Karlsruher Institut für Technologie

Algorithmen für Routenplanung

21. Vorlesung, Sommersemester 2019

Jonas Sauer | 22. Juli 2019

INSTITUT FÜR THEORETISCHE INFORMATIK · ALGORITHMIK · PROF. DR. DOROTHEA WAGNER

Inhalt heute

Karlsruhe Institute of Technology

Unbeschränktes Laufen:

- Limitierungen von transitiv abgeschlossenen Graphen
- Auswirkung von vollständigen Fußweggraphen

Verkehrsumlegung:

- Fahrzeugauslastungen bei statistischem Reiseaufkommen

Unbeschränktes Laufen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Fußwegegraphen

Bisher:

- Effiziente Algorithmen
- Eingeschränkte Modelle für Laufen

Einschränkungen:

RAPTOR	Transitiv abgeschlossen
CSA	Transitiv abgeschlossen
Trip-Based Routing	Transitiv abgeschlossen
Transfer Patterns	Max. 400 m
PTL	Nur Transfers aus ÖV-Netzwerk
MEAT	Kein Laufen möglich

Fußwegegraphen

Karlsruhe Institute of Technology

Argumente für Einschränkung:

- „Laufen lohnt sich nie“
- „Laufen vom Nutzer unerwünscht“
- „Laufen in der Praxis nicht relevant“

Aber:

- Relevanz nie bewiesen/widerlegt
- Algorithmus sollte über Relevanz entscheiden

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transitiver Abschluss

Frage: Bis zu welcher Fußweglänge ist transitiver Abschluss praktikabel?

Ansatz:

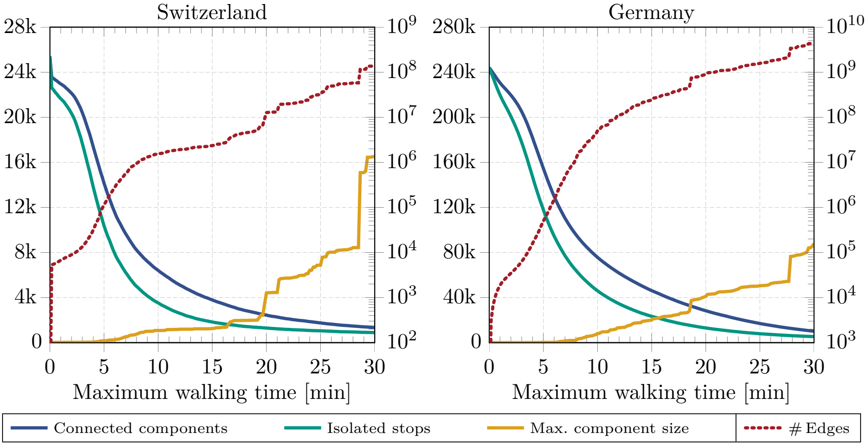
- Starte mit vollständigem Fußweggraphen
- Wähle maximale Laufdauer τ
- Verbinde Stops $\Leftrightarrow$ Laufdistanz $\leq \tau$
- Bilde transitiven Abschluss des resultierenden Graphen

- Wie groß wird dieser Graph?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transitiver Abschluss

Größe des transitiven Abschlusses:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Fazit

Beobachtung:

- Transitiver Abschluss nur für kurze Laufwege praktikabel
- Somit bestehende Algorithmen nicht anwendbar

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Fazit

Beobachtung:

- Transitiver Abschluss nur für kurze Laufwege praktikabel
- Somit bestehende Algorithmen nicht anwendbar

Frage: Machen längere Laufwege in der Praxis einen Unterschied?

Probleme:

- Hängt von der Tageszeit ab (Tag vs. Nacht)
- Hängt von Start und Ziel ab (ländlich vs. städtisch)
- Hängt von der Distanz zwischen Start und Ziel ab (lang vs. kurz)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Fazit

Beobachtung:

- Transitiver Abschluss nur für kurze Laufwege praktikabel
- Somit bestehende Algorithmen nicht anwendbar

Frage: Machen längere Laufwege in der Praxis einen Unterschied?

Probleme:

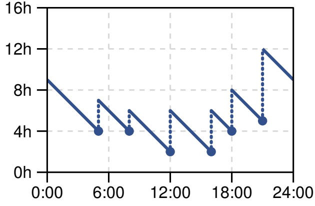
- Hängt von der Tageszeit ab (Tag vs. Nacht)
- Hängt von Start und Ziel ab (ländlich vs. städtisch)
- Hängt von der Distanz zwischen Start und Ziel ab (lang vs. kurz)

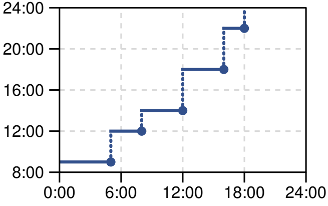
$\Rightarrow$ Auswertung von Reisezeitprofilen
auf Instanzen mit unterschiedlich vielen Fußwegen
für viele s–t-Paare

Profile & Profil-Algorithmen

Wiederholung Profil:

- Funktion, die Abfahrtszeit auf Reisezeit oder Ankunftszeit abbildet





Profil-Algorithmen für unbeschränktes Laufen:

- Die meisten Public-Transit-Algorithmen sind nicht geeignet
- Ausnahme: Multimodal Multicriteria RAPTOR (MCR)
  - Kann mit beliebigem Laufen umgehen
  - Unterstützt aber nur Earliest-Arrival-Queries

Profile – RAPTOR & MCR

MCR Profil-Algorithmus:

- MCR basiert auf dem RAPTOR-Algorithmus
- RAPTOR kann für Profilberechnung angepasst werden (rRAPTOR)
- Warum ist dies nicht auf MCR übertragbar?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Profile – RAPTOR & MCR

MCR Profil-Algorithmus:

- MCR basiert auf dem RAPTOR-Algorithmus
- RAPTOR kann für Profilberechnung angepasst werden (rRAPTOR)
- Warum ist dies nicht auf MCR übertragbar?

rRAPTOR Profil-Algorithmus: (Wiederholung)

- Profileinträge sind durch Abfahrten am Start begrenzt
- Sammle alle Abfahrtszeiten am Start ein
- Führe RAPTOR für jede Abfahrtszeit einmal aus

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Profile – RAPTOR & MCR

MCR Profil-Algorithmus:

- MCR basiert auf dem RAPTOR-Algorithmus
- RAPTOR kann für Profilberechnung angepasst werden (rRAPTOR)
- Warum ist dies nicht auf MCR übertragbar?

rRAPTOR Profil-Algorithmus: (Wiederholung)

- Profileinträge sind durch Abfahrten am Start begrenzt
- Sammle alle Abfahrtszeiten am Start ein
- Führe RAPTOR für jede Abfahrtszeit einmal aus

Probleme mit unbeschränktem Laufen/MCR:

- Beliebiges Laufen vor der ersten Fahrt möglich
- Jede Abfahrt im Netzwerk kann die erste genutzte sein

$\Rightarrow$ Zu viele RAPTOR-Anfragen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternating RAPTOR (aRAPTOR)

Ziel:

- Nutze Earliest-Arrival-Algorithmus (MCR) als Black Box
- Earliest-Arrival-Algorithmus berechnet einzelne Profileinträge
- Anzahl Ausführungen $\propto$ Anzahl Profileinträge

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternating RAPTOR (aRAPTOR)

Ziel:

- Nutze Earliest-Arrival-Algorithmus (MCR) als Black Box
- Earliest-Arrival-Algorithmus berechnet einzelne Profileinträge
- Anzahl Ausführungen $\propto$ Anzahl Profileinträge

Vorgehen:

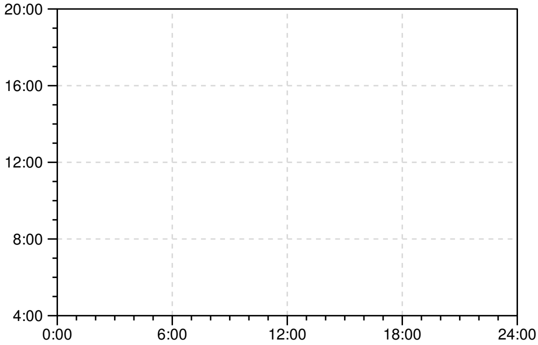
- Baue das Profil Eintrag um Eintrag auf
- Pro Profileintrag werden zwei Black-Box-Aufrufe genutzt:
  - Starte mit frühestmöglicher Abfahrtszeit
  - Vorwärtssuche $\Rightarrow$ Frühestmögliche Ankunftszeit
  - Rückwärtssuche $\Rightarrow$ Spätestmögliche Abfahrtszeit
  - Fahre mit der spätestmöglichen Abfahrtszeit + ε fort

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall [0:00, 24:00]



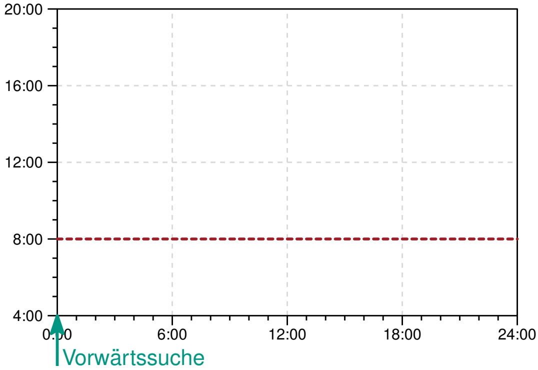
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall [0:00, 24:00]

Schritt 1: Vorwärtssuche für frühestmögliche Abfahrtszeit (0:00)

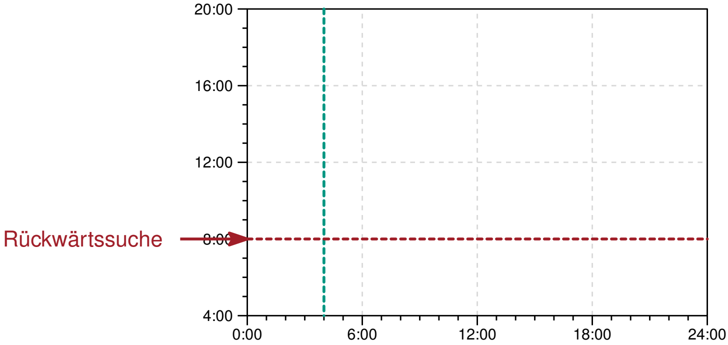


aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall [0:00, 24:00]

Schritt 2: Rückwärtssuche für die resultierende Ankunftszeit (8:00)

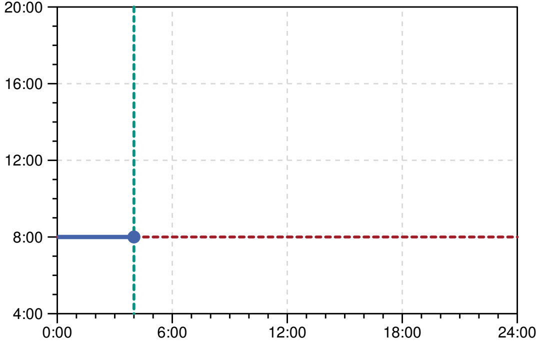


aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall [0:00, 24:00]

Schritt 3: Beide Ergebnisse zusammen bilden einen Profileintrag



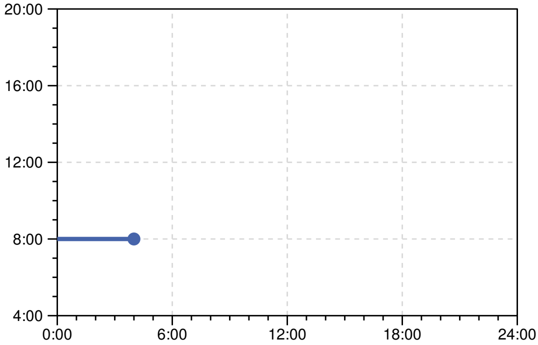
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall [0:00, 24:00]

Schritt 4: Das Profil ist schon korrekt für das Intervall [0:00, 4:00]

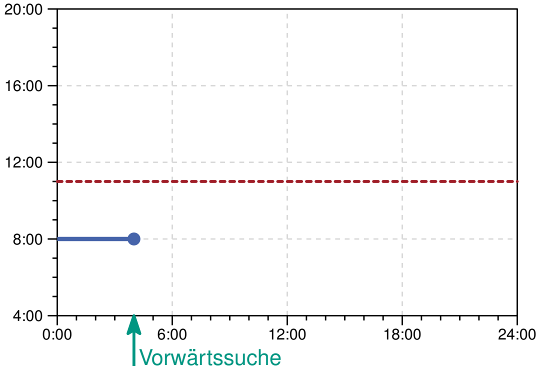


aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 1: Vorwärtssuche für frühestmögliche Abfahrtszeit (4:00 + ε)



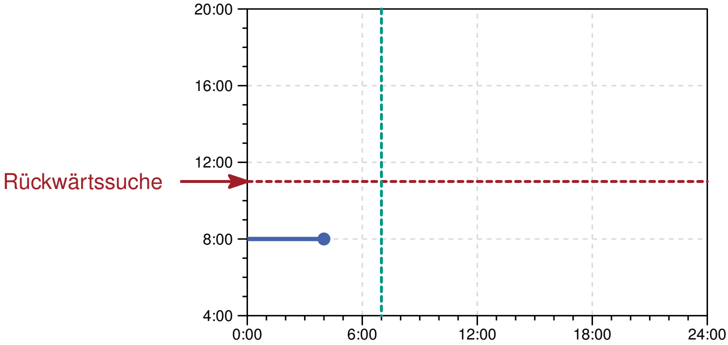
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 2: Rückwärtssuche für die resultierende Ankunftszeit (11:00)



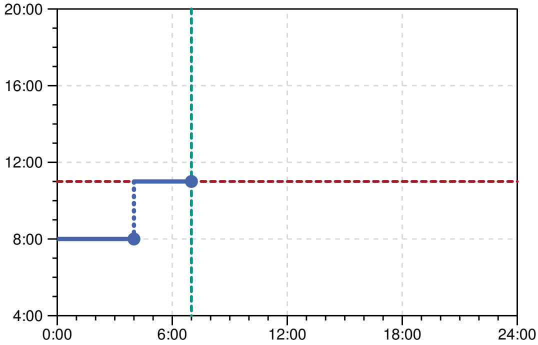
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 3: Beide Ergebnisse zusammen bilden einen Profileintrag



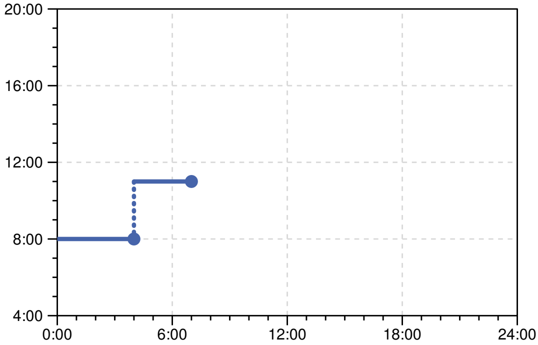
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 4: Das Profil ist schon korrekt für das Intervall [0:00, 7:00]



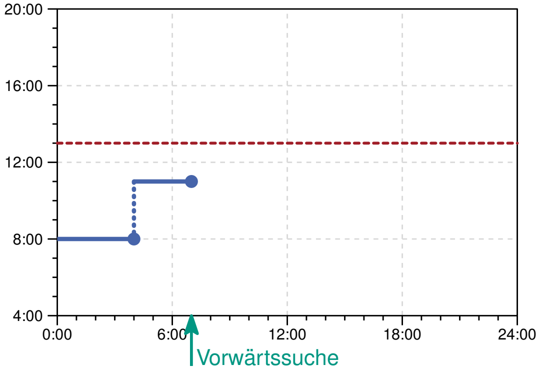
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall (7:00, 24:00]

Schritt 1: Vorwärtssuche für frühestmögliche Abfahrtszeit (7:00 + ε)



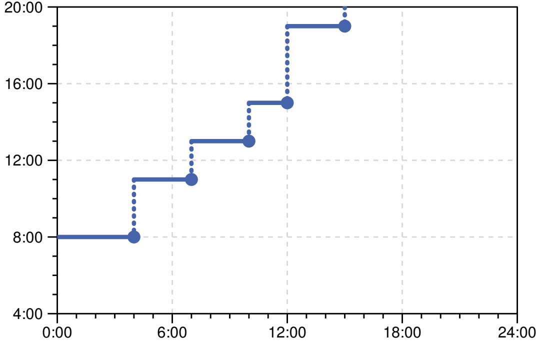
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Beispiel

Ziel:

- Berechne ein Profil für das Intervall (7:00, 24:00]

Ende: Der Algorithmus endet, wenn das gesamte Profil berechnet wurde



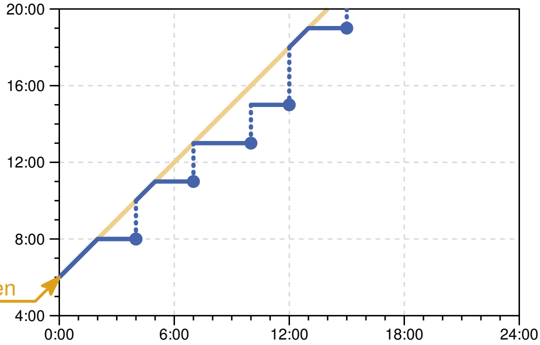
aRAPTOR – Unbeschränktes Laufen

Problem:

- Unbeschränktes Laufen verhindert Fortschritt

Beispiel:

Direktes Laufen dauert 6 Stunden



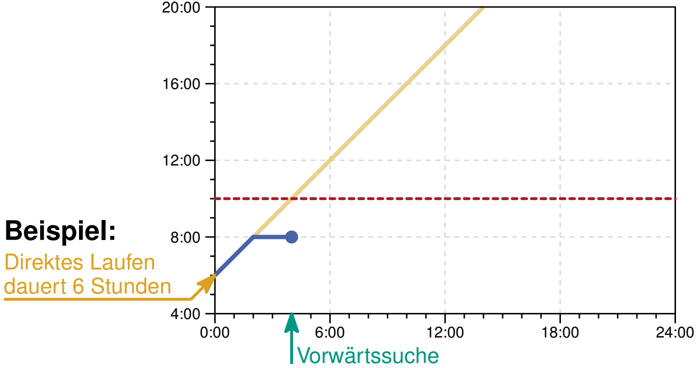
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Unbeschränktes Laufen

Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 1: Vorwärtssuche für frühestmögliche Abfahrtszeit (4:00 + ε)

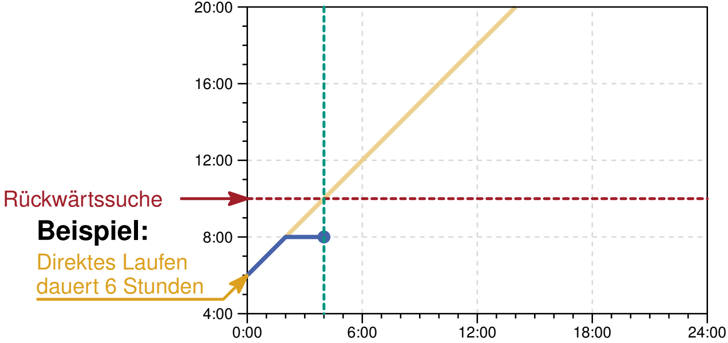


aRAPTOR – Unbeschränktes Laufen

Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 2: Rückwärtssuche für die resultierende Ankunftszeit (10:00)

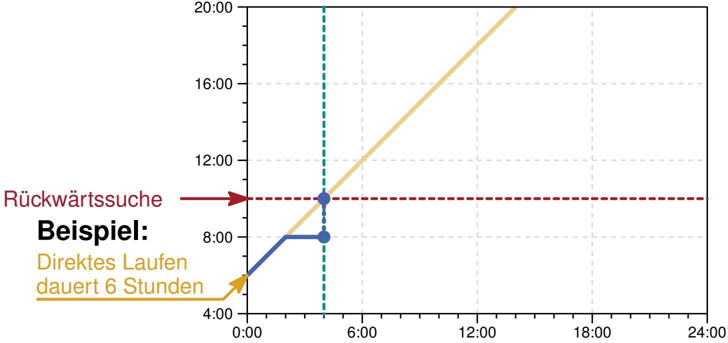


aRAPTOR – Unbeschränktes Laufen

Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 3: Beide Ergebnisse zusammen bilden einen Profileintrag



aRAPTOR – Unbeschränktes Laufen

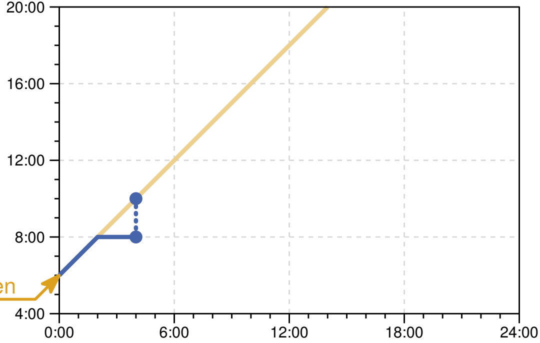
Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 4: Das Profil ist immer noch korrekt für das Intervall [0:00, 4:00]

Beispiel:

Direktes Laufen dauert 6 Stunden



aRAPTOR – Unbeschränktes Laufen

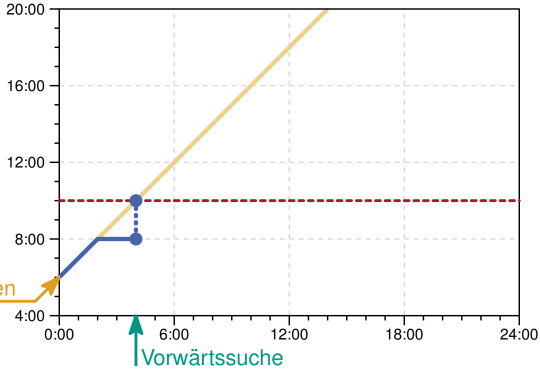
Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Schritt 1: Vorwärtssuche für frühestmögliche Abfahrtszeit (4:00 + ε)

Beispiel:

Direktes Laufen dauert 6 Stunden



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

aRAPTOR – Unbeschränktes Laufen

Karlsruhe Institute of Technology

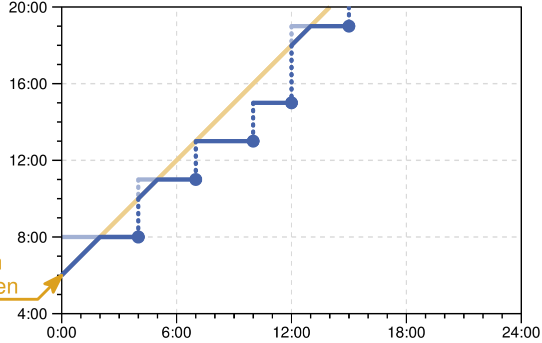
Ziel:

- Berechne ein Profil für das Intervall (4:00, 24:00]

Lösung: Berechne Profil ohne direktes Laufen,
füge den direkten Laufweg danach hinzu

Beispiel:

Direktes Laufen
dauert 6 Stunden



Auswirkung des Fußwegegraphen

Frage: Machen längere Laufwege in der Praxis einen Unterschied?

Ansatz:

- Vergleiche Reisezeiten in typischen Netzwerken
  - **original:** Nur Fußwege, die im Public-Transit-Netzwerk spezifiziert sind
  - **partial:** Transitiv abgeschlossener Fußwegegraph
  - **complete:** Vollständiger Fußwegegraph

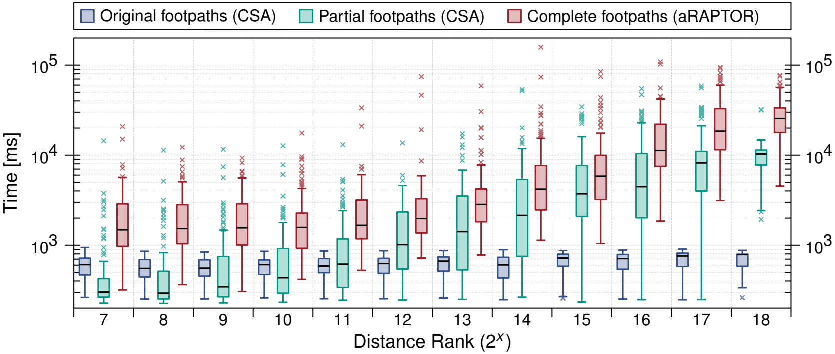
PT network	Footpaths	Stops	Vertices	Edges	Connections	Max. deg.
Switzerland	original	25 427	25 427	5 604	4 373 268	25
	partial	25 427	25 427	3 104 974	4 373 268	1 246
	complete	25 427	604 230	1 844 286	4 373 268	25
Germany	original	244 245	244 245	95 036	46 119 896	18
	partial	244 245	244 245	26 193 136	46 119 896	2 622
	complete	244 245	6 876 758	21 382 408	46 119 896	21

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente

Vergleich der Anfragezeiten:

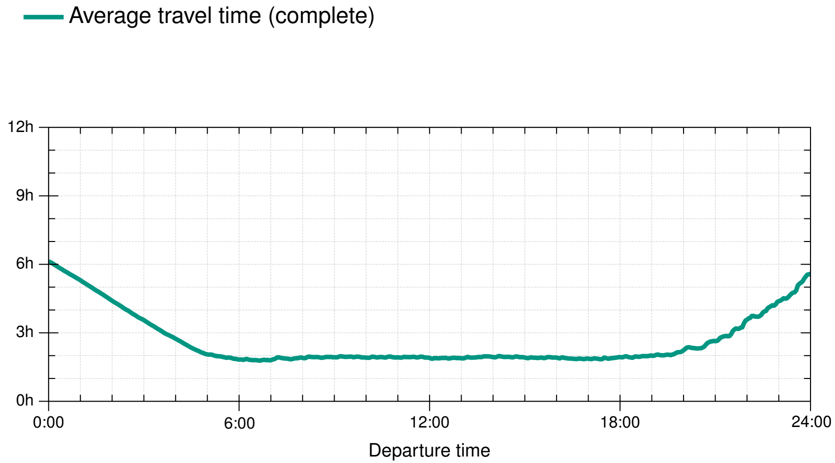
- Auf dem Schweizer Netzwerk
- Für Pareto-Profile (Reisezeit, Anzahl Umstiege)
- CSA berücksichtigt nur original-/partial-Fußwege
- aRAPTOR berücksichtigt complete-Fußwege



Experimente

Vergleich der Reisezeiten:

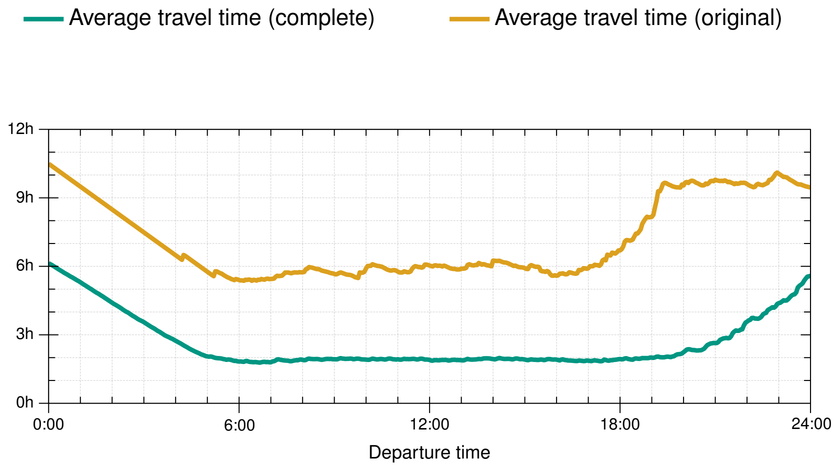
- Switzerland complete vs. Switzerland original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

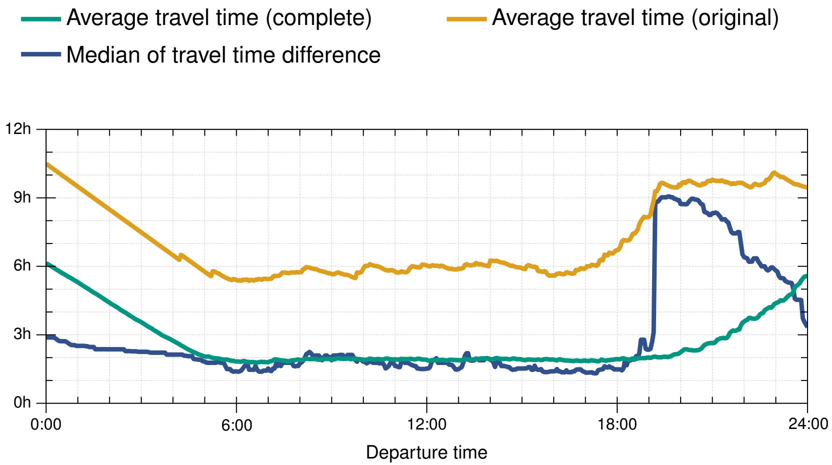
- Switzerland complete vs. Switzerland original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

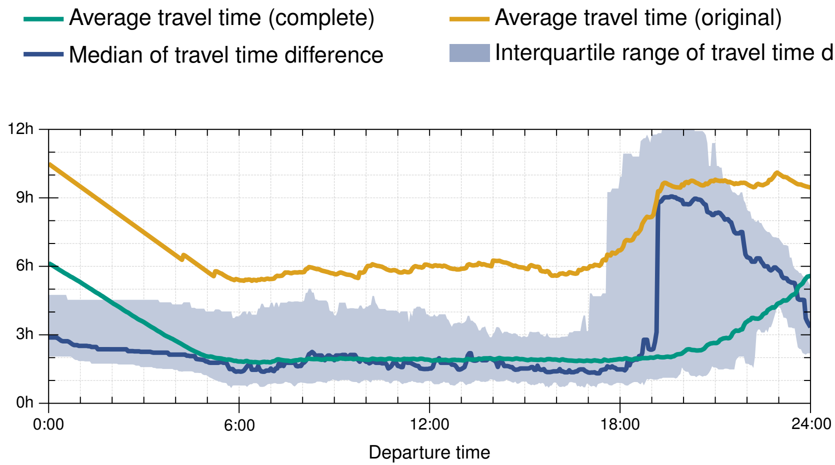
- Switzerland complete vs. Switzerland original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

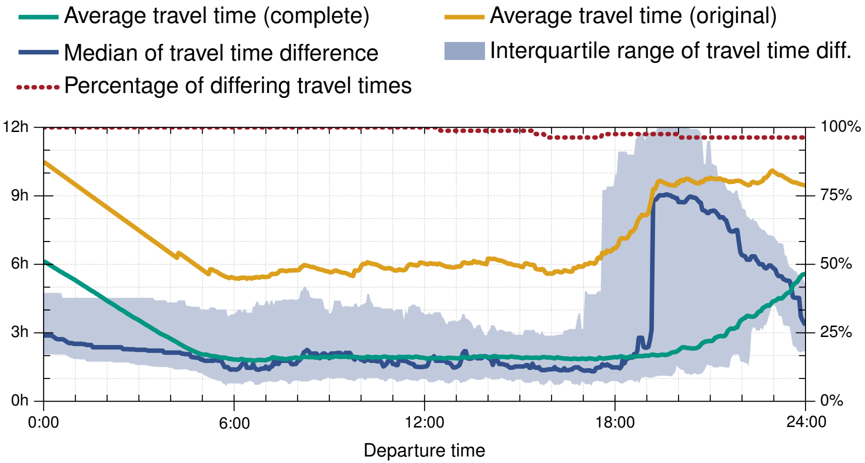
- Switzerland complete vs. Switzerland original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

- Switzerland complete vs. Switzerland original (Distance rank 16)

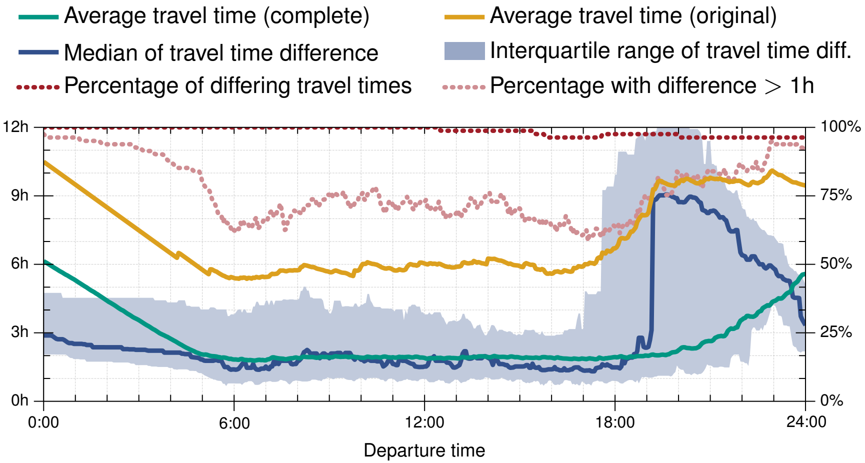


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente

Vergleich der Reisezeiten:

- Switzerland complete vs. Switzerland original (Distance rank 16)



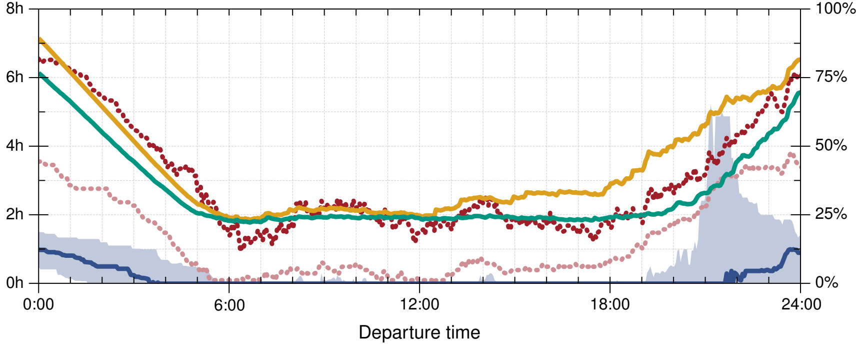
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente

Vergleich der Reisezeiten:

- Switzerland complete vs. Switzerland partial (Distance rank 16)

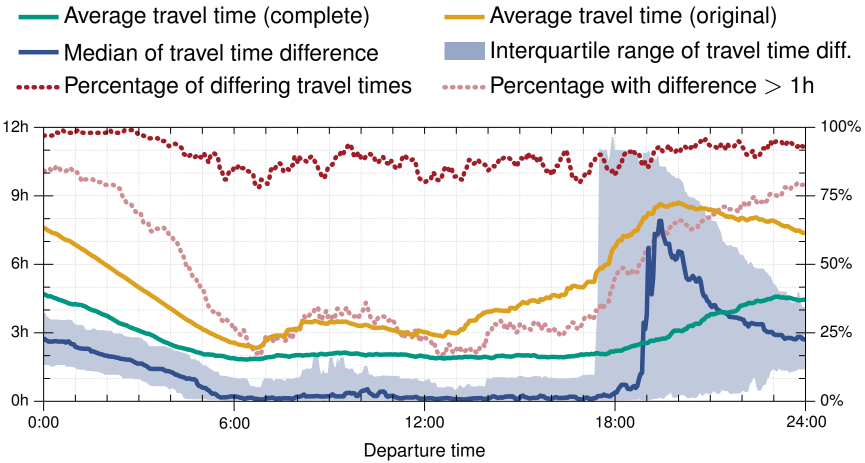
Average travel time (complete)
Average travel time (partial)
Median of travel time difference
Interquartile range of travel time diff.
Percentage of differing travel times
Percentage with difference > 1h



Experimente

Vergleich der Reisezeiten:

- Germany complete vs. Germany original (Distance rank 16)

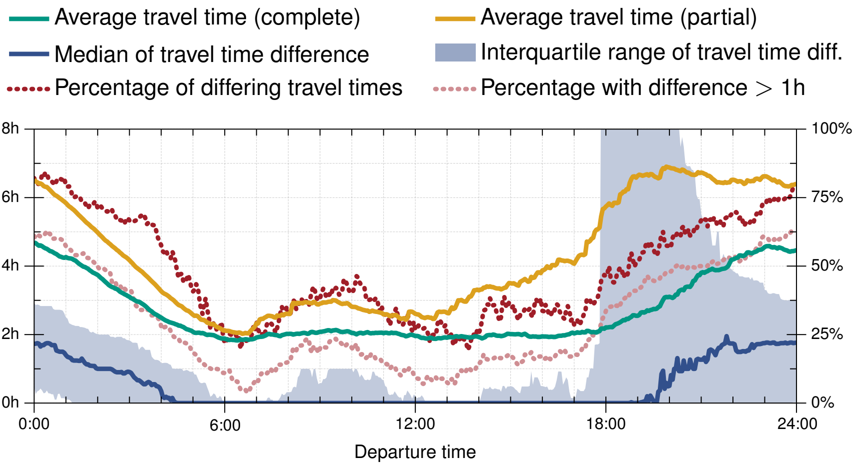


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente

Vergleich der Reisezeiten:

- Germany complete vs. Germany partial (Distance rank 16)



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Verkehrsumlegung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Verkehrsumlegung

Gegeben:

- Öffentliches Verkehrsnetz
- Liste mit erwarteter Nachfrage
  (Tupel aus: Startknoten, Zielknoten, Abfahrtszeit)

Gesucht:

- Auslastung der Fahrzeuge

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Verkehrsumlegung

Gegeben:

- Öffentliches Verkehrsnetz
- Liste mit erwarteter Nachfrage
  (Tupel aus: Startknoten, Zielknoten, Abfahrtszeit)

Gesucht:

- Auslastung der Fahrzeuge

Anwendung:

- Planung von neuen Linien
- Optimierung von Umleitungen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Verkehrsumlegung

Ansatz:

- Weise jedem Passagier (aus Nachfrage) eine Journey zu
- Algorithmus basiert auf CSA

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Verkehrsumlegung

Ansatz:

- Weise jedem Passagier (aus Nachfrage) eine Journey zu
- Algorithmus basiert auf CSA

Aber:

- Verhalten der Passagiere nicht immer eindeutig
- Erlaube suboptimale Journeys
- Erlaube (anteilig) mehrere Journeys pro Passagier

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Gefühlte Ankunftszeit (PAT)

Karlsruhe Institute of Technology

Idee:

- PAT für eine Connection *c* und Zielstop *d*
- Misst, wie nützlich *c* ist, um *d* zu erreichen
- Berücksichtigt vier Parameter:
  - Umstiegskosten λ_{trans}
  - Wartekosten λ_{wait}
  - Laufkosten λ_{walk}
  - Maximale erwartete Verspätung $\Delta_{\tau}^{\text{max}}$

Annahme:

- Passagiere versuchen, die PAT zu minimieren

Gefühlte Ankunftszeit (PAT)

Formale Definition:

$$\tau^{\mathrm{p}}_{\mathrm{arr}}(c, c', d) := \tau^{\mathrm{p}}_{\mathrm{trans}}(c, c') + \tau^{\mathrm{p}}_{\mathrm{wait}}(c, c') + \tau^{\mathrm{p}}_{\mathrm{arr}}(c', d)$$

$$\tau^{\mathrm{p}}_{\mathrm{arr}}(c, d \mid \mathrm{walk}) := \begin{cases} \tau_{\mathrm{arr}}(c) & \text{if } v_{\mathrm{arr}}(c) = d \\ \tau_{\mathrm{arr}}(c) + \lambda_{\mathrm{walk}} \cdot \tau_{\mathrm{trans}}(v_{\mathrm{arr}}(c), d) & \text{otherwise} \end{cases}$$

$$\mathcal{T}(c) := \{c' \in \mathcal{C} \mid \mathrm{trip}(c') = \mathrm{trip}(c) \wedge \tau_{\mathrm{dep}}(c') \geq \tau_{\mathrm{arr}}(c)\}$$

$$\tau^{\mathrm{p}}_{\mathrm{arr}}(c, d \mid \mathrm{trip}) := \begin{cases} \min\{\tau^{\mathrm{p}}_{\mathrm{arr}}(c', d) \mid c' \in \mathcal{T}(c)\} & \text{if } \mathcal{T}(c) \neq \emptyset \\ \infty & \text{otherwise} \end{cases}$$

$$\tau^{\mathrm{p}}_{\mathrm{arr}}(c, c', d) := \tau^{\mathrm{p}}_{\mathrm{trans}}(c, c') + \tau^{\mathrm{p}}_{\mathrm{wait}}(c, c') + \tau^{\mathrm{p}}_{\mathrm{arr}}(c', d)$$

$$\mathcal{R}(c) := \{c' \in \mathcal{C} \mid \tau_{\mathrm{wait}}(c, c') \geq 0\}$$

$$\mathcal{R}_{\mathrm{opt}}(c) := \{c' \in \mathcal{R}(c) \mid \forall \bar{c} \in \mathcal{R}(c) : \tau_{\mathrm{wait}}(c, \bar{c}) \geq \tau_{\mathrm{wait}}(c, c') \Rightarrow \tau^{\mathrm{p}}_{\mathrm{arr}}(c, \bar{c}, d) \geq \tau^{\mathrm{p}}_{\mathrm{arr}}(c, c', d)\}$$

$$\langle c_1, \ldots, c_k \rangle \text{ with } \forall i \in [1, k] \colon c_i \in \mathcal{R}_{\mathrm{opt}}(c) \wedge \forall i \in [2, k] \colon \tau_{\mathrm{wait}}(c, c_i) \geq \tau_{\mathrm{wait}}(c, c_{i-1})$$

$$\tau^{c}_{\mathrm{wait}}(i) := \begin{cases} \tau_{\mathrm{wait}}(c, c_i) & \text{if } i \in [1, k] \\ -\infty & \text{otherwise} \end{cases}$$

$$\tau^{\mathrm{p}}_{\mathrm{arr}}(c, d \mid \mathrm{trans}) := \begin{cases} \displaystyle\sum_{i=1}^{k} \left(\frac{P[\tau^{c}_{\mathrm{wait}}(i-1) < \Delta^{c}_{\tau} \leq \tau^{c}_{\mathrm{wait}}(i)]}{P[\Delta^{c}_{\tau} \leq \tau^{c}_{\mathrm{wait}}(k)]} \cdot \tau^{\mathrm{p}}_{\mathrm{arr}}(c, c_i, d) \right) & \text{if } k > 0 \\ \infty & \text{otherwise} \end{cases}$$

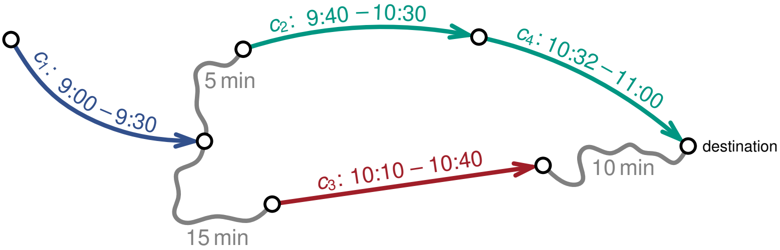
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min

Connection	PAT
c_4	
c_3	
c_2	
c_1	



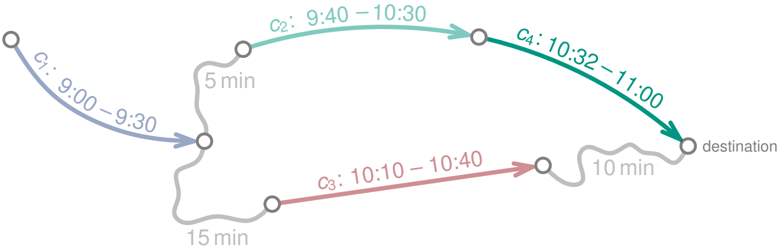
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 1:** Connection c erreicht Ziel
  $\Rightarrow$ PAT = arrival time $\tau_{\text{arr}}(c)$

Connection	PAT
c_4	11:00
c_3	
c_2	
c_1	



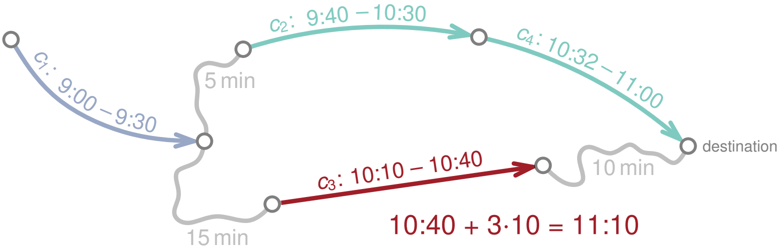
Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 2:** Laufen von Connection c zum Ziel

  $\Rightarrow$ PAT $= \tau_{\text{arr}}(c) + (\lambda_{\text{walk}} \cdot \tau_{\text{walk}})$

Connection	PAT
c_4	11:00
c_3	11:10
c_2	
c_1	



10:40 + 3·10 = 11:10

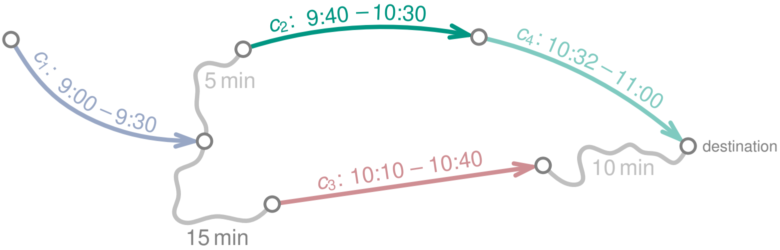
Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 3:** Weiterfahren mit Con. c' (gleicher Trip)

  $\Rightarrow$ PAT = PAT c'

Connection	PAT
c_4	11:00
c_3	11:10
c_2	11:00
c_1	



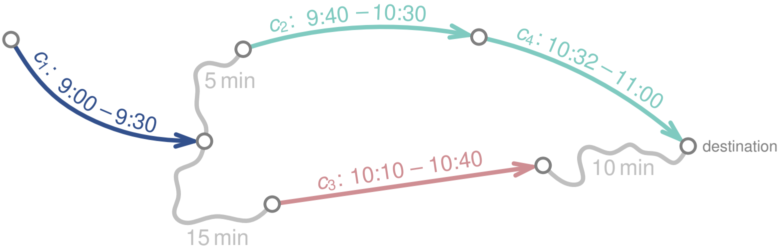
Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 4:** Weiterfahren mit Con. c' (anderer Trip)

  $\Rightarrow$ PAT = PAT $c' + \lambda_{\text{trans}} + (\lambda_{\text{walk}} \cdot \tau_{\text{walk}}) + (\lambda_{\text{wait}} \cdot \tau_{\text{wait}})$

Connection	PAT
c_4	11:00
c_3	11:10
c_2	11:00
c_1	



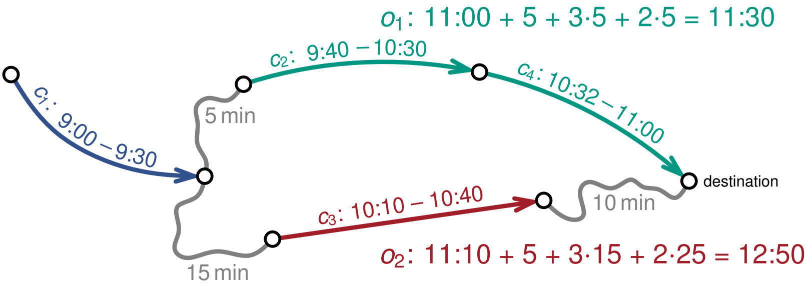
Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 4:** Weiterfahren mit Con. c' (anderer Trip)

  $\Rightarrow$ PAT = PAT $c' + \lambda_{\text{trans}} + (\lambda_{\text{walk}} \cdot \tau_{\text{walk}}) + (\lambda_{\text{wait}} \cdot \tau_{\text{wait}})$

Connection	PAT
c_4	11:00
c_3	11:10
c_2	11:00
c_1	



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel PAT-Berechnung

Beispiel:

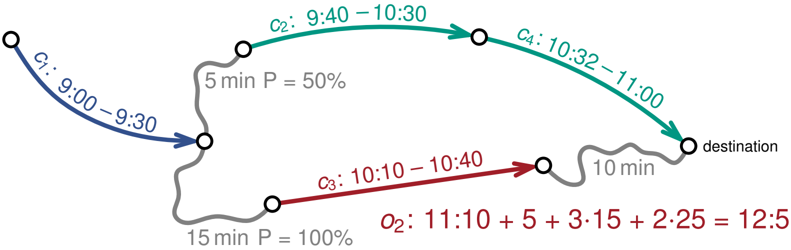
- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 4:** Weiterfahren mit einer Option o_i

  $\Rightarrow \text{PAT} = \sum_i \left(\text{transfer probability}(o_i) \cdot o_i\right)$

Connection	PAT
c_4	11:00
c_3	11:10
c_2	11:00
c_1	

P[1]·o_1 + (P[2] - P[1])·o_2

o_1: 11:00 + 5 + 3·5 + 2·5 = 11:30



o_2: 11:10 + 5 + 3·15 + 2·25 = 12:50

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel PAT-Berechnung

Beispiel:

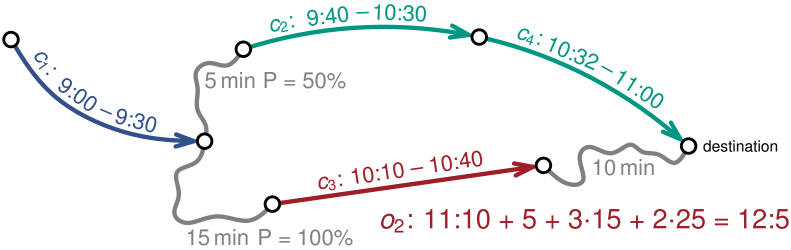
- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 4:** Weiterfahren mit einer Option o_i

  $\Rightarrow$ PAT $= \sum_i \left(\text{transfer probability}(o_i) \cdot o_i\right)$

Connection	PAT
c_4	11:00
c_3	11:10
c_2	11:00
c_1	12:10

$0.5 \cdot 11{:}30 + 0.5 \cdot 12{:}50 = 12{:}10$

o_1: $11{:}00 + 5 + 3 \cdot 5 + 2 \cdot 5 = 11{:}30$



o_2: $11{:}10 + 5 + 3 \cdot 15 + 2 \cdot 25 = 12{:}50$

Umlegungsalgorithmus

Ansatz:

- Simuliere Bewegung der Passagiere im Netzwerk
- Entscheide pro Connection c, wer c benutzt
- Passagiere mit selbem Ziel werden sich treffen
  - $\Rightarrow$ Müssen dieselben Entscheidungen treffen
  - $\Rightarrow$ Algorithmus profitiert von Synergieeffekten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umlegungsalgorithmus

Ansatz:

- Simuliere Bewegung der Passagiere im Netzwerk
- Entscheide pro Connection c, wer c benutzt
- Passagiere mit selbem Ziel werden sich treffen
  - $\Rightarrow$ Müssen dieselben Entscheidungen treffen
  - $\Rightarrow$ Algorithmus profitiert von Synergieeffekten

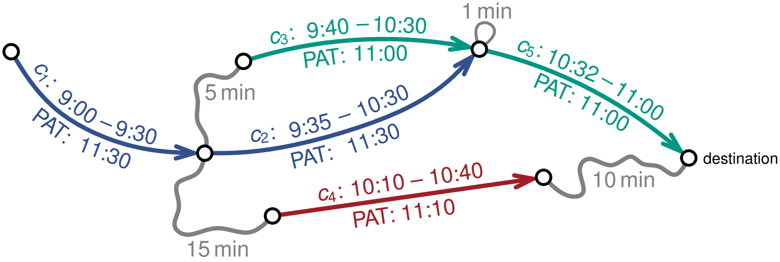
Überblick:

- Sortiere Passagiere nach Zielstop
- Berechne Umlegung pro Zielstop in 3 Schritten:
  - Berechne PATs für jede Connection
  - Simuliere Bewegung der Passagiere basierend auf PATs
  - Entferne überflüssige Kreise aus Journeys (optional)

Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

Time: 0:00

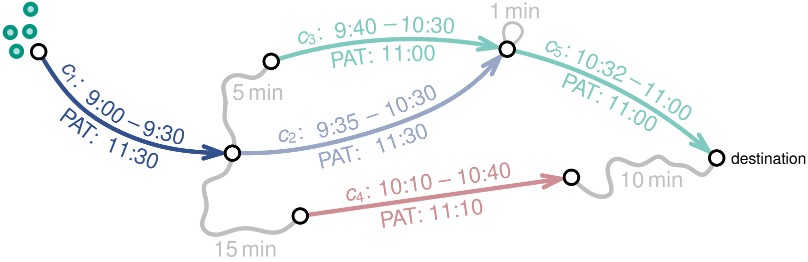


Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

1. Erzeuge Passagiere entsprechend der Nachfrage

Time: 9:00

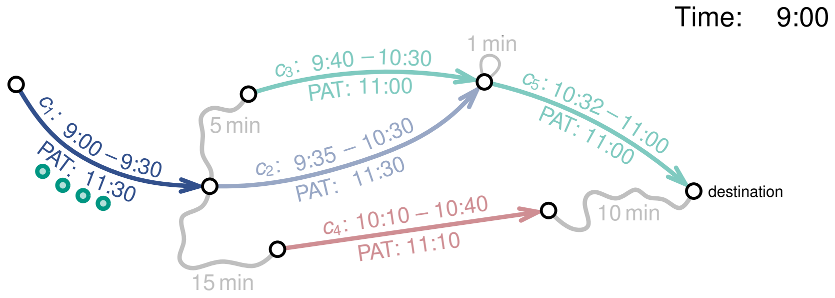


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

2. Entscheide, welche Passagiere einsteigen

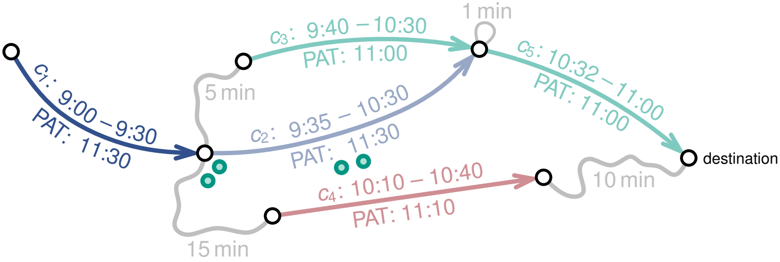


Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

3. Entscheide, welche Passagiere aussteigen

Time: 9:00



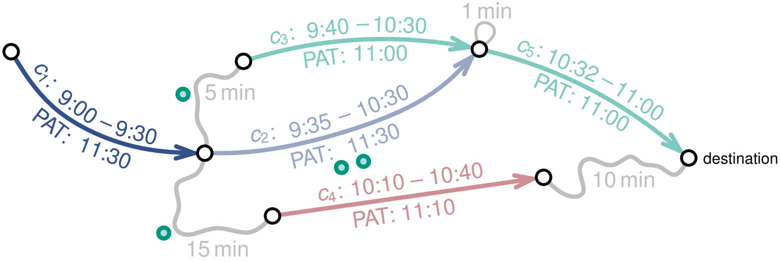
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

4. Verschiebe ausgestiegene Passagiere zum nächsten Stop

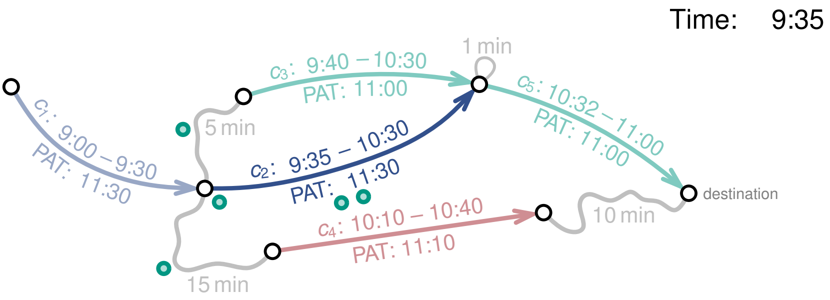
Time: 9:00



Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

1. Erzeuge Passagiere entsprechend der Nachfrage



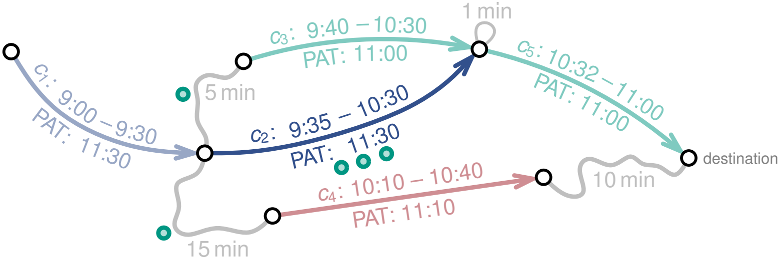
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

2. Entscheide, welche Passagiere einsteigen

Time: 9:35

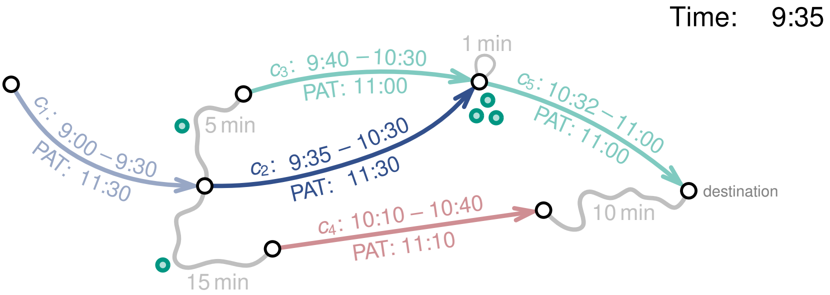


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

3. Entscheide, welche Passagiere aussteigen

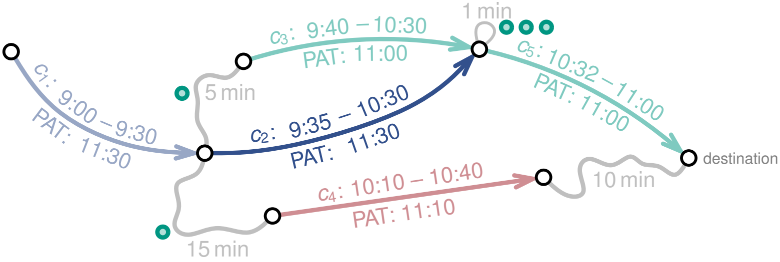


Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

4. Verschiebe ausgestiegene Passagiere zum nächsten Stop

Time: 9:35



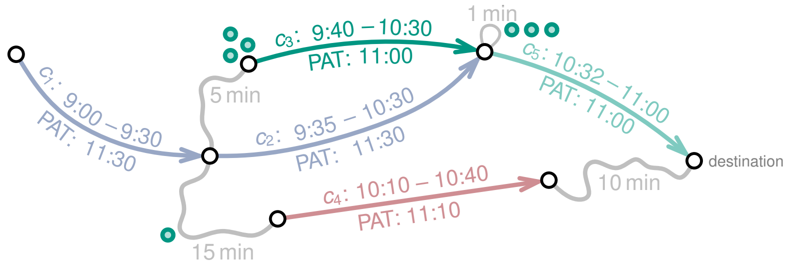
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

1. Erzeuge Passagiere entsprechend der Nachfrage

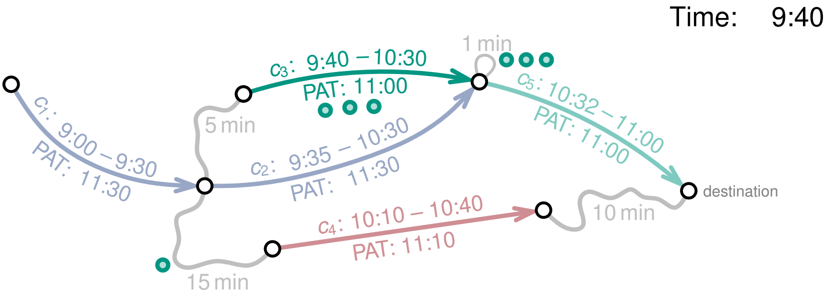
Time: 9:40



Beispiel Umlegungsberechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

2. Entscheide, welche Passagiere einsteigen

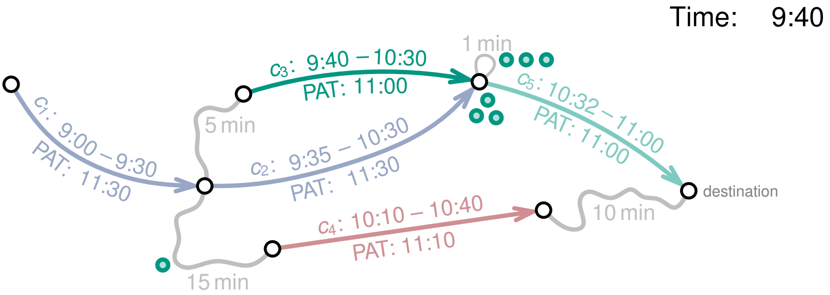


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel Umlegungsberechnung

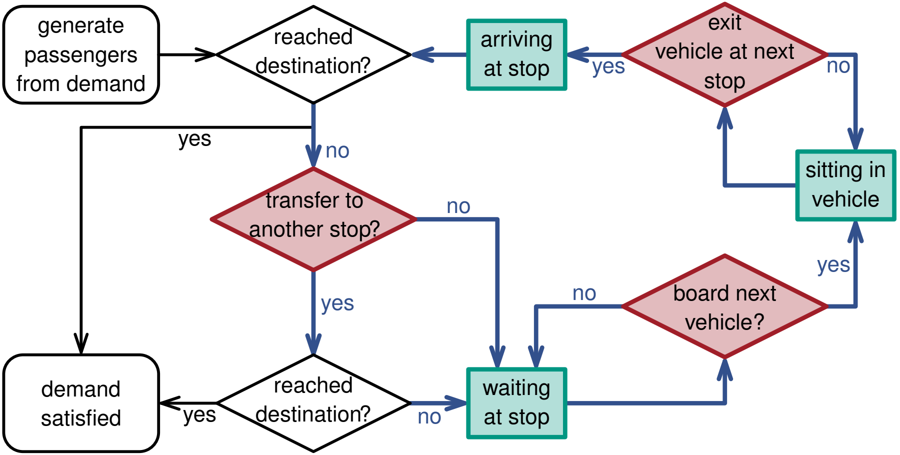
- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen

3. ...



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umlegungsberechnung Übersicht



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entscheidungsmodell

Ziel:

- Entscheidet, welche Connection ein Passagier nimmt
- Hängt von der Verspätungstoleranz $\lambda_{\Delta\text{max}}$ des Passagiers ab

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entscheidungsmodell

Ziel:

- Entscheidet, welche Connection ein Passagier nimmt
- Hängt von der Verspätungstoleranz $\lambda_{\Delta\max}$ des Passagiers ab

Definition:

- Gegeben sind die Optionen $o_1, \ldots, o_k$ und ihre PATs
- Bestimme den Nutzen $g(i)$ jeder Option i:

$$g(i) := \max(0, \min_{j \neq i}(\mathrm{PAT}(o_j) - \mathrm{PAT}(o_i) + \lambda_{\Delta\max})$$

- Die Wahrscheinlichkeit $P[i]$, dass ein Passagier Option i wählt, ist:

$$P[i] := \frac{g(i)}{\sum_{j=1}^{k} g(j)}$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

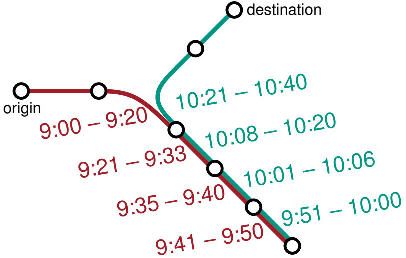
Kreise

- Journeys können Kreise enthalten, d.h. Stops mehrfach besuchen
- Umlegungen mit Kreisen können unerwünscht sein
- Eine Journey mit Kreisen kann optimal bezüglich PAT sein
- Hohe Wartekosten können zu Kreisen führen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

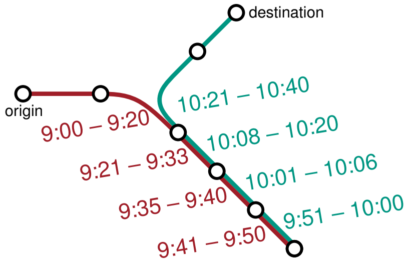
Kreise

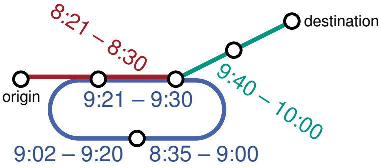
- Journeys können Kreise enthalten, d.h. Stops mehrfach besuchen
- Umlegungen mit Kreisen können unerwünscht sein
- Eine Journey mit Kreisen kann optimal bezüglich PAT sein
- Hohe Wartekosten können zu Kreisen führen



Kreise

- Journeys können Kreise enthalten, d.h. Stops mehrfach besuchen
- Umlegungen mit Kreisen können unerwünscht sein
- Eine Journey mit Kreisen kann optimal bezüglich PAT sein
- Hohe Wartekosten können zu Kreisen führen



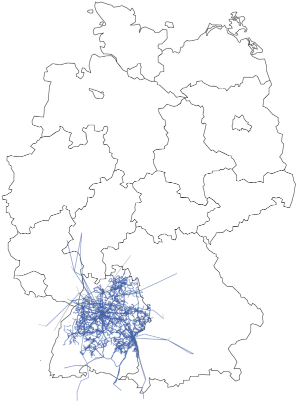


Auswertung

Instanzen:

- Großraum Stuttgart
- Enthält auch Frankfurt, Basel und München
- Beschreibt den Verkehr eines Tages

Anzahl Knoten	15 115
Anzahl Stops	13 941
Anzahl Kanten	33 890
Anzahl Kanten ohne Schlaufen	18 775
Anzahl Connections	780 042
Anzahl Trips	47 844
Anzahl Passagiere	1 249 910



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Auswertung – Laufzeiten

Benutzte Parameter:

- Laufkosten $\lambda_{\text{walk}} = 2$
- Wartekosten $\lambda_{\text{wait}} = 0.5$
- Umstiegskosten $\lambda_{\text{trans}} = 5$ min
- Verspätungstoleranz $\lambda_{\Delta\text{max}} = 5$ min
- Maximale erwartete Verspätung $\Delta_{\tau}^{\text{max}} = 1$ min

Laufzeitvergleich:

- Laufzeit VISUM ≈ 30 min (mit 8 Threads)
- PAT-basierte Umlegung: (mit Passenger Multiplier $= 10$)

Anzahl Threads	1	2	4
Laufzeit [sec]	108.92	65.57	38.41

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Auswertung – Umlegungsqualität

- Beide Umlegungen sind sehr ähnlich
- VISUM berechnet etwas kürzere Fahrzeiten
- PAT-basierter Algorithmus berechnet Journeys mit weniger Umstiegen

	VISUM			PAT-basierter Algorithmus		
Eigenschaft	min	mean	max	min	mean	max
Reisezeit [min]	2.98	46.885	429.00	2.98	47.199	429.00
Zeit im Fahrzeug [min]	0.02	21.059	380.00	0.02	21.231	323.97
Laufdauer [min]	2.00	22.394	149.00	2.00	22.476	149.00
Wartezeit [min]	0.00	3.432	217.02	0.00	3.492	217.02
Züge pro Passagier	1.00	1.771	6.00	1.00	1.746	8.00
Connections pro Passagier	1.00	9.396	109.00	1.00	9.474	97.00
Passagiere pro Connection	0.00	12.740	1 290.10	0.00	12.847	1 233.60

Institut für Theoretische Informatik
Lehrstuhl Algorithmik