

Algorithmen für Routenplanung

20. Vorlesung, Sommersemester 2024

Adrian Feilhauer | 1. Juli 2024



Gegeben:

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

Gesucht:

- Auslastung der Fahrzeuge

Gegeben:

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

Gesucht:

- Auslastung der Fahrzeuge

Anwendung:

- Planung von neuen Linien
- Optimierung von Umleitungen

Ansatz:

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

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

PAT (perceived arrival time):

- PAT für eine Connection c und Zielstop d
- Misst, wie nützlich c ist, um d zu erreichen
- Hauptbestandteil: Tatsächliche optimale Ankunftszeit an d
- Vier zusätzliche Parameter:
 - Umstiegskosten λ_{trans}
 - Wartekosten λ_{wait}
 - Laufkosten λ_{walk}
 - Maximale erwartete Verspätung $\Delta_{\tau}^{\text{max}}$

Annahme:

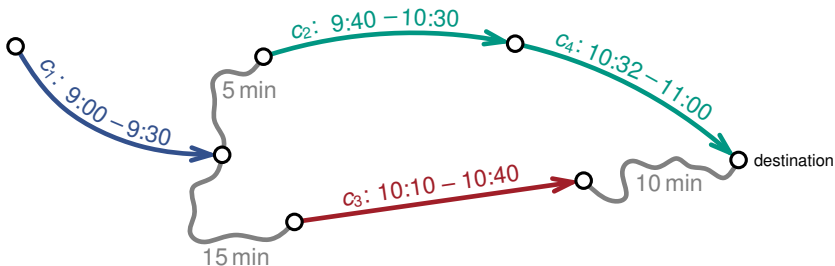
- Passagiere versuchen, die PAT zu minimieren

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	

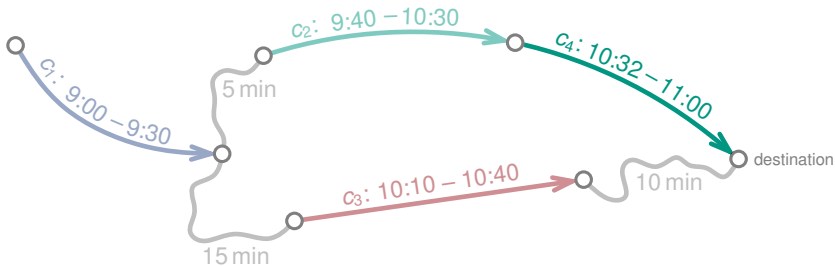


Beispiel PAT-Berechnung

Beispiel:

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

Connection	PAT
c_4	11:00
c_3	
c_2	
c_1	

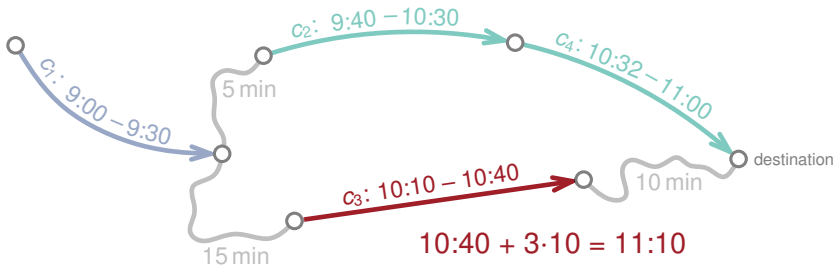


Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 2:** Laufen von Connection c zum Ziel
⇒ $\text{PAT} = \tau_{\text{arr}}(c) + (\lambda_{\text{walk}} \cdot \tau_{\text{walk}})$

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

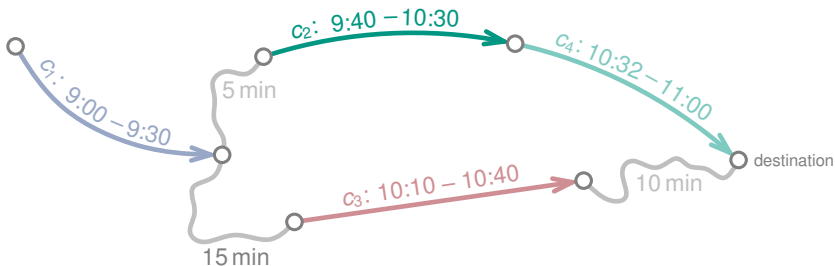


Beispiel PAT-Berechnung

Beispiel:

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

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

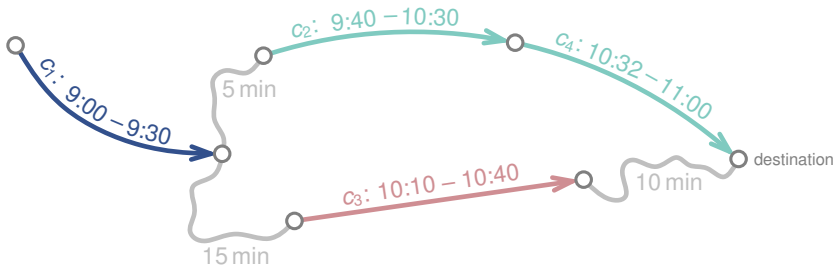


Beispiel PAT-Berechnung

Beispiel:

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

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

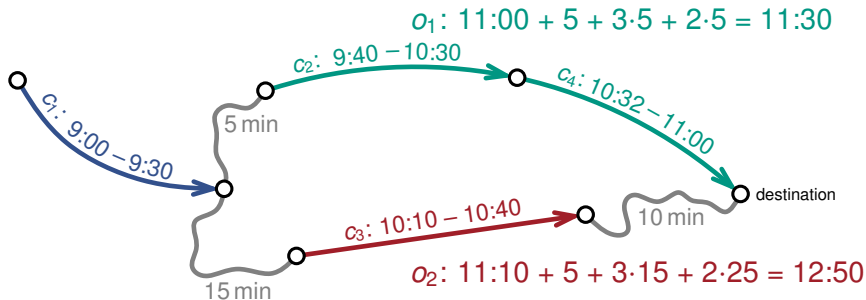


Beispiel PAT-Berechnung

Beispiel:

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

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



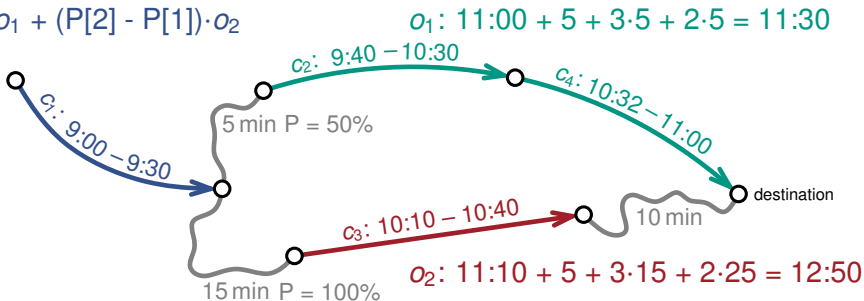
Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 4:** Weiterfahren mit einer Option o_i
 $\Rightarrow \text{PAT} = \sum_i (\text{transfer probability}(o_i) \cdot o_i)$

Connection	PAT
C_4	11:00
C_3	11:10
C_2	11:00
C_1	

$$P[1] \cdot o_1 + (P[2] - P[1]) \cdot o_2$$



Beispiel PAT-Berechnung

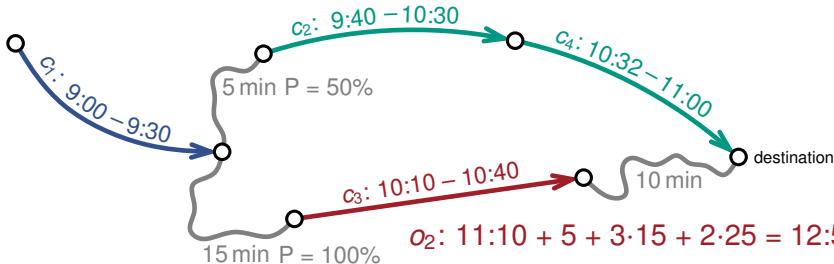
Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 4:** Weiterfahren mit einer Option o_i
 $\Rightarrow \text{PAT} = \sum_i (\text{transfer probability}(o_i) \cdot o_i)$

Connection	PAT
C_4	11:00
C_3	11:10
C_2	11:00
C_1	12:10

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

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



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

Ziel:

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

Ziel:

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

Definition:

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

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

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

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

Ansatz:

- Simuliere Bewegung der Passagiere im Netzwerk
- Entscheide pro Connection c , wer c benutzt
- Passagiere mit selbem Ziel werden sich treffen
 - ⇒ Müssen dieselben Entscheidungen treffen
 - ⇒ Algorithmus profitiert von Synergieeffekten
- **Passenger Multiplier** λ_{mul}
 - Generiere für jeden Passagier in der Nachfrage λ_{mul} Kopien
 - ⇒ Erlaubt (anteilig) mehrere Journeys pro Passagier

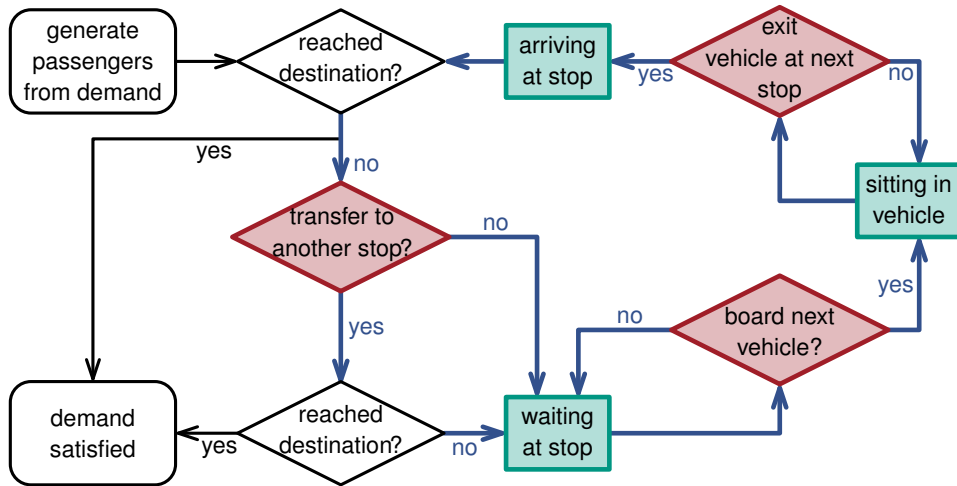
Ansatz:

- Simuliere Bewegung der Passagiere im Netzwerk
- Entscheide pro Connection c , wer c benutzt
- Passagiere mit selbem Ziel werden sich treffen
 - ⇒ Müssen dieselben Entscheidungen treffen
 - ⇒ Algorithmus profitiert von Synergieeffekten
- **Passenger Multiplier** λ_{mul}
 - Generiere für jeden Passagier in der Nachfrage λ_{mul} Kopien
 - ⇒ Erlaubt (anteilig) mehrere Journeys pro Passagier

Überblick:

- Gruppierere 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)

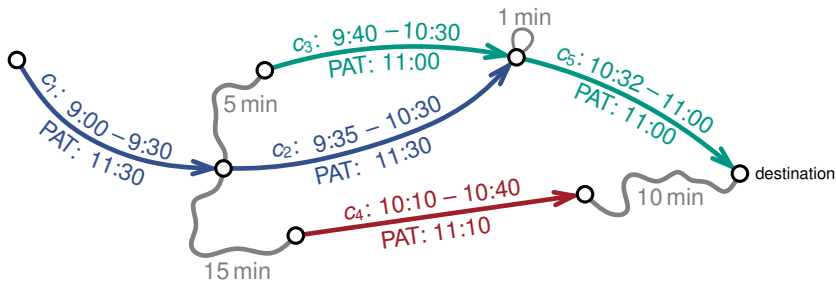
Umlegungsrechnung Übersicht



Beispiel Umlegungsrechnung

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

Time: 0:00

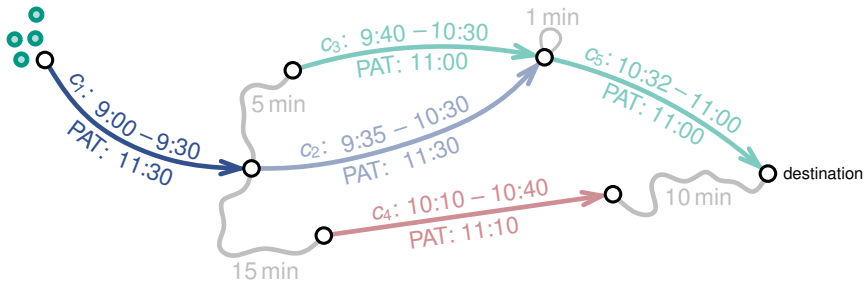


Beispiel Umlegungsrechnung

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

1. Erzeuge Passagiere entsprechend der Nachfrage

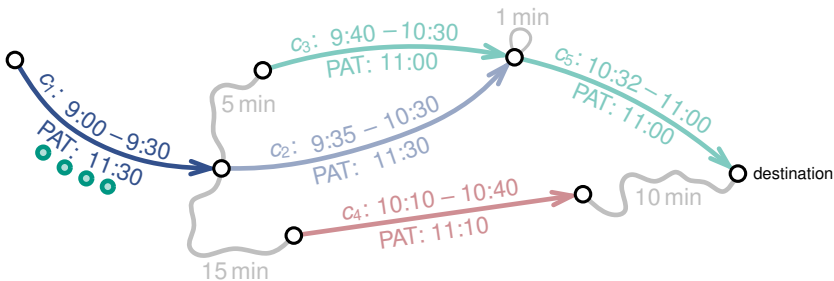
Time: 9:00



Beispiel Umlegungsrechnung

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

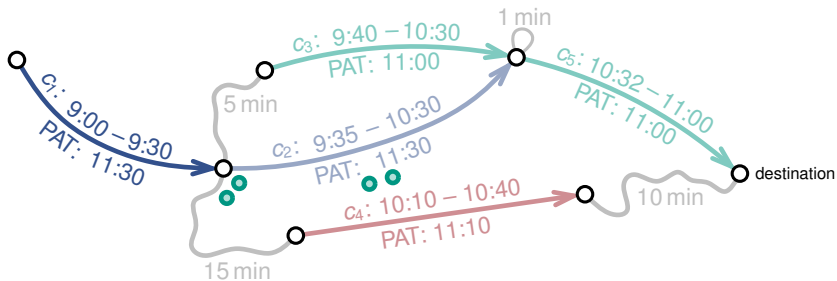
Time: 9:00



Beispiel Umlegungsrechnung

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

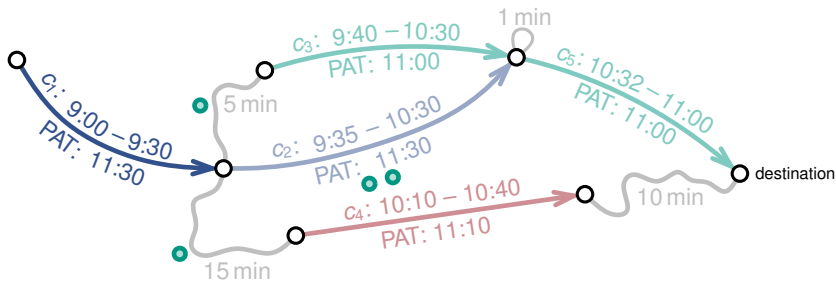
Time: 9:00



Beispiel Umlegungsrechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
 - Entscheide, welche Passagiere die Connection benutzen
4. Verschiebe ausgestiegene Passagiere zum nächsten Stop

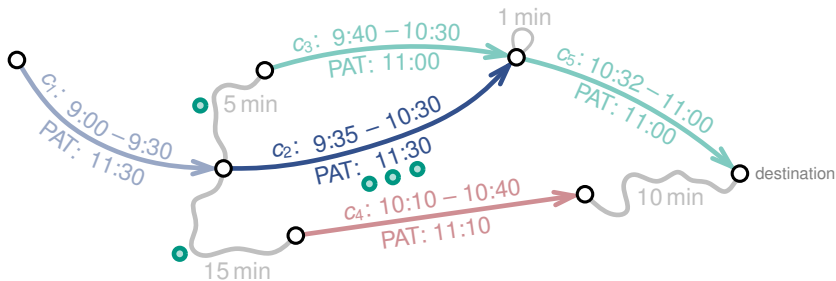
Time: 9:00



Beispiel Umlegungsrechnung

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

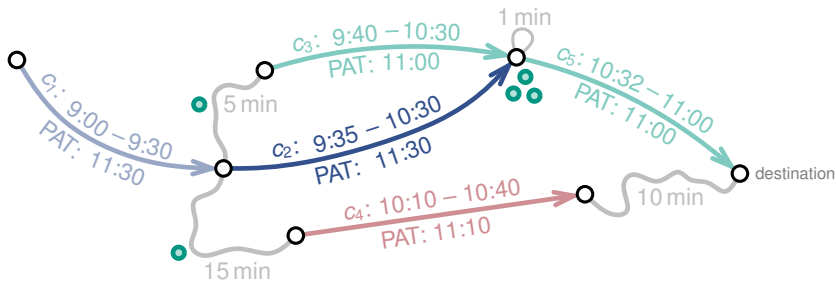
Time: 9:35



Beispiel Umlegungsrechnung

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

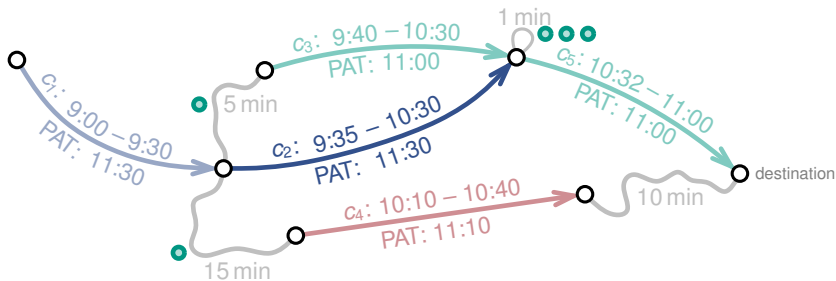
Time: 9:35



Beispiel Umlegungsrechnung

- Bearbeite Connections chronologisch (nach Abfahrtszeit)
 - Entscheide, welche Passagiere die Connection benutzen
4. Verschiebe ausgestiegene Passagiere zum nächsten Stop

Time: 9:35

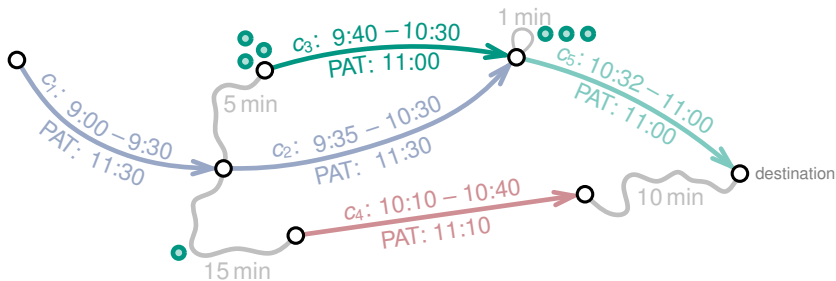


Beispiel Umlegungsrechnung

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

1. Erzeuge Passagiere entsprechend der Nachfrage

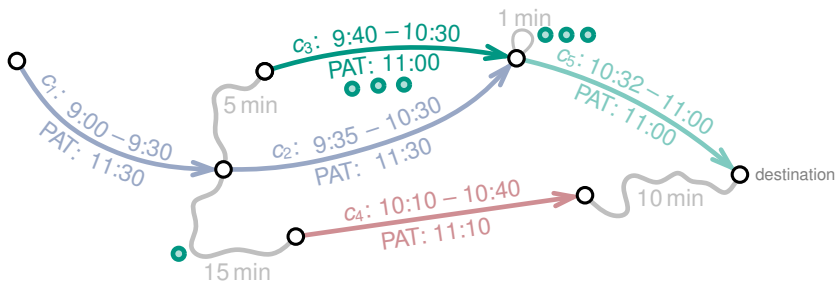
Time: 9:40



Beispiel Umlegungsrechnung

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

Time: 9:40

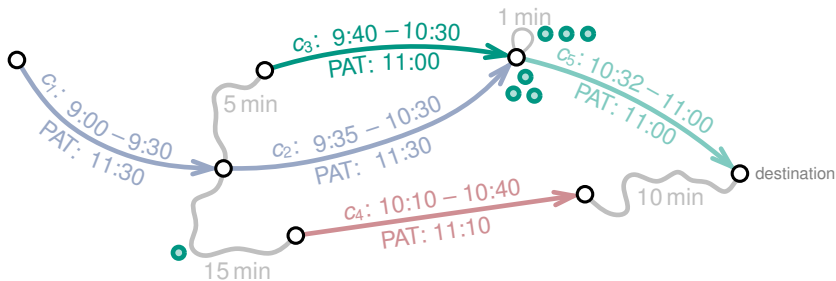


Beispiel Umlegungsrechnung

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

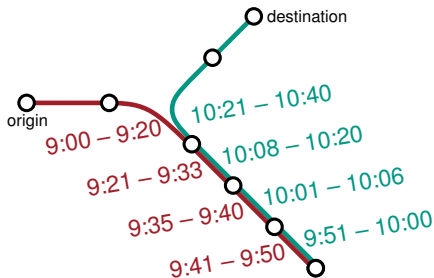
3. ...

Time: 9:40

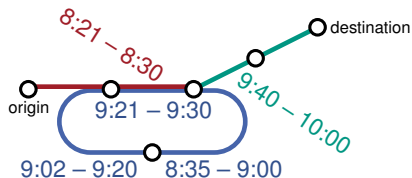
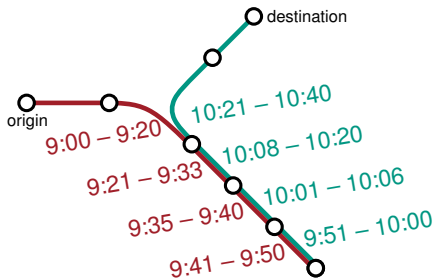


- Journeys können Kreise enthalten, d.h. Stops mehrfach besuchen
- Umlegungen mit Kreisen können unerwünscht sein
- Eine Journey mit Kreisen kann optimal bezüglich PAT sein
- Hohe Wartekosten können zu Kreisen führen

- 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



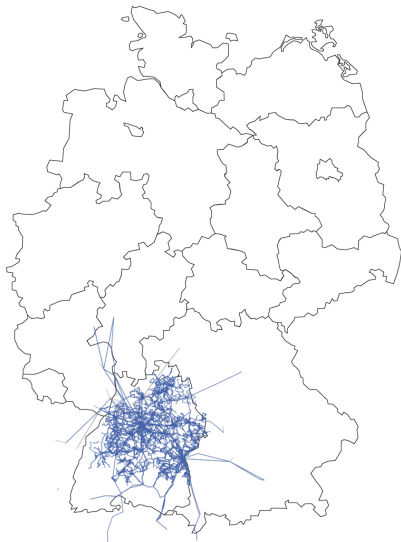
- Journeys können Kreise enthalten, d.h. Stops mehrfach besuchen
- Umlegungen mit Kreisen können unerwünscht sein
- Eine Journey mit Kreisen kann optimal bezüglich PAT sein
- Hohe Wartekosten können zu Kreisen führen



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 Schleifen	18 775
Anzahl Connections	780 042
Anzahl Trips	47 844
Anzahl Passagiere	1 249 910



Benutzte Parameter:

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

Laufzeitvergleich:

- Kommerzielles Tool VISUM: Laufzeit $\approx 30 \text{ min}$ (mit 8 Threads)
- PAT-basierte Umlegung: (mit $\lambda_{\text{mul}} = 10$)

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

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

Eigenschaft	VISUM			PAT-basierter Algorithmus		
	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



Lars Briem, Sebastian Buck, Holger Ebhart, Nicolai Mallig, Ben Strasser, Peter Vortisch, Dorothea Wagner, and Tobias Zündorf.
Efficient traffic assignment for public transit networks.

In *16th International Symposium on Experimental Algorithms (SEA 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

Formale Definition:

$$\tau_{\text{arr}}^{\text{p}}(c, d) := \min\{\tau_{\text{arr}}^{\text{p}}(c, d \mid \text{walk}), \tau_{\text{arr}}^{\text{p}}(c, d \mid \text{trip}), \tau_{\text{arr}}^{\text{p}}(c, d \mid \text{trans})\}$$

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

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

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

$$\tau_{\text{arr}}^{\text{p}}(c, c', d) := \tau_{\text{trans}}^{\text{p}}(c, c') + \tau_{\text{wait}}^{\text{p}}(c, c') + \tau_{\text{arr}}^{\text{p}}(c', d)$$

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

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

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

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

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