Karlsruher Institut für Technologie

Algorithmen für Routenplanung

22. Vorlesung, Sommersemester 2022

Jonas Sauer | 13. Juli 2022

Unbeschränktes Laufen

[WZ17]

Fußwegegraphen

Bisher:

- Effiziente Algorithmen für ÖV mit eingeschränktem Laufen
- Vergleichsweise langsame multimodale Algorithmen

Einschränkungen:

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

Fußwegegraphen

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

Transitiver Abschluss

Frage

Bis zu welcher Fußweglänge ist transitiver Abschluss praktikabel?

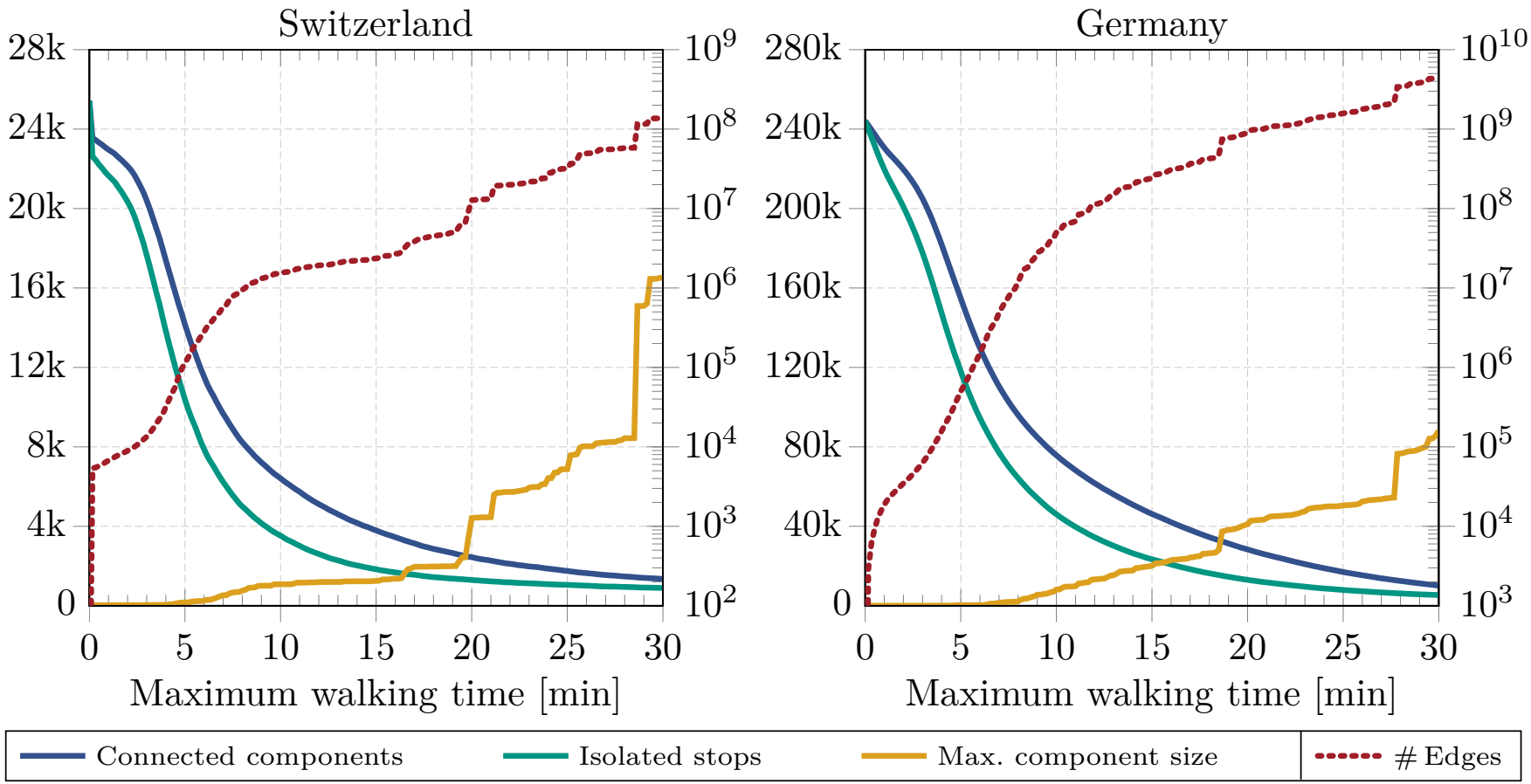
Ansatz:

- 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?

Transitiver Abschluss

Größe des transitiven Abschlusses:



Fazit

Beobachtung:

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

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)

Fazit

Beobachtung:

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

Frage

Machen längere Laufwege in der Praxis einen Unterschied?

Probleme:

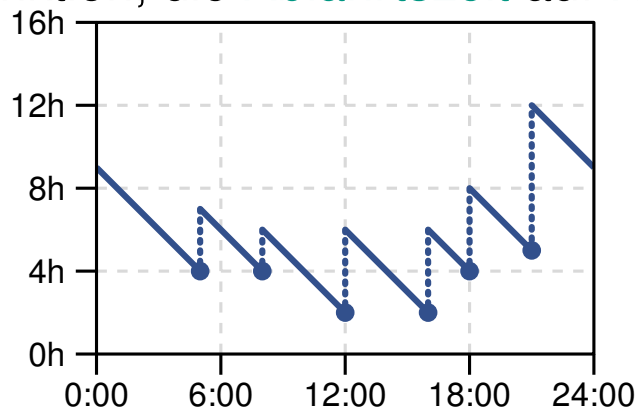
- Hängt von der Tageszeit ab (Tag vs. Nacht)
- Hängt von Start und Ziel ab (ländlich vs. städtisch)
- Hängt von der Distanz zwischen Start und Ziel ab (lang vs. kurz)

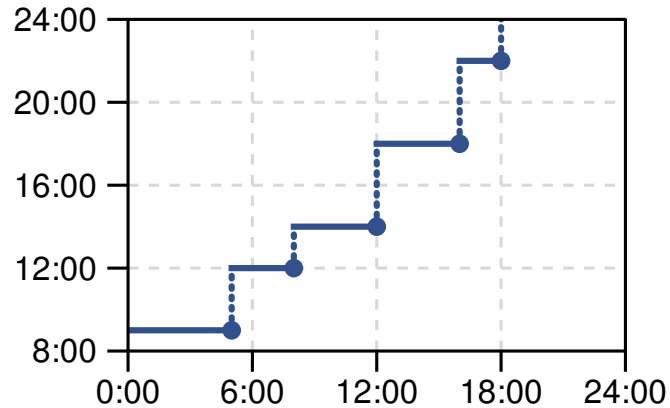
$\Rightarrow$ Auswertung von Reisezeitprofilen
auf Instanzen mit unterschiedlich vielen Fußwegen
für viele s–t-Paare

Profile & Profil-Algorithmen

Wiederholung Profil:

- Funktion, die Abfahrtszeit auf Reisezeit oder Ankunftszeit abbildet





Profil-Algorithmen für unbeschränktes Laufen:

- Reine Public-Transit-Algorithmen sind nicht geeignet
- Multimodal Multicriteria RAPTOR (MCR)
  - Kann mit unbeschränktem Laufen umgehen
  - Unterstützt aber nur Queries mit fester Abfahrtszeit

Profile – RAPTOR & MCR

MCR Profil-Algorithmus:

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

Profile – RAPTOR & MCR

MCR Profil-Algorithmus:

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

rRAPTOR Profil-Algorithmus: (Wiederholung)

- Profileinträge sind durch Abfahrtszeit des ersten Trips bestimmt
- Sammle alle Abfahrtszeiten am Start ein
- Führe RAPTOR für jede Abfahrtszeit einmal aus

Profile – RAPTOR & MCR

MCR Profil-Algorithmus:

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

rRAPTOR Profil-Algorithmus: (Wiederholung)

- Profileinträge sind durch Abfahrtszeit des ersten Trips bestimmt
- 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

Alternating RAPTOR (aRAPTOR)

Ziel:

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

Alternating RAPTOR (aRAPTOR)

Ziel:

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

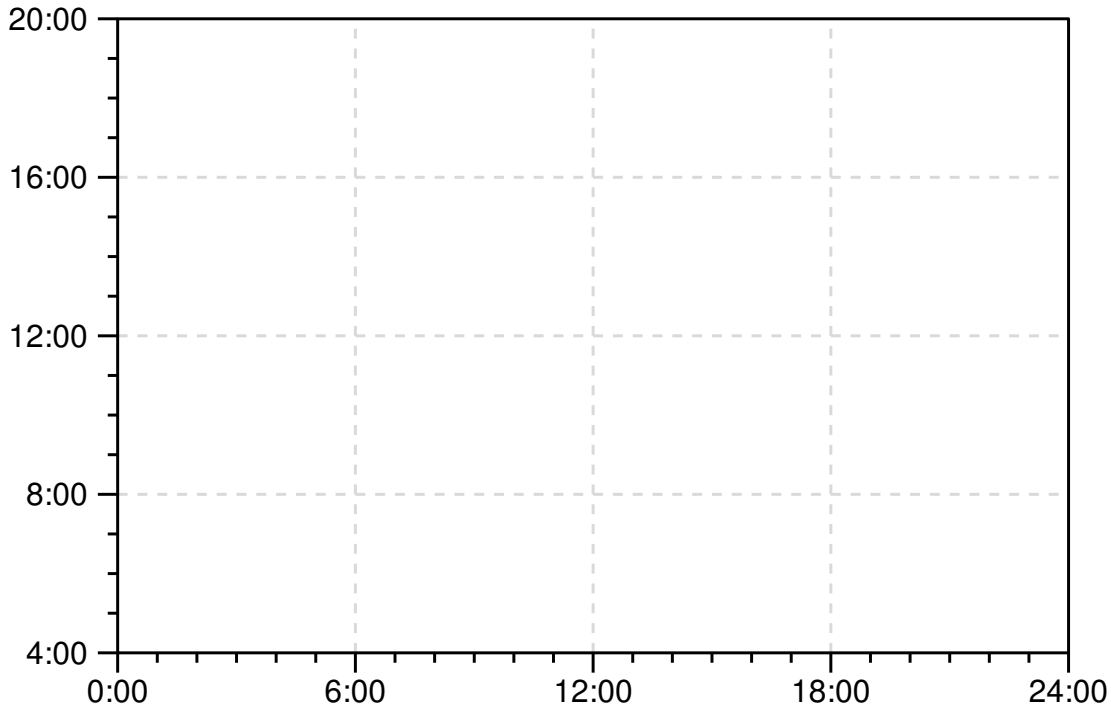
Vorgehen:

- 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

aRAPTOR – Beispiel

Ziel:

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

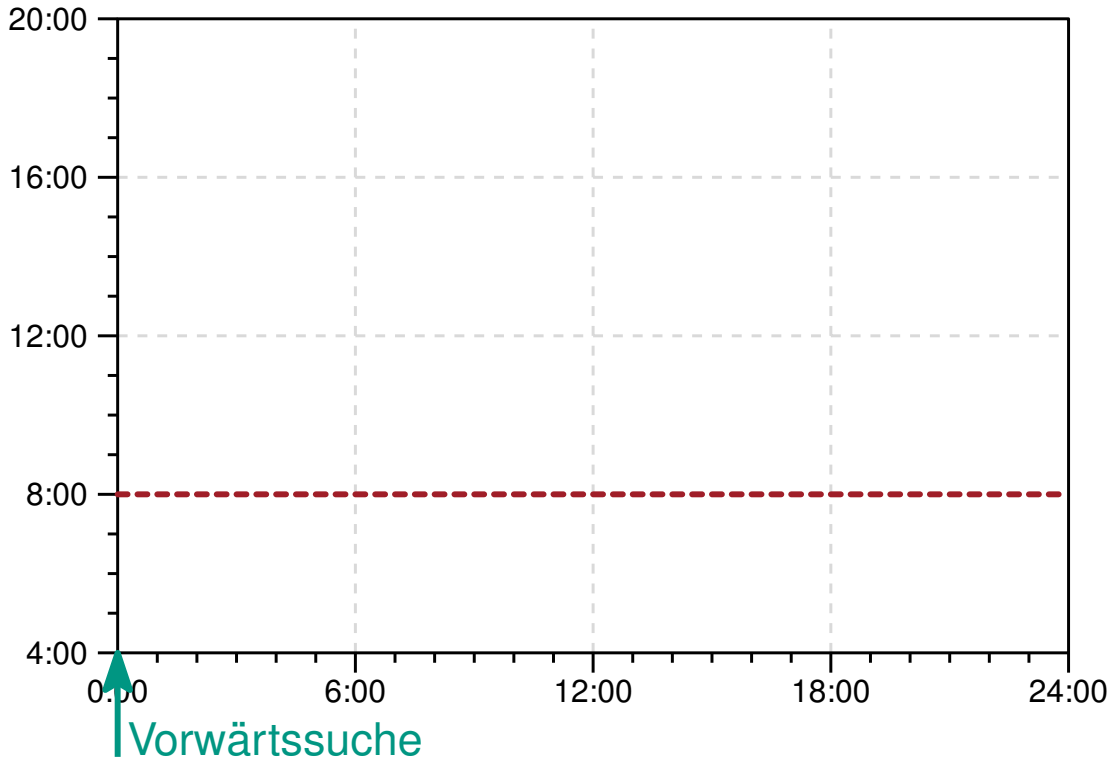


aRAPTOR – Beispiel

Ziel:

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

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

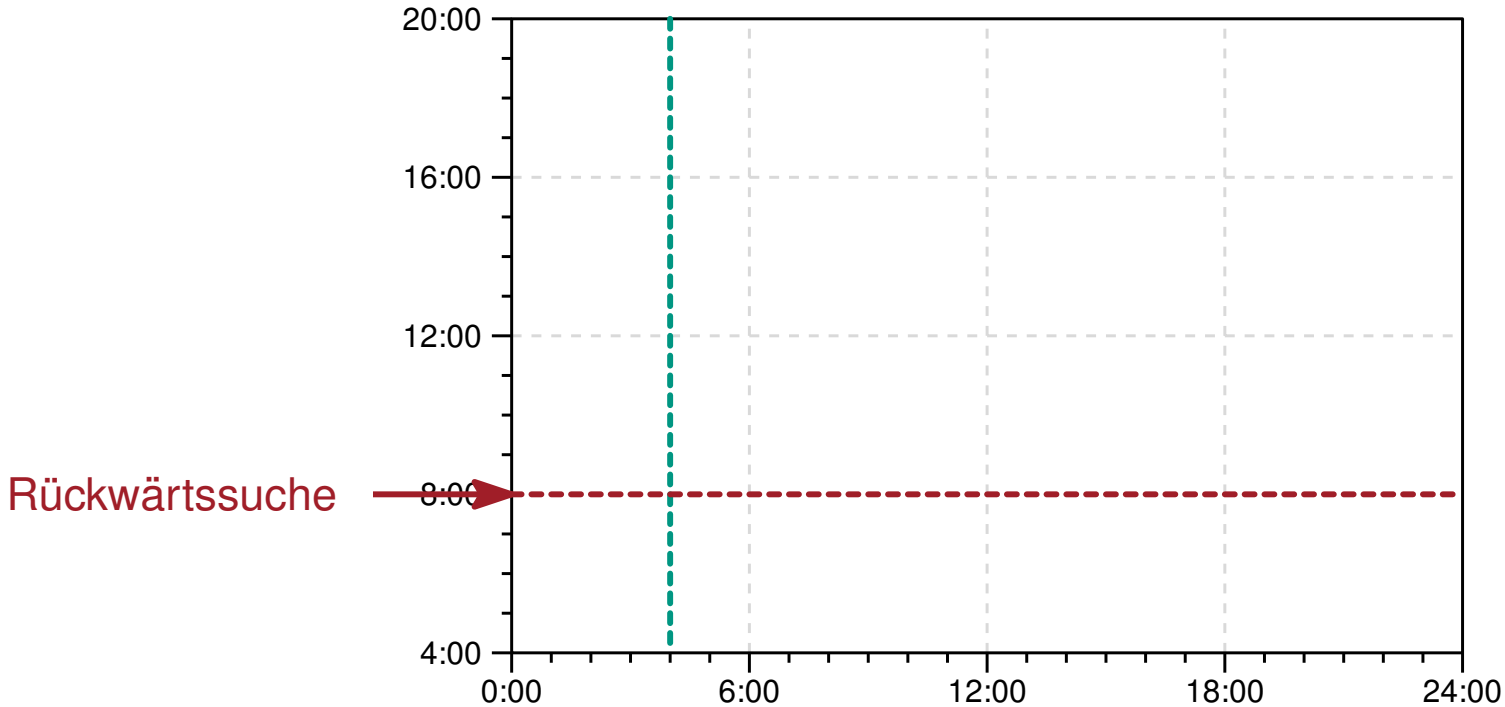


aRAPTOR – Beispiel

Ziel:

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

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

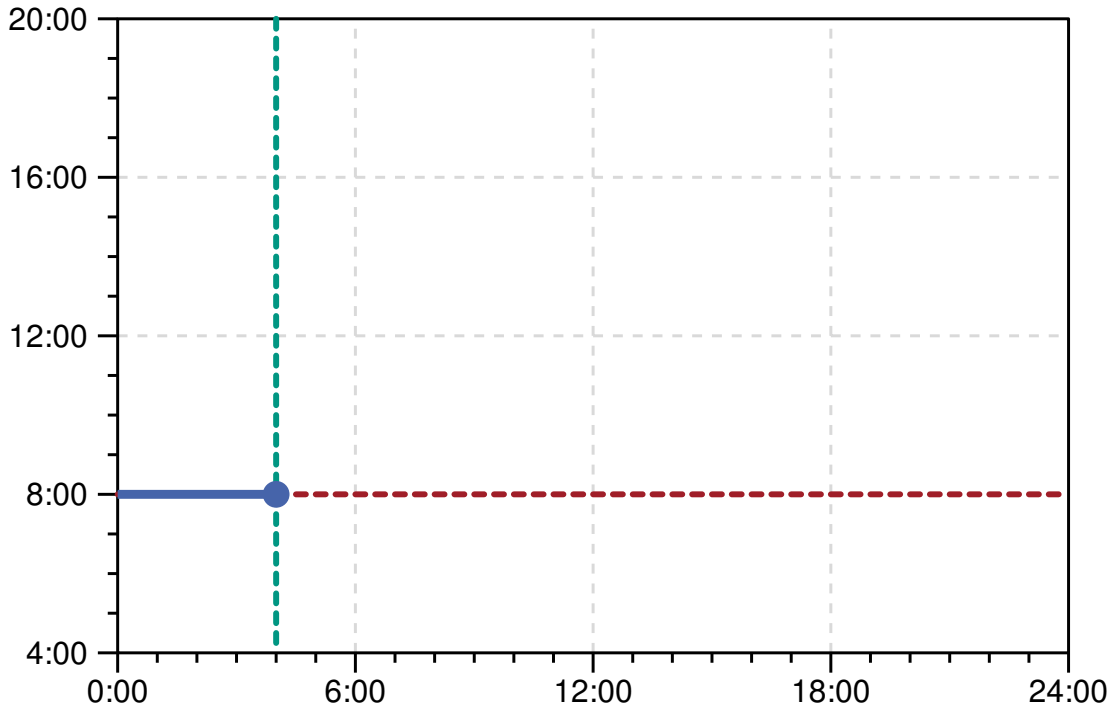


aRAPTOR – Beispiel

Ziel:

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

Schritt 3: Beide Ergebnisse zusammen bilden einen Profileintrag

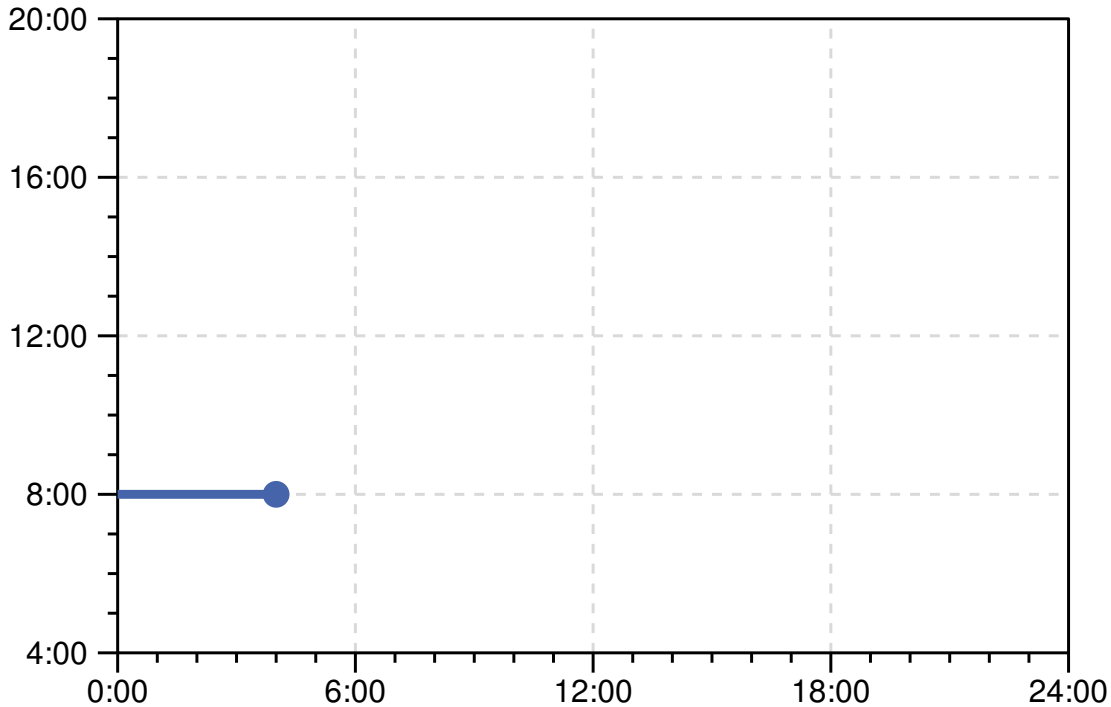


aRAPTOR – Beispiel

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]

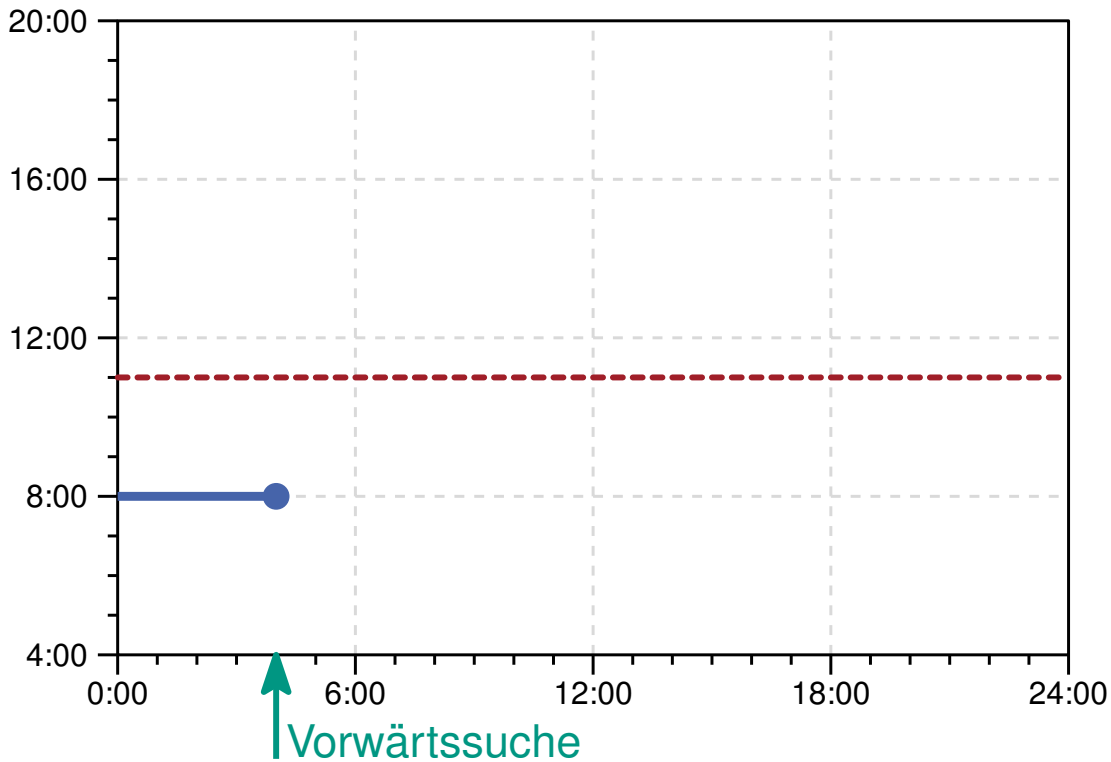


aRAPTOR – Beispiel

Ziel:

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

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

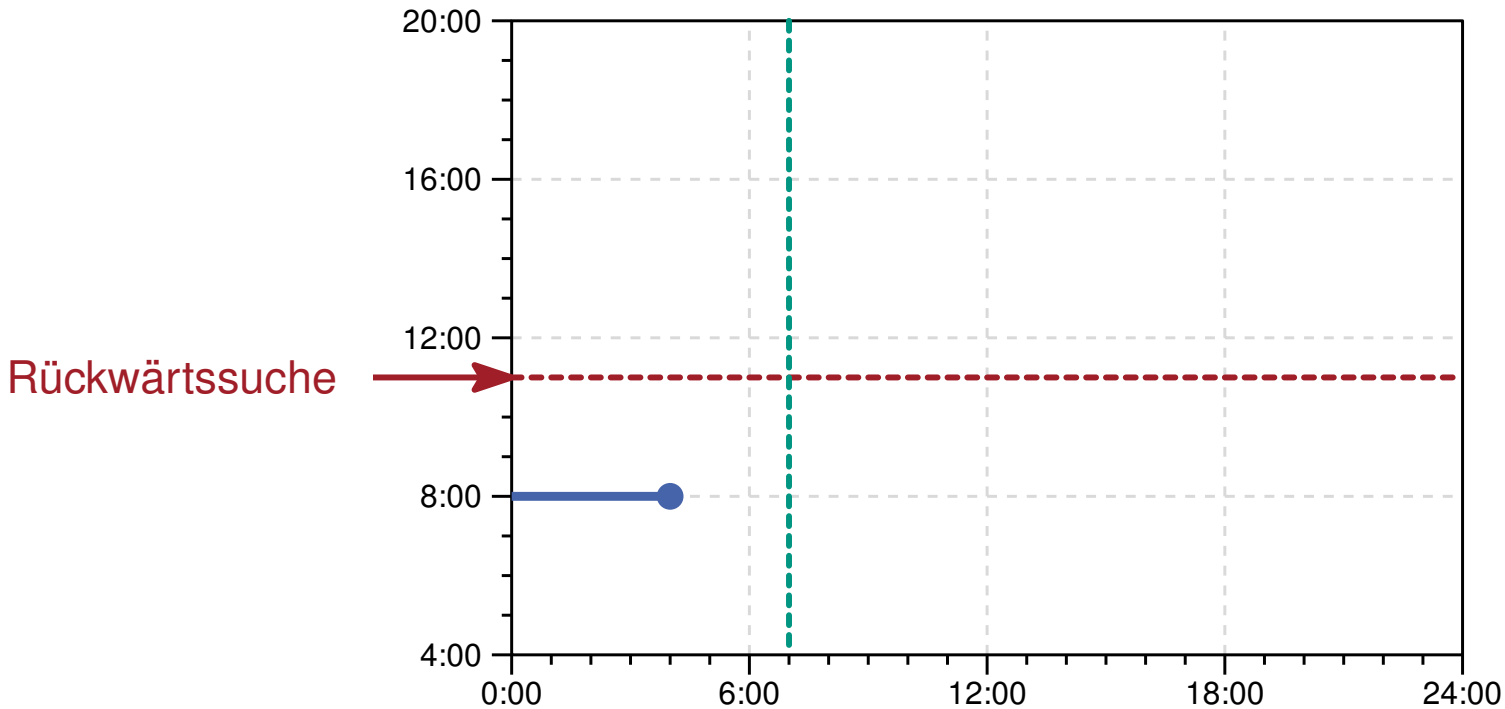


aRAPTOR – Beispiel

Ziel:

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

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

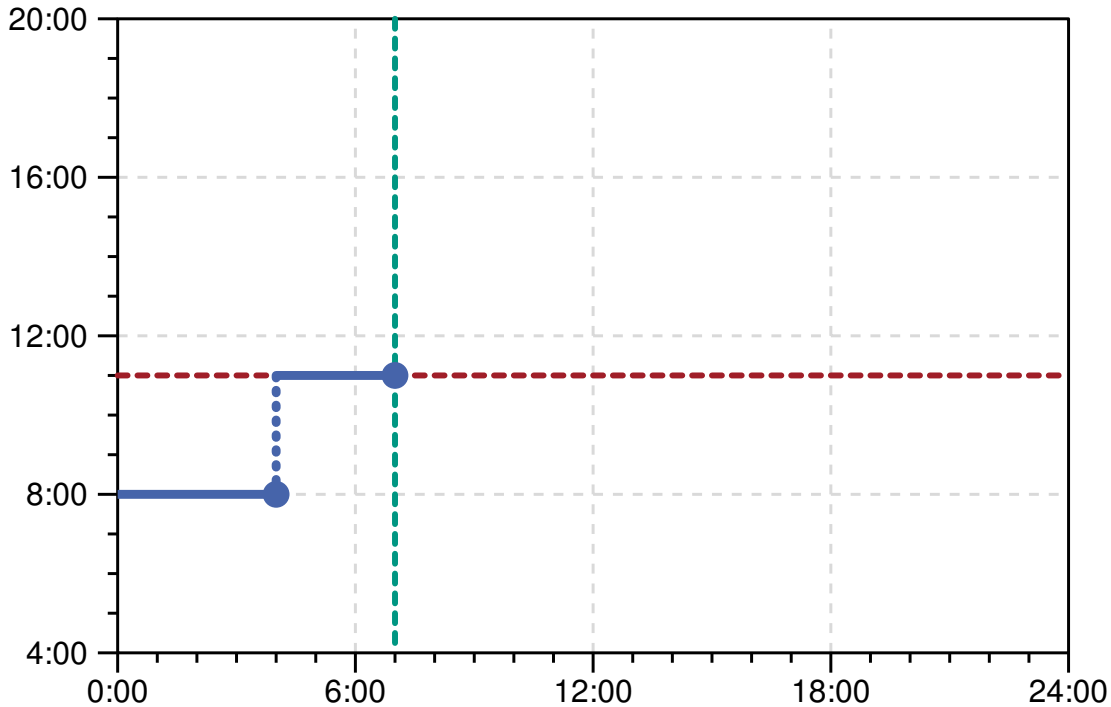


aRAPTOR – Beispiel

Ziel:

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

Schritt 3: Beide Ergebnisse zusammen bilden einen Profileintrag

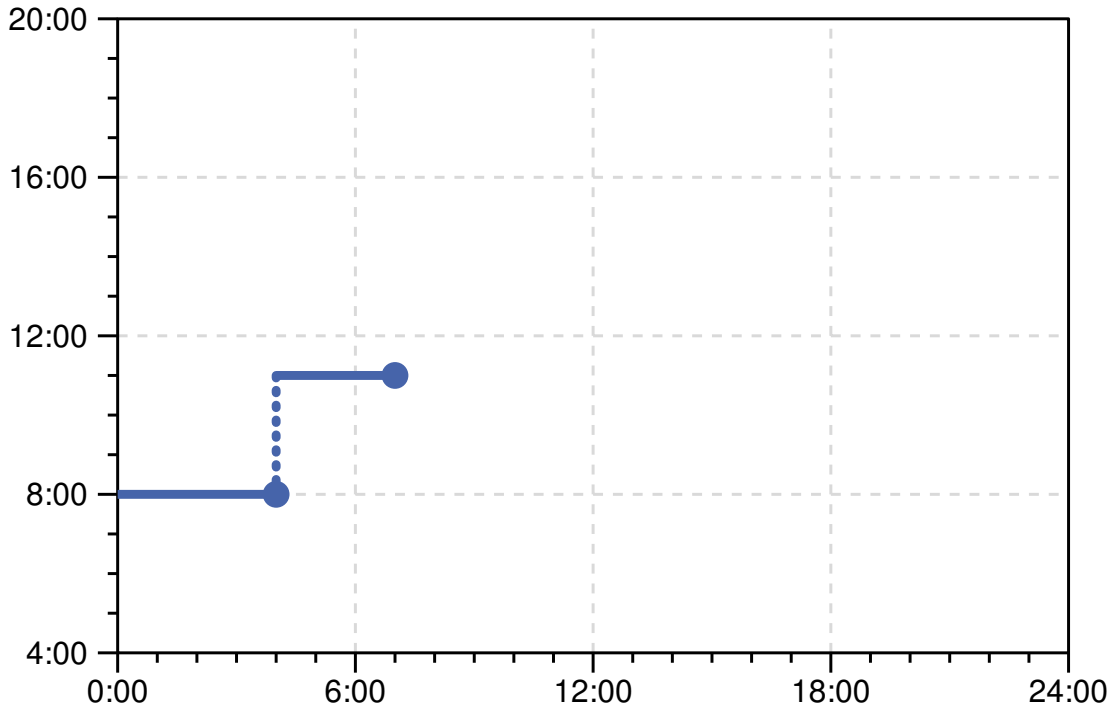


aRAPTOR – Beispiel

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]

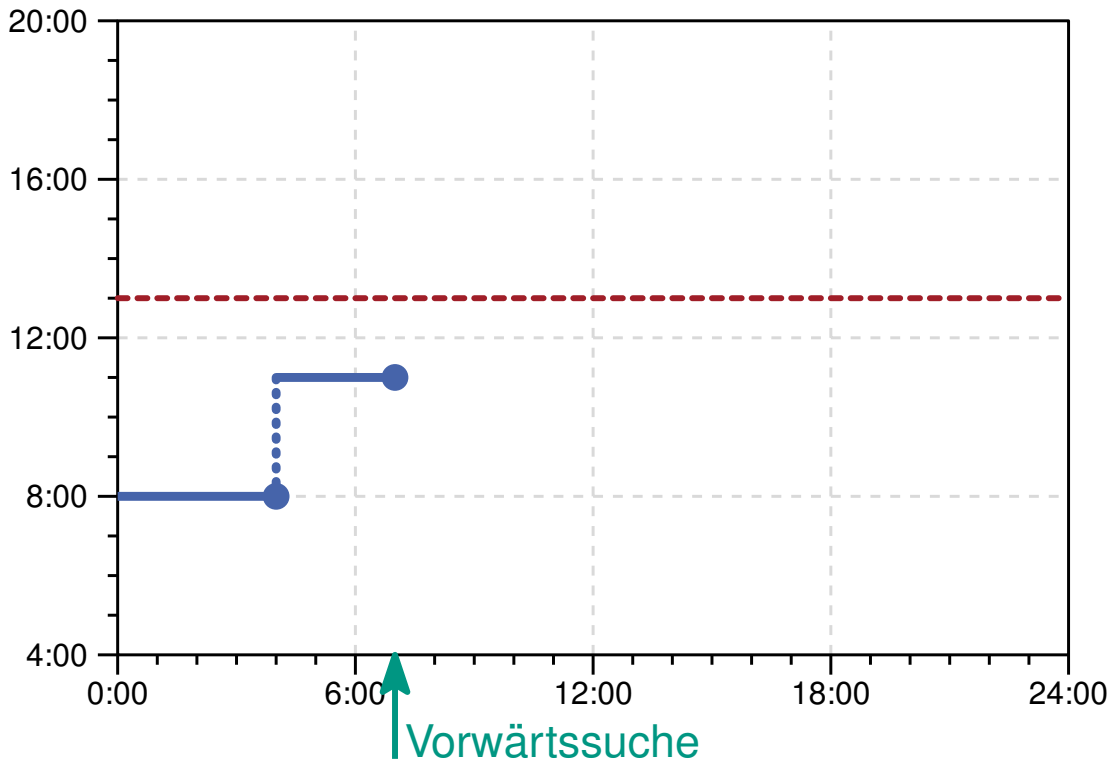


aRAPTOR – Beispiel

Ziel:

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

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

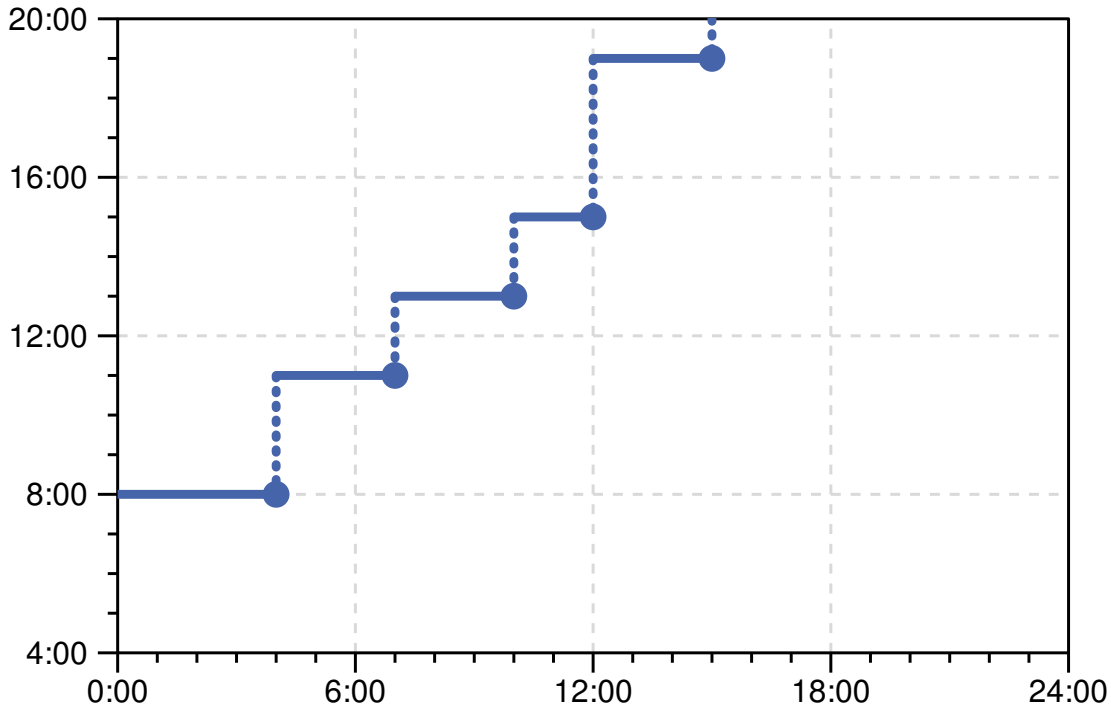


aRAPTOR – Beispiel

Ziel:

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

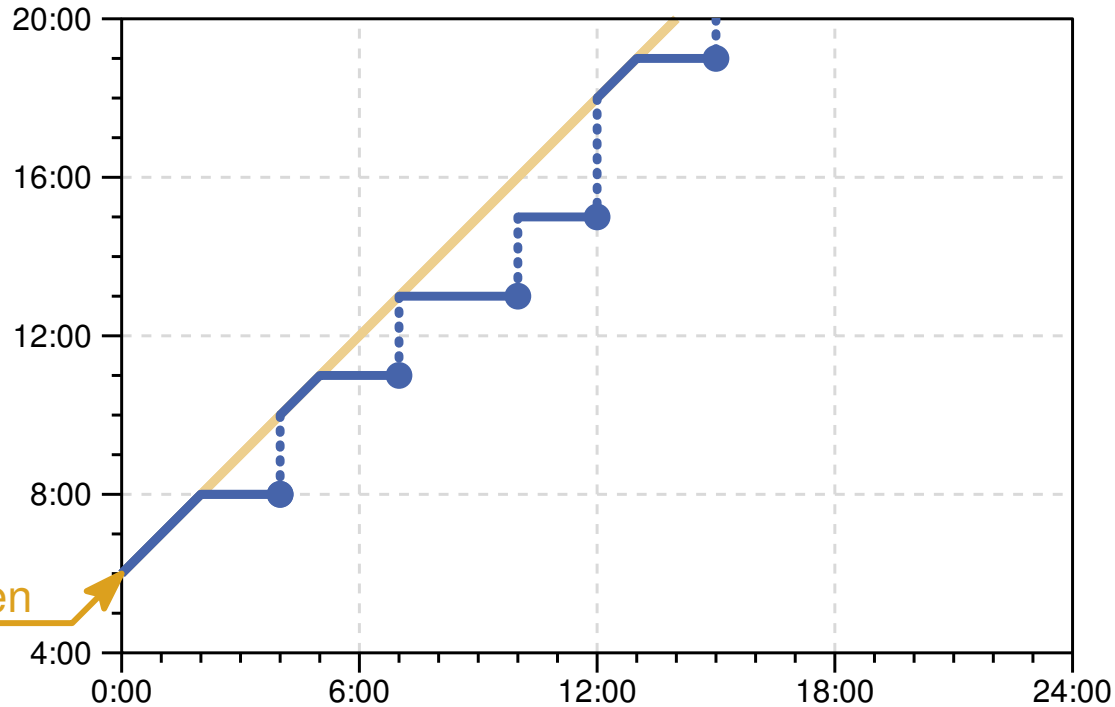
Ende: Der Algorithmus endet, wenn das gesamte Profil berechnet wurde



aRAPTOR – Direktes Laufen

Problem:

- Direktes Laufen verhindert Fortschritt



Beispiel:

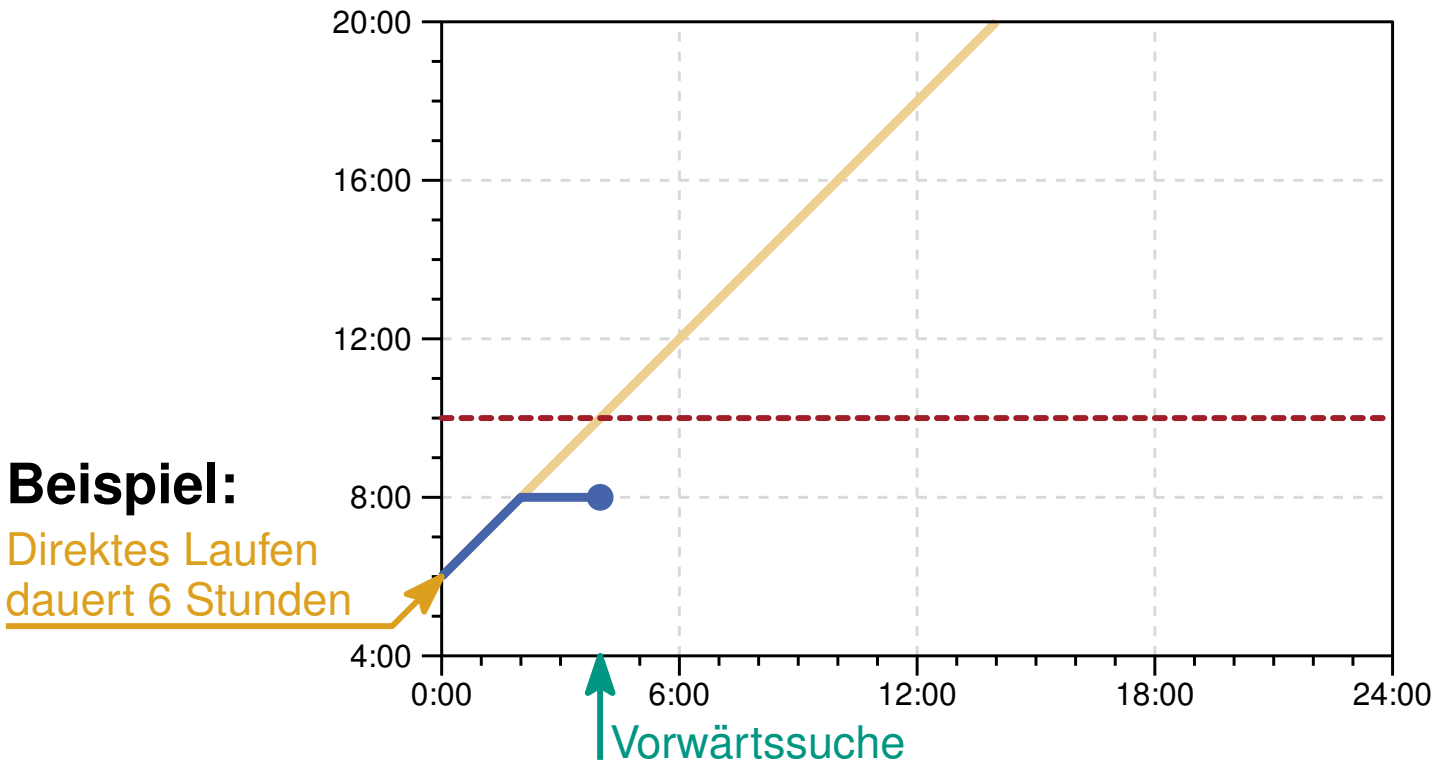
Direktes Laufen dauert 6 Stunden

aRAPTOR – Unbeschränktes Laufen

Ziel:

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

Schritt 1: Vorwärtssuche für frühestmögliche Abfahrtszeit ($4{:}00 + \varepsilon$)

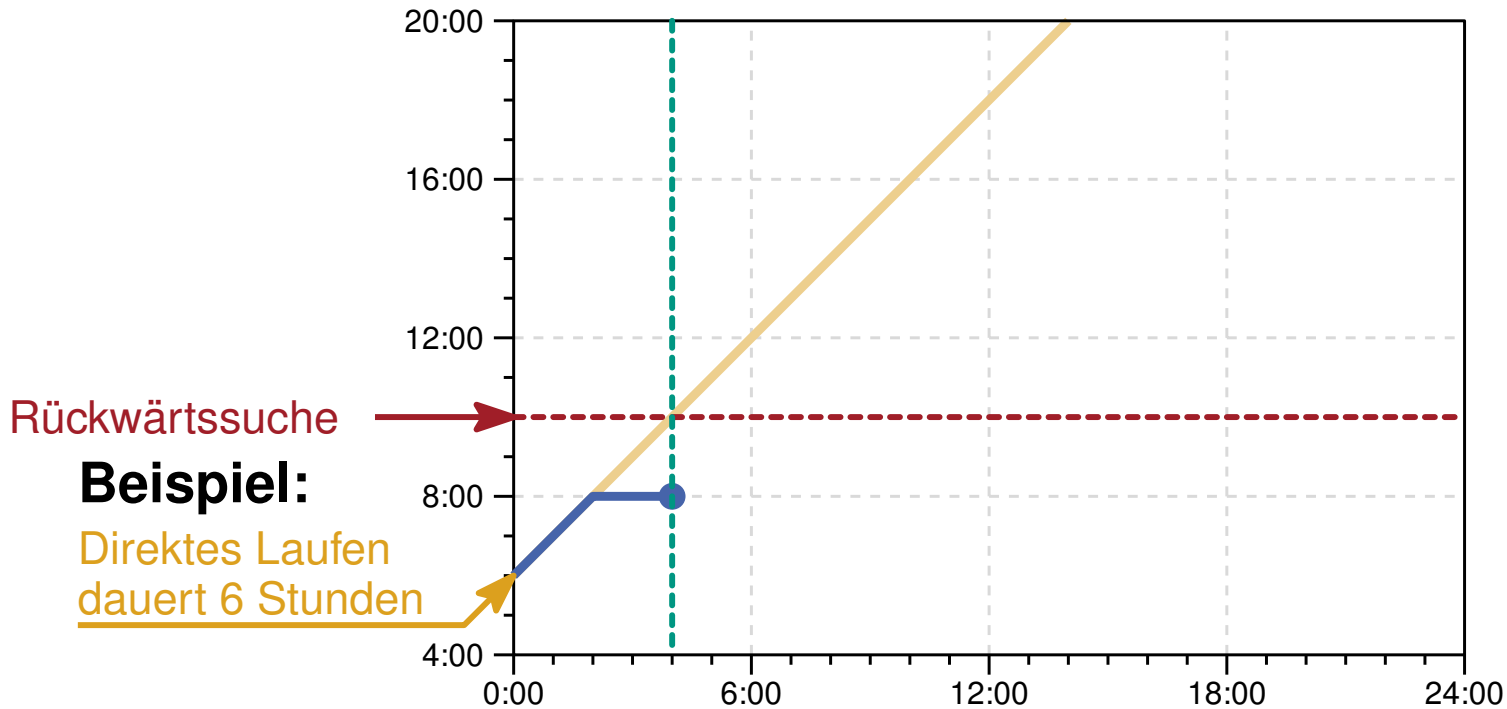


aRAPTOR – Unbeschränktes Laufen

Ziel:

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

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

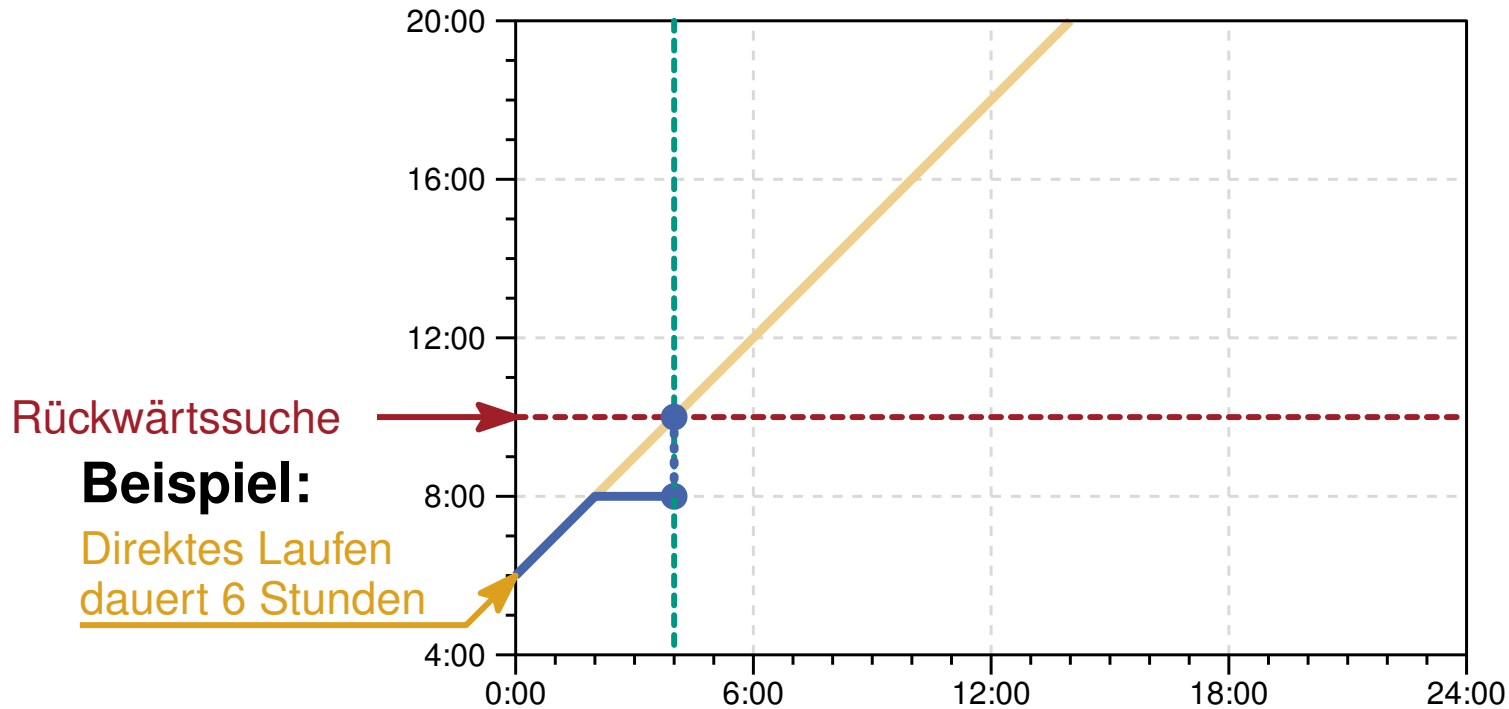


aRAPTOR – Unbeschränktes Laufen

Ziel:

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

Schritt 3: Beide Ergebnisse zusammen bilden einen Profileintrag

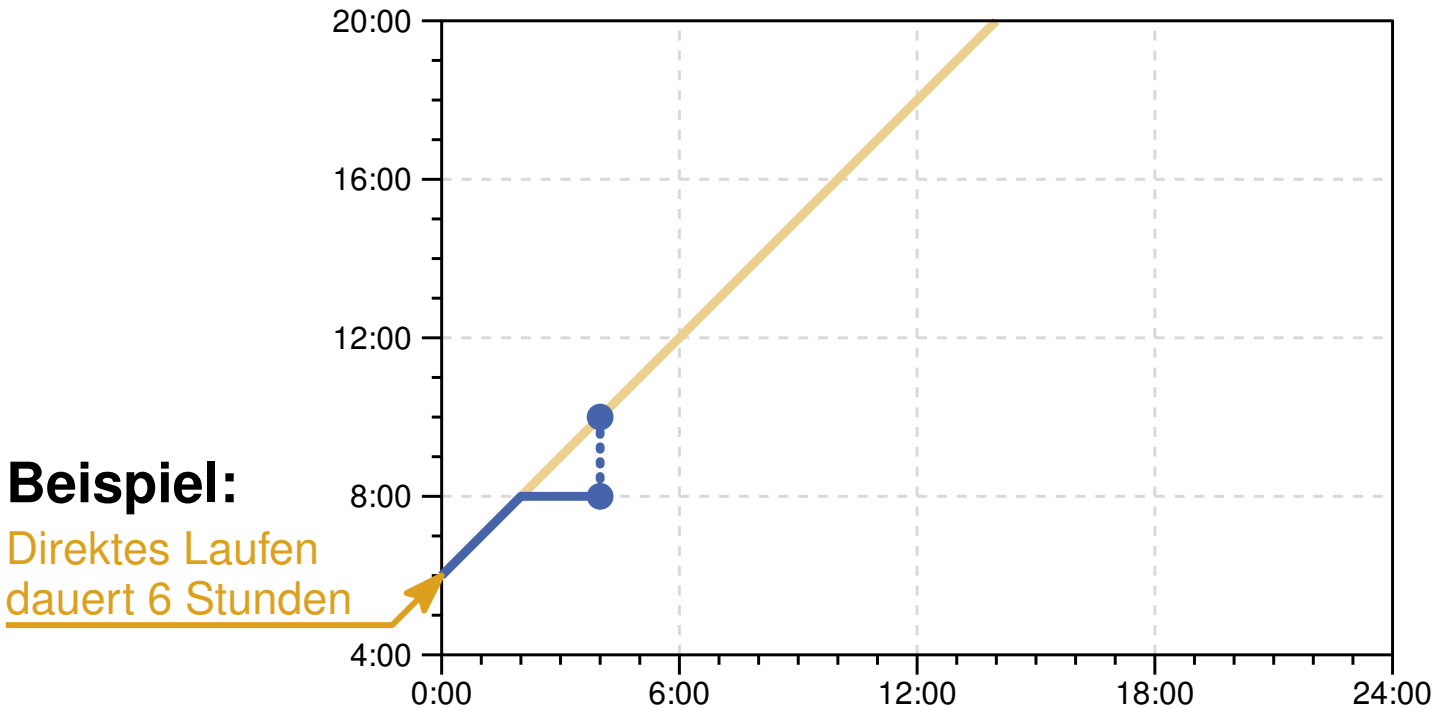


aRAPTOR – Unbeschränktes Laufen

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]

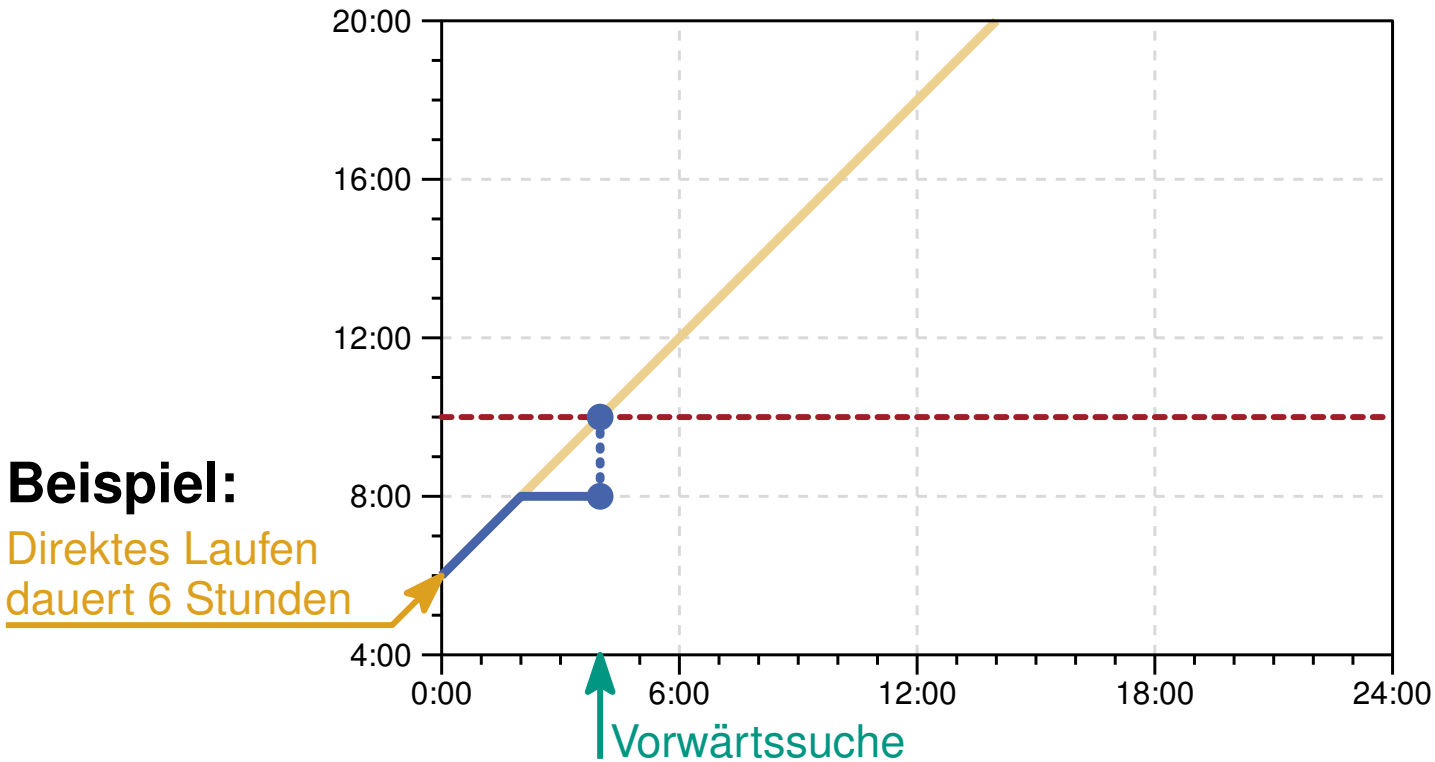


aRAPTOR – Unbeschränktes Laufen

Ziel:

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

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



aRAPTOR – Unbeschränktes Laufen

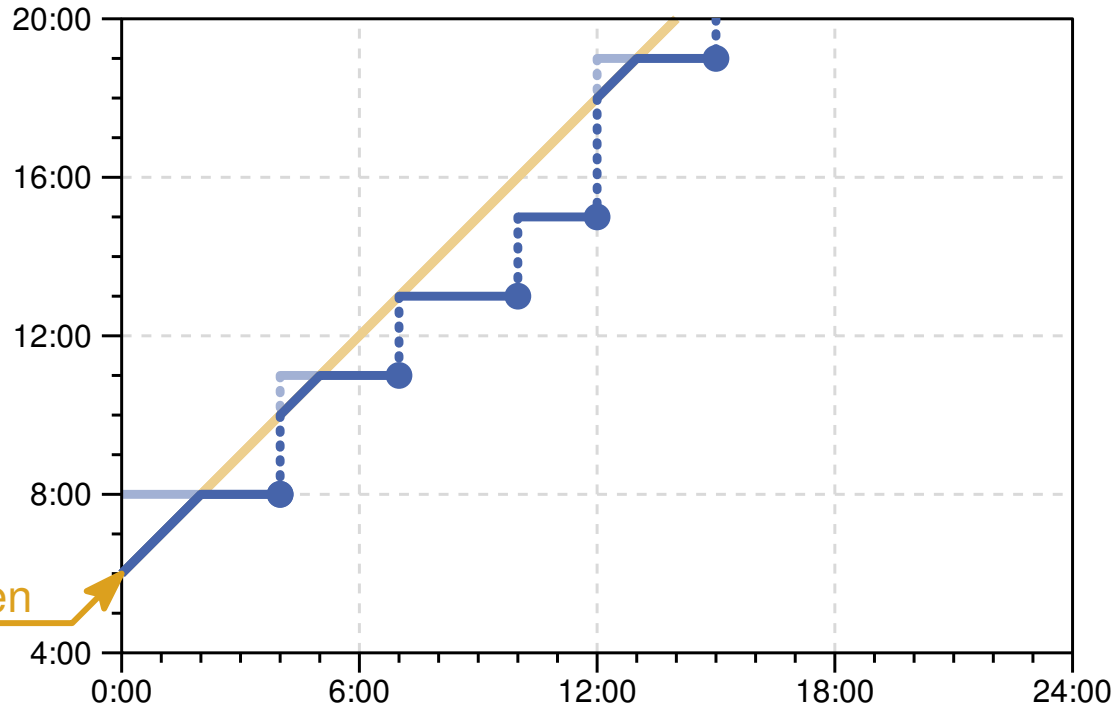
Ziel:

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

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 für verschiedene Fußwegegraphen

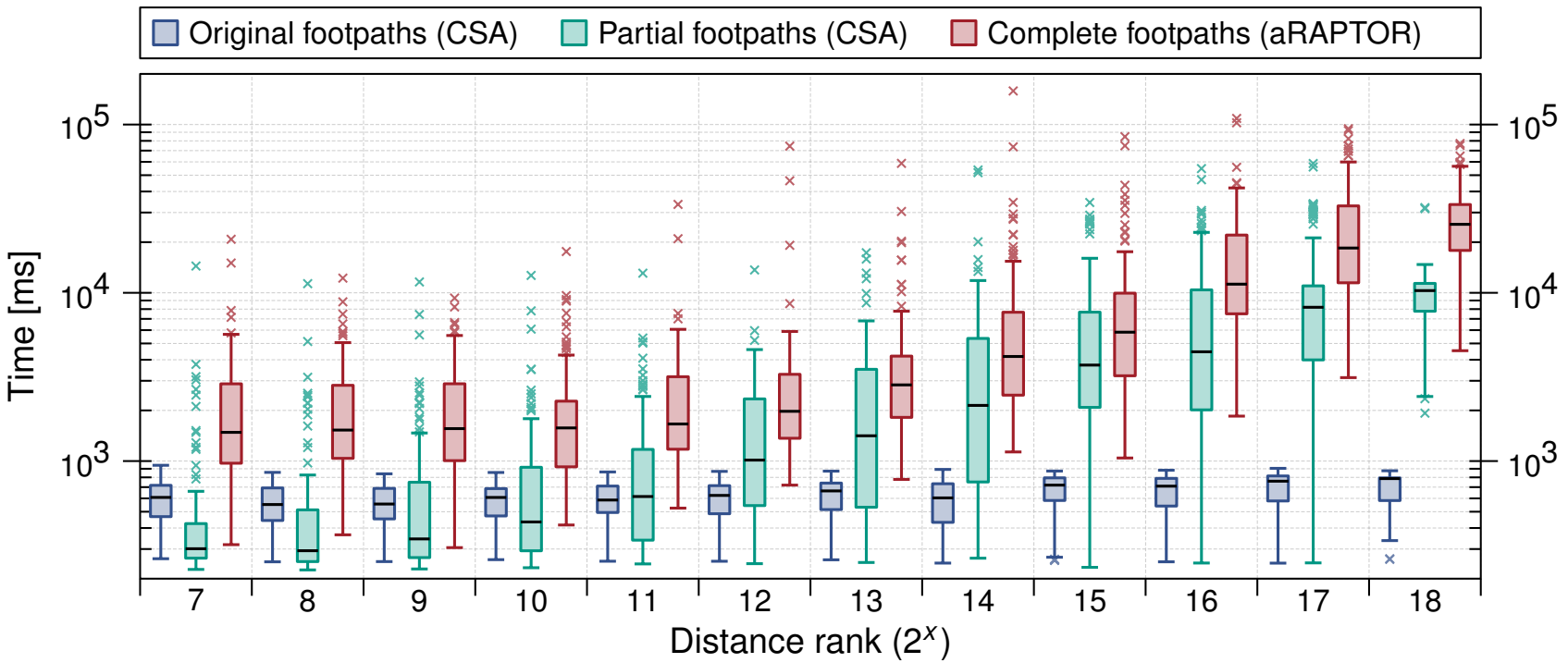
- **original:** Nur im PT-Netzwerk spezifizierte Fußwege
- **partial:** Transitiv abgeschlossener Fußwegegraph
- **complete:** Vollständiger Fußwegegraph

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

Experimente

Vergleich der Anfragezeiten:

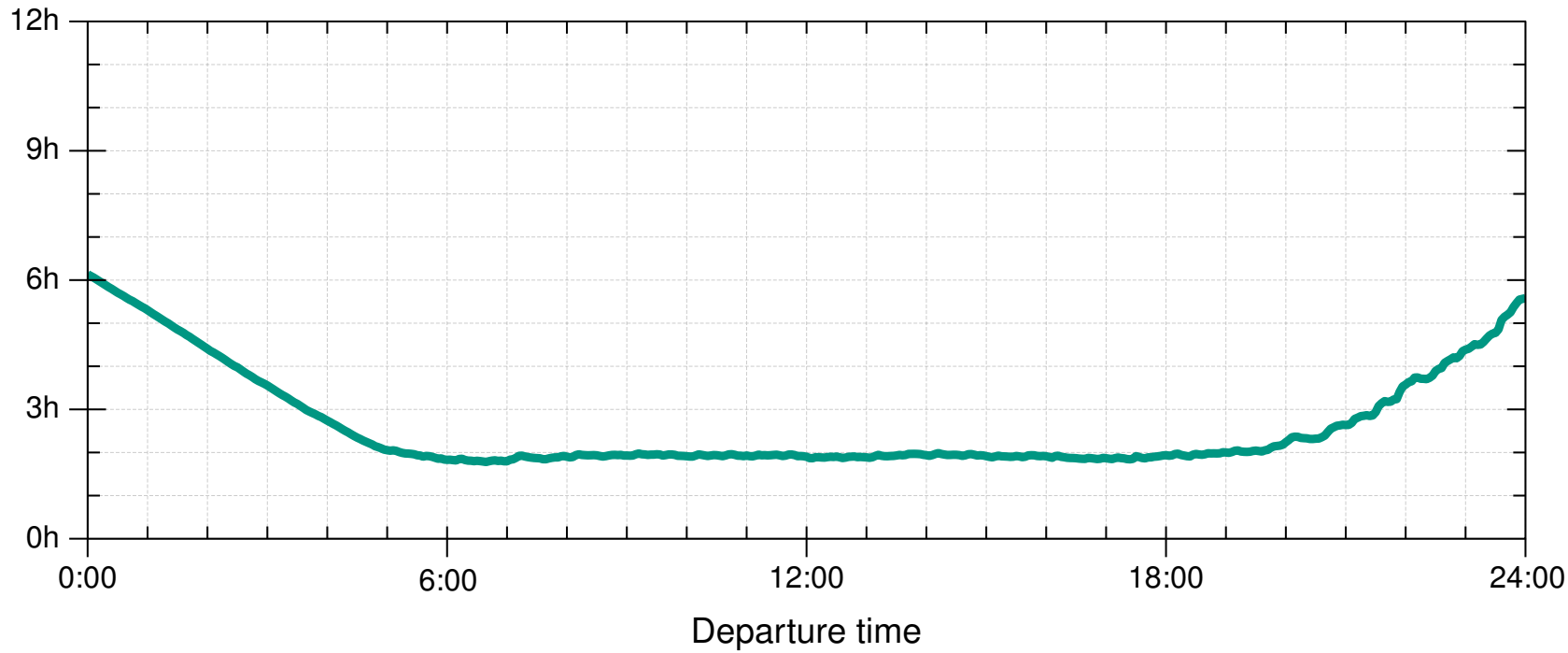
- Auf dem Schweizer Netzwerk
- Für Pareto-Profile (Reisezeit, Anzahl Umstiege)
- Profil-CSA für original-/partial-Fußwege
- aRAPTOR für complete-Fußwege



Experimente

Vergleich der Reisezeiten:

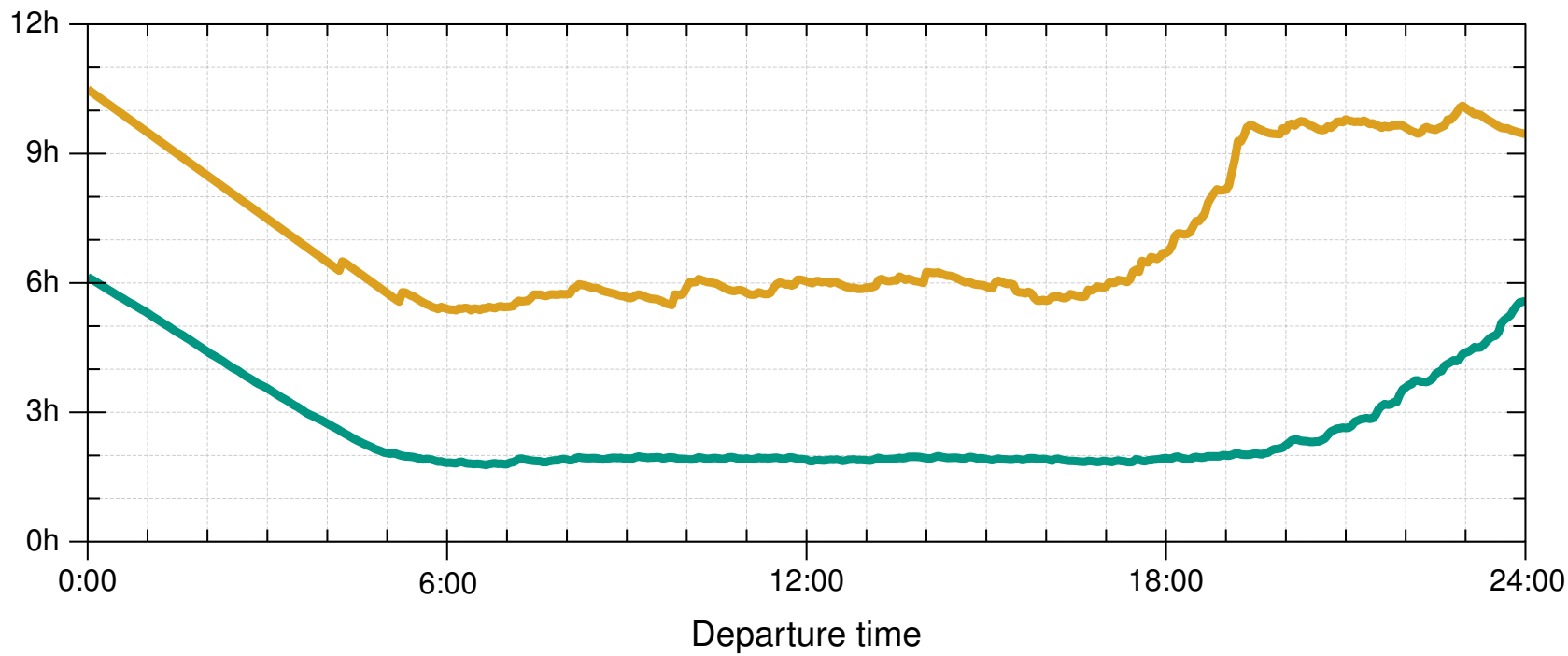
- Switzerland complete vs. original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

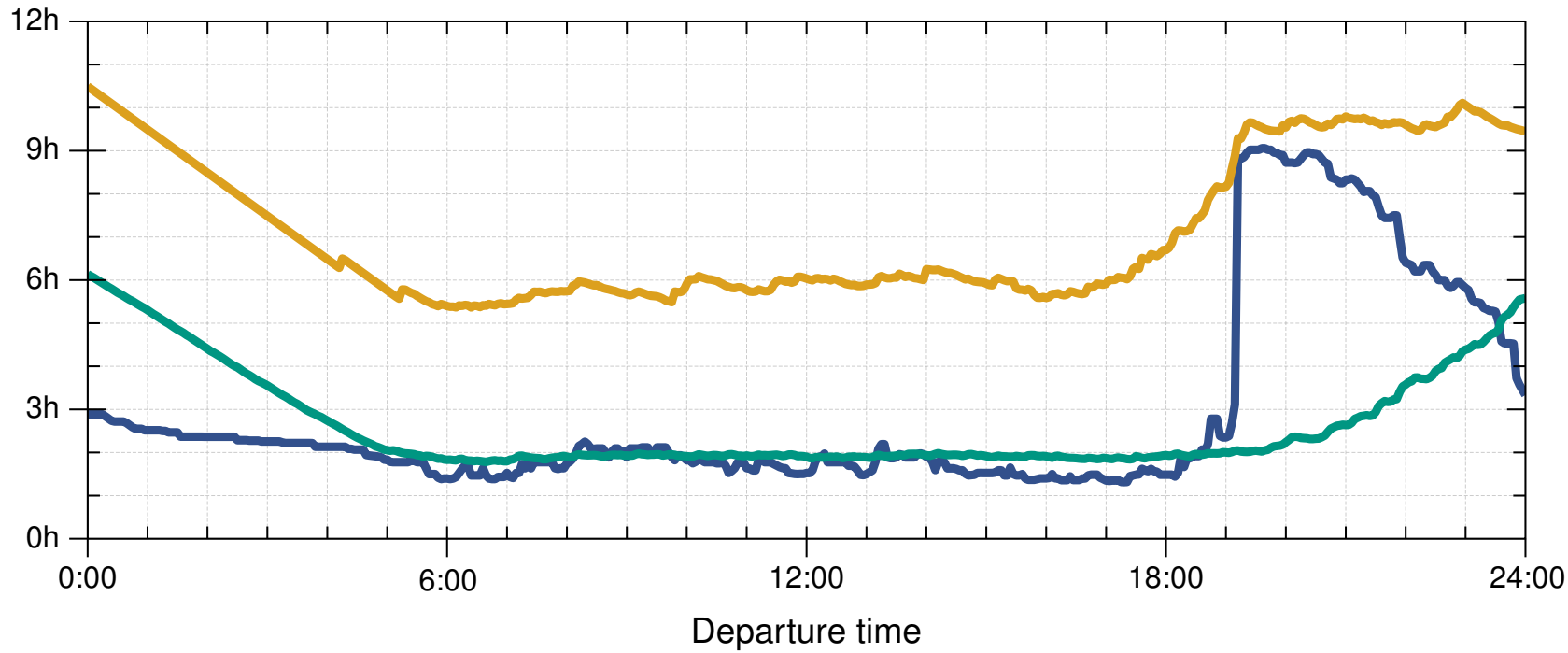
- Switzerland complete vs. original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

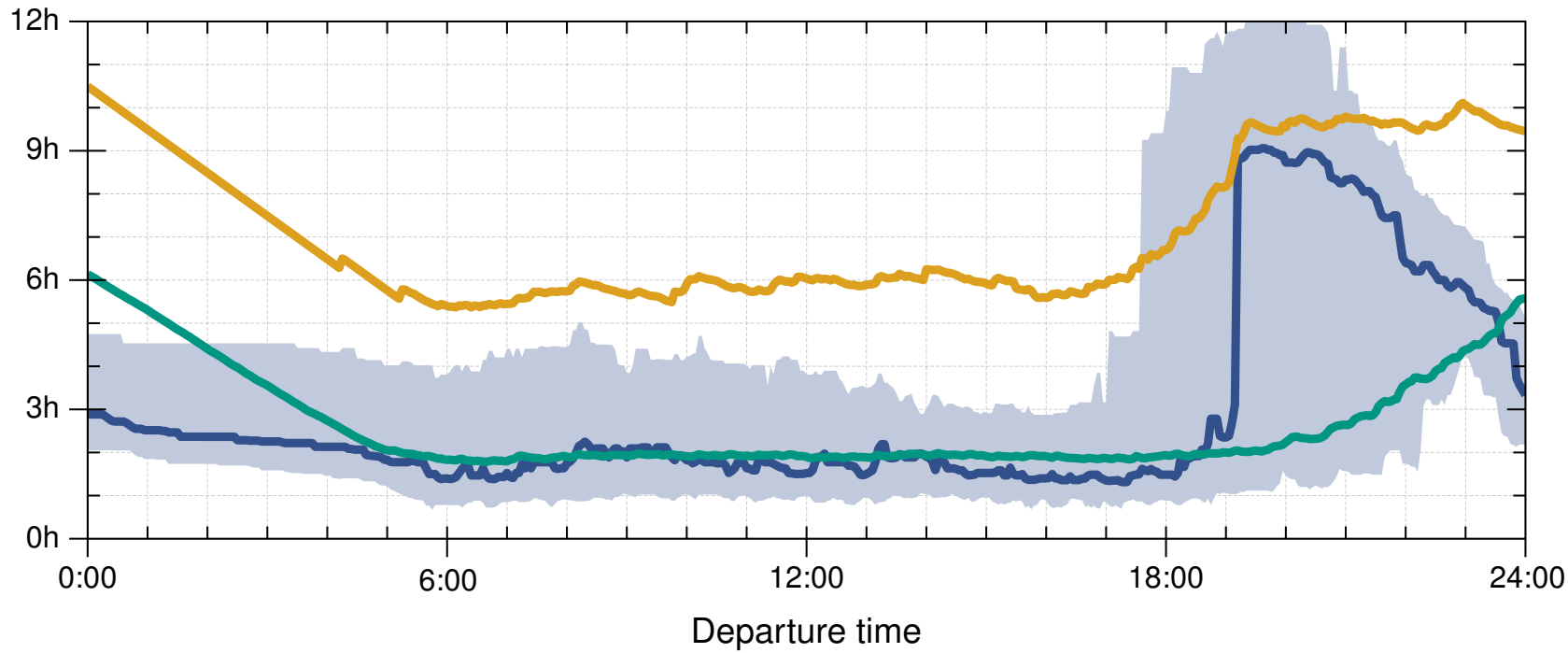
- Switzerland complete vs. original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

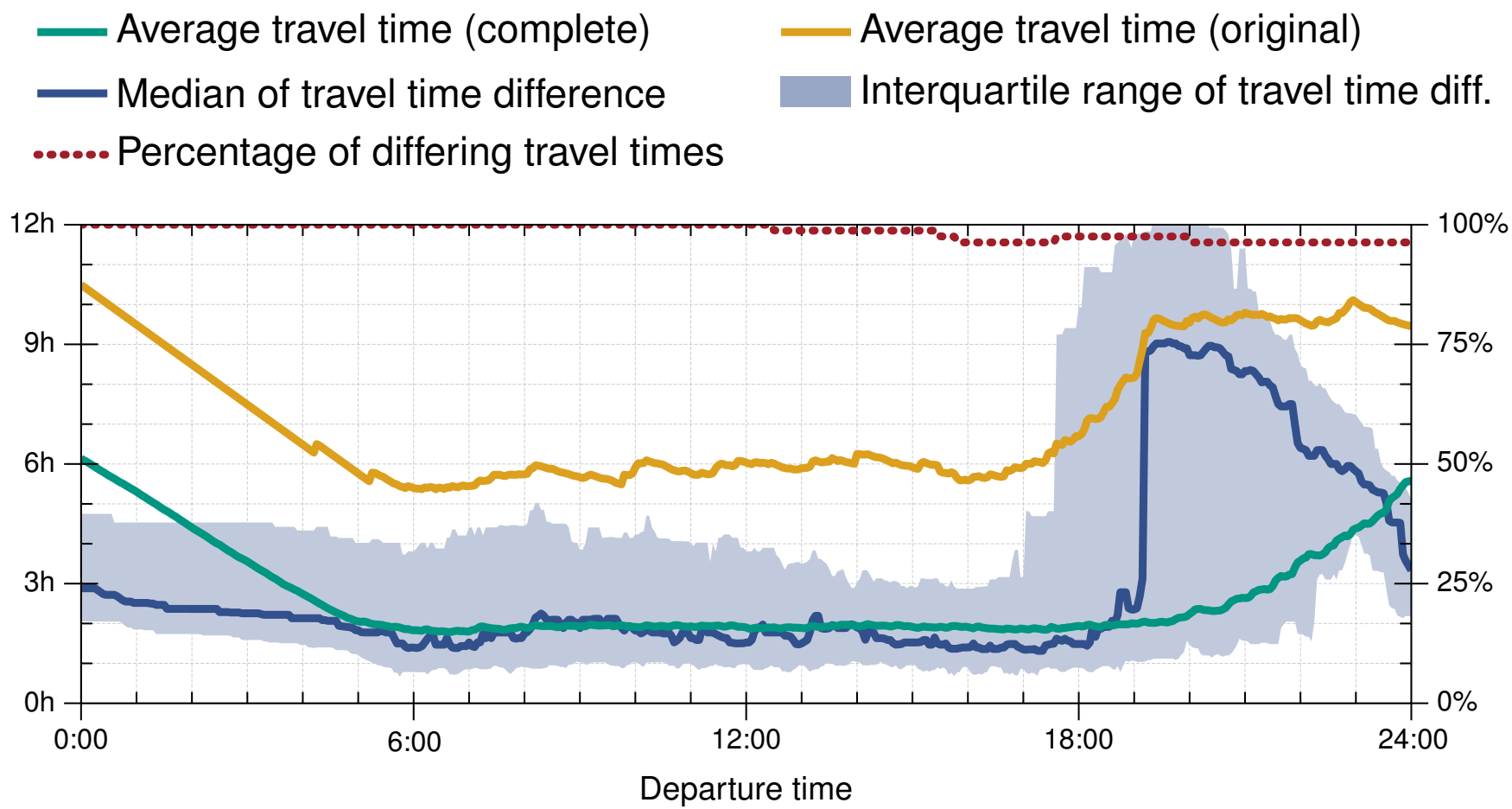
- Switzerland complete vs. original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

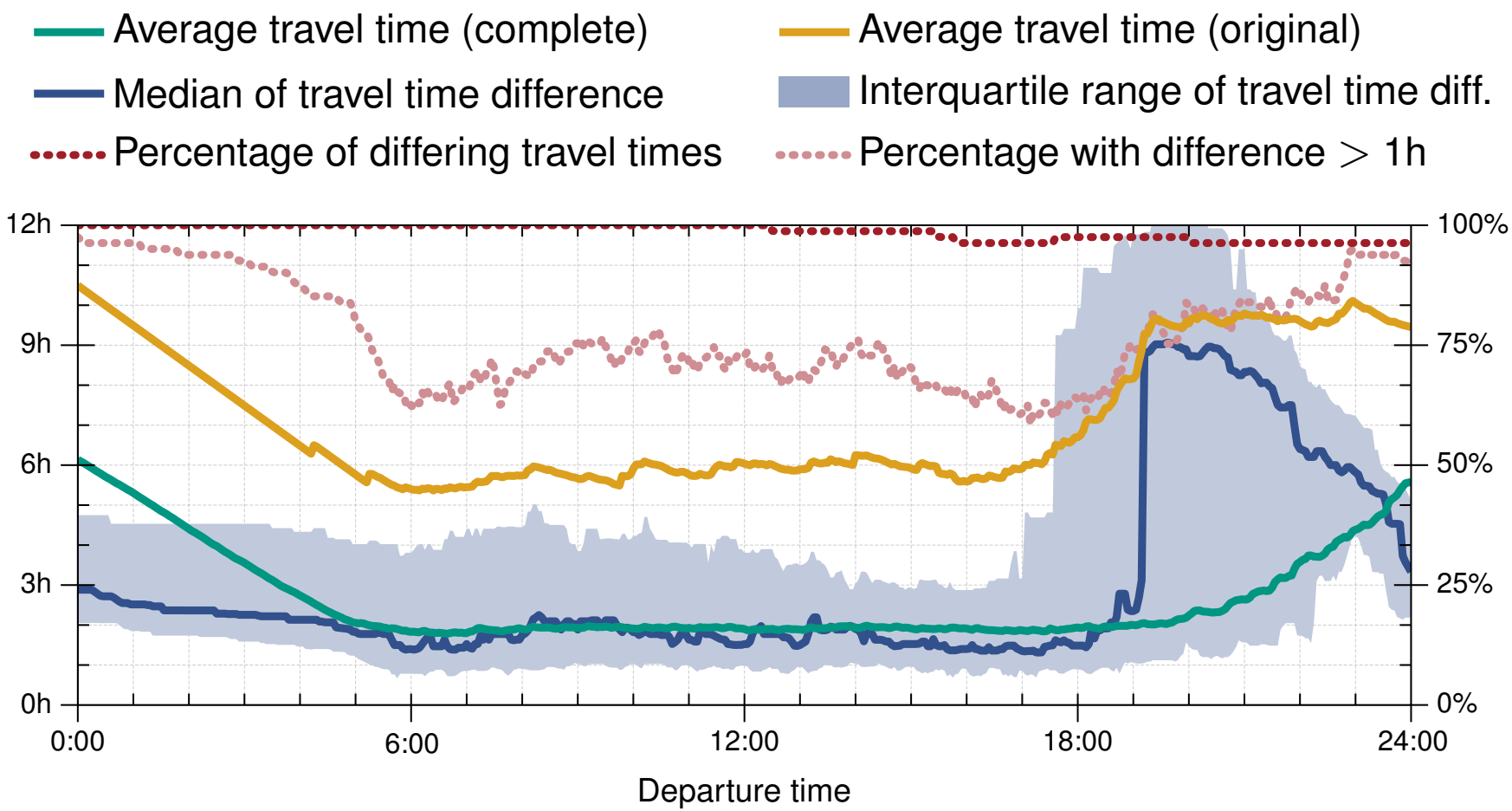
- Switzerland complete vs. original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

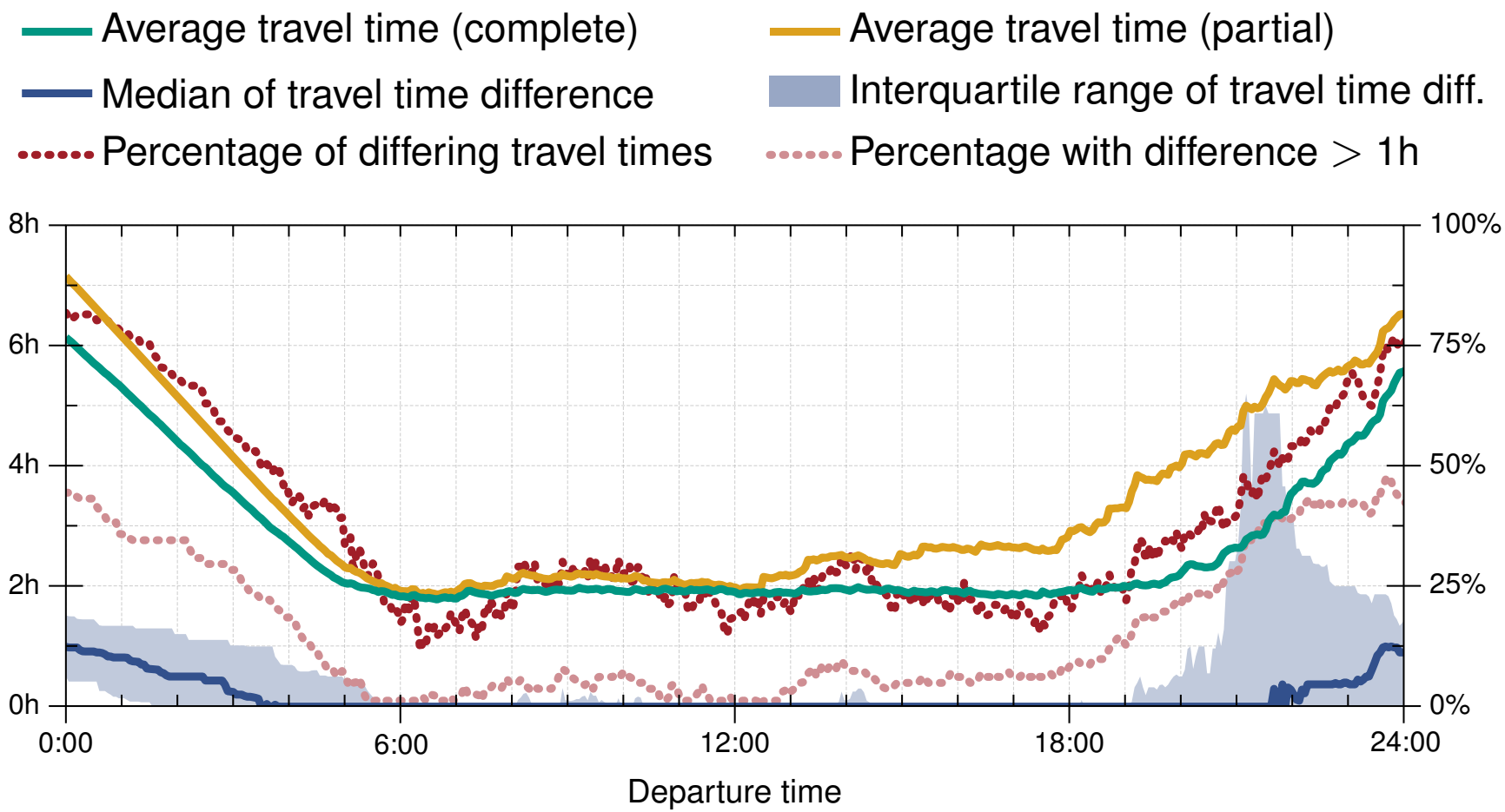
- Switzerland complete vs. original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

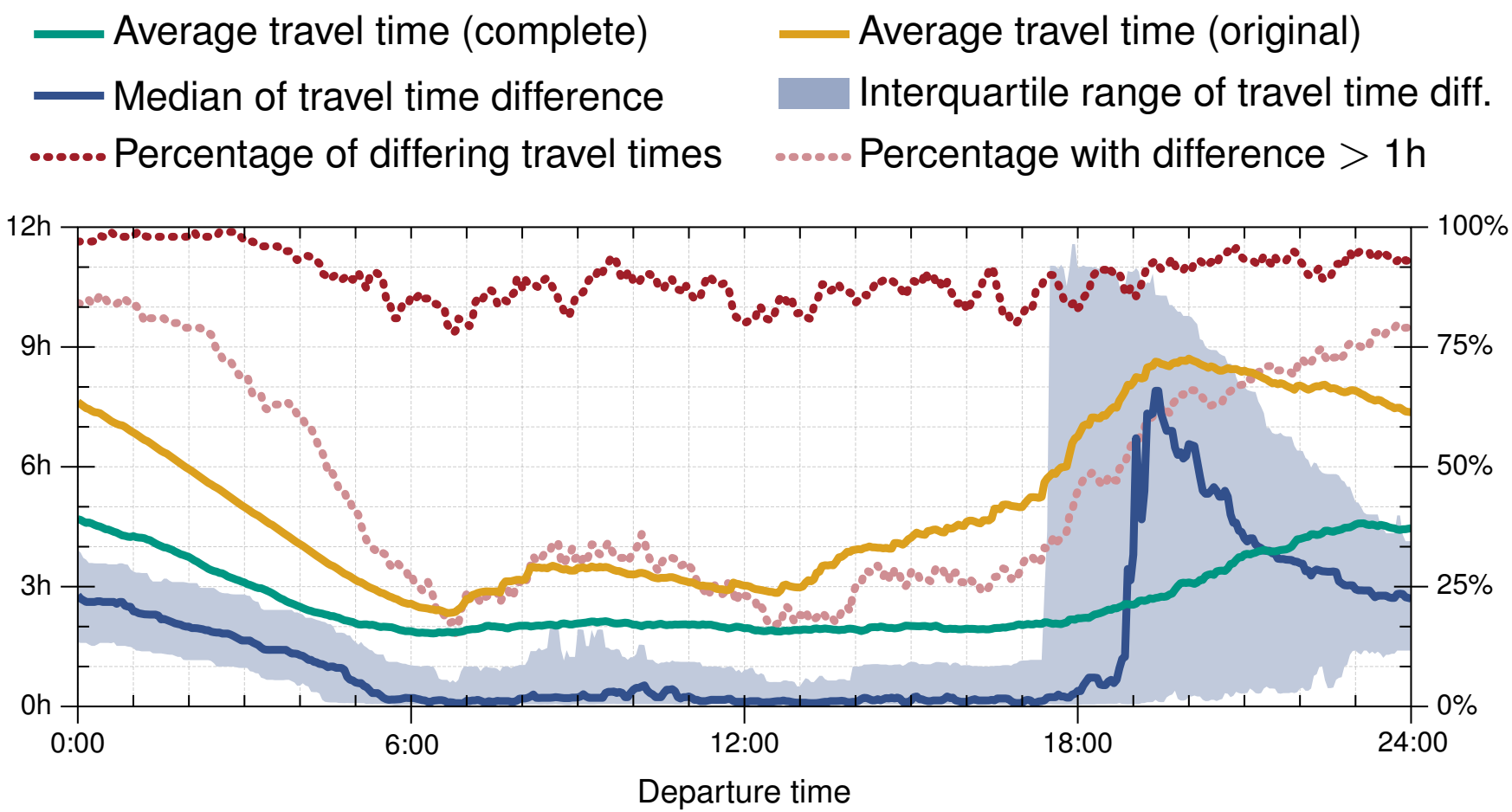
- Switzerland complete vs. partial (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

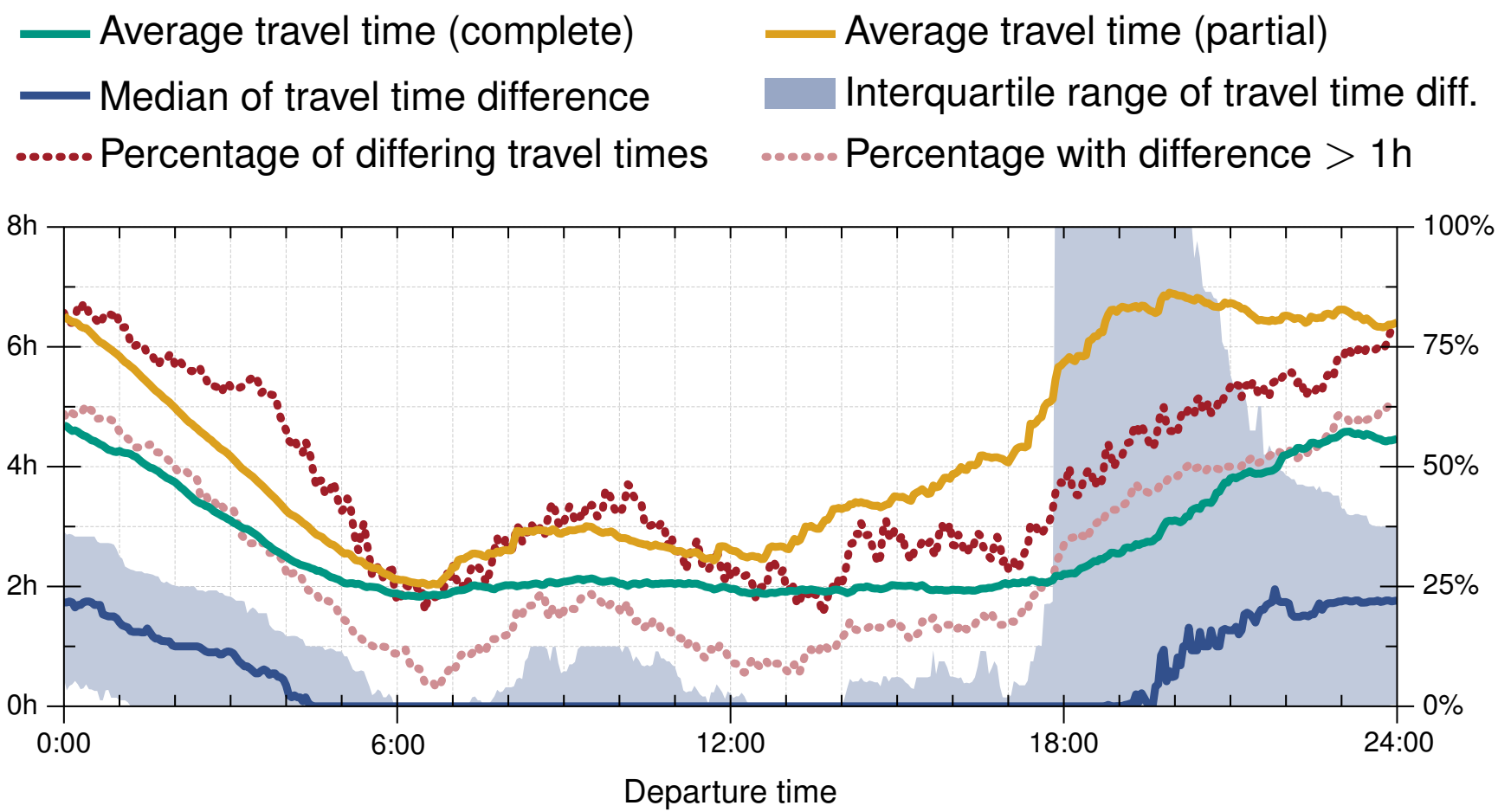
- Germany complete vs. original (Distance rank 16)



Experimente

Vergleich der Reisezeiten:

- Germany complete vs. partial (Distance rank 16)



Arten von Transfers

Frage

In welchem Teil der Journey wird unbeschränktes Laufen benötigt?

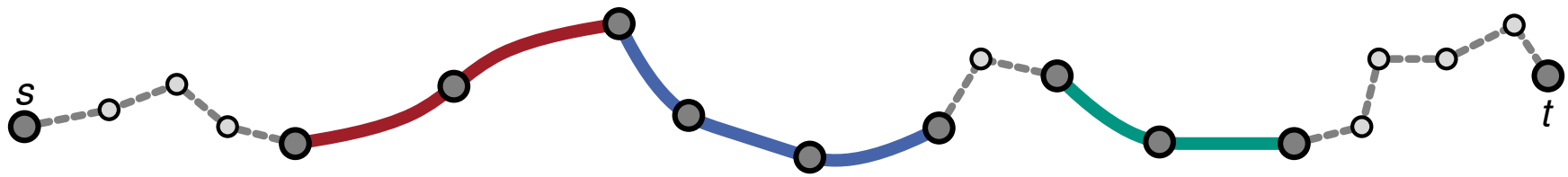
Arten von Transfers

Frage

In welchem Teil der Journey wird unbeschränktes Laufen benötigt?

Arten von Transfers:

- Initialer Transfer: (engl. *initial transfer*)
  Verbindet Startknoten mit erstem Stop
- Zwischentransfer: (engl. *intermediate transfer*)
  Verbindet zwei Trips
- Finaler Transfer: (engl. *final transfer*)
  Verbindet letzten Stop mit Zielknoten



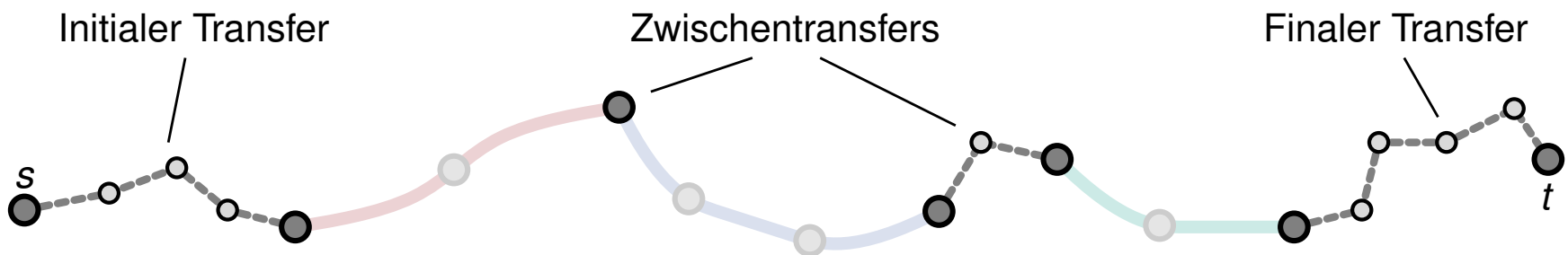
Arten von Transfers

Frage

In welchem Teil der Journey wird unbeschränktes Laufen benötigt?

Arten von Transfers:

- Initialer Transfer: (engl. *initial transfer*)
  Verbindet Startknoten mit erstem Stop
- Zwischentransfer: (engl. *intermediate transfer*)
  Verbindet zwei Trips
- Finaler Transfer: (engl. *final transfer*)
  Verbindet letzten Stop mit Zielknoten



Arten von Transfers

Hypothese

- Lange Laufwege hauptsächlich bei initialen und finalen Transfers (Anbindung von Start- und Zielpunkt ans ÖV-Netzwerk)
- Lange Zwischentransfers sind eher selten

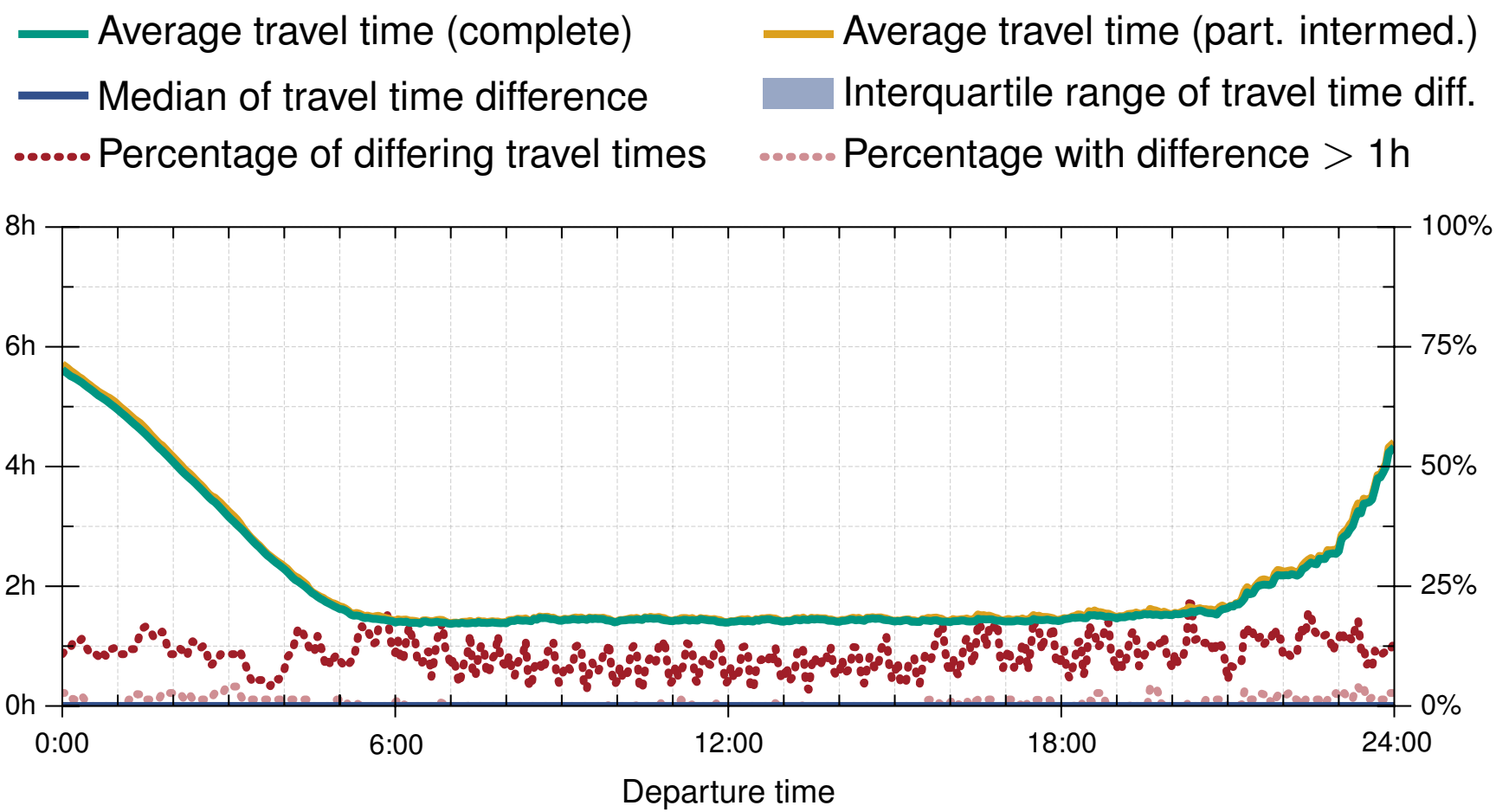
Weiteres Experiment:

- Neue Fußwege-Konfiguration **partial intermediate**:
  - Erlaube unbeschränktes Laufen nur für initiale und finale Transfers
  - Benutze transitiv abgeschlossenen Graph für Zwischentransfers
- Vergleiche mit komplett unbeschränktem Laufen
- Unterschied in Reisezeiten sollte deutlich schrumpfen

Experimente

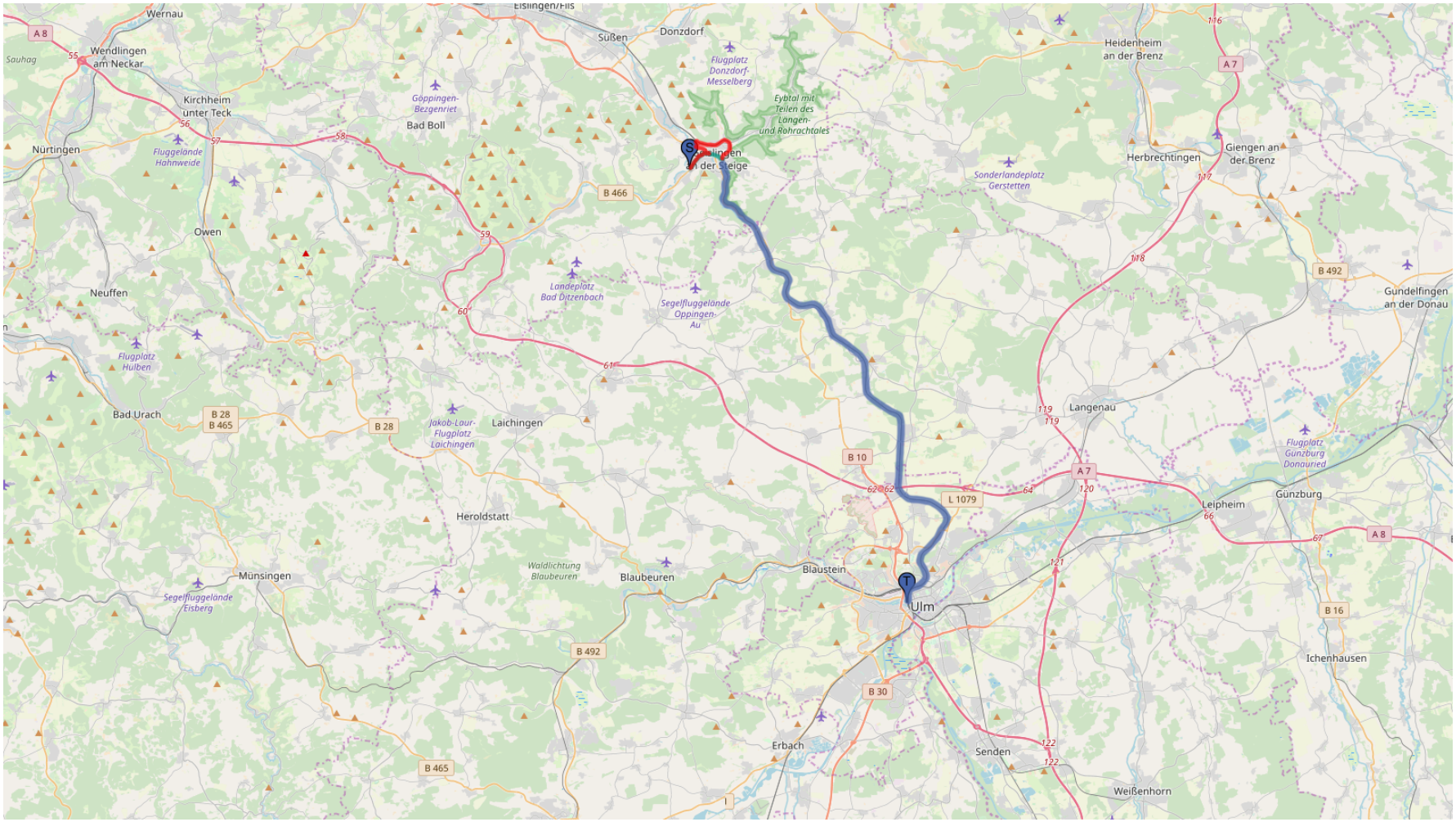
Vergleich der Reisezeiten:

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



Beispiel

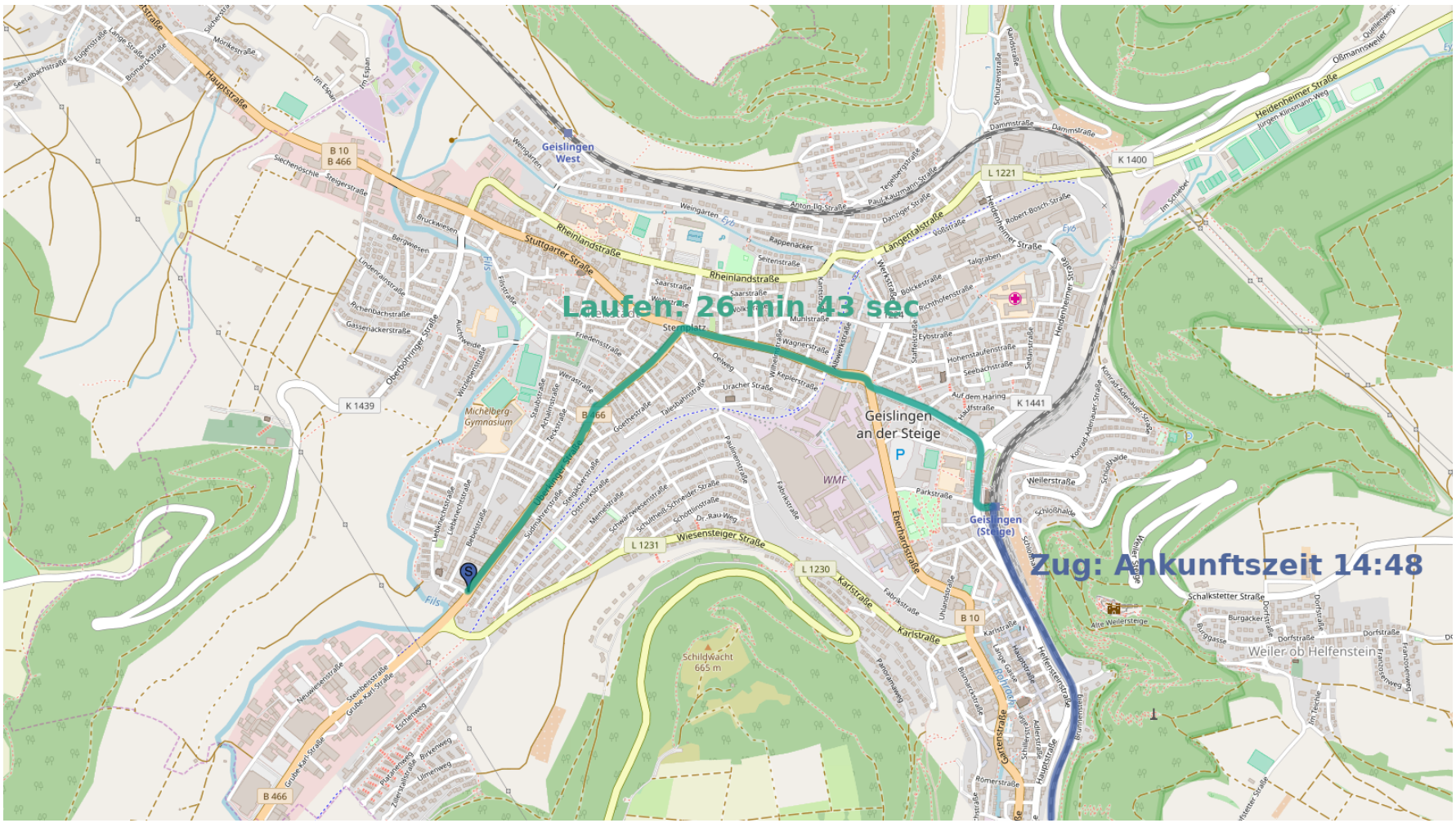
Query: Geislingen (Bebelstraße) nach Ulm Hbf um 13:10



Beispiel

Query: Geislingen (Bebelstraße) nach Ulm Hbf um 13:10

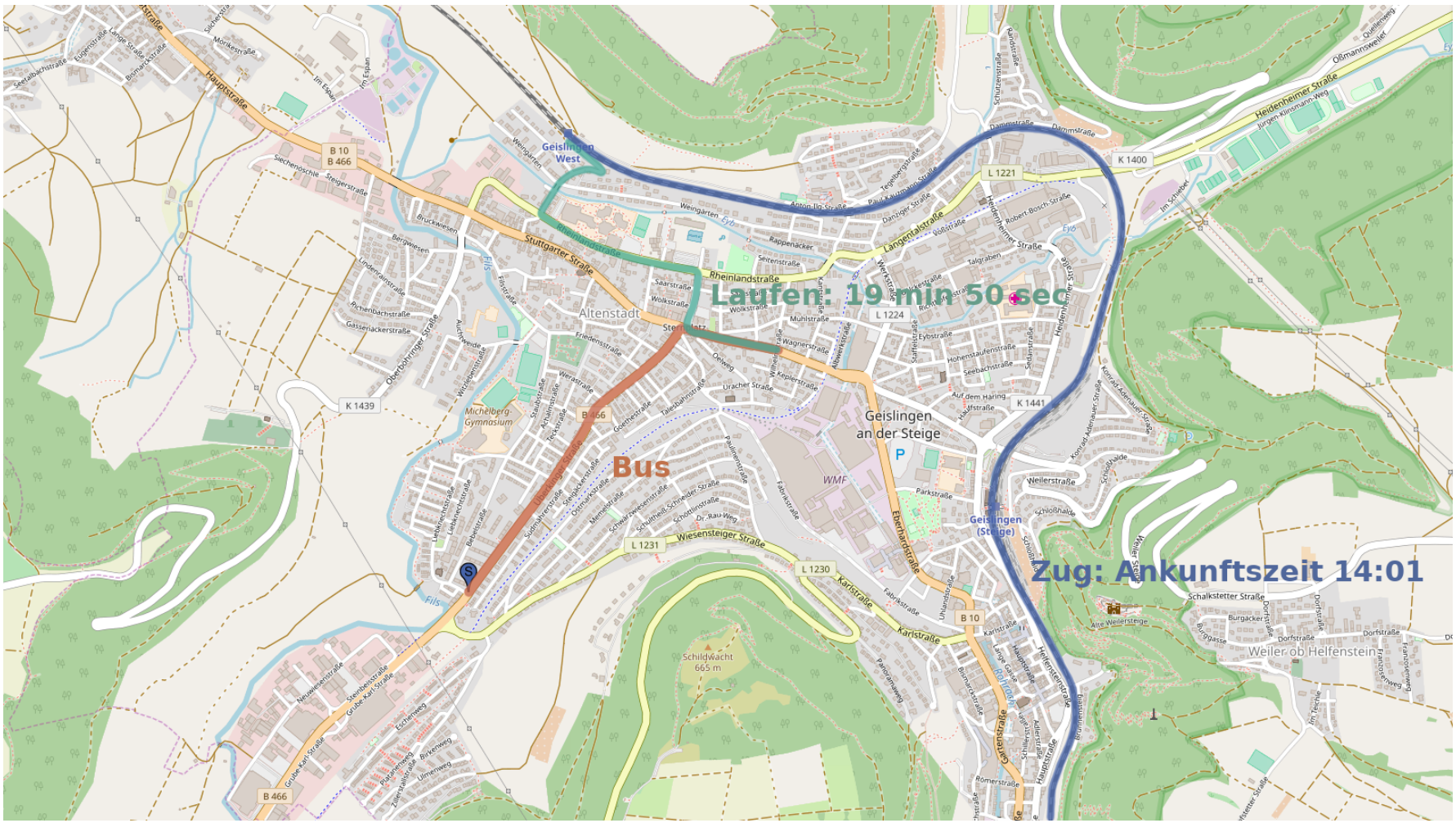
Zwischentransfers beschränkt:



Beispiel

Query: Geislingen (Bebelstraße) nach Ulm Hbf um 13:10

Zwischentransfers unbeschränkt:



ULTRA
(UnLimited TRAnsfers)

[BBS+19]

Problemstellung

Gegeben:

- Public-Transit-Netzwerk
  - Alle fahrplanbasierten Transportmodi
  - Zug, Straßenbahn, Bus, ...
- Unbeschränkter Transfergraph
  - Ein nicht-fahrplanbasierter Transportmodus
  - Laufen, Fahrrad, E-Scooter, ...
- Startknoten s, Zielknoten t, Abfahrtszeit τ_{dep}

Gesucht:

- Pareto-Menge von s-t-Journeys
- Kriterien: Ankunftszeit und Anzahl benutzter Trips

Problemstellung

Gegeben:

- Public-Transit-Netzwerk
  - Alle fahrplanbasierten Transportmodi
  - Zug, Straßenbahn, Bus, ...
- Unbeschränkter Transfergraph
  - Ein nicht-fahrplanbasierter Transportmodus
  - Laufen, Fahrrad, E-Scooter, ...
- Startknoten s, Zielknoten t, Abfahrtszeit τ_{dep}

Gesucht:

- Pareto-Menge von s-t-Journeys
- Kriterien: Ankunftszeit und Anzahl benutzter Trips

Zielsetzung

Bisher:

- Reine PT-Algorithmen können nur beschränkte Transfers
- MCR mit zwei Kriterien (MR-∞) braucht teure Dijkstra-Phasen

Ziel:

- Unbeschränkte Transfers ohne Performance-Einbußen in der Query
- Nehme dafür Vorberechnungsphase in Kauf

Folgerungen aus Experimenten:

- Unbeschränktes Laufen lohnt sich oft
- Bei schnellerem Transfermodus (z.B. Fahrrad) erst recht
- Behandle initiale/finale Transfers anders als Zwischentransfers

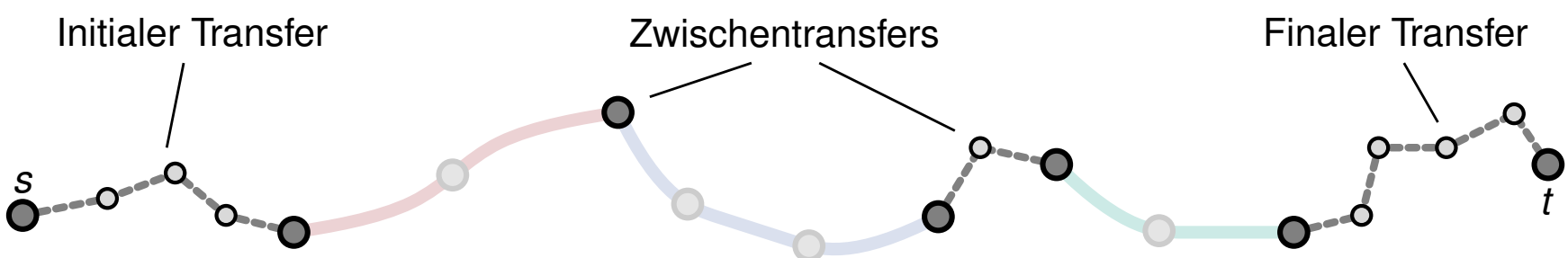
Ansatz

Initiale/finale Transfers:

- – Häufig
- – Oft lang
- \+ Ein Endpunkt bekannt (s oder t)

Zwischentransfers:

- \+ Selten
- \+ Meistens kurz
- – Beide Endpunkte unbekannt



Ansatz

Initiale/finale Transfers:

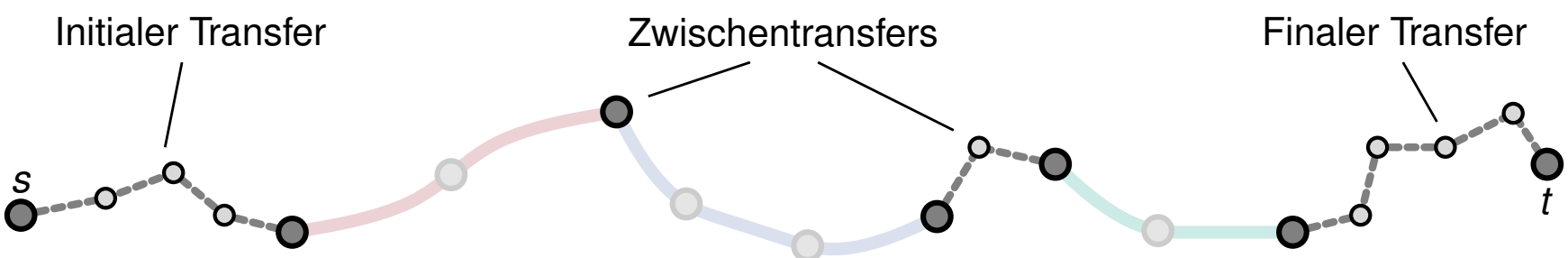
- – Häufig
- – Oft lang
- \+ Ein Endpunkt bekannt (*s* oder *t*)

- Benutze schnelle One-to-Many-Technik
- Bucket-CH

Zwischentransfers:

- \+ Selten
- \+ Meistens kurz
- – Beide Endpunkte unbekannt

- Berechne für alle relevanten Zwischentransfers Shortcuts vor
- 1-Hop-Transfers in der Query



Vorberechnung – Grundidee

Erste Idee:

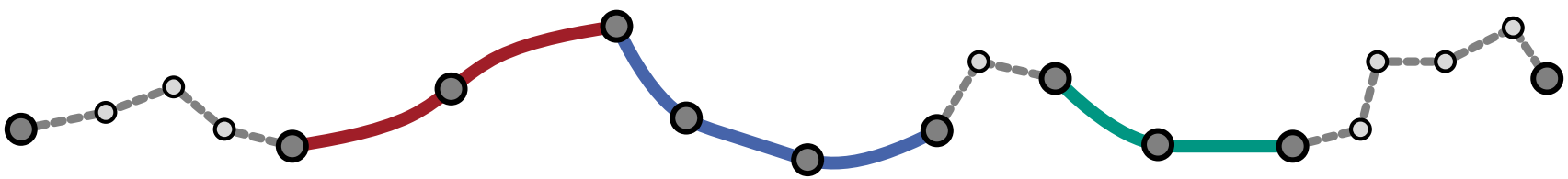
- Zähle alle Pareto-optimalen Journeys auf
- Füge für jeden benutzten Zwischentransfer einen Shortcut ein

Nutze Subpfadeigenschaft aus:

- Es reicht, Journeys mit genau 2 Trips aufzuzählen

Naive Implementierung:

- Profil-Variante von MCR beschränkt auf zwei Runden
- Für jeden Stop: Profilsuche zu allen anderen Stops
- Aber vorhin gesehen: Profil-MCR ist ineffizient



Vorberechnung – Grundidee

Erste Idee:

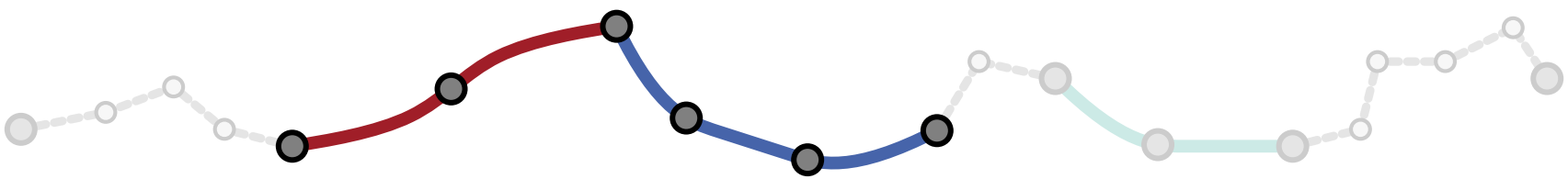
- Zähle alle Pareto-optimalen Journeys auf
- Füge für jeden benutzten Zwischentransfer einen Shortcut ein

Nutze Subpfadeigenschaft aus:

- Es reicht, Journeys mit genau 2 Trips aufzuzählen

Naive Implementierung:

- Profil-Variante von MCR beschränkt auf zwei Runden
- Für jeden Stop: Profilsuche zu allen anderen Stops
- Aber vorhin gesehen: Profil-MCR ist ineffizient



Vorberechnung – Grundidee

Erste Idee:

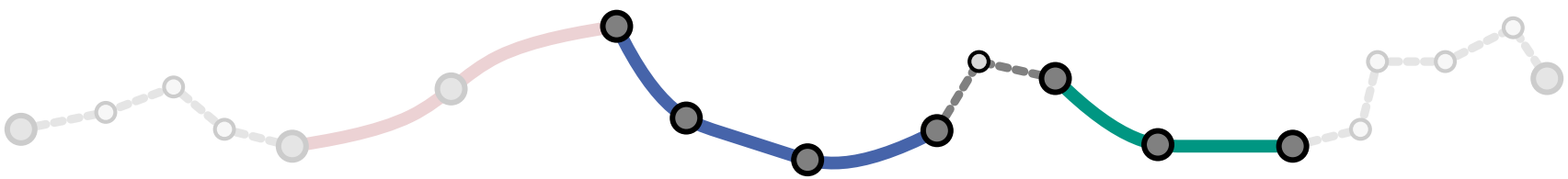
- Zähle alle Pareto-optimalen Journeys auf
- Füge für jeden benutzten Zwischentransfer einen Shortcut ein

Nutze Subpfadeigenschaft aus:

- Es reicht, Journeys mit genau 2 Trips aufzuzählen

Naive Implementierung:

- Profil-Variante von MCR beschränkt auf zwei Runden
- Für jeden Stop: Profilsuche zu allen anderen Stops
- Aber vorhin gesehen: Profil-MCR ist ineffizient



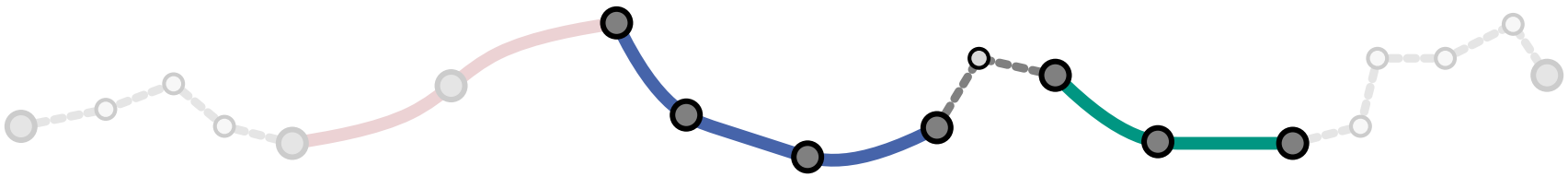
Vorberechnung – Kandidaten

Beobachtung:

- Wir müssen nicht *alle* Pareto-optimalen 2-Trip-Journeys aufzählen
- Wir müssen nur alle benötigten Zwischentransfers finden

Zwei Typen von Journeys:

- Kandidaten haben die Form Trip – Transfer – Trip
- Zeugen sind alle anderen Journeys mit höchstens 2 Trips



Kandidat wird von einem Zeugen dominiert $\Leftrightarrow$ Kein Shortcut benötigt

(Vereinfachende Annahme: Keine zwei Journeys sind in beiden Kriterien gleichwertig)

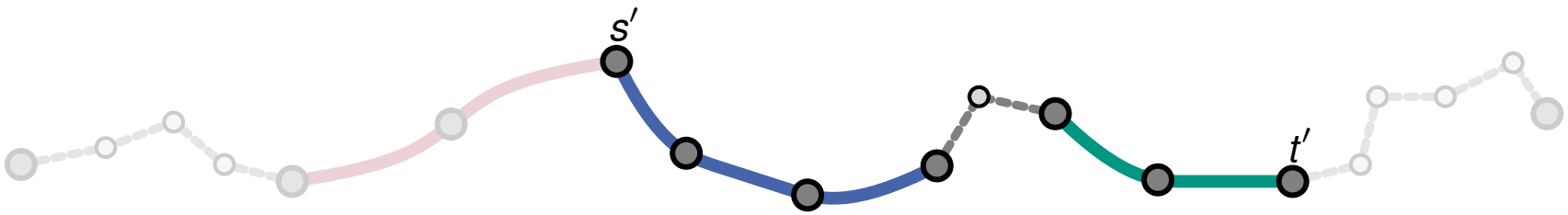
Vorberechnung – Kandidaten

Beobachtung:

- Wir müssen nicht *alle* Pareto-optimalen 2-Trip-Journeys aufzählen
- Wir müssen nur alle benötigten Zwischentransfers finden

Zwei Typen von Journeys:

- Kandidaten haben die Form Trip – Transfer – Trip
- Zeugen sind alle anderen Journeys mit höchstens 2 Trips



Kandidat wird von einem Zeugen dominiert $\Leftrightarrow$ Kein Shortcut benötigt

(Vereinfachende Annahme: Keine zwei Journeys sind in beiden Kriterien gleichwertig)

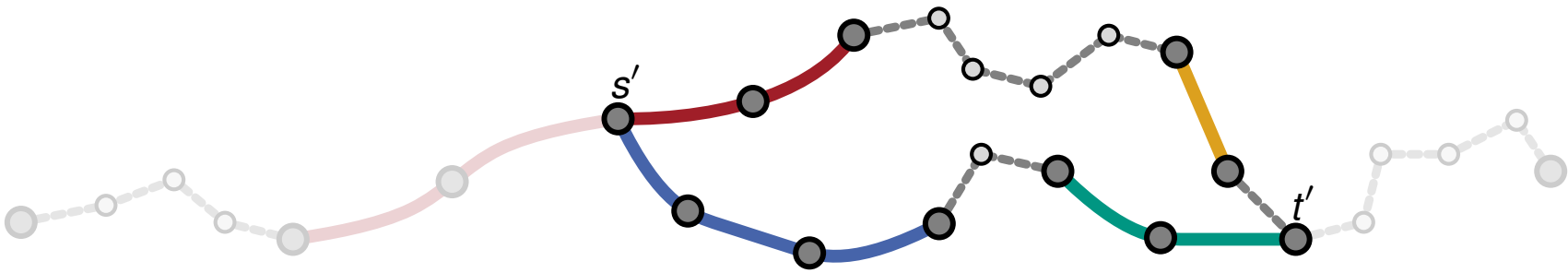
Vorberechnung – Kandidaten

Beobachtung:

- Wir müssen nicht *alle* Pareto-optimalen 2-Trip-Journeys aufzählen
- Wir müssen nur alle benötigten Zwischentransfers finden

Zwei Typen von Journeys:

- Kandidaten haben die Form Trip – Transfer – Trip
- Zeugen sind alle anderen Journeys mit höchstens 2 Trips



Kandidat wird von einem Zeugen dominiert ⇔ Kein Shortcut benötigt

(Vereinfachende Annahme: Keine zwei Journeys sind in beiden Kriterien gleichwertig)

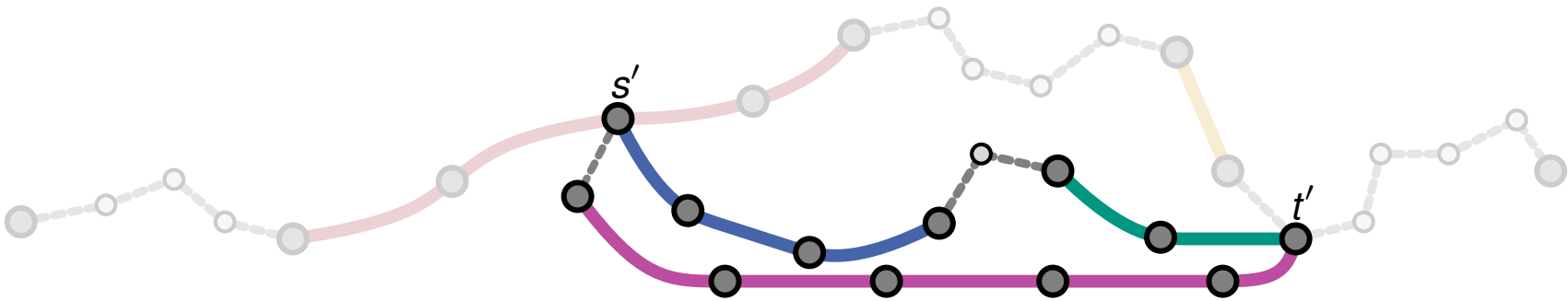
Vorberechnung – Kandidaten

Beobachtung:

- Wir müssen nicht *alle* Pareto-optimalen 2-Trip-Journeys aufzählen
- Wir müssen nur alle benötigten Zwischentransfers finden

Zwei Typen von Journeys:

- Kandidaten haben die Form Trip – Transfer – Trip
- Zeugen sind alle anderen Journeys mit höchstens 2 Trips



Kandidat wird von einem Zeugen dominiert ⇔ Kein Shortcut benötigt

(Vereinfachende Annahme: Keine zwei Journeys sind in beiden Kriterien gleichwertig)

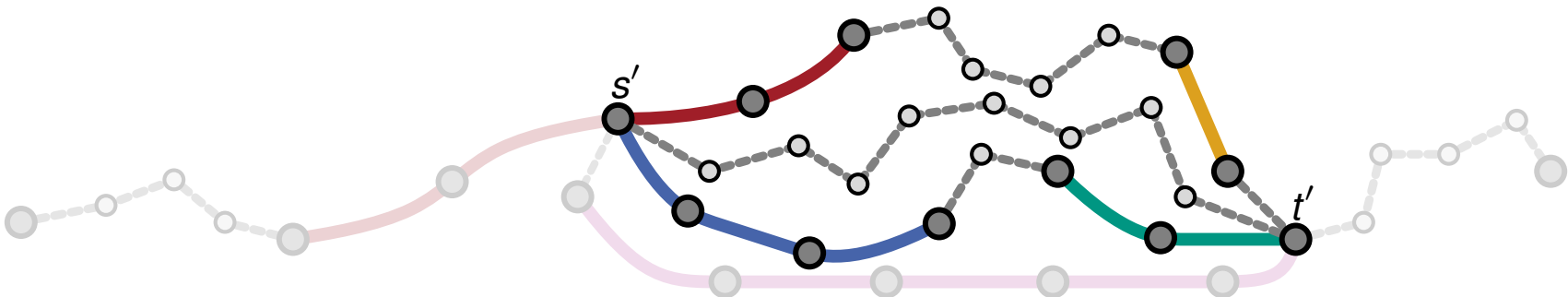
Vorberechnung – Kandidaten

Beobachtung:

- Wir müssen nicht *alle* Pareto-optimalen 2-Trip-Journeys aufzählen
- Wir müssen nur alle benötigten Zwischentransfers finden

Zwei Typen von Journeys:

- Kandidaten haben die Form Trip – Transfer – Trip
- Zeugen sind alle anderen Journeys mit höchstens 2 Trips



Kandidat wird von einem Zeugen dominiert ⇔ Kein Shortcut benötigt

(Vereinfachende Annahme: Keine zwei Journeys sind in beiden Kriterien gleichwertig)

Aufzählen der Kandidaten

Wir wollen aufzählen:

1. Alle optimalen Kandidaten
2. Für jeden dominierten Kandidaten: ≥ 1 dominierenden Zeugen

Finde Kandidaten:

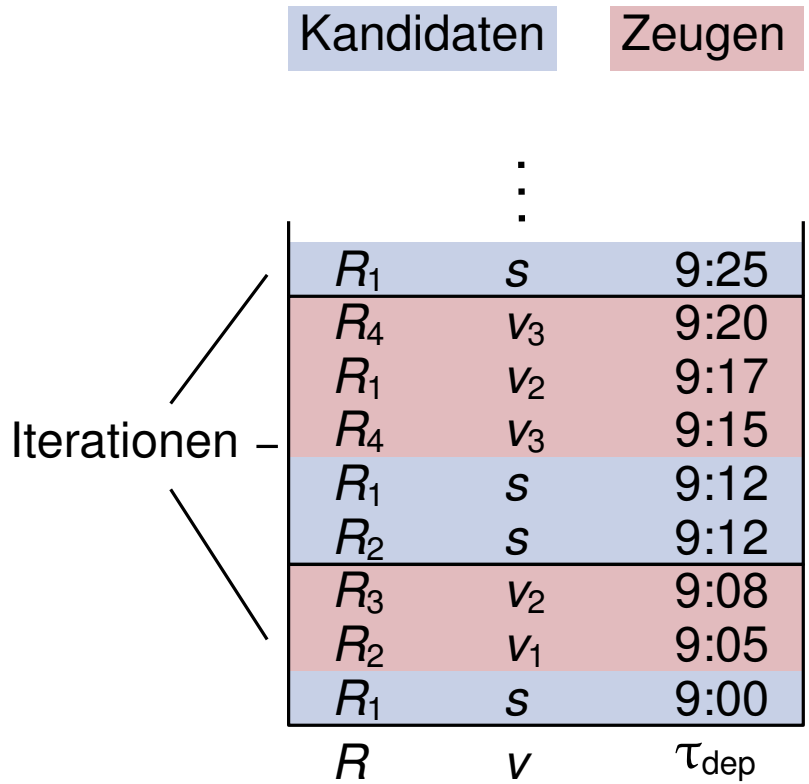
- Keine initialen Transfers $\rightarrow$ Abfahrt immer an s
- Eine RAPTOR-Iteration pro Abfahrt direkt an s

Finde dominierende Zeugen für Kandidaten mit Abfahrtszeit τ:

- Normale RAPTOR-Query für Abfahrtszeit τ
- Kann in RAPTOR-Iteration für τ integriert werden

Einsammeln der Abfahrtszeiten

- Erzeuge Tupel $(R, v, \tau_{\text{dep}})$ für jede
  - Abfahrt von Route R
  - an Stop v
  - zum Zeitpunkt $\tau_{\text{dep}} + d(s, v)$
- Sortiere Tupel absteigend nach τ_{dep}
- Tupel mit $v = s$:
  - Potenzieller Kandidat
  - RAPTOR-Iteration für τ_{dep}
- Tupel mit $v \neq s$:
  - Potenzieller Zeuge
- RAPTOR-Iteration für τ:
  - Sammle Tupel mit $\tau_{\text{dep}} \geq \tau$
  - Scanne jeweils Route R ab Stop v
  - Lösche Tupel



Vorberechnung – Pseudocode

```
for jeden Stop s do // trivial parallelisierbar
    Relaxiere initiale Transfers von s zu allen Stops
    Sammle Abfahrtszeiten ein
    for jede Abfahrtszeit τ an s absteigend do
        // Runde 1
        Sammle initiale Routen ein
        Scanne initiale Routen
        Relaxiere Zwischentransfers

        // Runde 2
        Scanne finale Routen // Erzeugt Kandidaten
        Relaxiere finale Transfers // Prunt Kandidaten

        for jeden optimalen Kandidaten do
            Entpacke Shortcut
```

Vorberechnung – Kandidaten vs. Zeugen

Karlsruher Institut für Technologie

Während der RAPTOR-Iteration:

- Kandidaten brauchen Parent-Pointer zum Entpacken der Shortcuts
- Wie Zeugen von Kandidaten unterscheiden?

$\Rightarrow$ Zeugen bekommen Dummy-Wert als Parent-Pointer

Shortcuts entpacken:

- Iteriere über in Runde 2 erzeugte Labels
- Parent-Pointer nicht Dummy $\Rightarrow$ optimaler Kandidat
- Verfolge Parent-Pointer und extrahiere Shortcut

Noch besseres Pruning:

- Journey ist nur Kandidat, wenn Shortcut noch nicht hinzugefügt wurde

Vorberechnung – Transfer-Phasen

Relaxiere Transfers:

- Dijkstra auf partiell kontrahiertem Core-Graph (wie UCCH & MCR)
- Trotz Kontraktion teuer $\rightsquigarrow$ beschränken?

Initiale Transfers:

- Werden nur einmal pro Stop exploriert

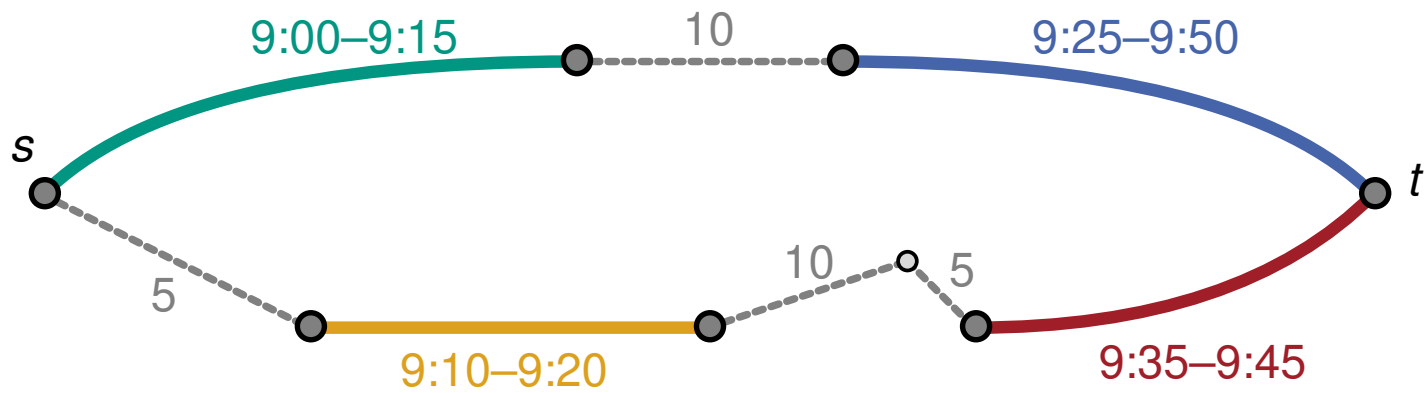
Finale Transfers:

- Routenscan in Runde 2: Merke Stops, die über Kandidat erreicht werden
- Abbruchkriterium: Dijkstra hat alle diese Stops erreicht
- Korrektheit: Danach kann kein Kandidat mehr dominiert werden

Vorberechnung – Zwischentransfers

Zwischentransfers:

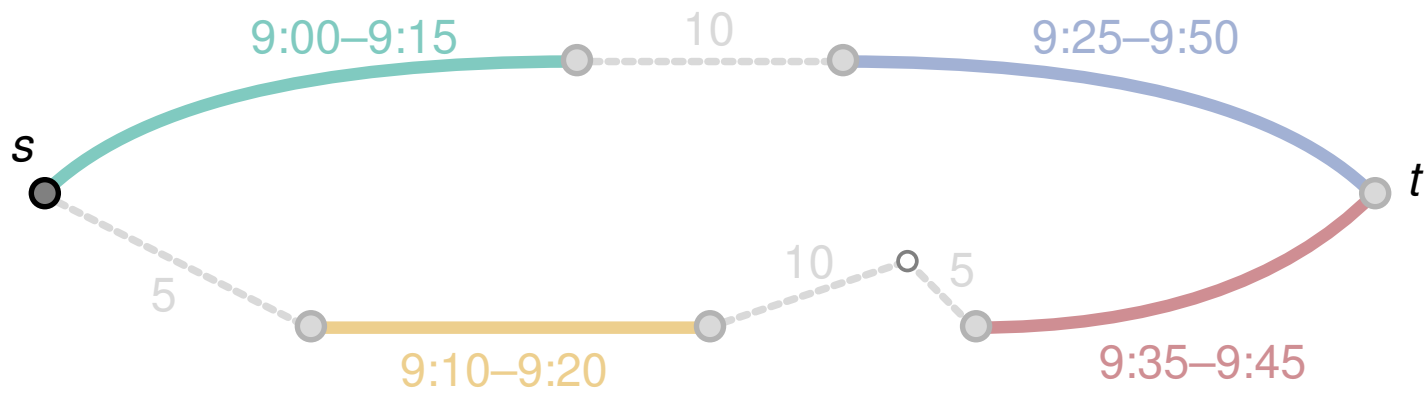
- Breche ab, wenn keine Kandidaten mehr in Queue sind?
- Kann zu nicht gefundenen Zeugen führen $\rightarrow$ überflüssige Shortcuts



Vorberechnung – Zwischentransfers

Zwischentransfers:

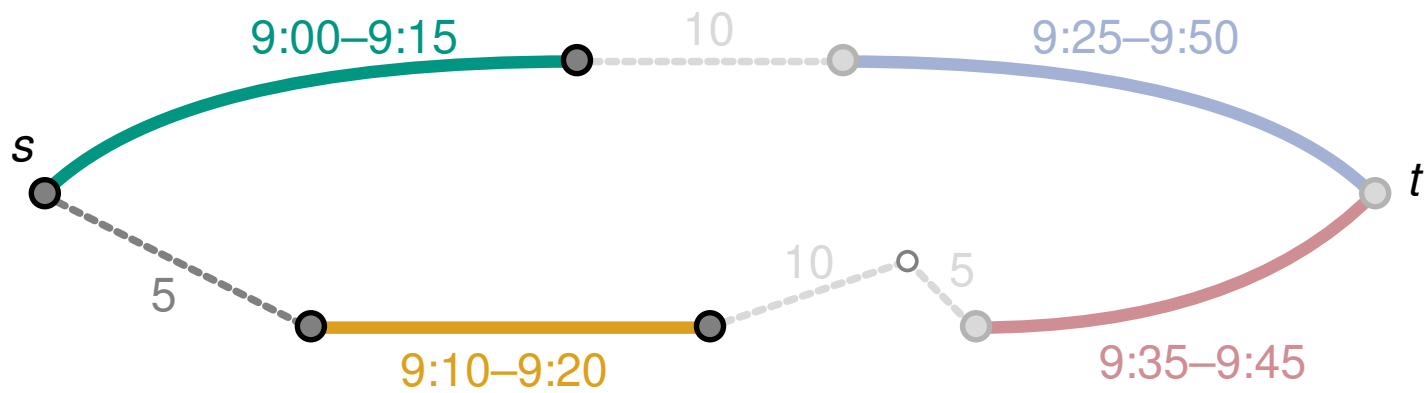
- Breche ab, wenn keine Kandidaten mehr in Queue sind?
- Kann zu nicht gefundenen Zeugen führen $\rightarrow$ überflüssige Shortcuts



Vorberechnung – Zwischentransfers

Zwischentransfers:

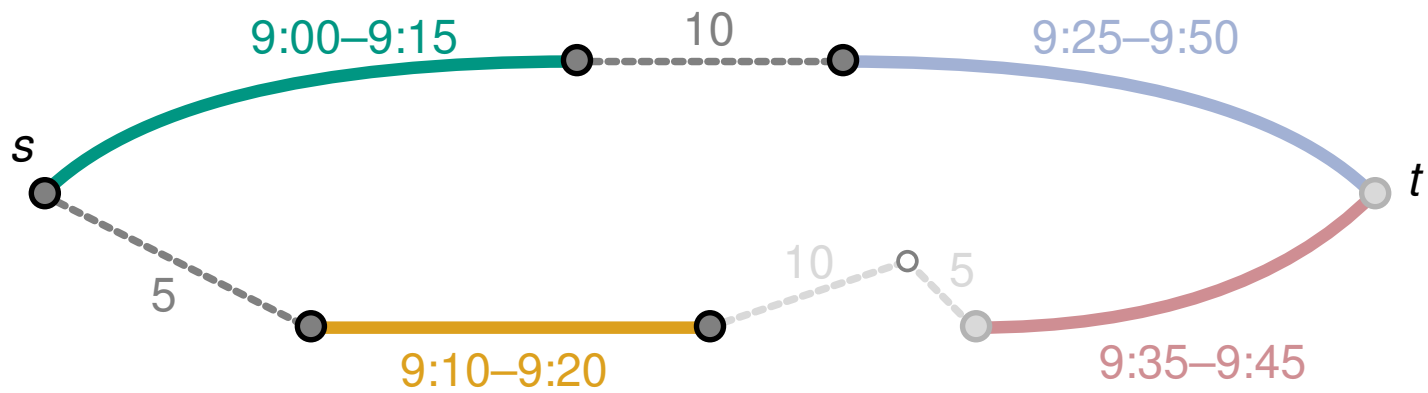
- Breche ab, wenn keine Kandidaten mehr in Queue sind?
- Kann zu nicht gefundenen Zeugen führen $\rightarrow$ überflüssige Shortcuts



Vorberechnung – Zwischentransfers

Zwischentransfers:

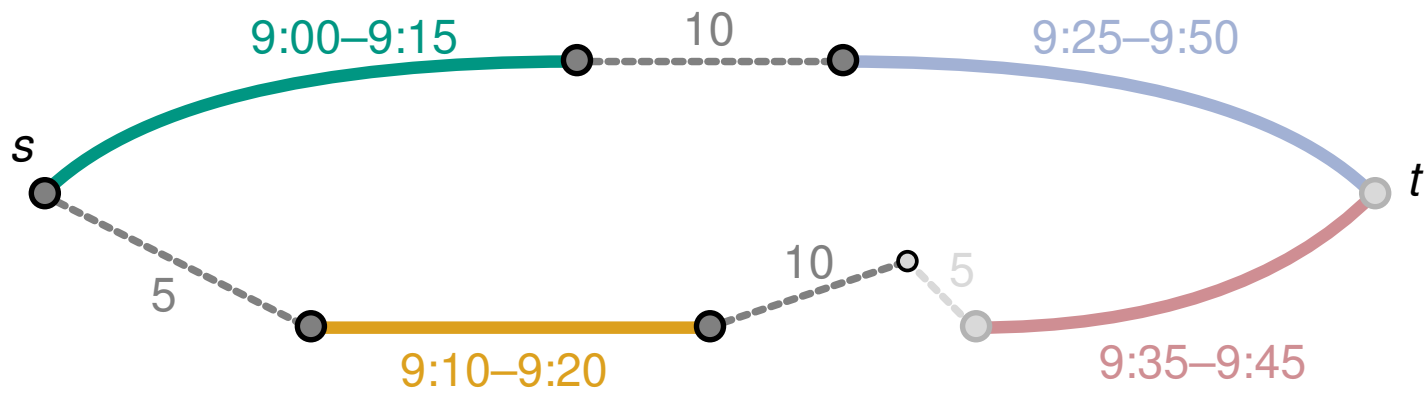
- Breche ab, wenn keine Kandidaten mehr in Queue sind?
- Kann zu nicht gefundenen Zeugen führen $\rightarrow$ überflüssige Shortcuts



Vorberechnung – Zwischentransfers

Zwischentransfers:

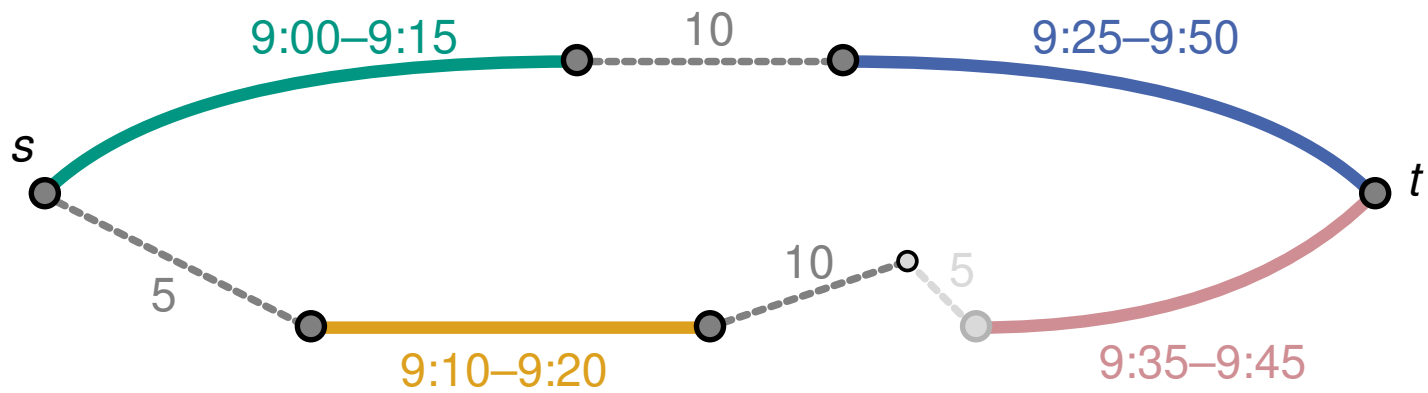
- Breche ab, wenn keine Kandidaten mehr in Queue sind?
- Kann zu nicht gefundenen Zeugen führen $\rightarrow$ überflüssige Shortcuts



Vorberechnung – Zwischentransfers

Zwischentransfers:

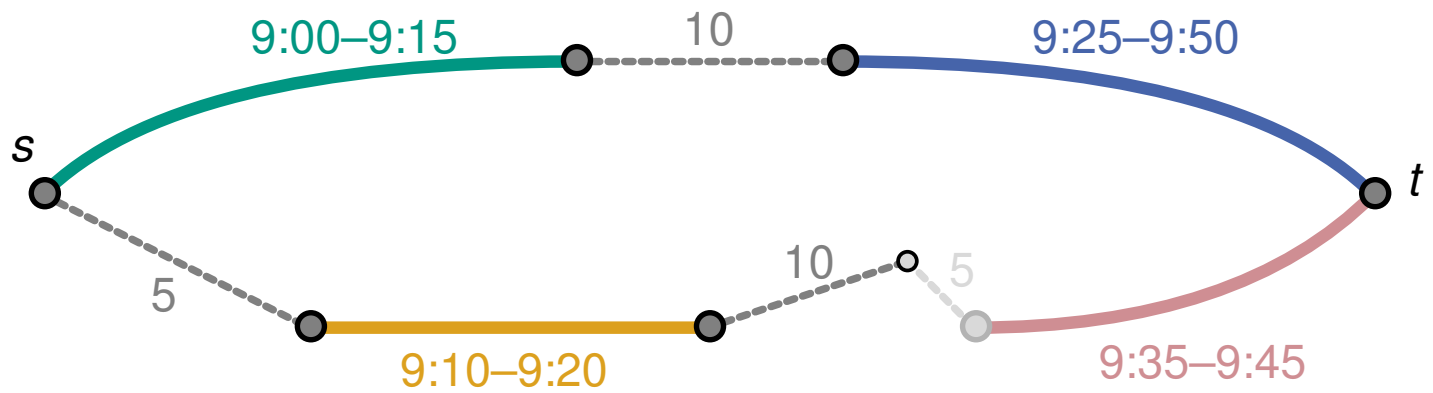
- Breche ab, wenn keine Kandidaten mehr in Queue sind?
- Kann zu nicht gefundenen Zeugen führen $\rightarrow$ überflüssige Shortcuts



Vorberechnung – Zwischentransfers

Zwischentransfers:

- Breche ab, wenn keine Kandidaten mehr in Queue sind?
- Kann zu nicht gefundenen Zeugen führen $\rightarrow$ überflüssige Shortcuts



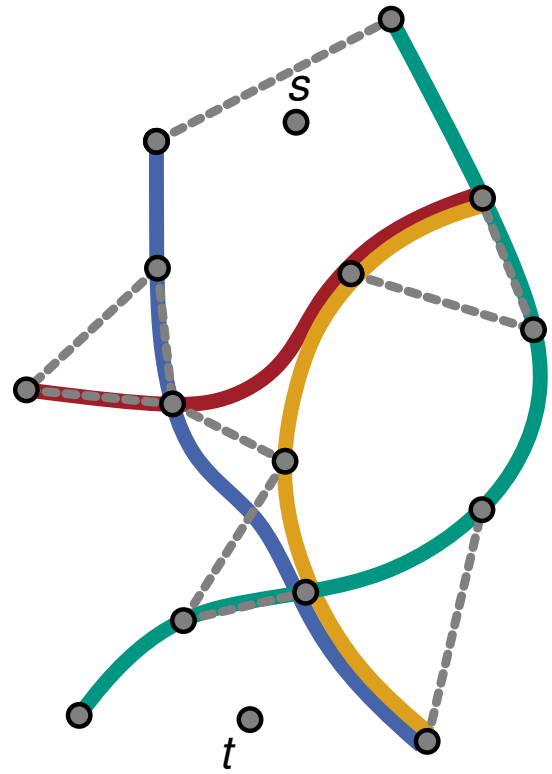
Witness Limit $\overline{\tau}$:

- τ_{arr}: Späteste optimale Ankunftszeit ohne initialen Transfer
- Abbruchkriterium: Queue-Key $> \tau_{\text{arr}} + \overline{\tau}$
- Tradeoff Laufzeit vs. Anzahl Shortcuts

Query – Black-Box

Black-Box-Query:

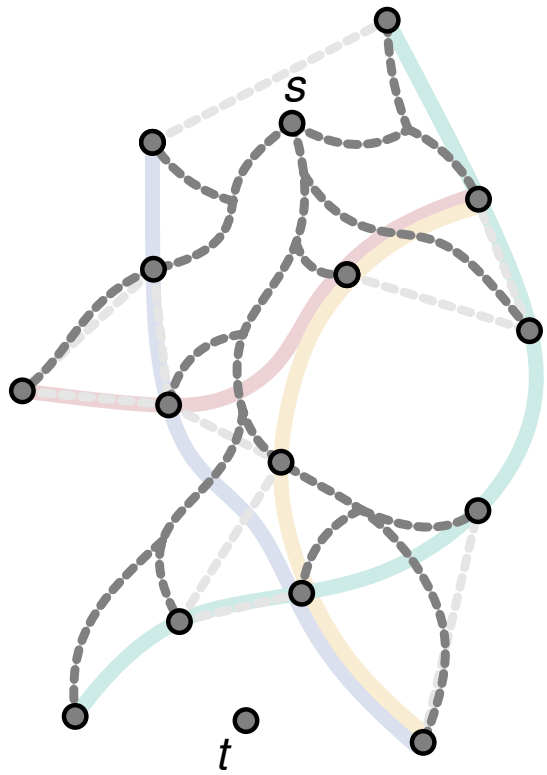
- Berechne initiale/finale Transfers mit One-to-Many-Algorithmus
  - One = Start/Ziel
  - Many = Stops
  - Schnelle Implementierung: Bucket-CH
- Baue temporären Transfer-Graph G' mit:
  - Vorberechneten Shortcuts
  - Initialen Transfers
  - Finalen Transfers
- Lasse beliebigen PT-Algorithmus auf PT-Netzwerk + G' laufen



Query – Black-Box

Black-Box-Query:

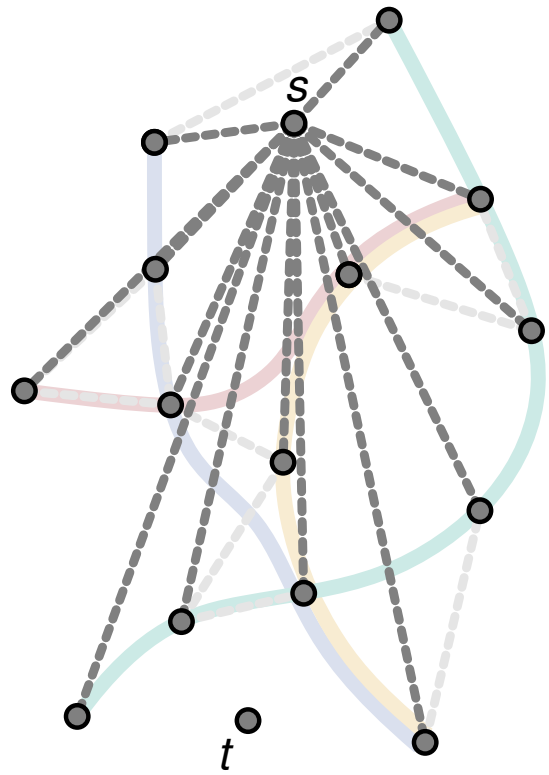
- Berechne initiale/finale Transfers mit One-to-Many-Algorithmus
  - One = Start/Ziel
  - Many = Stops
  - Schnelle Implementierung: Bucket-CH
- Baue temporären Transfer-Graph G' mit:
  - Vorberechneten Shortcuts
  - Initialen Transfers
  - Finalen Transfers
- Lasse beliebigen PT-Algorithmus auf PT-Netzwerk + G' laufen



Query – Black-Box

Black-Box-Query:

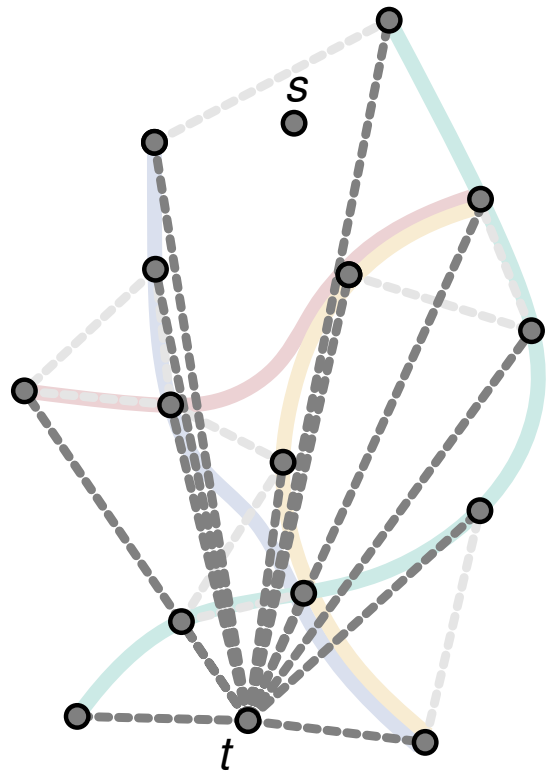
- Berechne initiale/finale Transfers mit One-to-Many-Algorithmus
  - One = Start/Ziel
  - Many = Stops
  - Schnelle Implementierung: Bucket-CH
- Baue temporären Transfer-Graph G' mit:
  - Vorberechneten Shortcuts
  - Initialen Transfers
  - Finalen Transfers
- Lasse beliebigen PT-Algorithmus auf PT-Netzwerk + G' laufen



Query – Black-Box

Black-Box-Query:

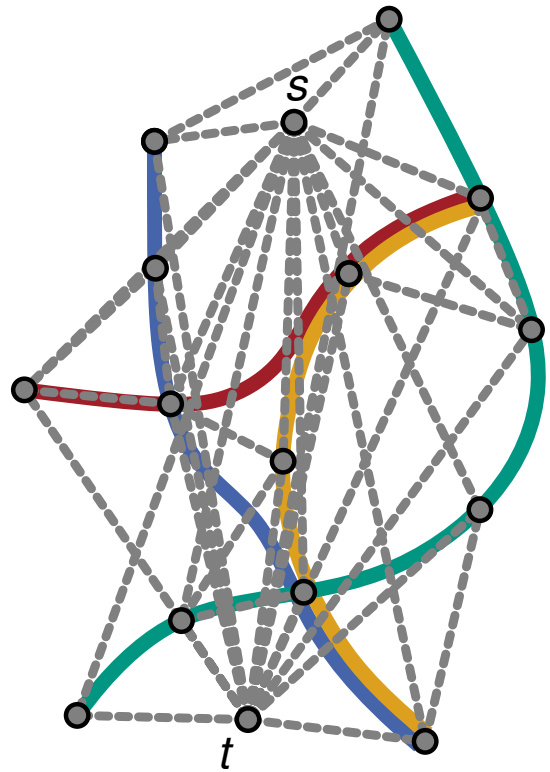
- Berechne initiale/finale Transfers mit One-to-Many-Algorithmus
  - One = Start/Ziel
  - Many = Stops
  - Schnelle Implementierung: Bucket-CH
- Baue temporären Transfer-Graph G' mit:
  - Vorberechneten Shortcuts
  - Initialen Transfers
  - Finalen Transfers
- Lasse beliebigen PT-Algorithmus auf PT-Netzwerk + G' laufen



Query – Black-Box

Black-Box-Query:

- Berechne initiale/finale Transfers mit One-to-Many-Algorithmus
  - One = Start/Ziel
  - Many = Stops
  - Schnelle Implementierung: Bucket-CH
- Baue temporären Transfer-Graph G' mit:
  - Vorberechneten Shortcuts
  - Initialen Transfers
  - Finalen Transfers
- Lasse beliebigen PT-Algorithmus auf PT-Netzwerk + G' laufen



Query – Direkte Integration

Temporärer Graph G' unterscheidet nicht zwischen Transfertypen

$\rightarrow$ direkte Integration in PT-Algorithmus

Query – Direkte Integration

Temporärer Graph G' unterscheidet nicht zwischen Transfertypen

$\rightarrow$ direkte Integration in PT-Algorithmus

Initiale Transfers:

- Liefern Distanz $d(s, u)$ zu jedem erreichbaren Stop u
- Initialisiere jeden erreichbaren Stop u mit $\tau_{\text{dep}} + d(s, u)$
- Beispiel RAPTOR: Aktualisiere Label, sammle Routen für u ein

Query – Direkte Integration

Temporärer Graph G' unterscheidet nicht zwischen Transfertypen

$\rightarrow$ direkte Integration in PT-Algorithmus

Initiale Transfers:

- Liefern Distanz $d(s, u)$ zu jedem erreichbaren Stop u
- Initialisiere jeden erreichbaren Stop u mit $\tau_{\text{dep}} + d(s, u)$
- Beispiel RAPTOR: Aktualisiere Label, sammle Routen für u ein

Finale Transfers:

- On-the-fly während PT-Query
- Wenn Query Stop v mit $d(v, t) < \infty$ erreicht:
  Überprüfe, ob finaler Transfer Ankunftszeit an t verbessert
- Beispiel RAPTOR: Beim Erreichen von v in der Routenphase

Query – Bucket-CH

Vorwärts-Bucket $\beta^{\uparrow}(u)$:

- Enthält Eintrag $(v, d(u, v))$ für jeden erreichbaren Stop v
- Befüllt durch Rückwärtssuchen von allen Stops
- Sortiere Einträge aufsteigend nach $d(u, v)$

Query – Bucket-CH

Vorwärts-Bucket $\beta^{\uparrow}(u)$:

- Enthält Eintrag $(v, d(u, v))$ für jeden erreichbaren Stop v
- Befüllt durch Rückwärtssuchen von allen Stops
- Sortiere Einträge aufsteigend nach $d(u, v)$

Bucket-Auswertung mit Target Pruning:

- CH-Suche zwischen s und $t \Rightarrow d(s, t)$ bekannt
- Scanne $\beta^{\uparrow}(u)$ nur, falls $d(s, u) \leq d(s, t)$
- Das gilt genau dann, wenn CH-Aufwärtssuche u besucht hat
- Scannen von $\beta^{\uparrow}(u)$:
  - Iteriere über Einträge $(v, d(u, v))$
  - Update $d(s, v)$ mit $d(s, u) + d(u, v)$
  - Breche ab, sobald $d(s, u) + d(u, v) > d(s, t)$

Query – Bucket-CH

Vorwärts-Bucket $\beta^{\uparrow}(u)$:

- Enthält Eintrag $(v, d(u, v))$ für jeden erreichbaren Stop v
- Befüllt durch Rückwärtssuchen von allen Stops
- Sortiere Einträge aufsteigend nach $d(u, v)$

Bucket-Auswertung mit Target Pruning:

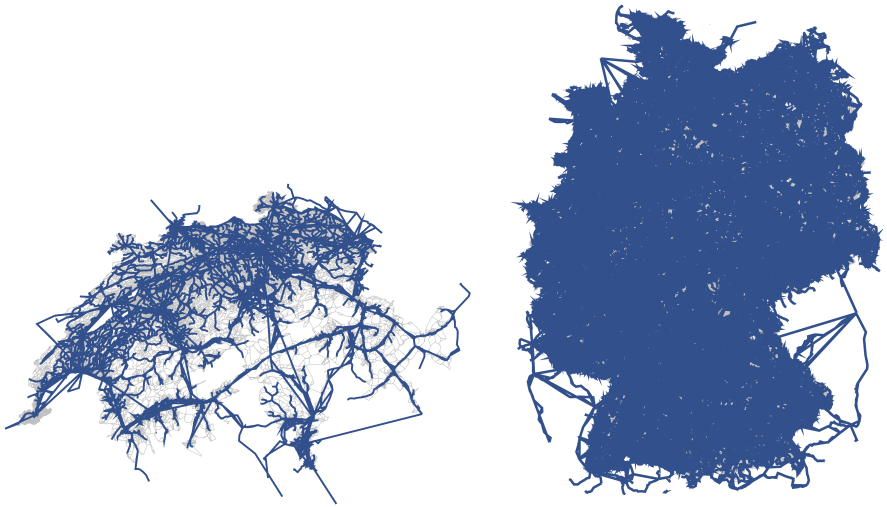
- CH-Suche zwischen s und $t \Rightarrow d(s, t)$ bekannt
- Scanne $\beta^{\uparrow}(u)$ nur, falls $d(s, u) \leq d(s, t)$
- Das gilt genau dann, wenn CH-Aufwärtssuche u besucht hat
- Scannen von $\beta^{\uparrow}(u)$:
  - Iteriere über Einträge $(v, d(u, v))$
  - Update $d(s, v)$ mit $d(s, u) + d(u, v)$
  - Breche ab, sobald $d(s, u) + d(u, v) > d(s, t)$

Analog für Rückwärts-Buckets $\beta^{\downarrow}(u)$

Experimente – Instanzen

Instanzen:

- Fahrpläne über zwei Tage
  - Schweiz (GTFS-Feed)
  - Deutschland (von DB)
- Transfer-Graphen von OSM
  - Straßen und Fußgängerzonen
  - Beachte Tempolimits
- Transitive Graphen zum Vergleich
  - Beschränkte Laufdauer
  - Avg. Knotengrad ≈ 100



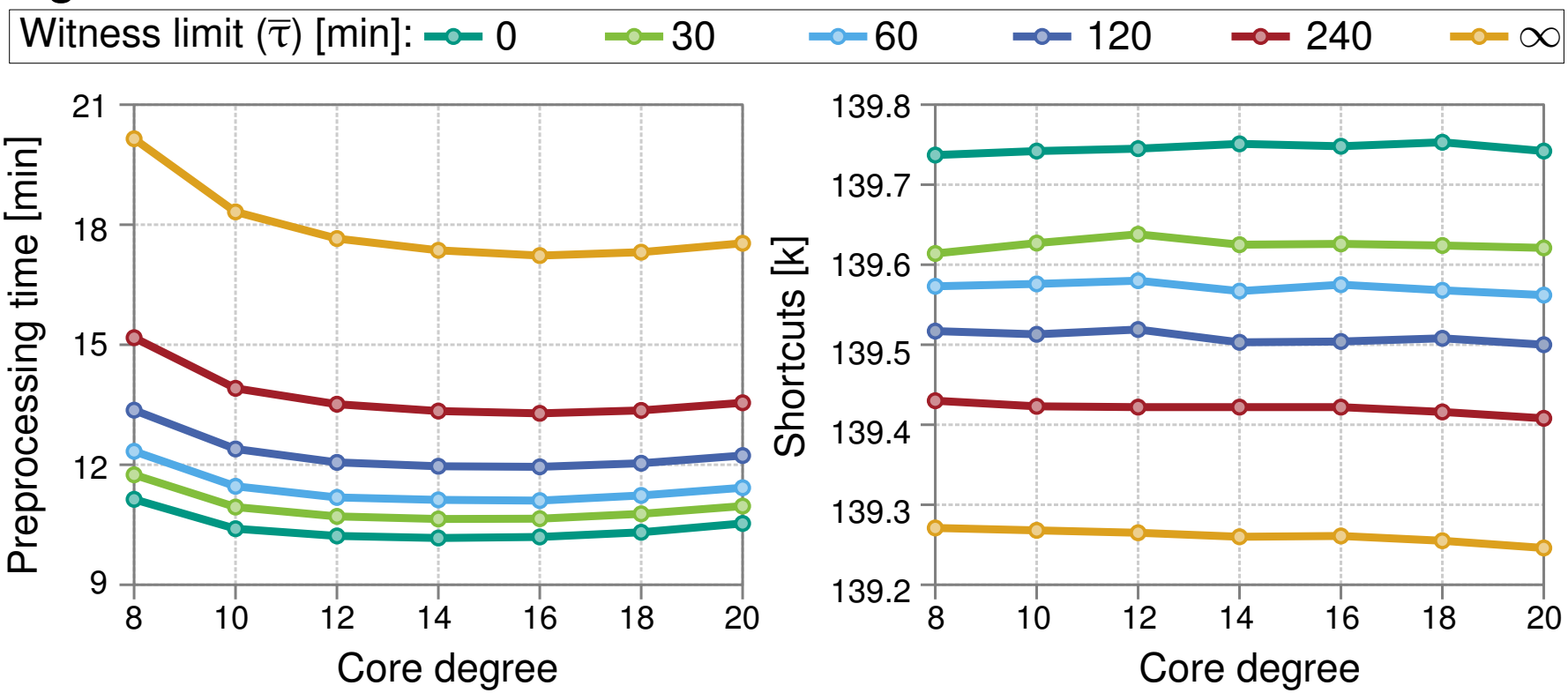
Network	Switzerland	Germany
Stops	25 426	244 055
Routes	13 934	231 089
Trips	369 534	2 387 297
Stop events	4 740 929	48 495 169
Vertices	604 167	6 872 105
Full edges	1 847 140	21 372 360
Transitive edges	4 687 016	22 645 480

Vorberechnung – Witness Limit

Aufbau:

- Schweiz mit Laufen als Transfermodus (4.5 km/h)
- Parallel mit 16 Kernen

Ergebnis:



Vorberechnung – Parallelisierung

Aufbau:

- Schweiz mit Laufen als Transfermodus (4.5 km/h)
- Witness Limit $\overline{\tau} = 15$ min (ab jetzt immer)

Ergebnis:

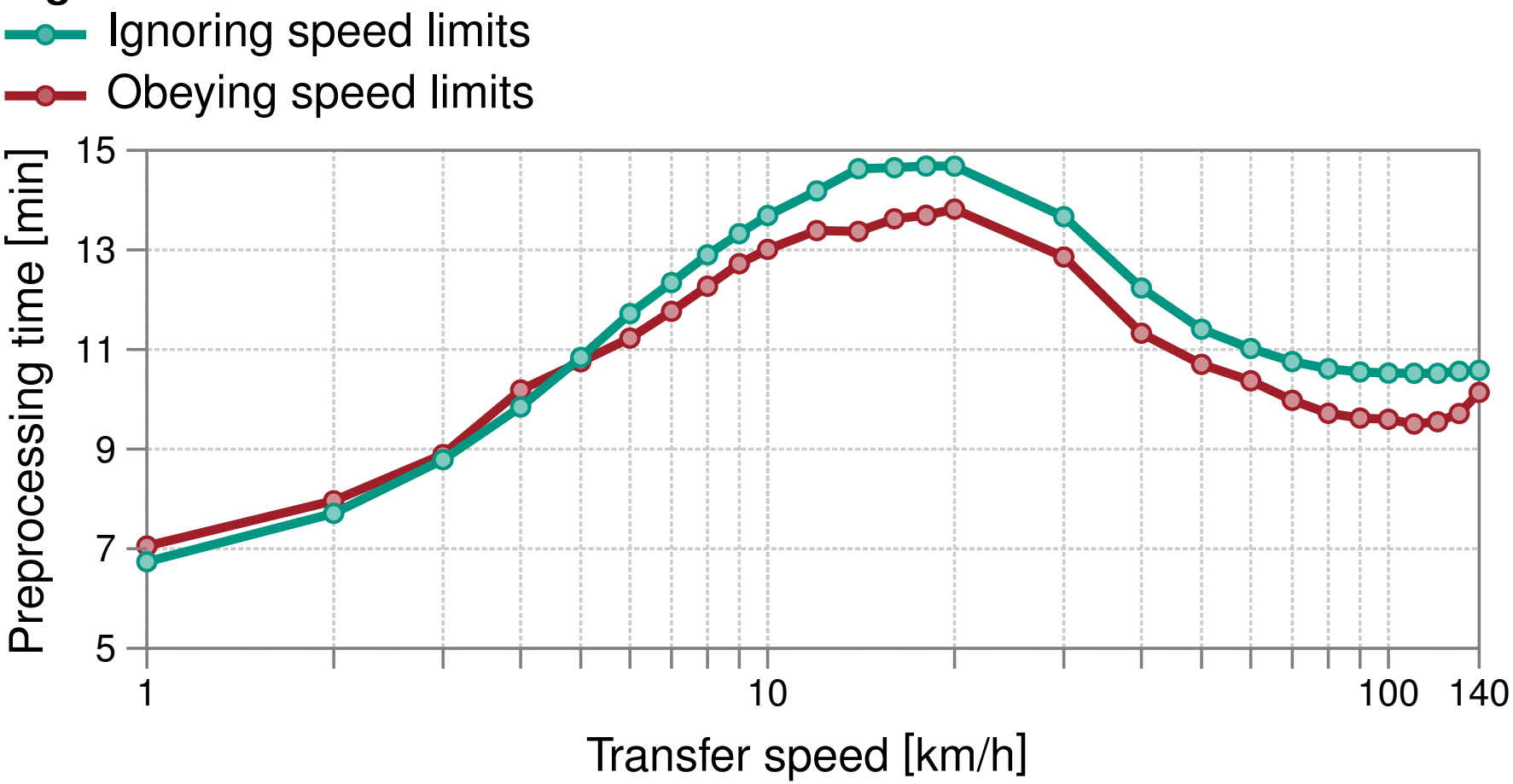
Number of threads	1	2	4	8	16
Preprocessing time [mm:ss]	2:00:56	58:03	31:11	17:29	10:12
Speed-up factor	1	2.08	3.88	6.92	11.85

Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph

Ergebnis:

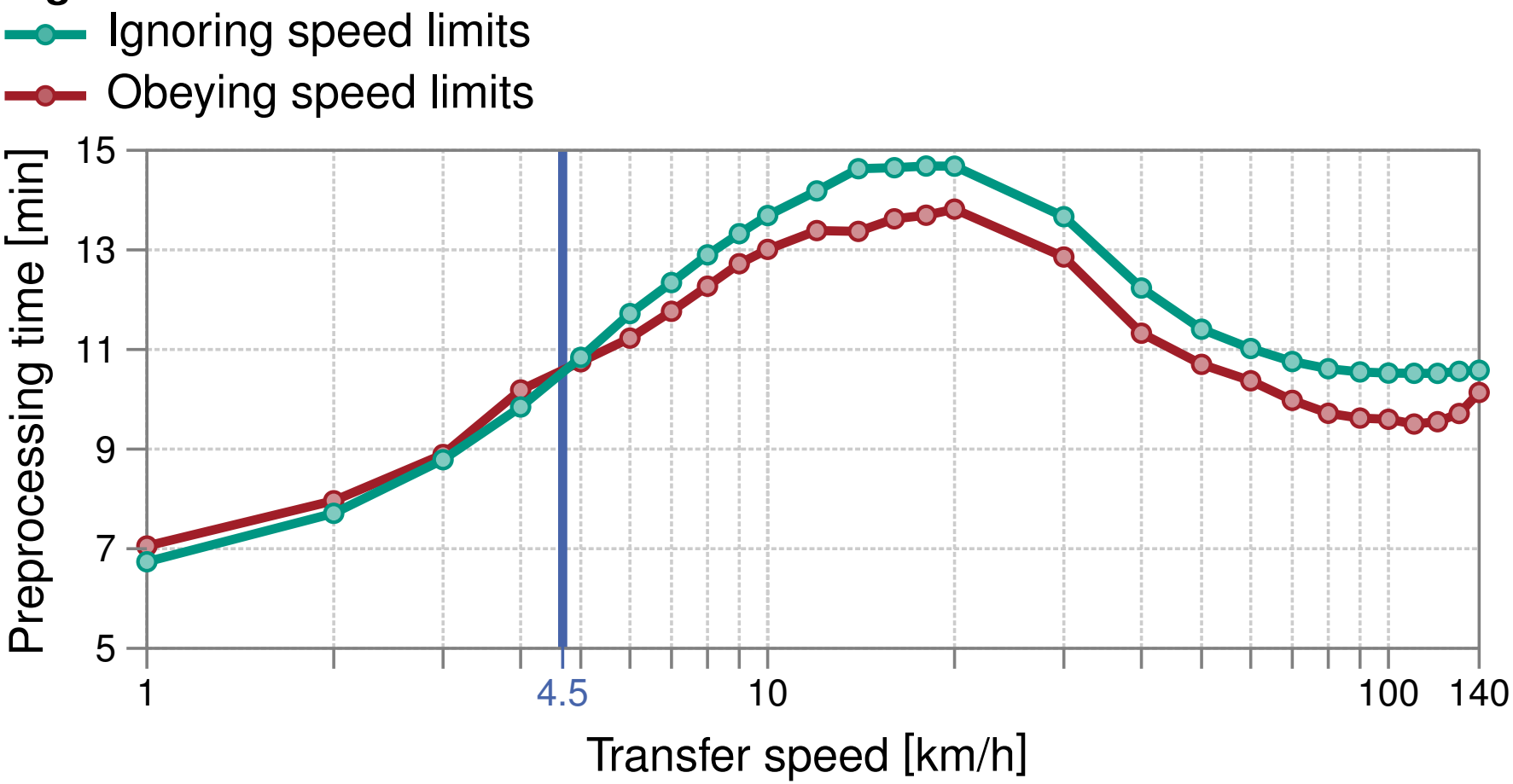


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph

Ergebnis:

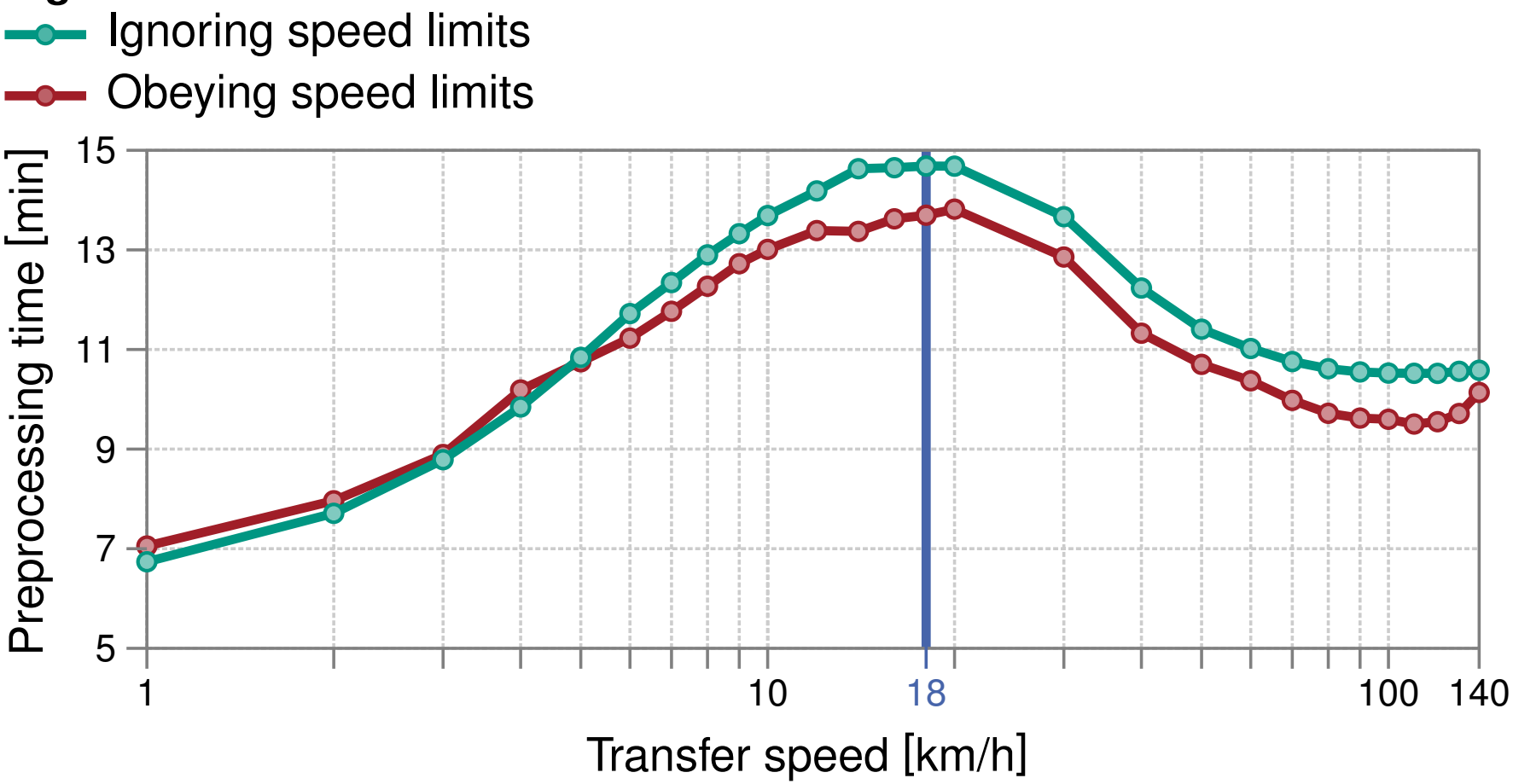


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph

Ergebnis:

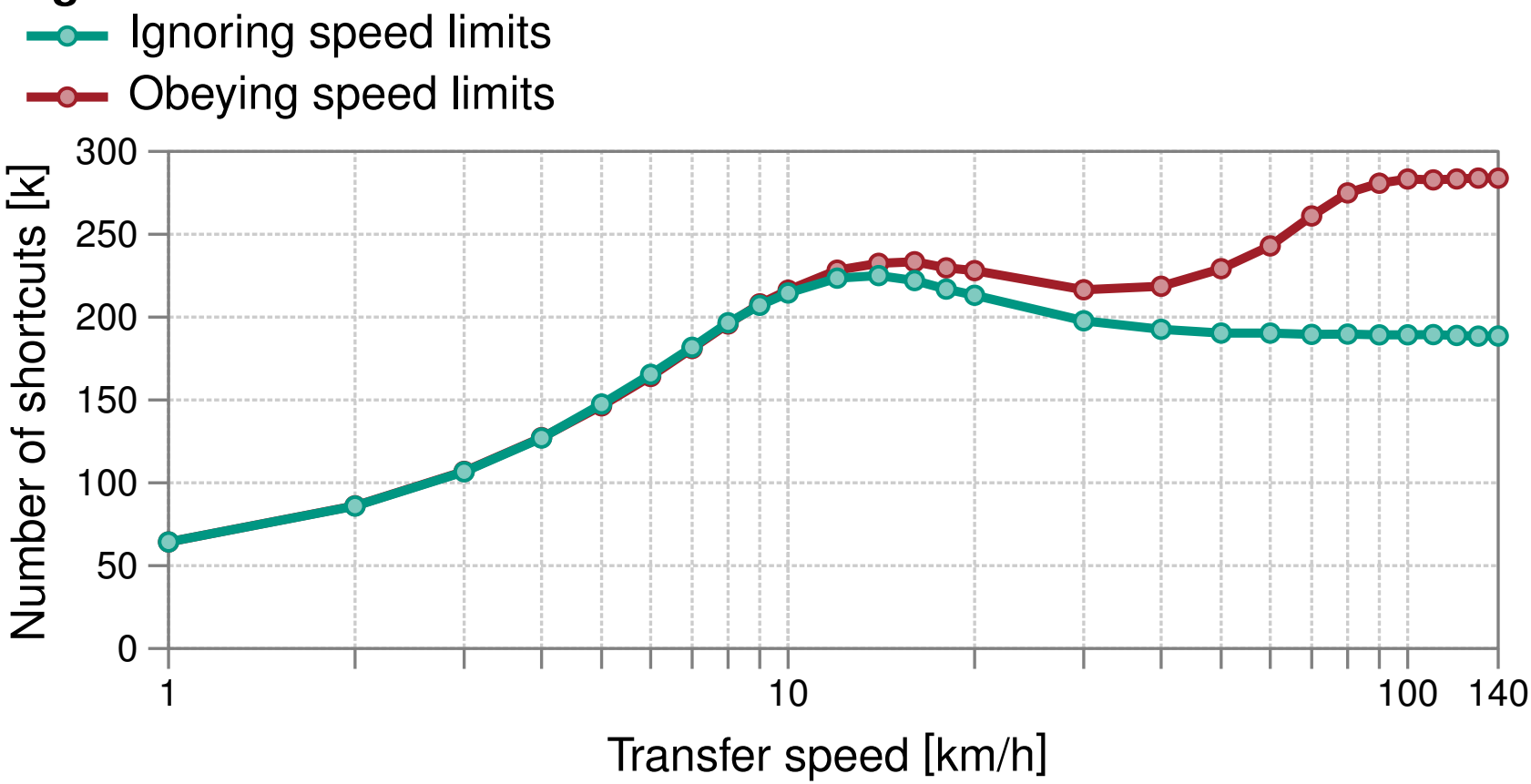


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph

Ergebnis:

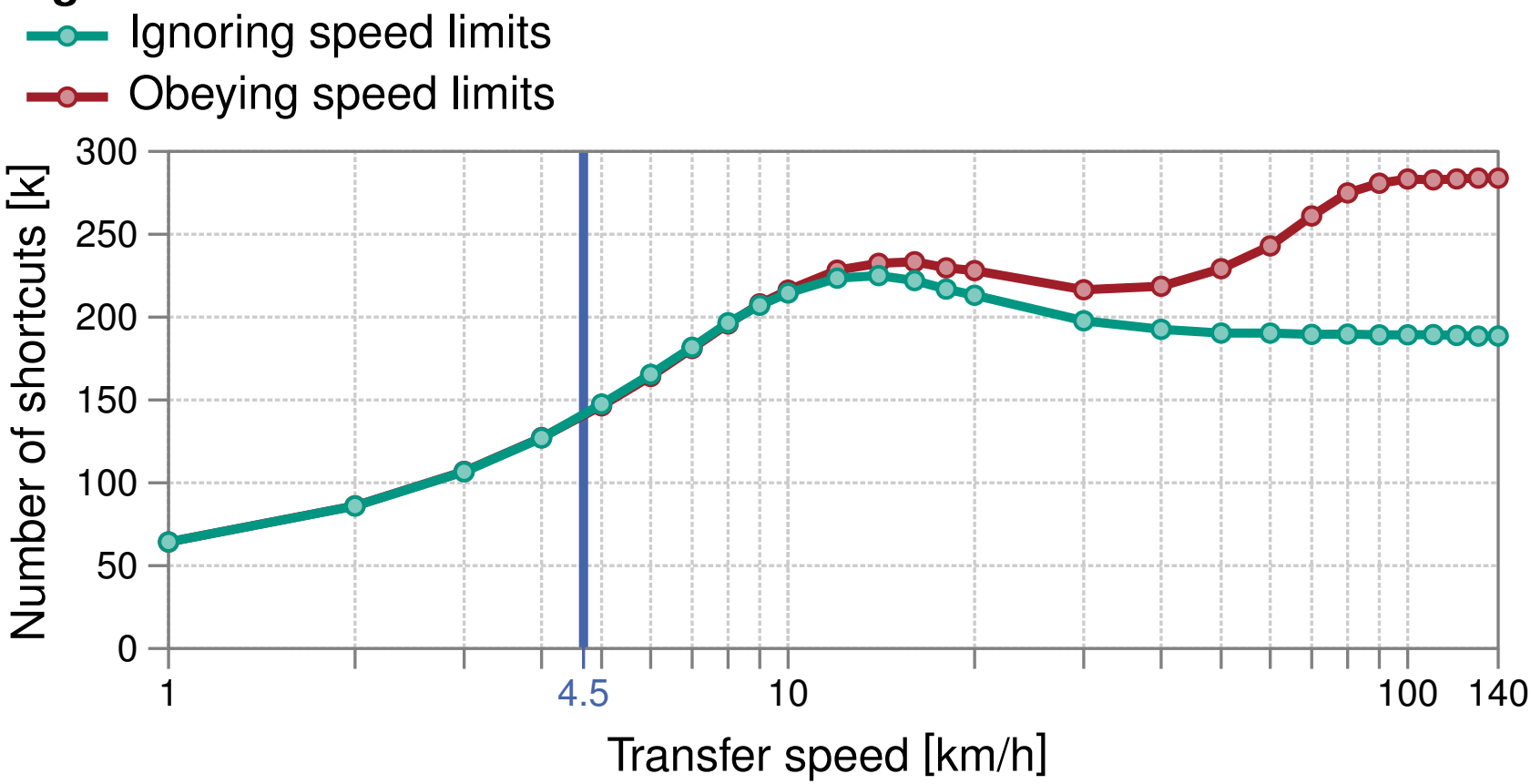


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph

Ergebnis:

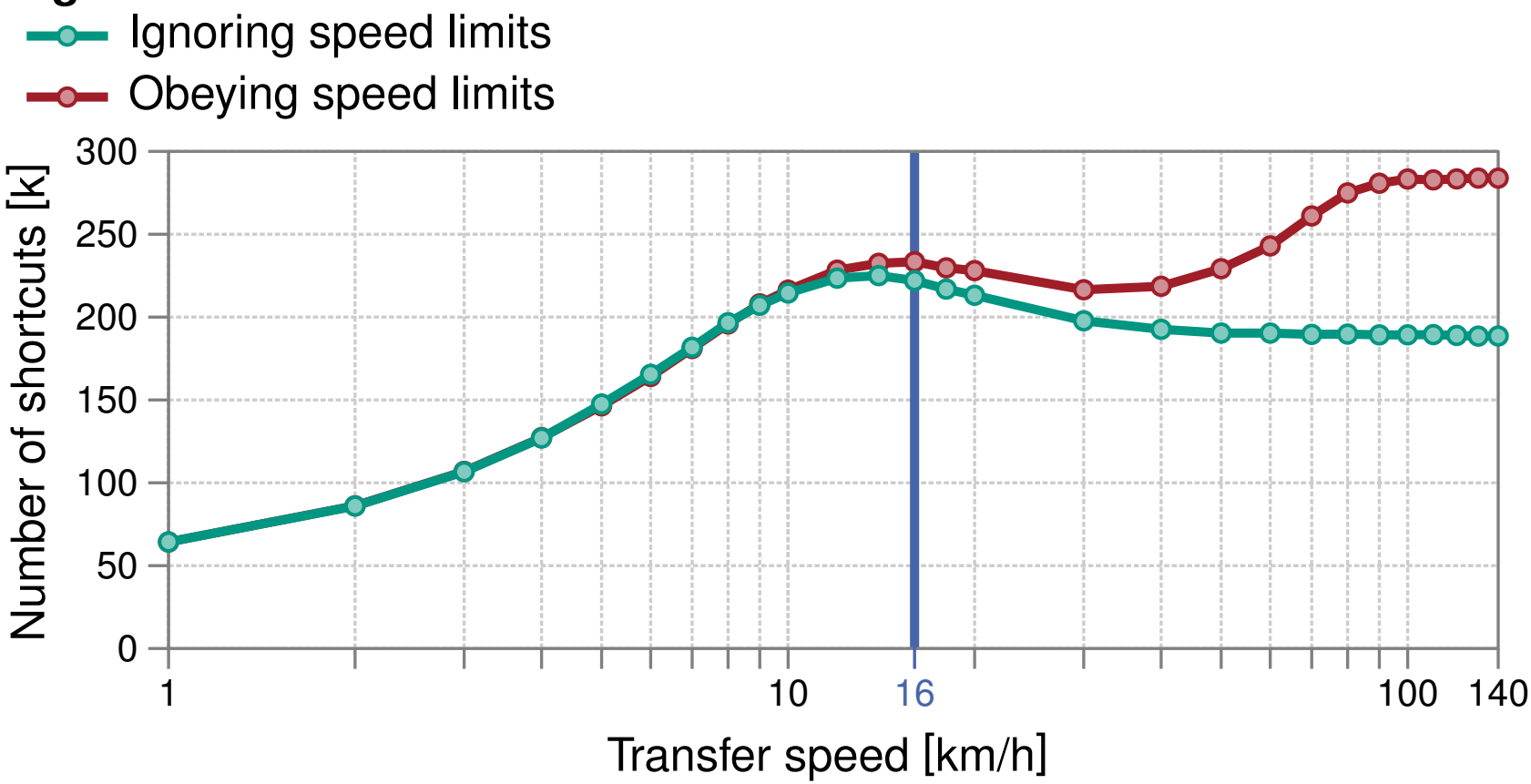


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph

Ergebnis:

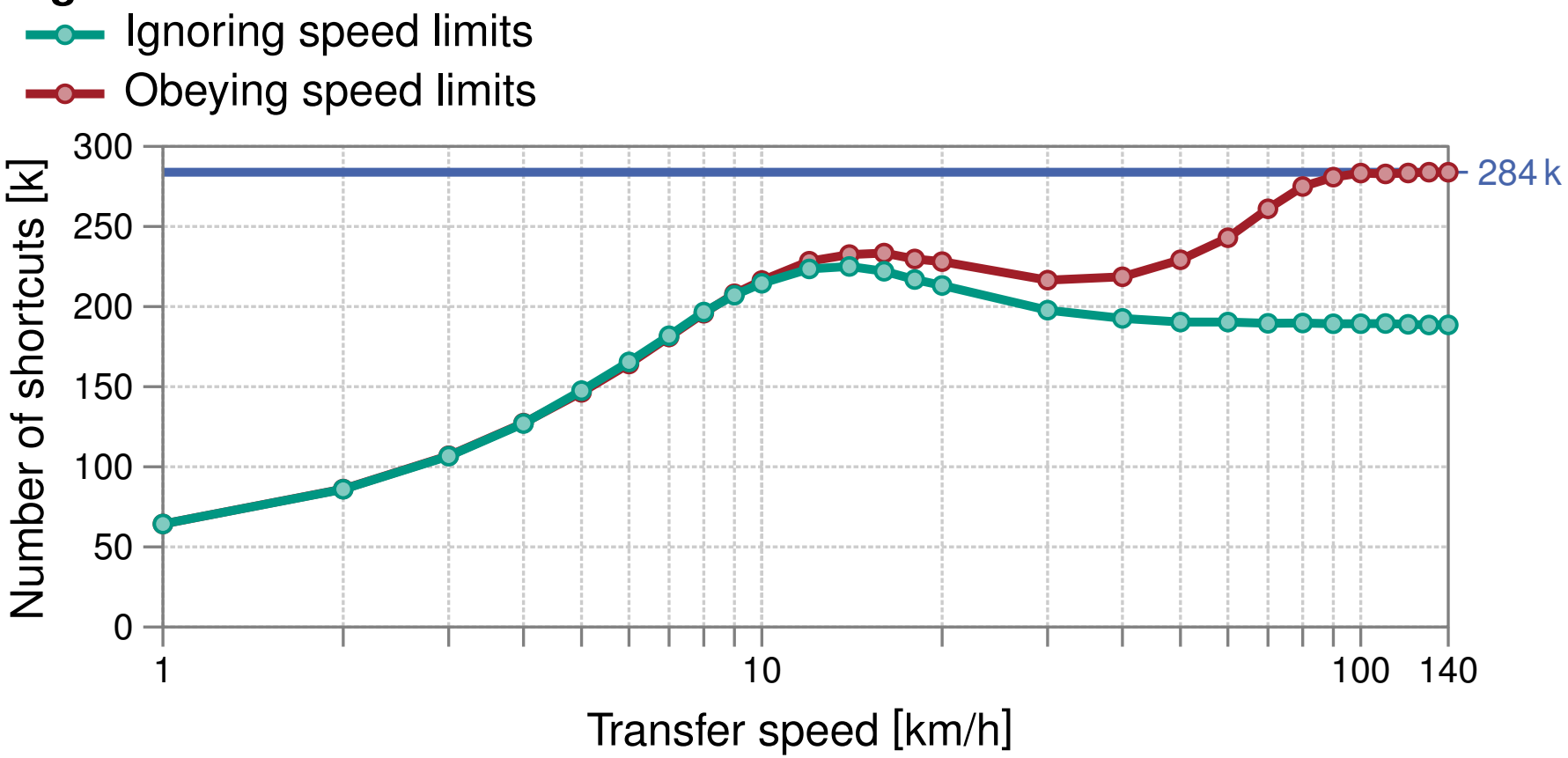


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph

Ergebnis:

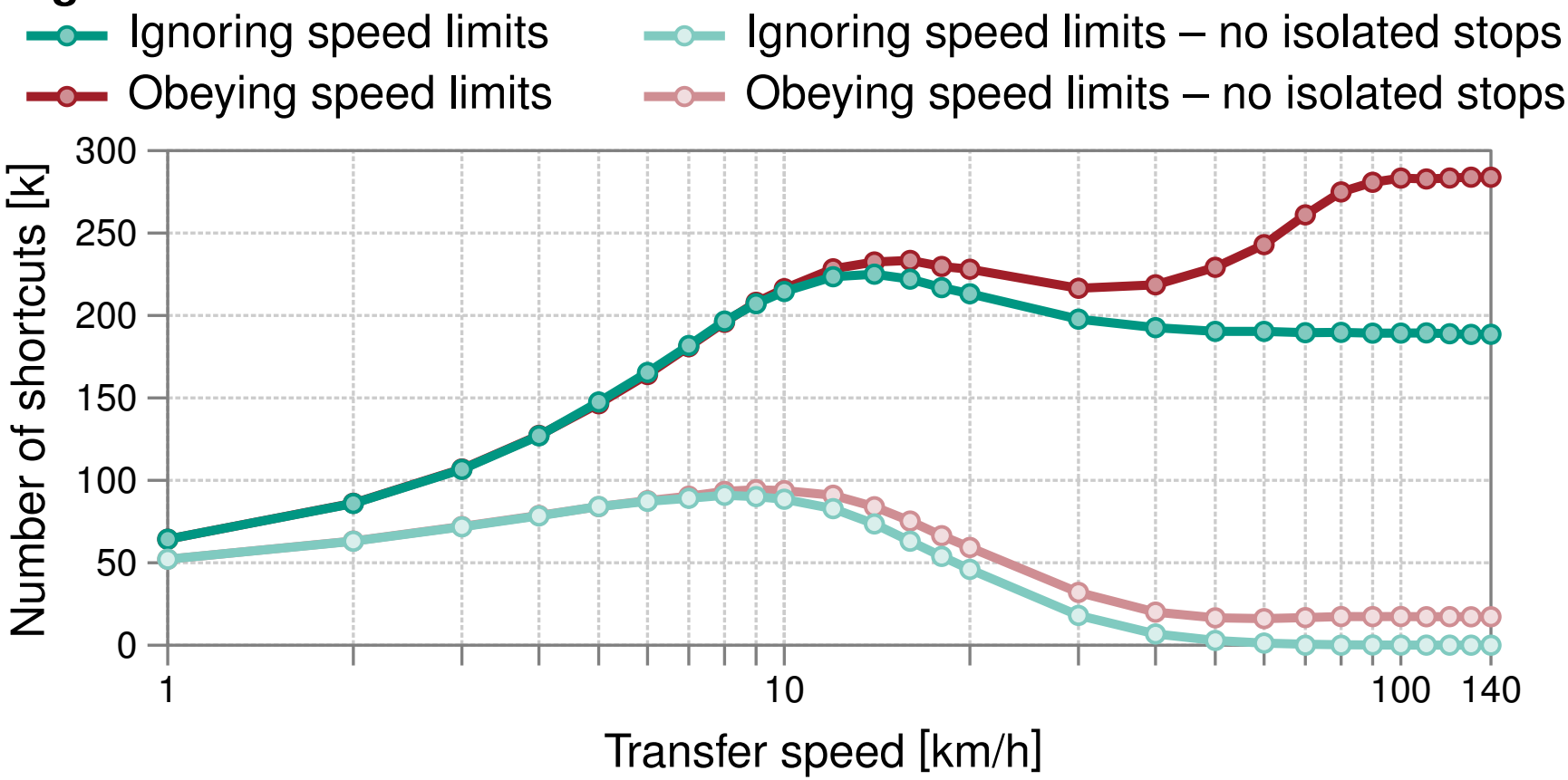


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph
- Im Transfergraph isolierte Stops wurden herausgefiltert

Ergebnis:

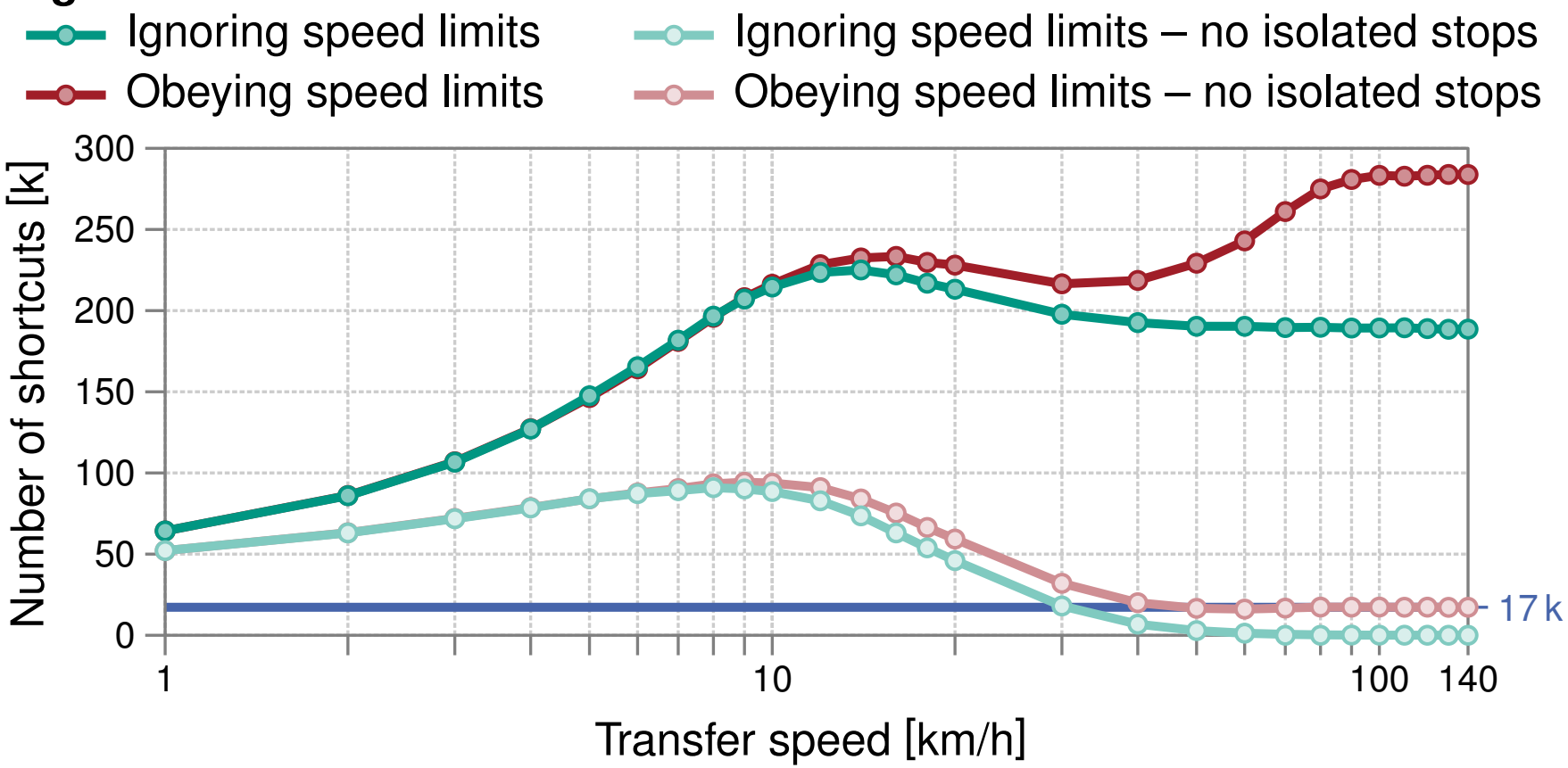


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph
- Im Transfergraph isolierte Stops wurden herausgefiltert

Ergebnis:

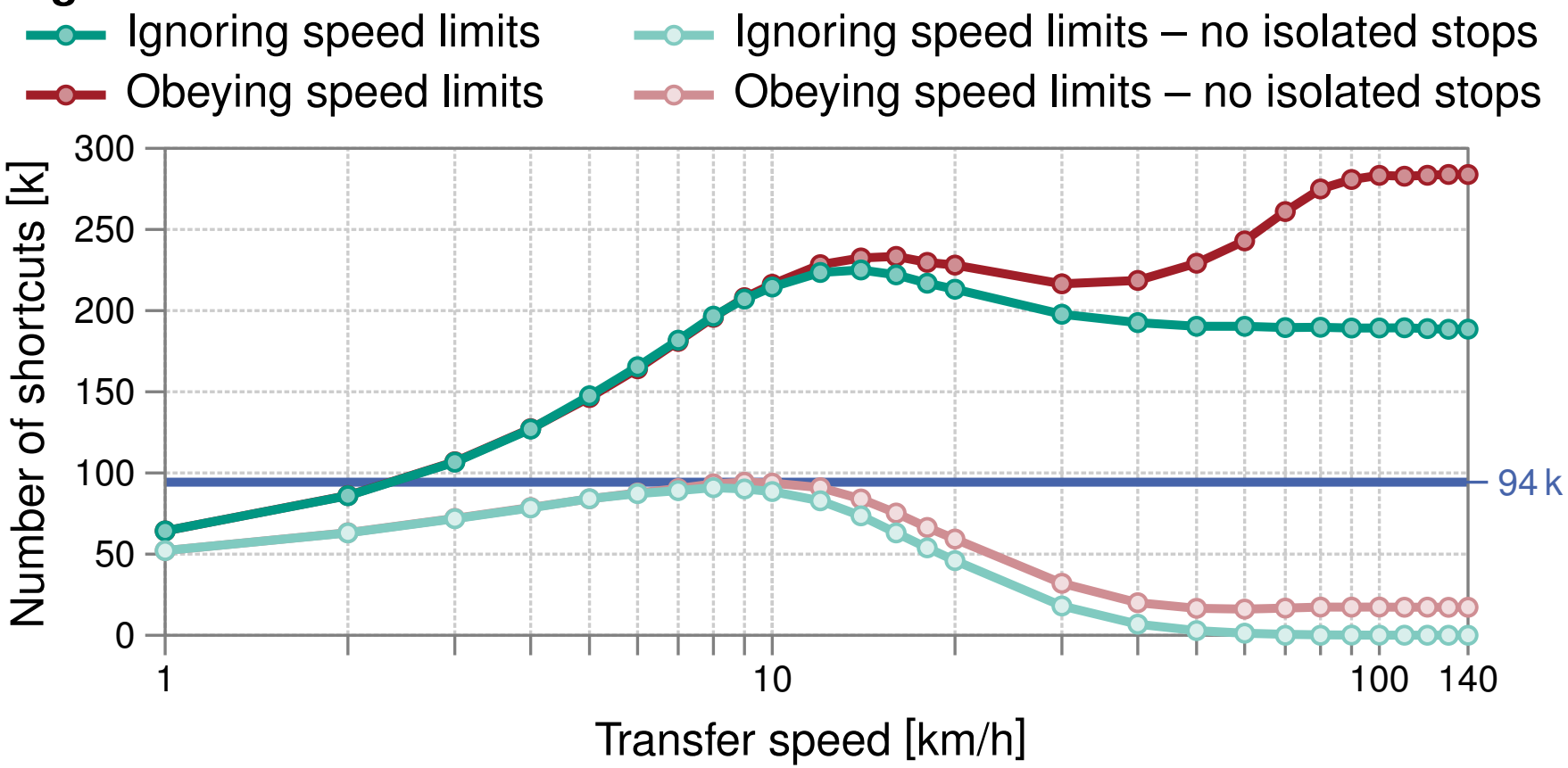


Vorberechnung

Aufbau:

- Schweiz mit verschiedenen Geschwindigkeiten für den Transfergraph
- Im Transfergraph isolierte Stops wurden herausgefiltert

Ergebnis:



Query – ULTRA-CSA

Setup:

- CSA-basierte Queries, optimieren nur Ankunftszeit
- MCSA kombiniert CSA mit Dijkstra (ähnlich MCR)
- Query-Typ für CSA*: Stop-zu-Stop
- Query-Typ für MCSA, ULTRA-CSA: Knoten-zu-Knoten

Network	Algorithm	Scans [k]		Time [ms]		
		Connections	Edges	Init.	Scan	Total
Switzerland (4.5 km/h)	CSA*	124.7	1 294	0.1	6.0	6.2
	MCSA	85.3	244	9.9	9.0	18.8
	ULTRA-CSA	84.7	80	1.3	4.2	5.6
Germany (4.5 km/h)	CSA*	2 564.0	6 269	1.7	145.8	147.5
	MCSA	1 527.8	3 182	148.2	185.9	334.1
	ULTRA-CSA	1 523.4	933	23.3	119.7	143.0

Query – ULTRA-RAPTOR

Setup:

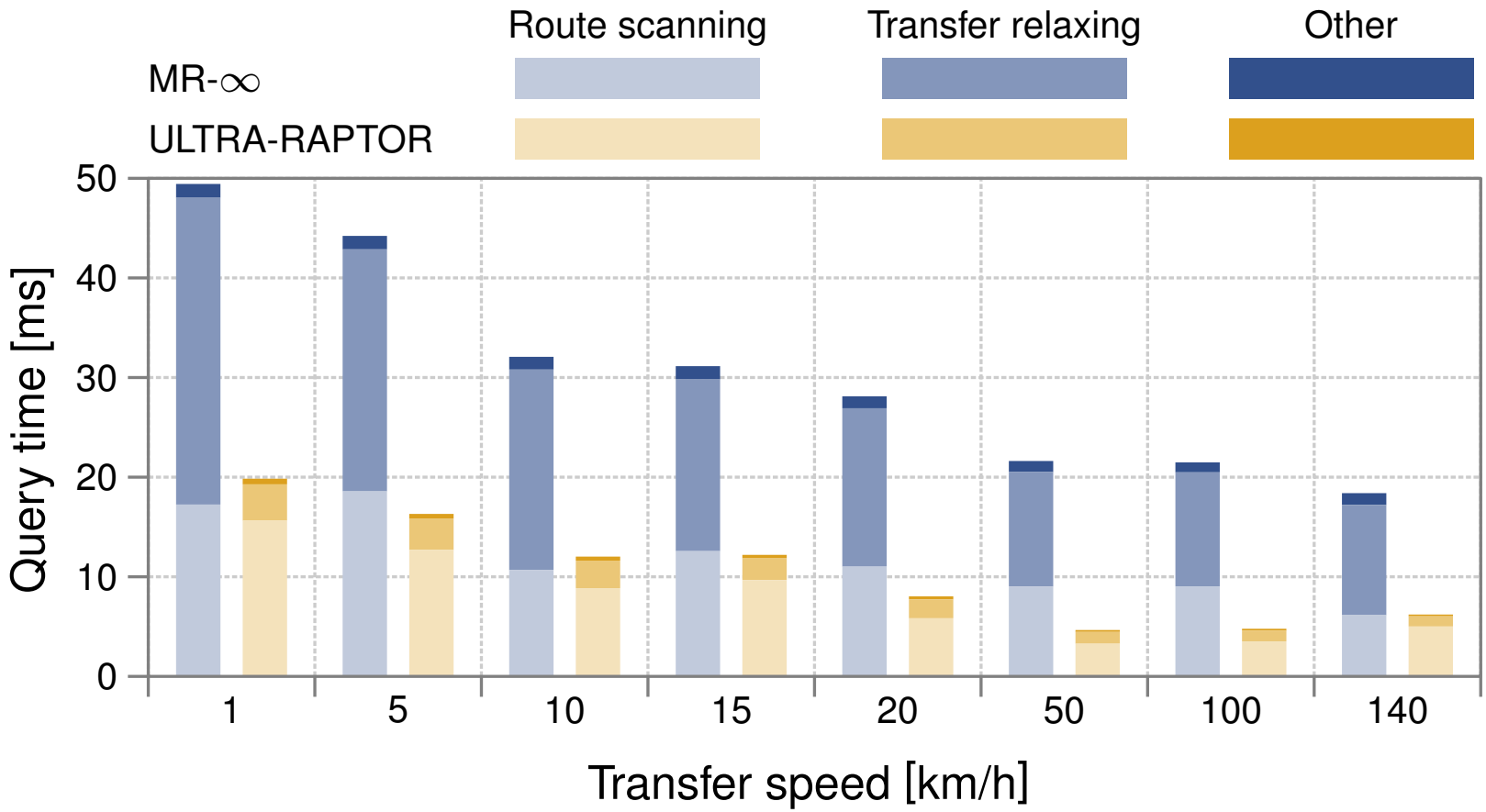
- RAPTOR-basierte Queries, optimieren Ankunftszeit und Anzahl Trips
- MR-∞ ist MCR mit unbeschränktem Laufen
- Query-Typ für RAPTOR*: Stop-zu-Stop
- Query-Typ für MR-∞, ULTRA-RAPTOR: Knoten-zu-Knoten

Network	Algorithm	Scans [k]		Time [ms]				
		Routes	Edges	Init.	Collect	Scan	Relax	Total
Switzerland (4.5 km/h)	RAPTOR*	27.2	3 527	0.0	3.7	6.4	7.8	18.4
	MR-∞	34.9	769	11.6	5.9	8.2	12.3	39.3
	ULTRA-RAPTOR	37.7	148	1.6	4.9	7.9	1.9	16.7
Germany (4.5 km/h)	RAPTOR*	480.4	25 798	0.0	166.9	178.0	85.1	436.5
	MR-∞	555.8	12 571	191.1	250.7	202.2	272.2	944.1
	ULTRA-RAPTOR	610.6	2 224	26.8	204.5	202.9	37.0	477.8

Query – ULTRA-RAPTOR

Setup:

- RAPTOR-basierte Queries, optimieren Ankunftszeit und Anzahl Trips



Ausblick

Erweiterungen der ULTRA-Vorberechnung:

- Zusätzliche Kriterien
  - Laufdistanz/Transferzeit [PS22]
  - Preis Offen
  - Fahrzeugbelegung In Arbeit
- Shortcuts mit Verspätungstoleranz In Arbeit
- Mehrere Transfermodi In Arbeit
- Kompliziertere Transfermodi (Bike-Sharing-Stationen) [SWZ20b]

Anwendungen der ULTRA-Shortcuts:

- Multi-modale Umlegung [SWZ19]
- Andere PT-Algorithmen (Trip-Based Routing, . . .) [SWZ20c]
- One-to-Many-Queries [SWZ20a]
- POI-Queries In Arbeit

Ende

Lehrveranstaltungen im WS 22/23

Lehrstuhl Algorithmik (Wagner):

- Vorlesung: Algorithmen zur Visualisierung von Graphen
- Praktikum: Algorithm Engineering (Routenplanung)
- Seminar: Algorithmentechnik
- Seminar: Energieinformatik

Lehrstuhl Scalable Algorithms (Bläsius):

- Vorlesung: Parametrisierte Algorithmen
- Praktikum: Fortgeschrittenes Algorithmisches Programmieren

Abschlussarbeiten:

- Einfach bei uns nachfragen

Praktikum Algorithm Engineering

Karlsruher Institut für Technologie

Praktikum zur Routenplanung

- 6 ECTS-Punkte
- Implementierung von Algorithmen aus der Vorlesung, z.B.
  - CRP
  - CCH
  - PHAST
  - ULTRA
  - CHArge
  - ...
- In 2er- oder 3er-Teams
- Möglichkeit, aktuelle Forschung praktisch auszuprobieren!

Details werden auf unserer Seite bekanntgegeben unter

`http://i11www.iti.kit.edu`

Literatur I

Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf.
UnLimited TRAnsfers for Multi-Modal Route Planning: An Efficient Solution.
In *27th Annual European Symposium on Algorithms (ESA 2019)*, pages 14:1–14:16, 2019.

Moritz Potthoff and Jonas Sauer.
Fast Multimodal Journey Planning for Three Criteria.
In *Proceedings of the 24th Workshop on Algorithm Engineering and Experiments (ALENEX 2022)*, pages 145–157, 2022.

Jonas Sauer, Dorothea Wagner, and Tobias Zündorf.
Efficient Computation of Multi-Modal Public Transit Traffic Assignments Using ULTRA.
In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 524–527, 2019.

Jonas Sauer, Dorothea Wagner, and Tobias Zündorf.
An Efficient Solution for One-to-Many Multi-Modal Journey Planning.
In *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)*, pages 1:1–1:15, 2020.

Literatur II

Jonas Sauer, Dorothea Wagner, and Tobias Zündorf.

Faster Multi-Modal Route Planning With Bike Sharing Using ULTRA.

In *18th International Symposium on Experimental Algorithms (SEA 2020)*, pages 16:1–16:14, 2020.

Jonas Sauer, Dorothea Wagner, and Tobias Zündorf.

Integrating ULTRA and Trip-Based Routing.

In *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)*, pages 4:1–1:15, 2020.

Dorothea Wagner and Tobias Zündorf.

Public Transit Routing with Unrestricted Walking.

In *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, 2017.