Karlsruher Institut für Technologie

Algorithmen für Routenplanung

20. Vorlesung, Sommersemester 2022

Adrian Feilhauer | 4. Juli 2022

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

Verkehrsumlegung

Gegeben:

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

Gesucht:

- Auslastung der Fahrzeuge

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

Verkehrsumlegung

Ansatz:

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

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

Gefühlte Ankunftszeit (PAT)

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 $\Delta_{\tau}^{\max}$

Annahme:

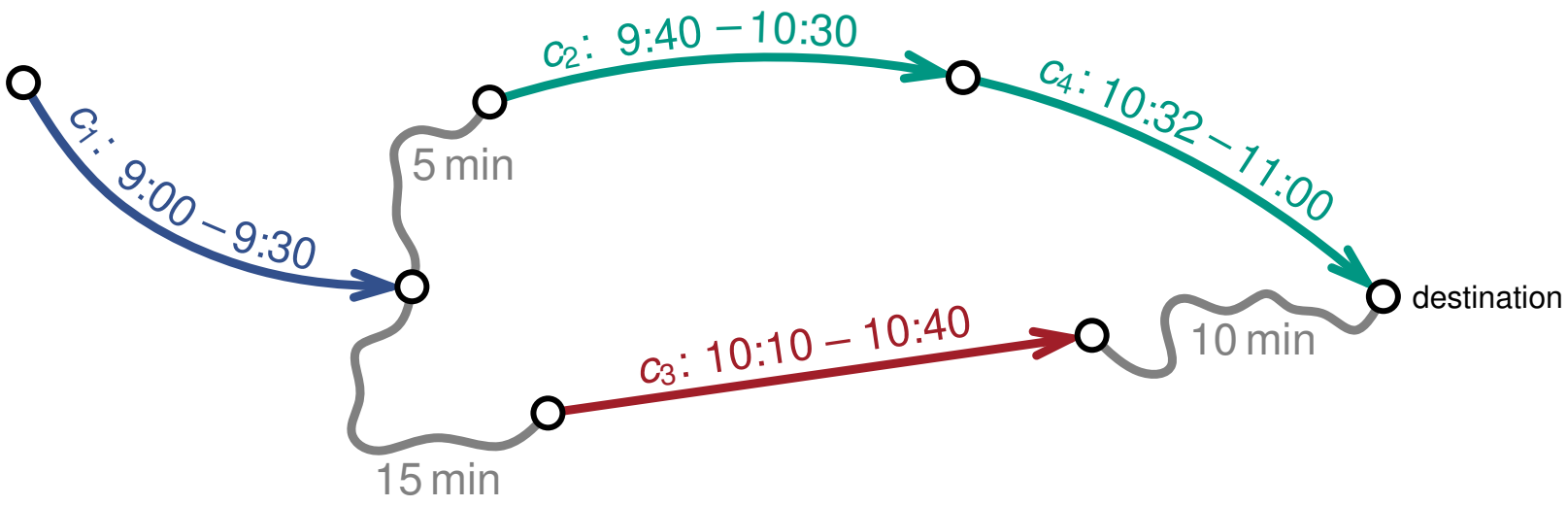
- Passagiere versuchen, die PAT zu minimieren

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	



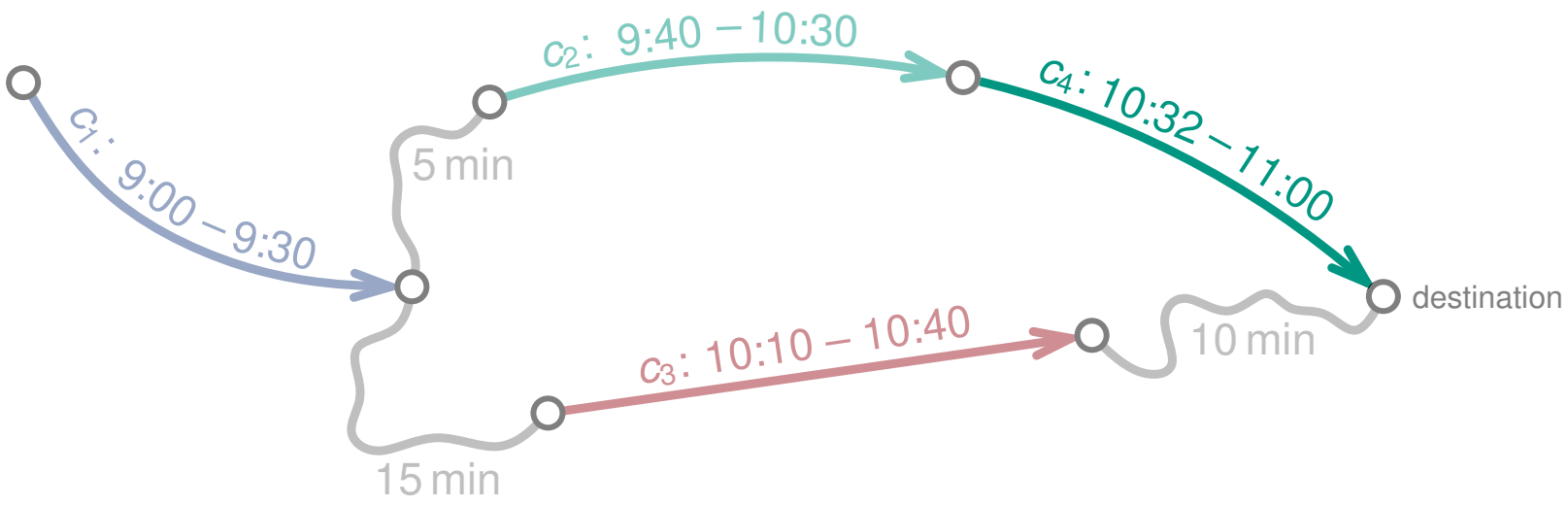
Beispiel PAT-Berechnung

Beispiel:

- $\lambda_{\text{walk}} = 3$, $\lambda_{\text{wait}} = 2$, $\lambda_{\text{trans}} = 5$ min
- **Fall 1:** Connection c erreicht Ziel

  $\Rightarrow$ PAT = arrival time $\tau_{\text{arr}}(c)$

Connection	PAT
c_4	11:00
c_3	
c_2	
c_1	



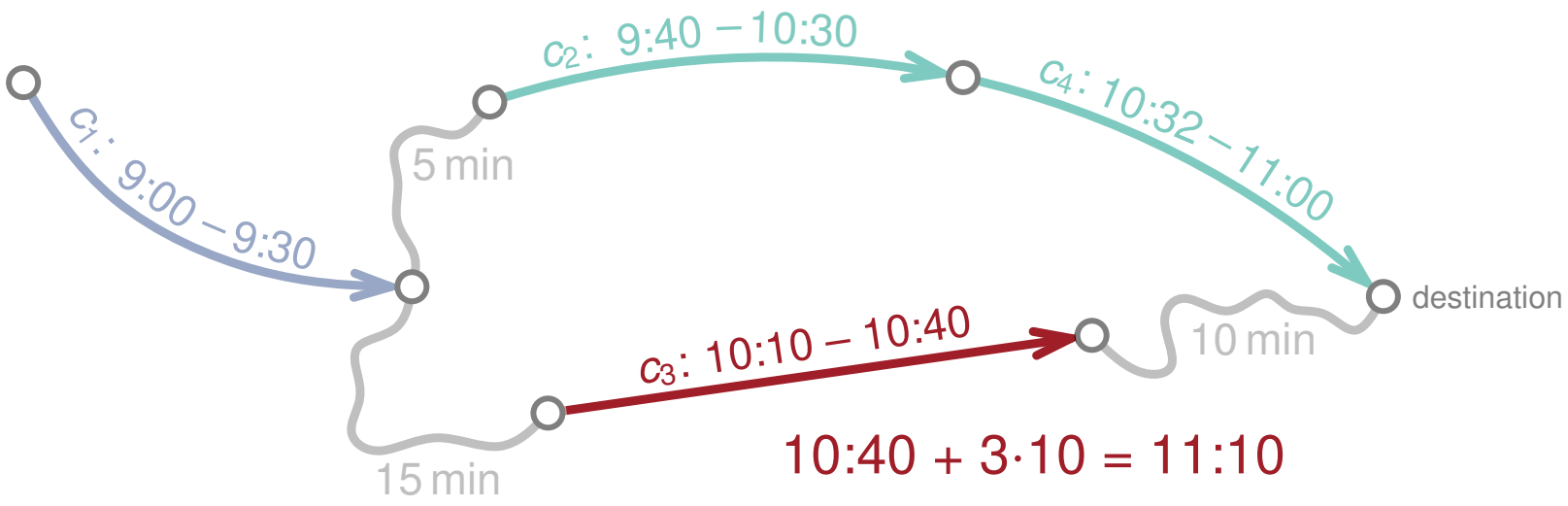
Beispiel PAT-Berechnung

Beispiel:

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

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

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



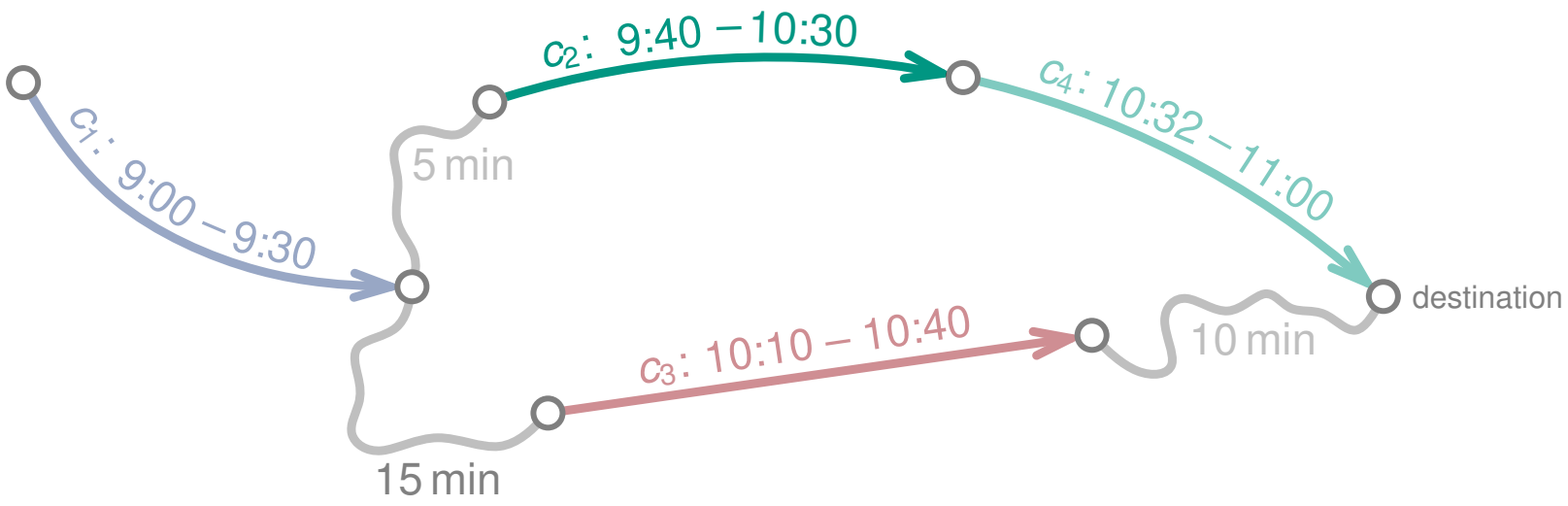
Beispiel PAT-Berechnung

Beispiel:

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

  $\Rightarrow$ PAT = PAT c'

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



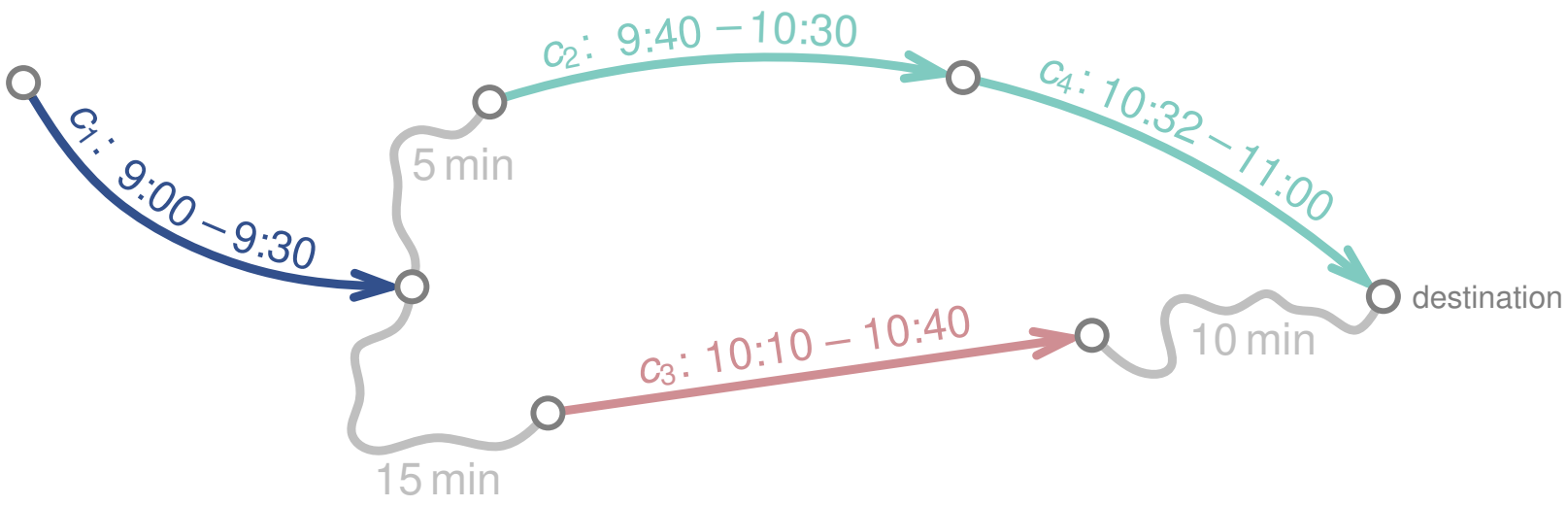
Beispiel PAT-Berechnung

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	



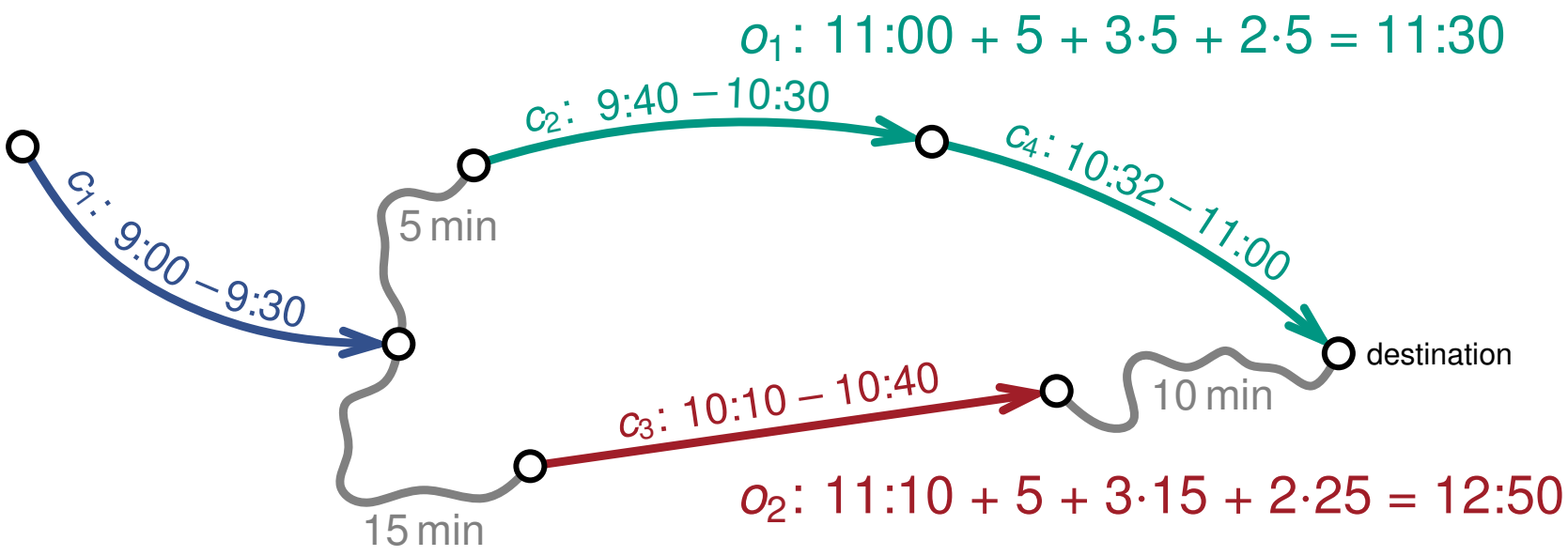
Beispiel PAT-Berechnung

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	



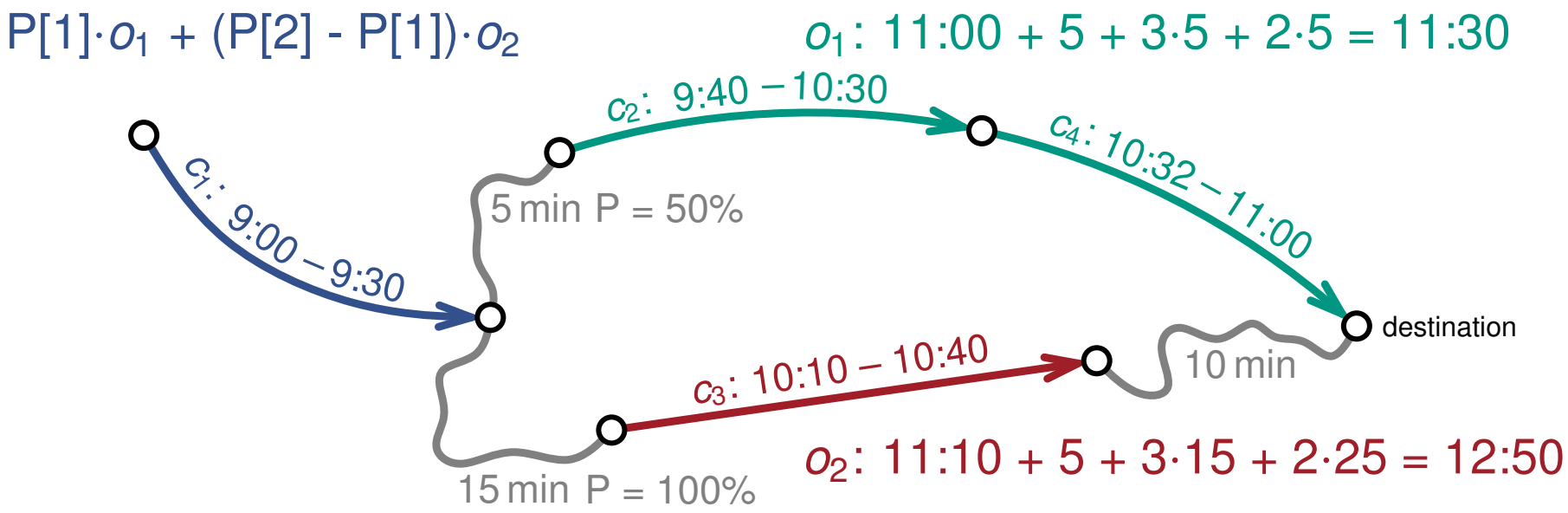
Beispiel PAT-Berechnung

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	



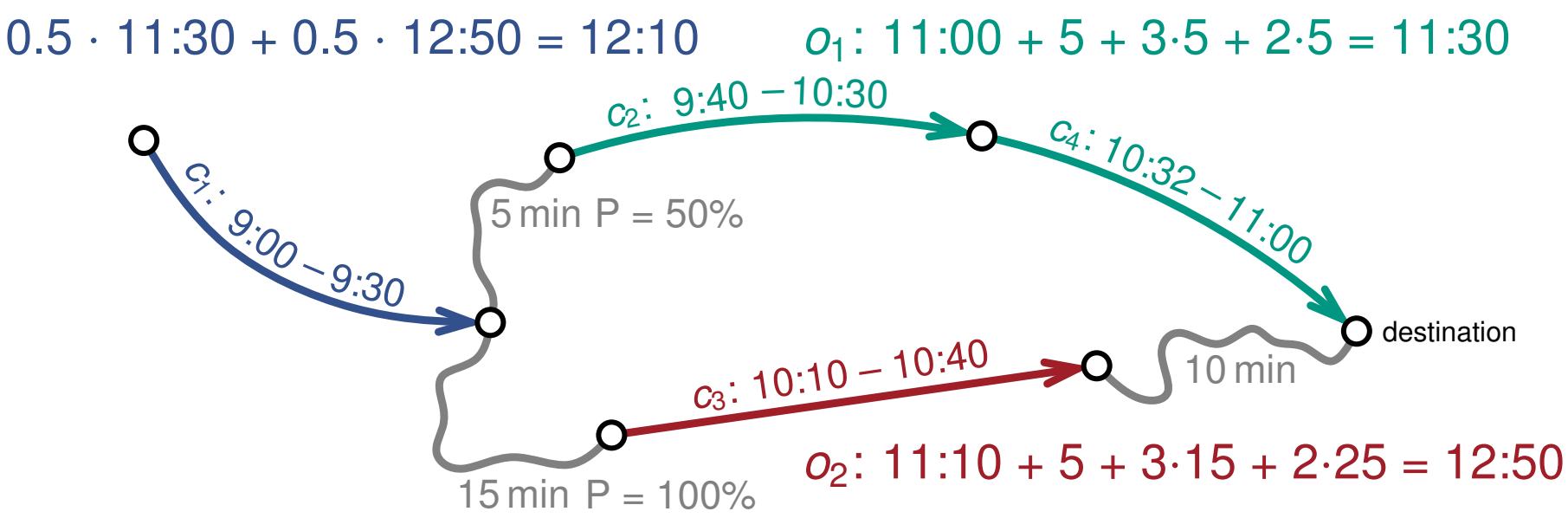
Beispiel PAT-Berechnung

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	12:10



Entscheidungsmodell

Ziel:

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

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:

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

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
- Passenger Multiplier λ_{mul}
  - Generiere für jeden Passagier in der Nachfrage λ_{mul} Kopien
  - $\Rightarrow$ Erlaubt (anteilig) mehrere Journeys pro Passagier

Umlegungsalgorithmus

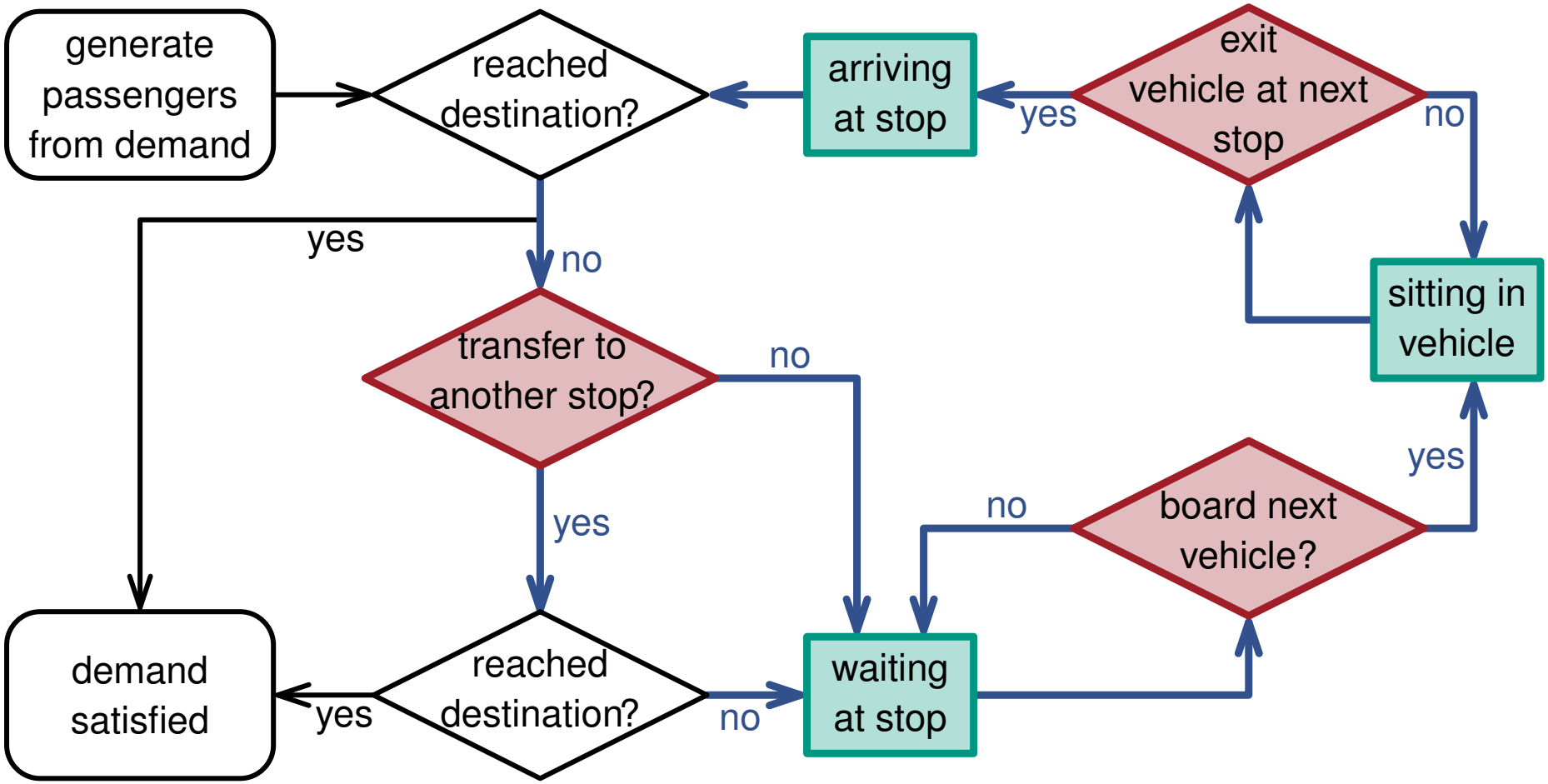
Ansatz:

- 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
- Passenger Multiplier λ_{mul}
  - Generiere für jeden Passagier in der Nachfrage λ_{mul} Kopien
  - $\Rightarrow$ Erlaubt (anteilig) mehrere Journeys pro Passagier

Überblick:

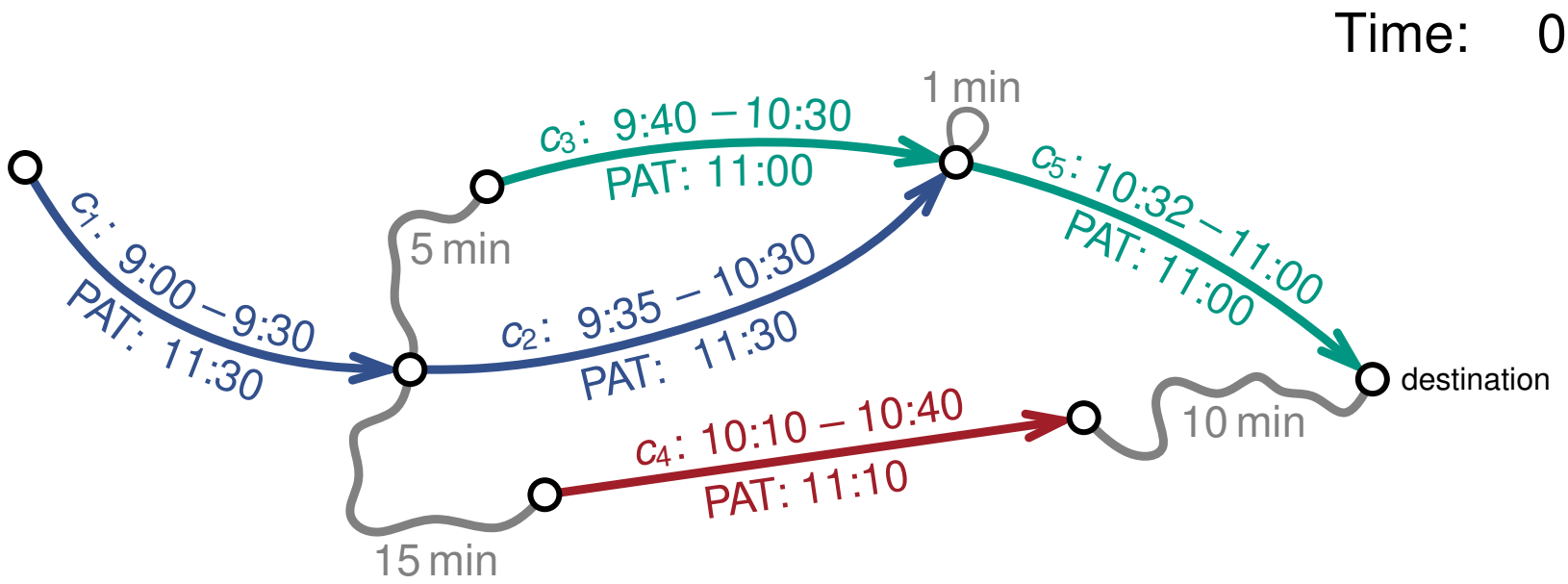
- Gruppiere Passagiere nach Zielstop
- Berechne Umlegung pro Zielstop in 3 Schritten:
  - Berechne PATs für jede Connection
  - Simuliere Bewegung der Passagiere basierend auf PATs
  - Entferne überflüssige Kreise aus Journeys (optional)

Umlegungsberechnung Übersicht



Beispiel Umlegungsberechnung

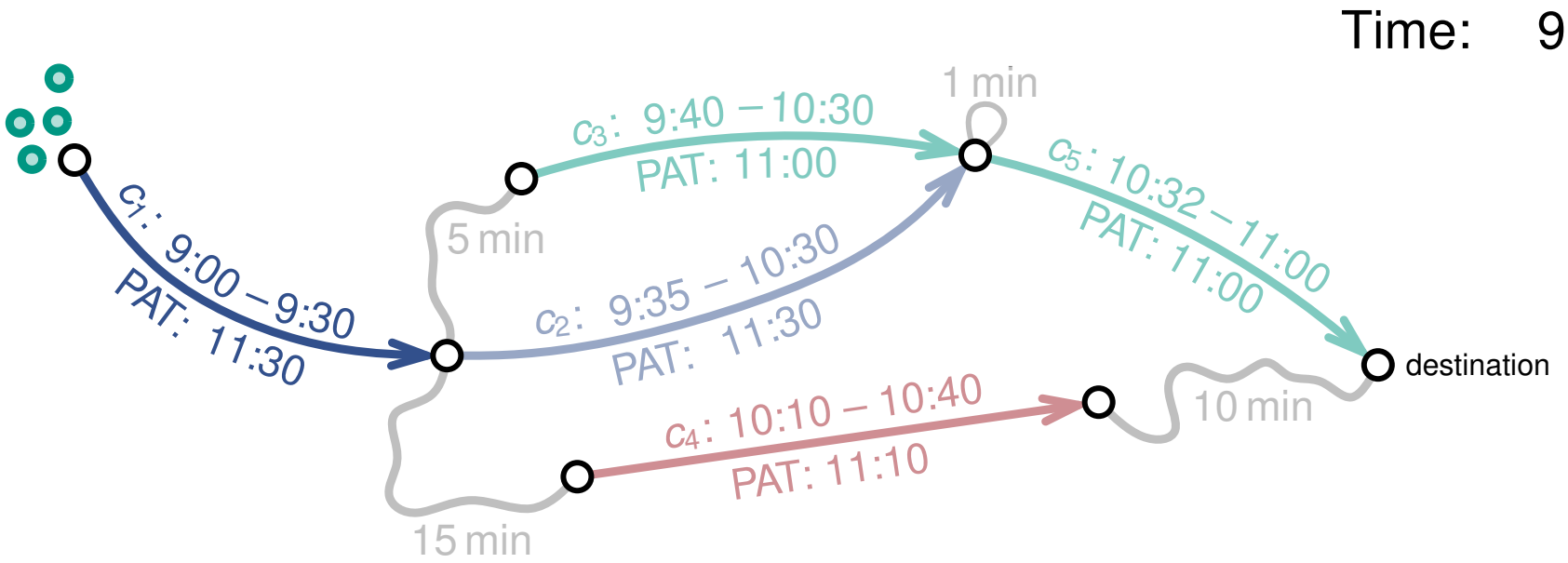
- Bearbeite Connections chronologisch (nach Abfahrtszeit)
- Entscheide, welche Passagiere die Connection benutzen



Beispiel Umlegungsberechnung

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

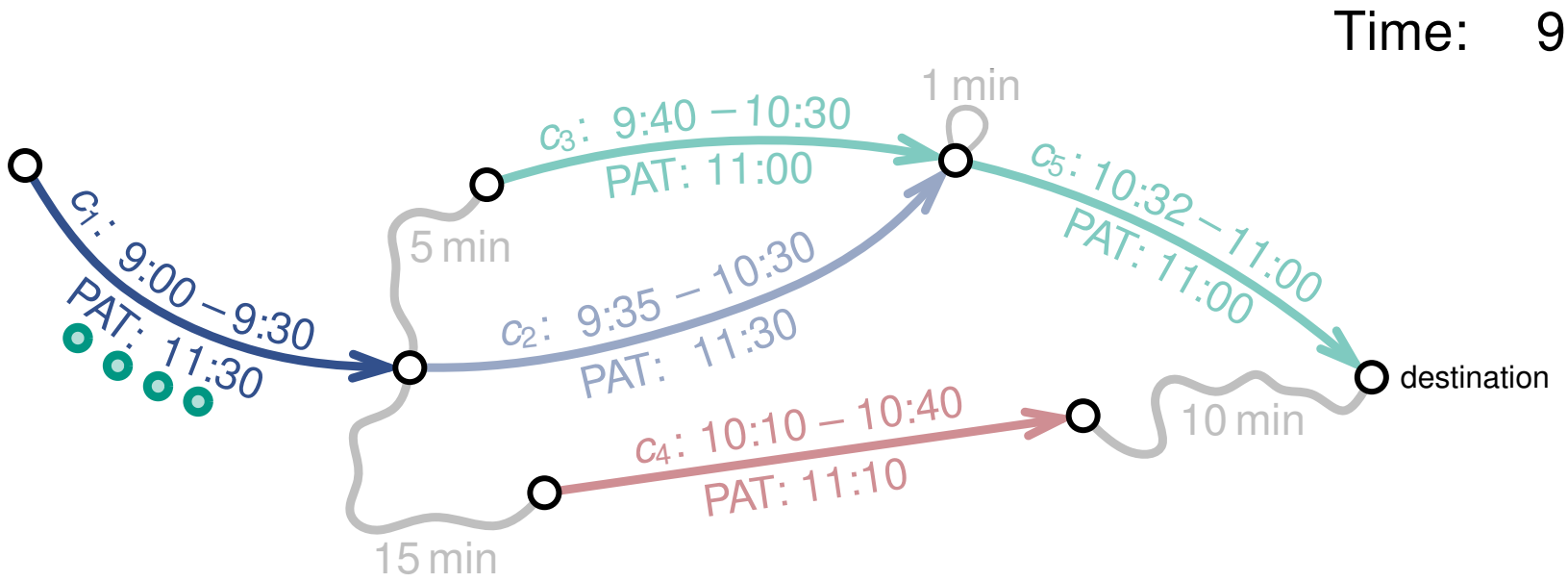
1. Erzeuge Passagiere entsprechend der Nachfrage



Beispiel Umlegungsberechnung

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

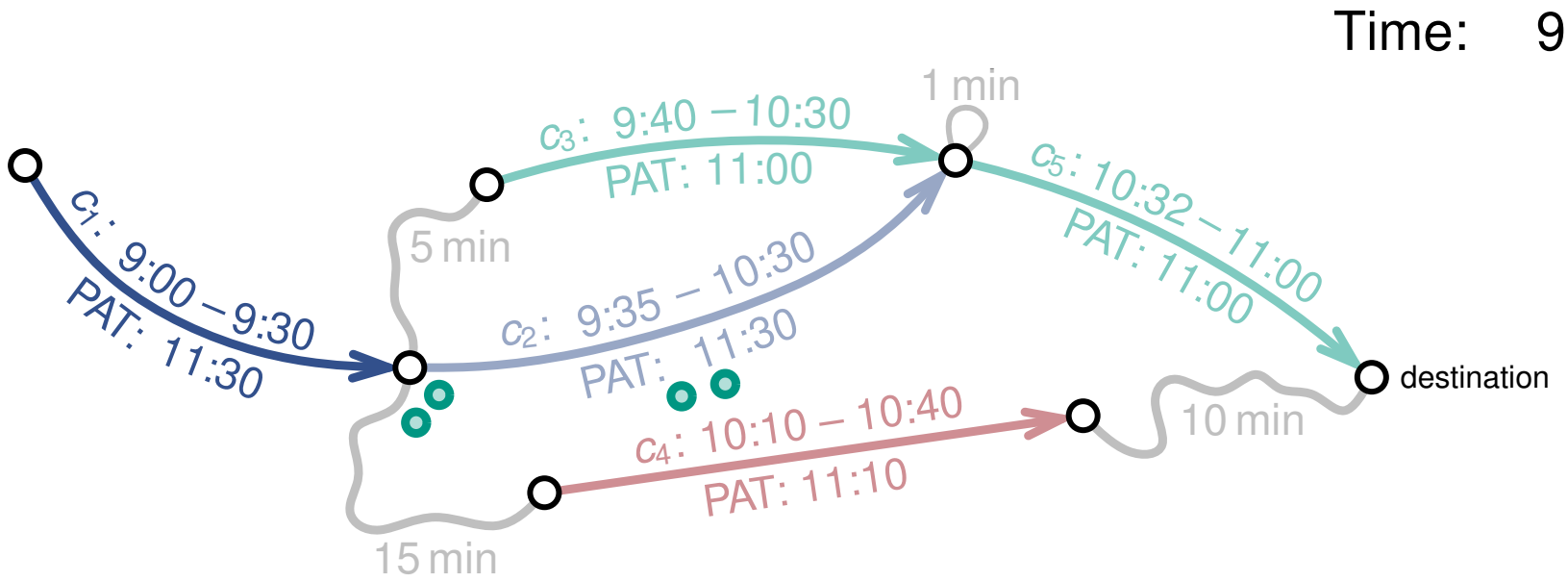
2. Entscheide, welche Passagiere einsteigen



Beispiel Umlegungsberechnung

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

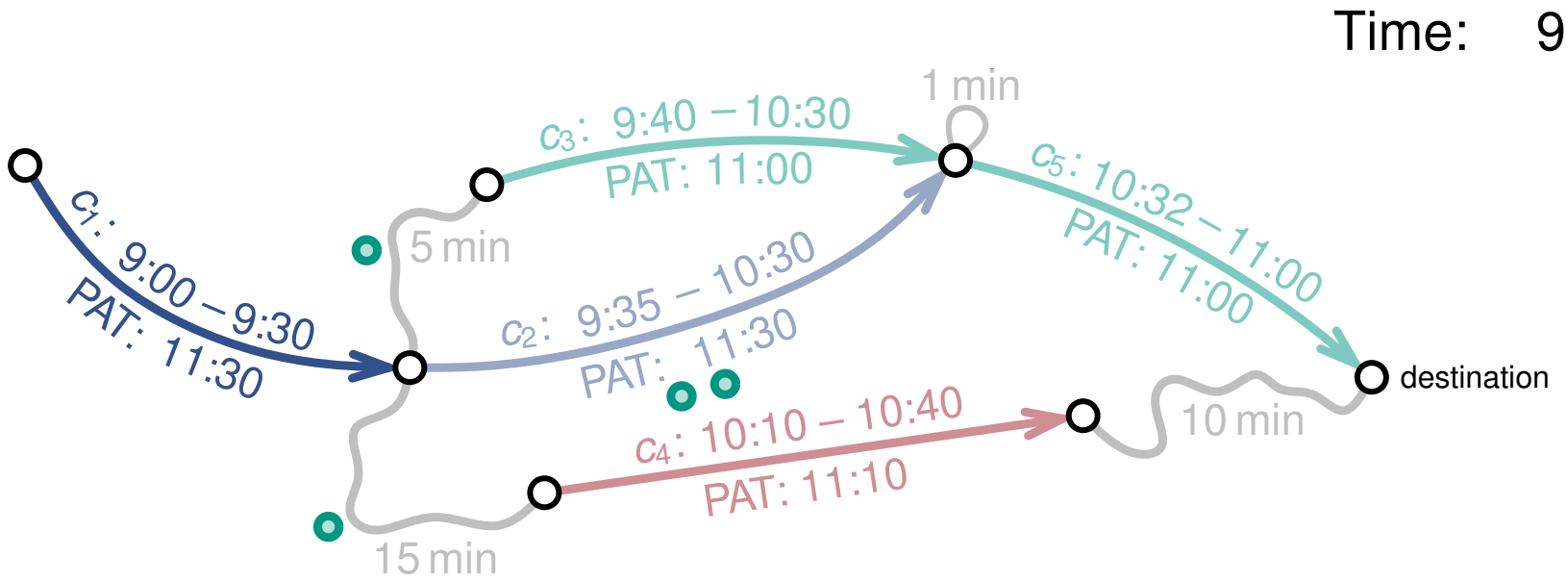
3. Entscheide, welche Passagiere aussteigen



Beispiel Umlegungsberechnung

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

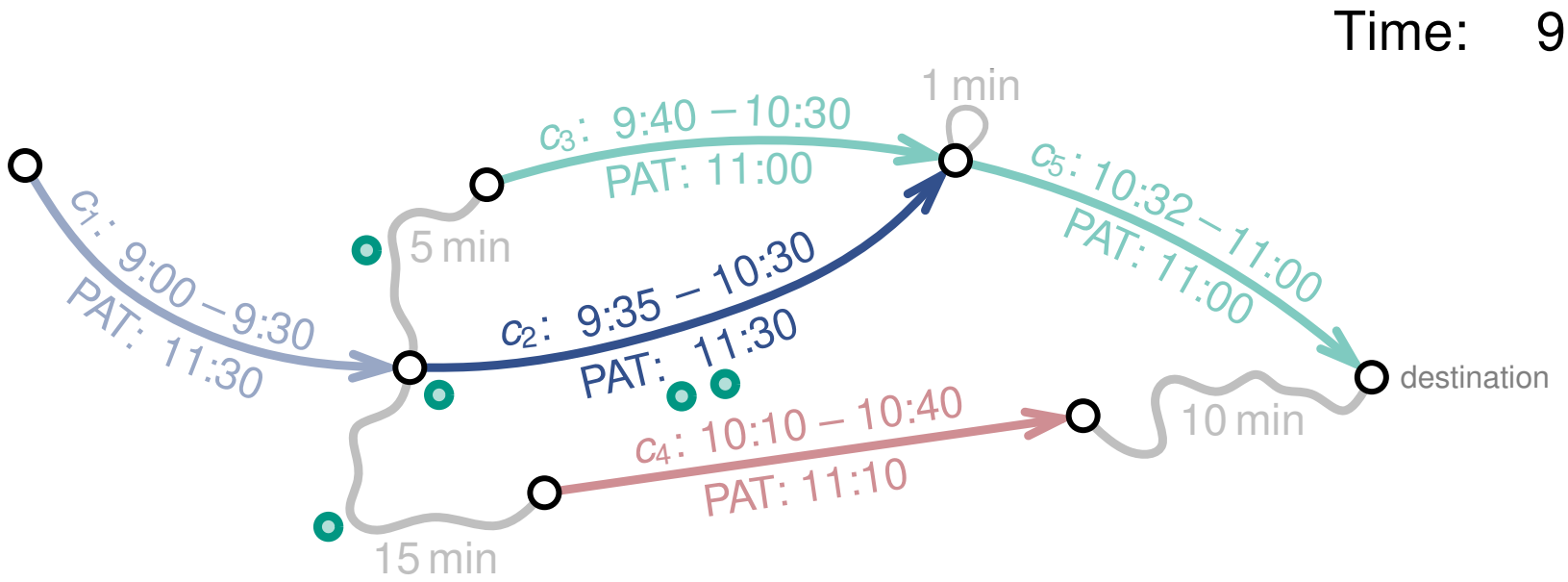
4. Verschiebe ausgestiegene Passagiere zum nächsten Stop



Beispiel Umlegungsberechnung

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

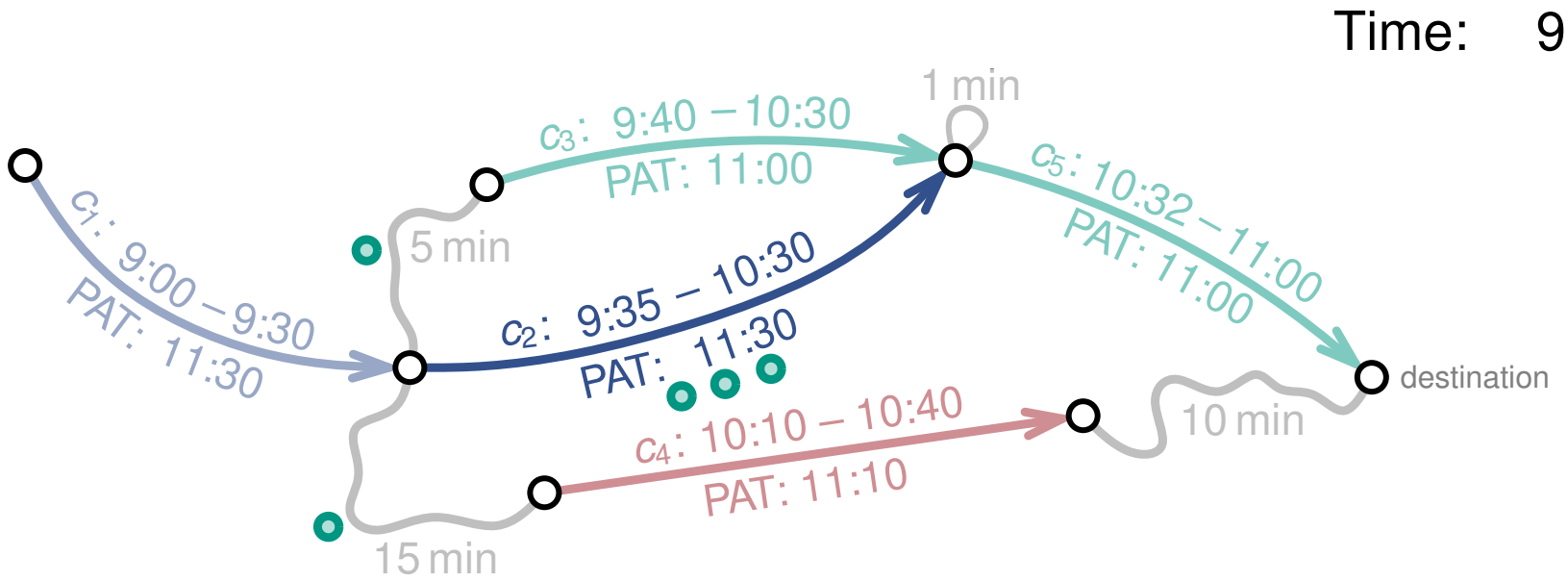
1. Erzeuge Passagiere entsprechend der Nachfrage



Beispiel Umlegungsberechnung

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

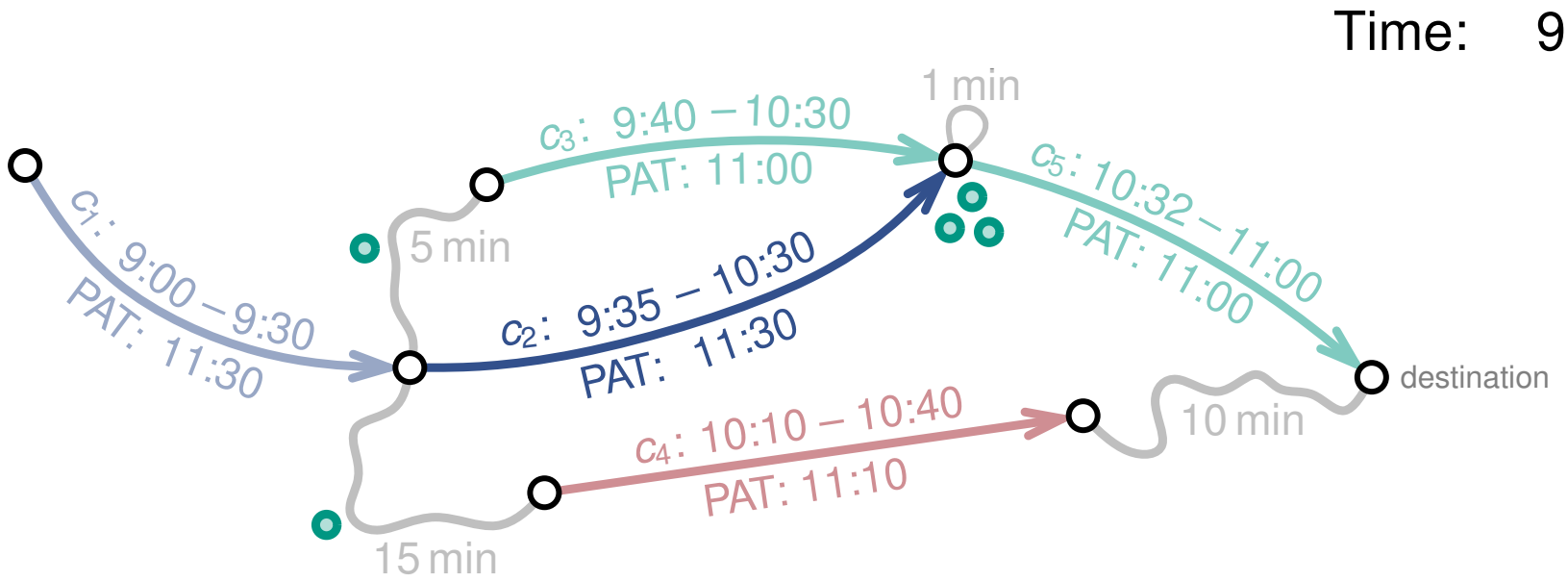
2. Entscheide, welche Passagiere einsteigen



Beispiel Umlegungsberechnung

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

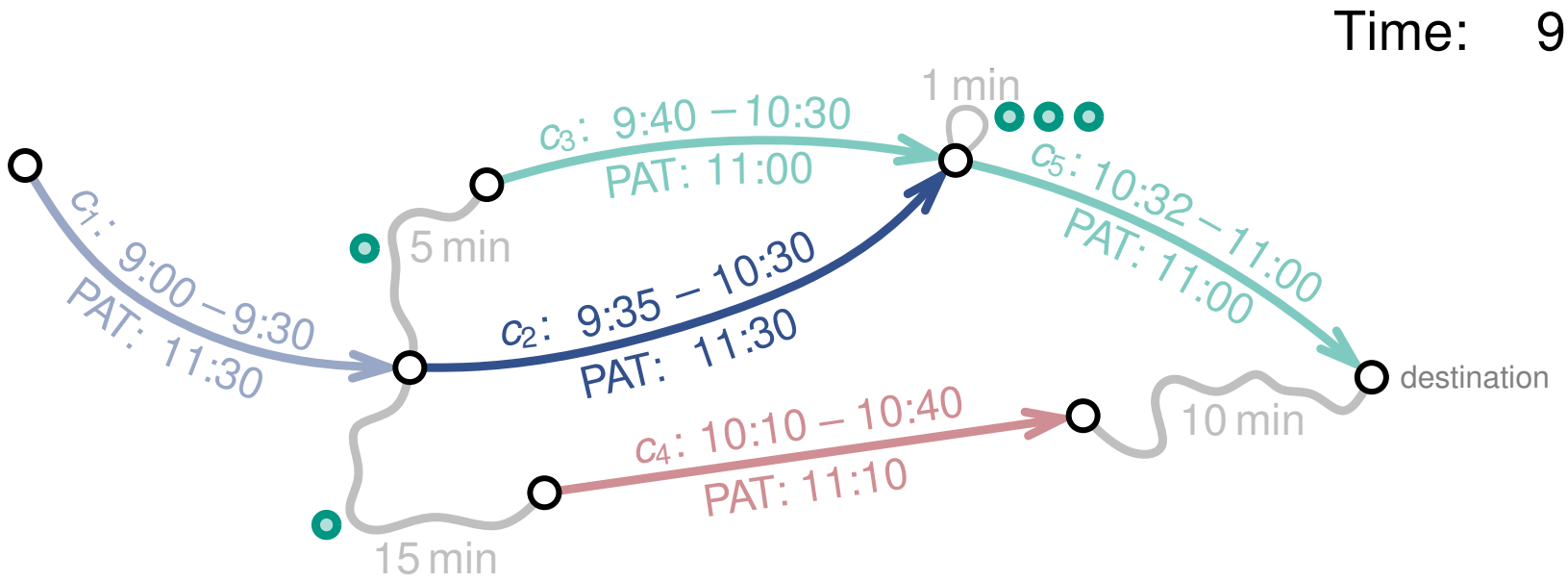
3. Entscheide, welche Passagiere aussteigen



Beispiel Umlegungsberechnung

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

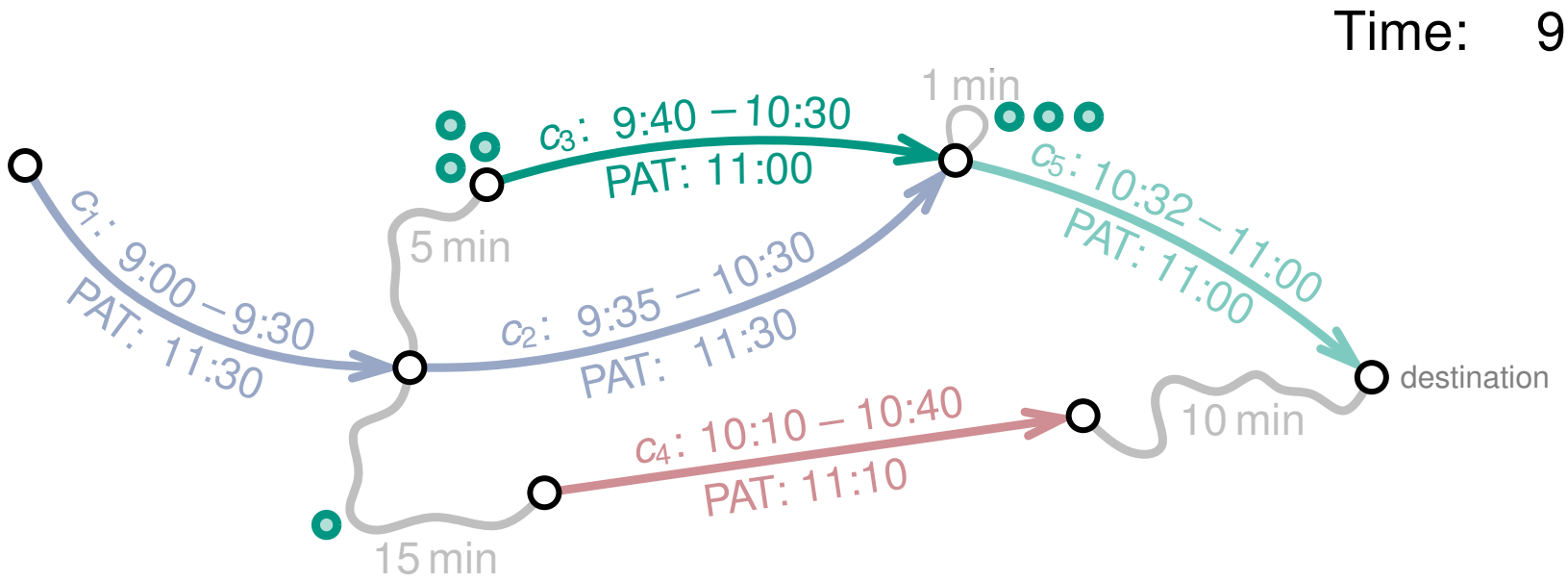
4. Verschiebe ausgestiegene Passagiere zum nächsten Stop



Beispiel Umlegungsberechnung

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

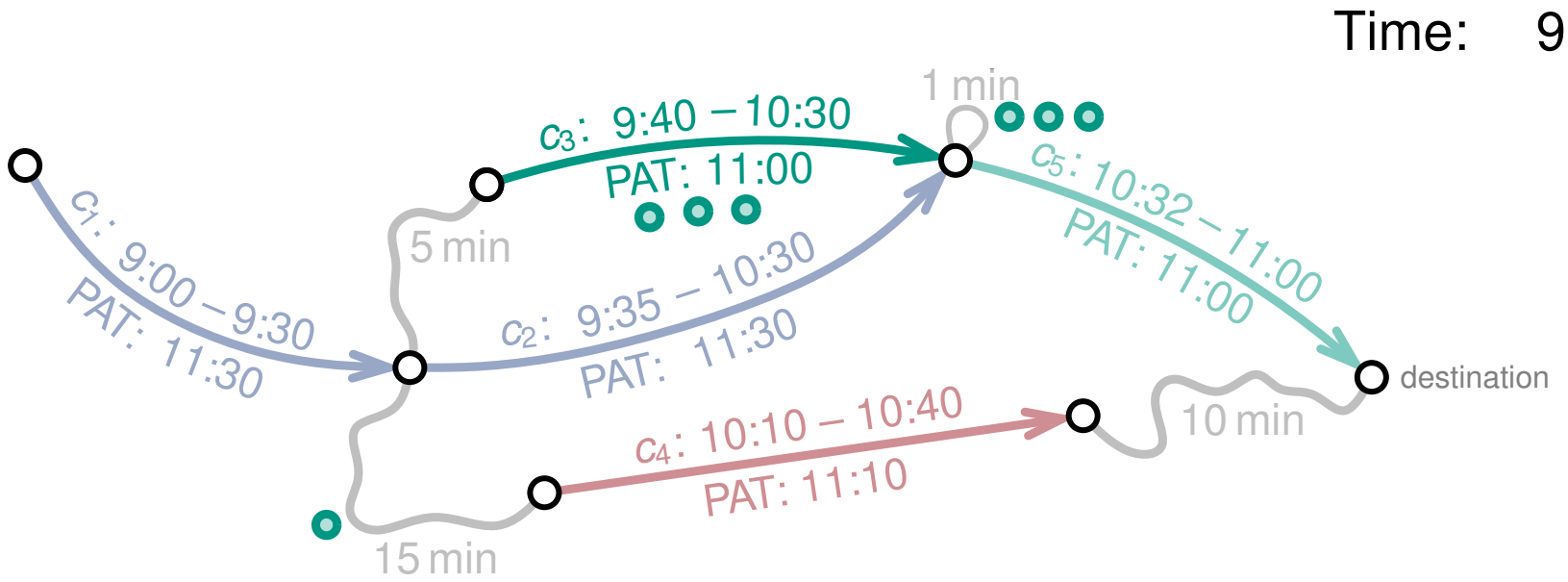
1. Erzeuge Passagiere entsprechend der Nachfrage



Beispiel Umlegungsberechnung

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

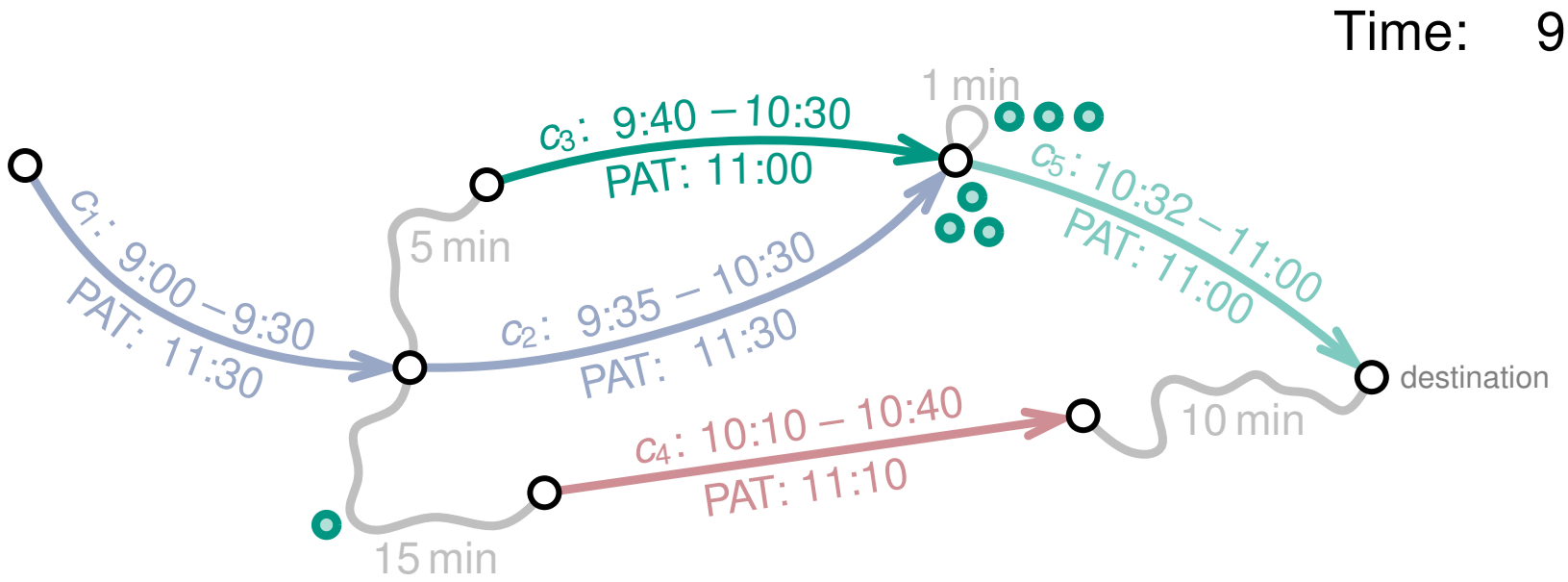
2. Entscheide, welche Passagiere einsteigen



Beispiel Umlegungsberechnung

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

3. ...

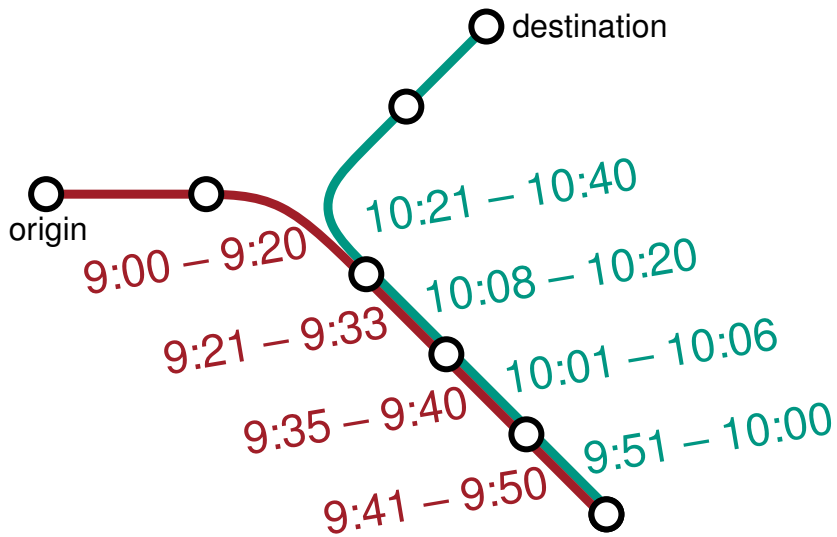


Kreise

- 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

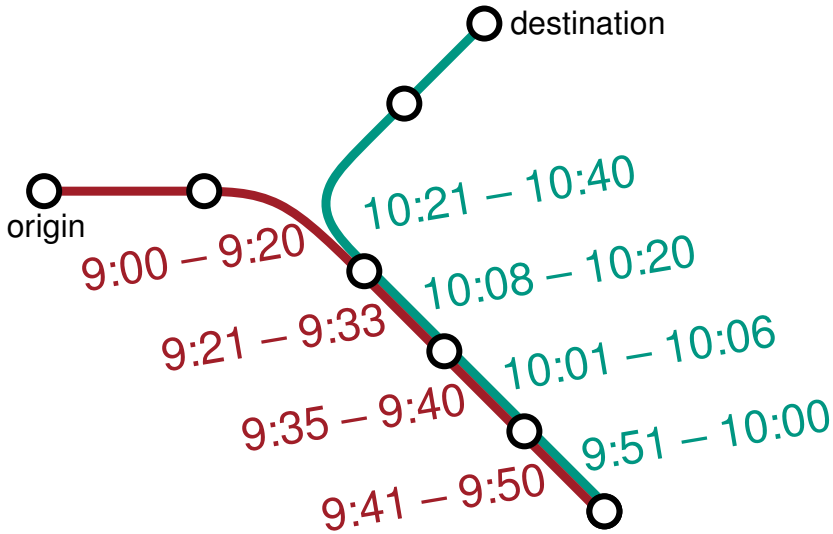
Kreise

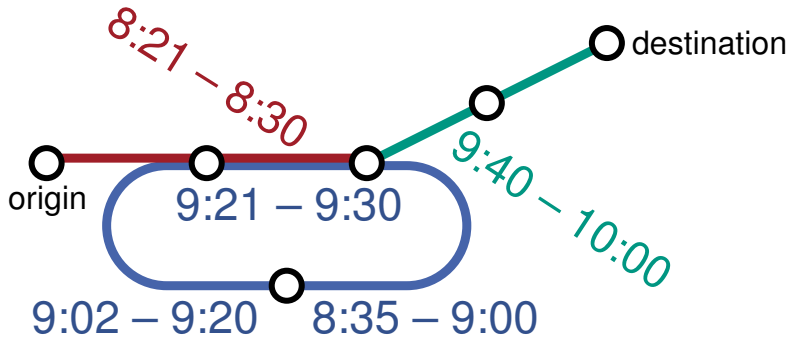
- Journeys können Kreise enthalten, d.h. Stops mehrfach besuchen
- Umlegungen mit Kreisen können unerwünscht sein
- Eine Journey mit Kreisen kann optimal bezüglich PAT sein
- Hohe Wartekosten können zu Kreisen führen



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



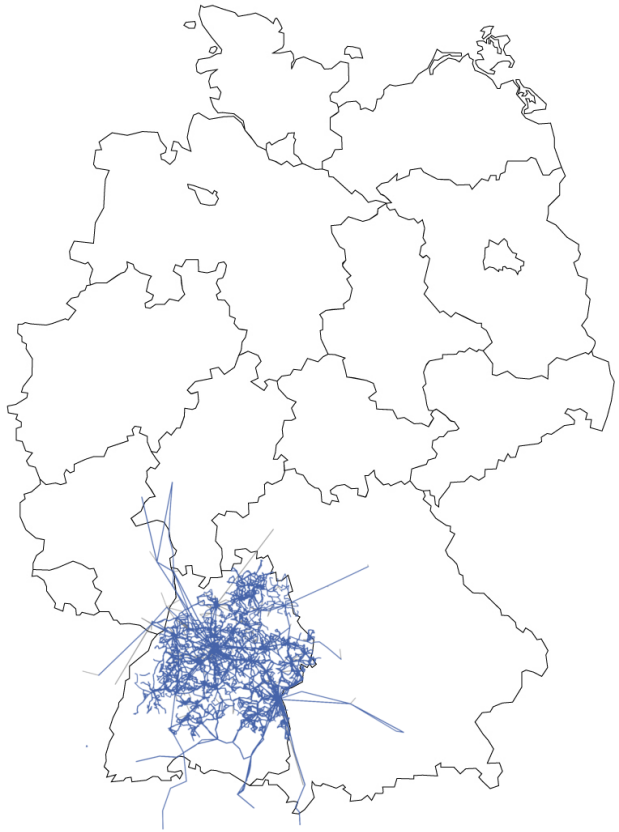


Auswertung

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



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:

- Kommerzielles Tool VISUM: Laufzeit $\approx$ 30 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

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

Literatur I

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.

Appendix: Gefühlte Ankunftszeit (PAT)

Formale Definition:

$$\tau^{\mathrm{p}}_{\mathrm{arr}}(c, d) := \min\{\tau^{\mathrm{p}}_{\mathrm{arr}}(c, d \mid \mathrm{walk}), \tau^{\mathrm{p}}_{\mathrm{arr}}(c, d \mid \mathrm{trip}), \tau^{\mathrm{p}}_{\mathrm{arr}}(c, d \mid \mathrm{trans})\}$$

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

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

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

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

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

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

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

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

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