) \cdot \frac{w_{i+1} - w_i}{t_{i+1} - t_i}$$



Problem:

- Finden von t_i und t_{i+1}
- Theoretisch:
  - Lineare Suche: $\mathcal{O}(|I|)$
  - Binäre Suche: $\mathcal{O}(\log_2 |I|)$
- Praktisch:
  - $|I| < 30 \Rightarrow$ lineare Suche
  - Sonst: Lineare Suche mit Startpunkt $\frac{\tau}{\Pi} \cdot |I|$
    wobei Π die Periodendauer ist

Linken

KIT
Karlsruher Institut für Technologie

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

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

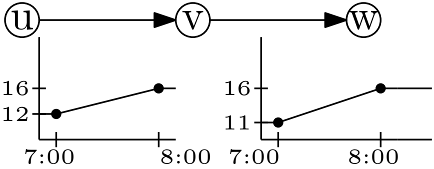
Oder

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

Straße: Link

Linken zweier Funktionen f und g

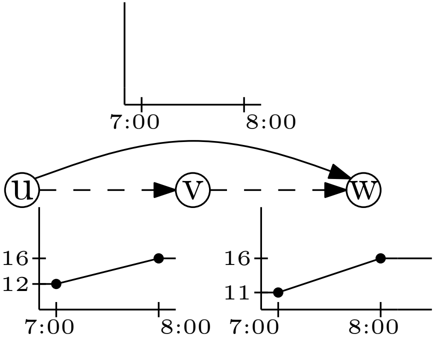
- $f \oplus g$ enthält auf jeden Fall $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_I^f, w_I^f + g(t_I^f + w_I^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Straße: Link

Linken zweier Funktionen f und g

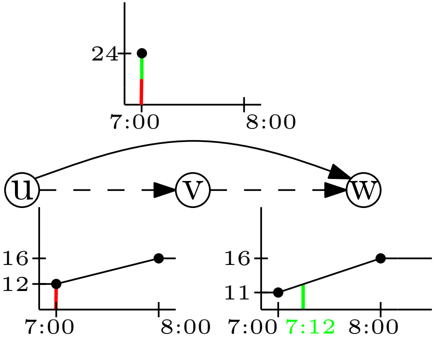
- $f \oplus g$ enthält auf jeden Fall $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_I^f, w_I^f + g(t_I^f + w_I^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Straße: Link

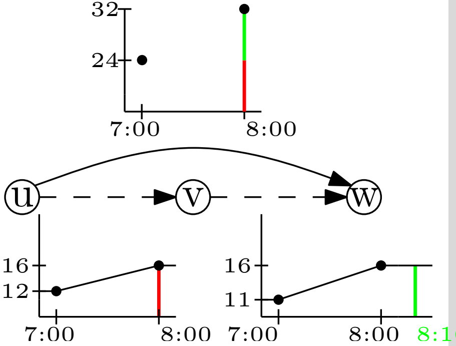
Linken zweier Funktionen f und g

- $f \oplus g$ enthält auf jeden Fall
  $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_l^f, w_l^f + g(t_l^f + w_l^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Linken zweier Funktionen f und g

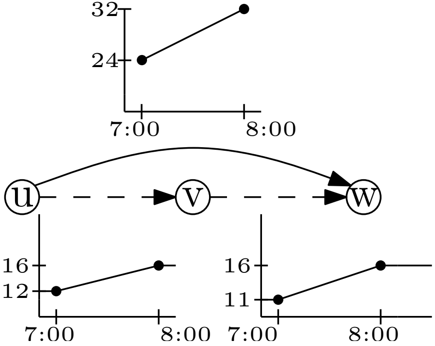
- $f \oplus g$ enthält auf jeden Fall
  $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_l^f, w_l^f + g(t_l^f + w_l^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Straße: Link

Linken zweier Funktionen f und g

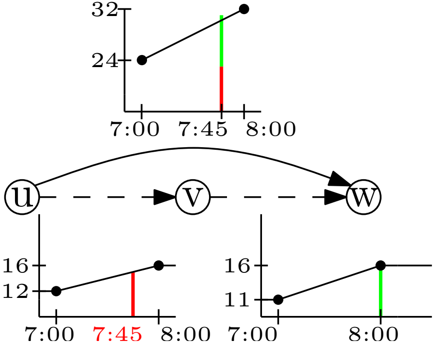
- $f \oplus g$ enthält auf jeden Fall
  $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_l^f, w_l^f + g(t_l^f + w_l^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Straße: Link

Linken zweier Funktionen f und g

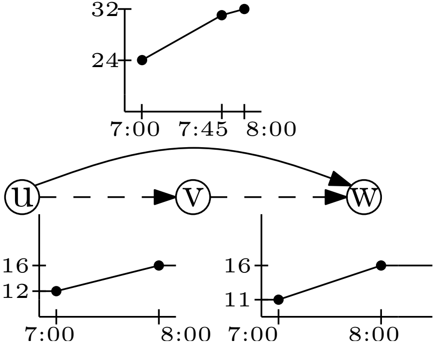
- $f \oplus g$ enthält auf jeden Fall $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_l^f, w_l^f + g(t_l^f + w_l^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Straße: Link

Linken zweier Funktionen f und g

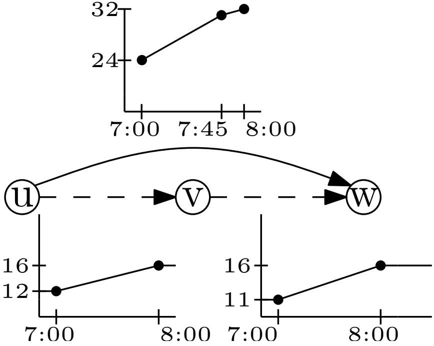
- $f \oplus g$ enthält auf jeden Fall $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_l^f, w_l^f + g(t_l^f + w_l^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Straße: Link

Linken zweier Funktionen f und g

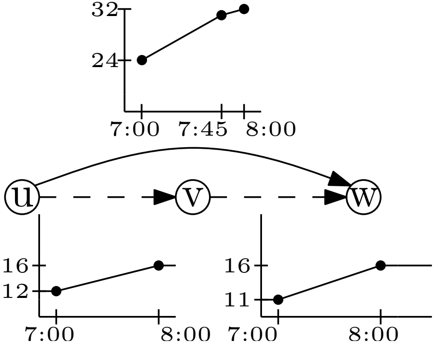
- $f \oplus g$ enthält auf jeden Fall $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_I^f, w_I^f + g(t_I^f + w_I^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Straße: Link

Linken zweier Funktionen f und g

- $f \oplus g$ enthält auf jeden Fall $\{(t_1^f, w_1^f + g(t_1^f + w_1^f)), \ldots, (t_l^f, w_l^f + g(t_l^f + w_l^f))\}$
- Zusätzliche Interpolationspunkte an t_j^{-1} mit $f(t_j^{-1}) + t_j^{-1} = t_j^g$
- Füge $(t_j^{-1}, f(t_j^{-1}) + w_j^g)$ für alle Punkte von g zu $f \oplus g$
- Durch linearen Sweeping-Algorithmus implementierbar



Straße: Diskussion Link

Laufzeit

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

Speicherverbrauch

- Gelinkte Funktion hat $\approx |I^f| + |I^g|$ Interpolationspunkte

Problem:

- Während Profilsuche kann ein Pfad mehreren Tausend Kanten entsprechen...
- Shortcuts...
- es kommt noch schlimmer...

Straße: Diskussion Link

Laufzeit

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

Speicherverbrauch

- Gelinkte Funktion hat $\approx |I^f| + |I^g|$ Interpolationspunkte

Problem:

- Während Profilsuche kann ein Pfad mehreren Tausend Kanten entsprechen…
- Shortcuts…
- es kommt noch schlimmer…

Straße: Diskussion Link

Laufzeit

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

Speicherverbrauch

- Gelinkte Funktion hat $\approx |I^f| + |I^g|$ Interpolationspunkte

Problem:

- Während Profilsuche kann ein Pfad mehreren Tausend Kanten entsprechen...
- Shortcuts...
- es kommt noch schlimmer...

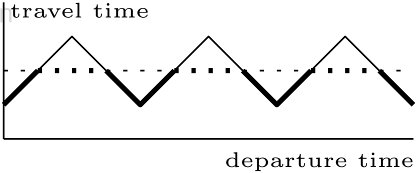
Straße: Merge

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 $w_j^g < f(t_j^g)$
- Schnittpunkte müssen ebenfalls eingefügt werden

Vorgehen:

- Linearer sweep
- Evaluiere, welcher Abschnitt oben
- Checke ob Schnittpunkt existiert



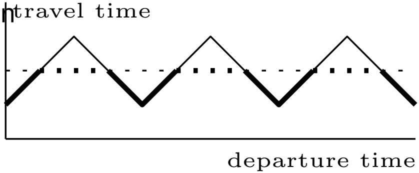
Straße: Merge

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 $w_j^g < f(t_j^g)$
- Schnittpunkte müssen ebenfalls eingefügt werden

Vorgehen:

- Linearer sweep
- Evaluiere, welcher Abschnitt oben
- Checke ob Schnittpunkt existiert



Straße: Diskussion Merge

Laufzeit

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

Speicherverbrauch

- Minimum-Funktion kann mehr als $|I^f| + |I^g|$ Interpolationspunkte enthalten

Problem:

- Während Profilsuche werden Funktionen gemergt
- Laufzeit der Profilsuchen wird durch diese Operationen dominiert

Straße: Diskussion Merge

Laufzeit

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

Speicherverbrauch

- Minimum-Funktion kann mehr als $|I^f| + |I^g|$ Interpolationspunkte enthalten

Problem:

- Während Profilsuche werden Funktionen gemergt
- Laufzeit der Profilsuchen wird durch diese Operationen dominiert

Straße: Diskussion Merge

Laufzeit

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

Speicherverbrauch

- Minimum-Funktion kann mehr als $|I^f| + |I^g|$ Interpolationspunkte enthalten

Problem:

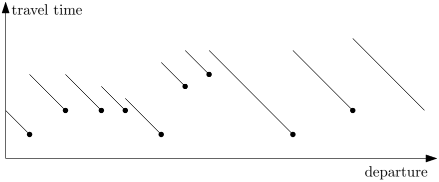
- Während Profilsuche werden Funktionen gemergt
- Laufzeit der Profilsuchen wird durch diese Operationen dominiert

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:

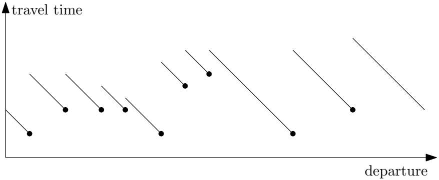
- Finden von t_i und t_{i+1}
- Theoretisch:
  - Lineare Suche: $\mathcal{O}(|I|)$
  - Binäre Suche: $\mathcal{O}(\log_2 |I|)$
- praktisch:
  - $|I| < 30$: Lineare Suche
  - Sonst: Lineare Suche mit Startpunkt $\frac{\tau}{\Pi} \cdot |I|$

Public Transport: Auswertung

Evaluation von $f(\tau)$:

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

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



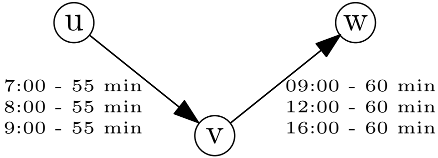
Problem:

- 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 $\frac{\tau}{\Pi} \cdot |I|$

Public Transport: Link

Linken zweier Funktionen f und g

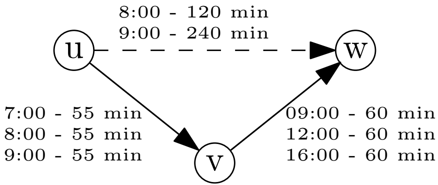
- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann
- Füge $(t_i^f, t_j^g + w_j^g - t_i^f)$ hinzu
- Wenn zwei Punkte den gleichen Verbindungspunkt haben, behalte nur den mit größerem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



Public Transport: Link

Linken zweier Funktionen f und g

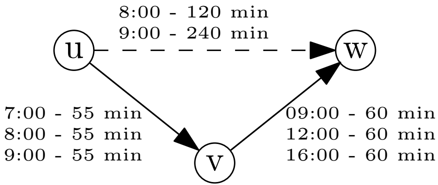
- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann
- Füge $(t_i^f, t_j^g + w_j^g - t_i^f)$ hinzu
- Wenn zwei Punkte den gleichen Verbindungspunkt haben, behalte nur den mit größerem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



Public Transport: Link

Linken zweier Funktionen f und g

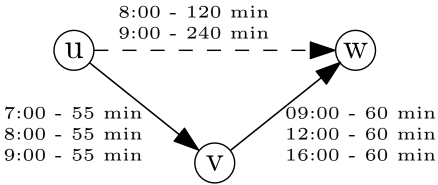
- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann
- Füge $(t_i^f, t_j^g + w_j^g - t_i^f)$ hinzu
- Wenn zwei Punkte den gleichen Verbindungspunkt haben, behalte nur den mit größerem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



Public Transport: Link

Linken zweier Funktionen f und g

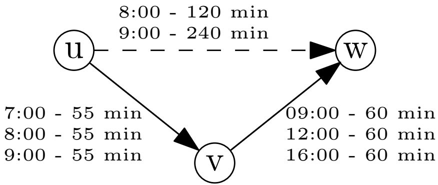
- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann
- Füge $(t_i^f, t_j^g + w_j^g - t_i^f)$ hinzu
- Wenn zwei Punkte den gleichen Verbindungspunkt haben, behalte nur den mit größerem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



Public Transport: Link

Linken zweier Funktionen f und g

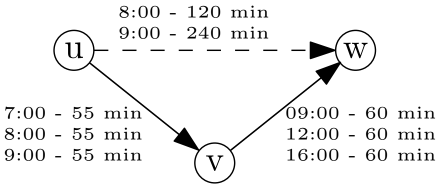
- Für jeden Punkt (t_i^f, w_i^f) bestimme den Verbindungspunkt (t_j^g, w_j^g) mit $t_j^g - t_i^f - w_i^f \geq 0$ minimal
- Erste Verbindung, die man auf g erreichen kann
- Füge $(t_i^f, t_j^g + w_j^g - t_i^f)$ hinzu
- Wenn zwei Punkte den gleichen Verbindungspunkt haben, behalte nur den mit größerem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



Public Transport: 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

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: 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

Somit:

- Deutlich gutmütiger als Straßengraph-Funktionen

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

Somit:

- Deutlich gutmütiger als Straßengraph-Funktionen

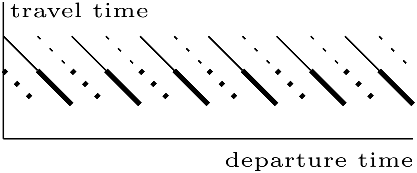
Public Transport: Merge

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



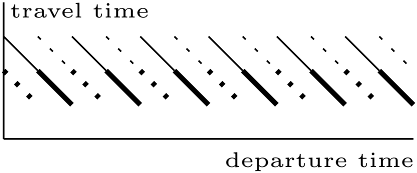
Public Transport: Merge

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



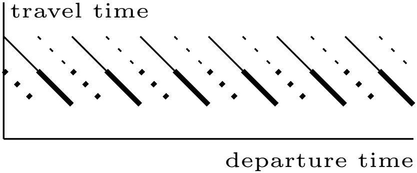
Public Transport: Merge

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

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

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

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

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

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

Eingabe

Straße:

- Netzwerk Deutschland $|V| \approx 4.7$ Mio., $|E| \approx 10.8$ Mio.
- 5 Verkehrszenarien:
  - Montag: $\approx 8\%$ Kanten zeitabhängig
  - Dienstag - Donnerstag: $\approx 8\%$
  - Freitag: $\approx 7\%$
  - Samstag: $\approx 5\%$
  - Sonntag: $\approx 3\%$

Schiene:

- Europa Fernverbindungen
- 30 156 Stationen, 1.8 Millionen Verbindungen
- $|V| = 0.4$ Mio., $|E| = 1.4$ Mio.

"Grad" der Zeitabhängigkeit

	#delete mins	slow-down	time [ms]	slow-down
kein	2,239,500	0.00%	1219.4	0.00%
Montag	2,377,830	6.18%	1553.5	27.40%
DiDo	2,305,440	2.94%	1502.9	23.25%
Freitag	2,340,360	4.50%	1517.2	24.42%
Samstag	2,329,250	4.01%	1470.4	20.59%
Sonntag	2,348,470	4.87%	1464.4	20.09%

Beobachtung:

- kaum Veränderung in Suchraum
- Anfragen etwas langsamer durch Auswertung

Profilsuchen Straße

Karlsruher Institut für Technologie

☹

Beobachtung:

- Nicht durchführbar durch zu großen Speicherbedarf ($>$ 32 GiB RAM)
- Interpoliert:
  - Suchraum steigt um ca. 10%
  - Suchzeiten um einen Faktor von bis zu 2 500

$\Rightarrow$ inpraktikabel

Profilsuchen Schiene

	#delete mins	time [ms]
Zeit-Anfragen	260 095	125
Profil-Anfragen	1 919 662	5 327

Beobachtung:

- Deletemins steigen an (ungefähr Faktor 8)
- Queryzeit steigt an um Faktor 42
- Verlust für Operationen ist ca. 5

Zusammenfassung

Zeitabhängige Netzwerke (Basics)

- Funktionen statt Konstanten an Kanten
- Operationen werden teurer
  - $\mathcal{O}(\log |I|)$ für Auswertung
  - $\mathcal{O}(|I^f| + |I^g|)$ für Linken und Minimum
  - Straßennetzwerke: Speicherverbrauch explodiert
  - Eisenbahn: gutartiger
- Zeitanfragen:
  - Normaler Dijkstra
  - Kaum langsamer (lediglich Auswertung)
- Profilanfragen
  - In Public Transportation gut nutzbar
  - Straßennetzwerke nicht zu handhaben

Ende

Literatur (Zeitabhängige Routenplanung):

- Daniel Delling:
  Enginering and Augmenting Route Planning Algorithms
  Ph.D. Thesis, Universität Karlsruhe (TH), 2009.