

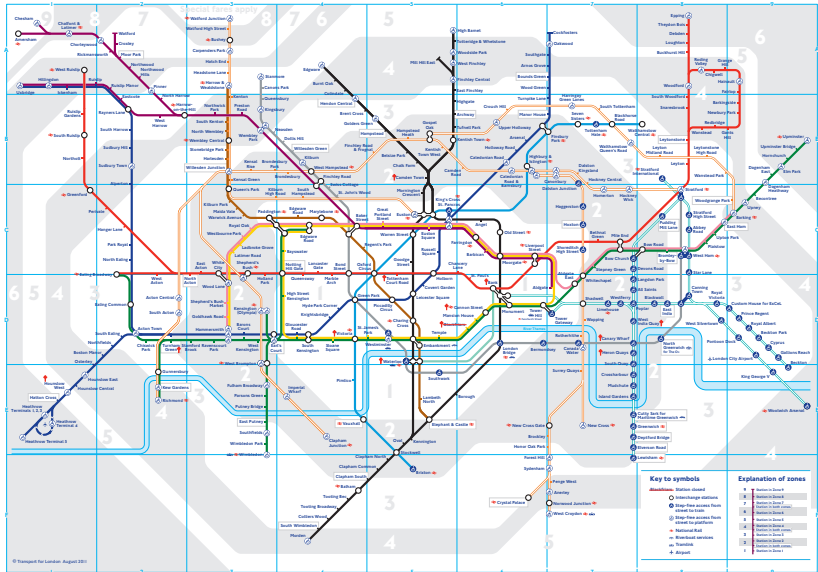
# Algorithmen für Routenplanung

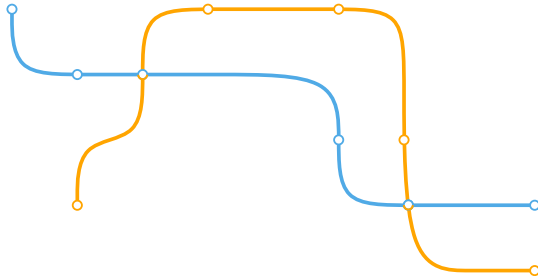
18. Vorlesung, Sommersemester 2019

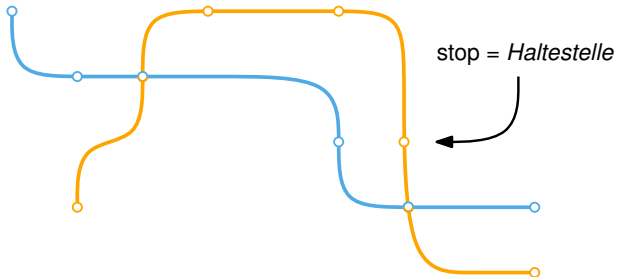
Jonas Sauer | 10. Juli 2019

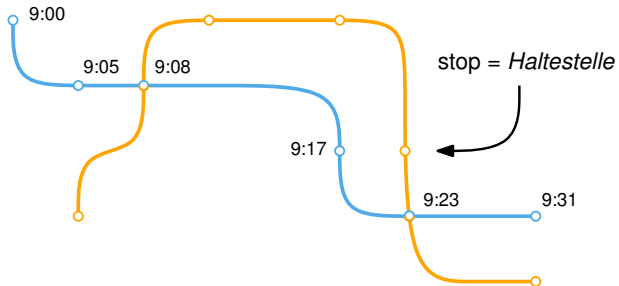
INSTITUT FÜR THEORETISCHE INFORMATIK · ALGORITHMIK · PROF. DR. DOROTHEA WAGNER

# Fahrplanauskunft

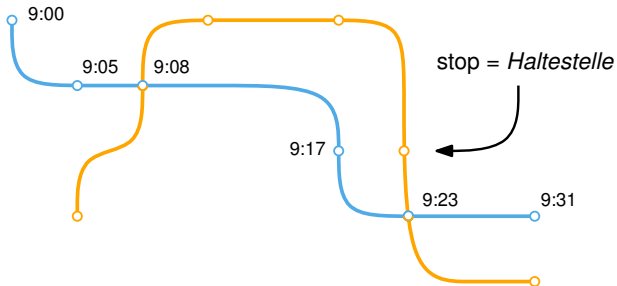
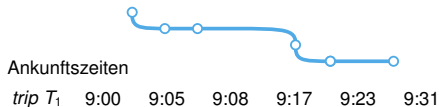




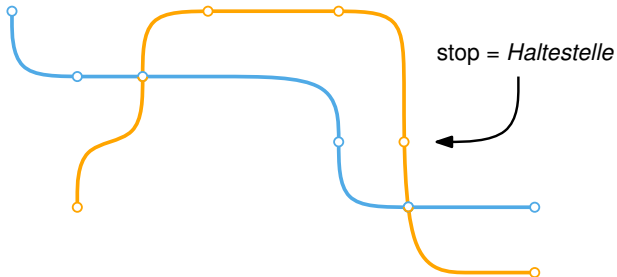
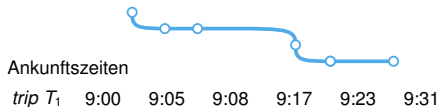




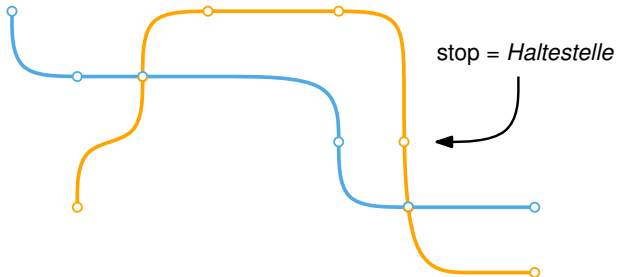
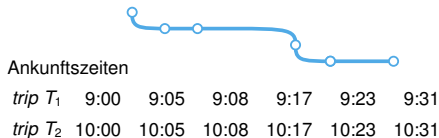
# Wdh. Begriffe



# Wdh. Begriffe




# Wdh. Begriffe



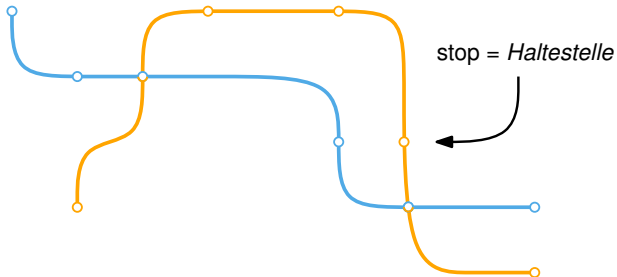


# Wdh. Begriffe



Ankunftszeiten

trip $T_1$	9:00	9:05	9:08	9:17	9:23	9:31
trip $T_2$	10:00	10:05	10:08	10:17	10:23	10:31
trip $T_3$	10:30	10:35	10:38	—	—	10:53



# Wdh. Begriffe

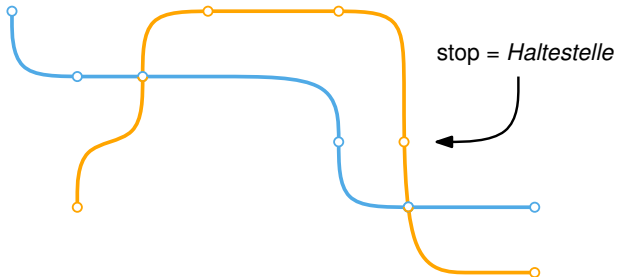


Ankunftszeiten

<i>trip</i> $T_1$	9:00	9:05	9:08	9:17	9:23	9:31
<i>trip</i> $T_2$	10:00	10:05	10:08	10:17	10:23	10:31
<i>trip</i> $T_3$	10:30	10:35	10:38	—	—	10:53

Route  $R_1$

Route  $R_2$



# Wdh. Begriffe

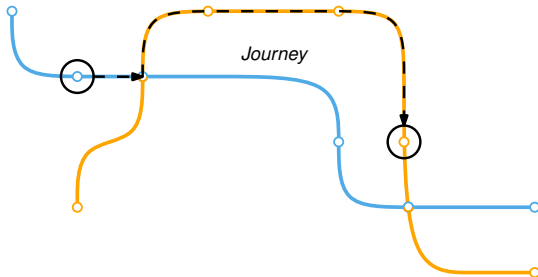


Ankunftszeiten

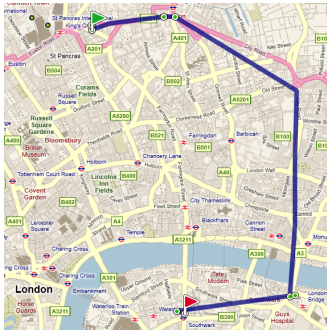
trip $T_1$	9:00	9:05	9:08	9:17	9:23	9:31
trip $T_2$	10:00	10:05	10:08	10:17	10:23	10:31
trip $T_3$	10:30	10:35	10:38	—	—	10:53

Route  $R_1$

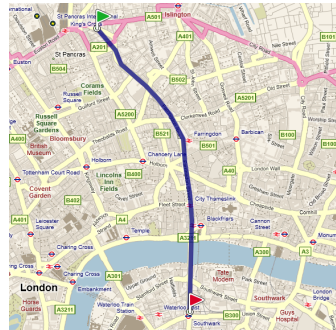
Route  $R_2$



**Gesucht:** „Gute“ Routen für Ankunftszeit *und* Anzahl Umstiege

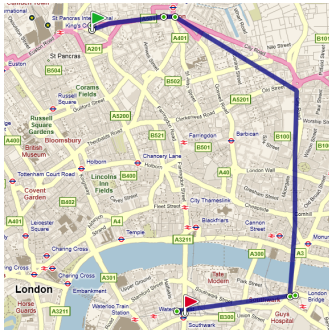


Ankunft 11:08 Uhr, 2 Umstiege

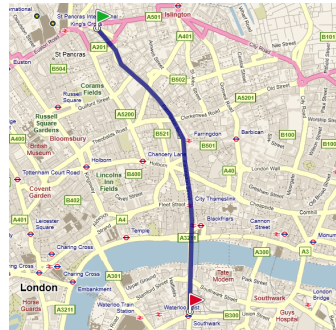


Ankunft 11:09 Uhr, 0 Umstiege

**Gesucht:** „Gute“ Routen für Ankunftszeit *und* Anzahl Umstiege



Ankunft 11:08 Uhr, 2 Umstiege



Ankunft 11:09 Uhr, 0 Umstiege

**Problem:**

Dijkstra-basierter Multi-Label-Correcting-Ansatz zu langsam

# Graph-Modelle?

## Bis jetzt:

- Modelliere Fahrplan als gerichteten Graphen
- Zeitexpandiert vs. zeitabhängig
- Verschiedene Varianten von Dijkstras Algorithmus
- Earliest Arrival-, Profil-, Multi-Criteria-Suchen

## Bis jetzt:

- Modelliere Fahrplan als gerichteten Graphen
- Zeitexpandiert vs. zeitabhängig
- Verschiedene Varianten von Dijkstras Algorithmus
- Earliest Arrival-, Profil-, Multi-Criteria-Suchen

## Probleme:

- Viele Knoten und Kanten
- Overhead von Priority Queue
- Wenig explizites Ausnutzen der Fahrplanstruktur
- Dynamische Szenarien erfordern Updates der Graph-Topologie
- Außerdem: Beschleunigungstechniken funktionieren nicht gut

## Bis jetzt:

- Modelliere Fahrplan als gerichteten Graphen
- Zeitexpandiert vs. zeitabhängig
- Verschiedene Varianten von Dijkstras Algorithmus
- Earliest Arrival-, Profil-, Multi-Criteria-Suchen

## Probleme:

- Viele Knoten und Kanten
- Overhead von Priority Queue
- Wenig explizites Ausnutzen der Fahrplanstruktur
- Dynamische Szenarien erfordern Updates der Graph-Topologie
- Außerdem: Beschleunigungstechniken funktionieren nicht gut

Sind Graphen die beste Art, Fahrpläne zu modellieren?



## Anforderungen:

## Anforderungen:

- Berechnet Pareto-Mengen  
mindestens Ankunftszeit und # Umstiege

## Anforderungen:

- Berechnet Pareto-Mengen  
mindestens Ankunftszeit und # Umstiege
- Nutzt die Struktur der Fahrpläne aus  
benutzt Routen und Trips explizit?

## Anforderungen:

- Berechnet Pareto-Mengen  
mindestens Ankunftszeit und # Umstiege
- Nutzt die Struktur der Fahrpläne aus  
benutzt Routen und Trips explizit?
- Funktioniert in dynamischen Szenarien  
Verspätungen, Zugausfälle, Routenänderungen; keine Vorberechnung

## Anforderungen:

- Berechnet Pareto-Mengen  
mindestens Ankunftszeit und # Umstiege
- Nutzt die Struktur der Fahrpläne aus  
benutzt Routen und Trips explizit?
- Funktioniert in dynamischen Szenarien  
Verspätungen, Zugausfälle, Routenänderungen; keine Vorberechnung
- Kann auf zusätzliche Kriterien erweitert werden  
z.B. Tarifzonen, Umstiegssicherheit, etc.

## Anforderungen:

- Berechnet Pareto-Mengen  
mindestens Ankunftszeit und # Umstiege
- Nutzt die Struktur der Fahrpläne aus  
benutzt Routen und Trips explizit?
- Funktioniert in dynamischen Szenarien  
Verspätungen, Zugausfälle, Routenänderungen; keine Vorberechnung
- Kann auf zusätzliche Kriterien erweitert werden  
z.B. Tarifzonen, Umstiegssicherheit, etc.
- ... und ist hinreichend schnell  
für interaktive Szenarien

## Anforderungen:

- Berechnet Pareto-Mengen  
mindestens Ankunftszeit und # Umstiege
- Nutzt die Struktur der Fahrpläne aus  
benutzt Routen und Trips explizit?
- Funktioniert in dynamischen Szenarien  
Verspätungen, Zugausfälle, Routenänderungen; keine Vorberechnung
- Kann auf zusätzliche Kriterien erweitert werden  
z.B. Tarifzonen, Umstiegssicherheit, etc.
- ... und ist hinreichend schnell  
für interaktive Szenarien

**RAPTOR: Round-bAsed Public Transit Optimized Router**

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.



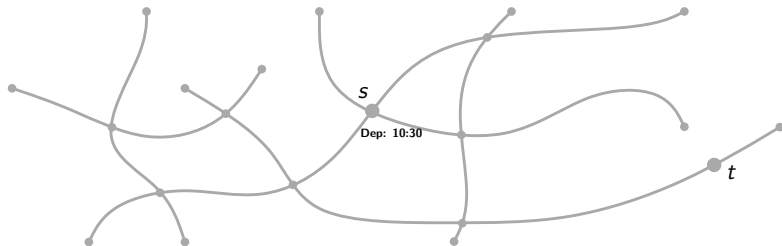
# Runden

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

**Idee:** Eine **Runde** für jeden genommenen Trip.

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

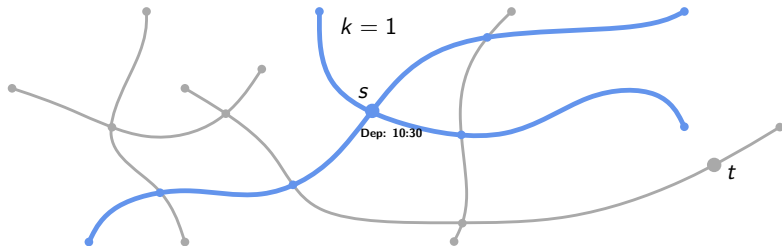
**Idee:** Eine **Runde** für jeden genommenen Trip.



**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

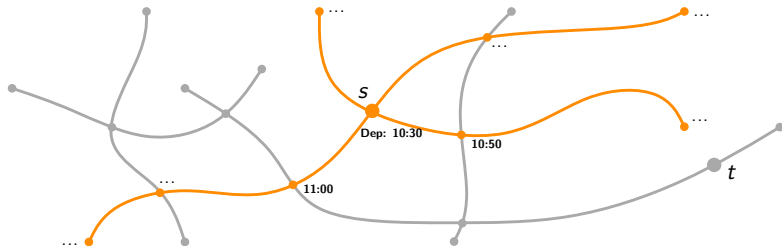
**Idee:** Eine **Runde** für jeden genommenen Trip.



**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

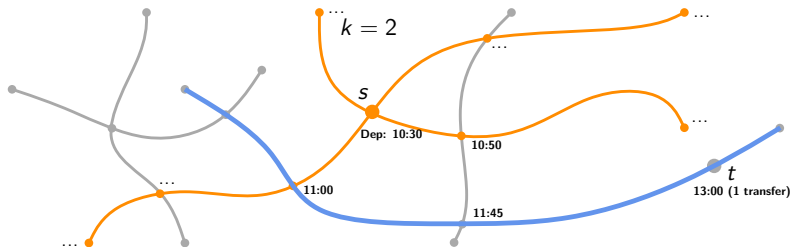
**Idee:** Eine **Runde** für jeden genommenen Trip.



**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

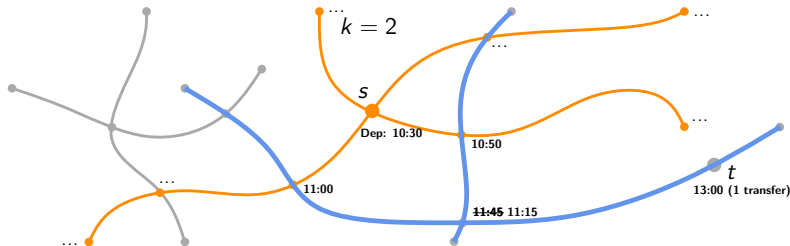
**Idee:** Eine **Runde** für jeden genommenen Trip.



**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

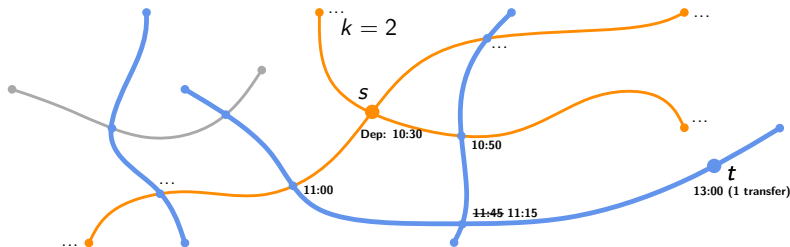
**Idee:** Eine **Runde** für jeden genommenen Trip.



**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

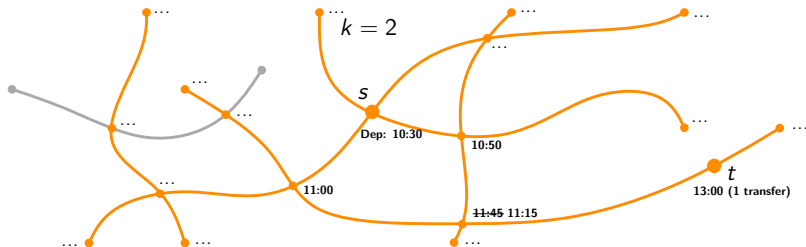
**Idee:** Eine **Runde** für jeden genommenen Trip.



**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

**Idee:** Eine **Runde** für jeden genommenen Trip.

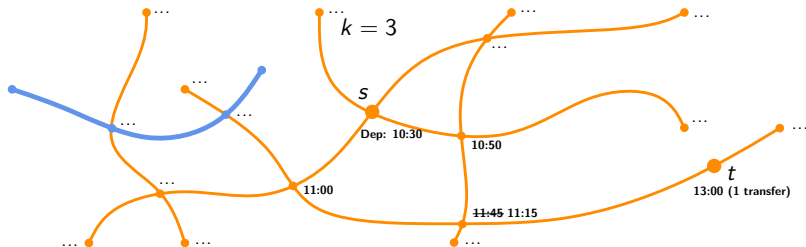


**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.



**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

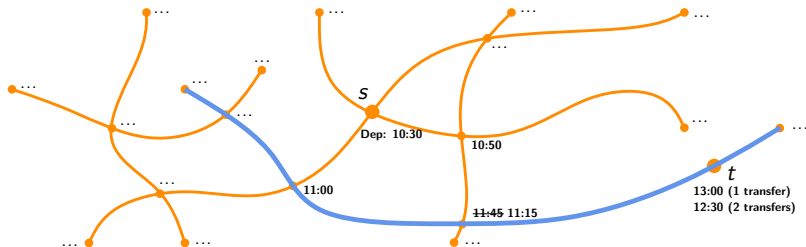
**Idee:** Eine **Runde** für jeden genommenen Trip.



**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.

**Beobachtung:** Wechseln zw. Trips führt immer zu einem Umstieg.

**Idee:** Eine **Runde** für jeden genommenen Trip.

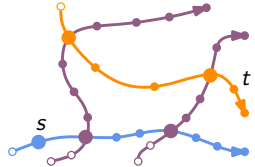


**Ansatz:** In Runde  $k$  werden Ankunftszeiten für  $k$  Trips berechnet.

Scanne jede **Route** höchstens einmal pro Runde.

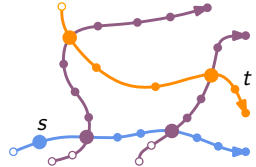
Für jede Runde  $k \leftarrow 1, 2, \dots$

- 1 Scanne jede Route
- 2 Relaxiere Fußwege



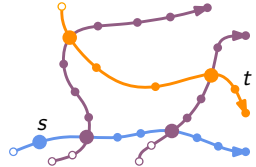
Für jede Runde  $k \leftarrow 1, 2, \dots$

- 1 Scanne jede Route
- 2 Relaxiere Fußwege



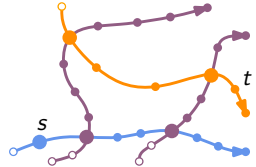
Für jede Runde  $k \leftarrow 1, 2, \dots$

- 1 Scanne jede Route
- 2 Relaxiere Fußwege



Für jede Runde  $k \leftarrow 1, 2, \dots$

- 1 Scanne jede Route
- 2 Relaxiere Fußwege



Terminiere, wenn ... ?

Some route

Current Trip:  $\perp$



$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	$\infty$	8:19	7:12	$\infty$	$\infty$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip:  $\perp$



$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	$\infty$	8:19	7:12	$\infty$	$\infty$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$



Some route

Current Trip:  $\perp$



	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	$\infty$	8:19	7:12	$\infty$	$\infty$
	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 13



9:20

	...	...	...	...	...	...
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$
$k$	$\infty$	$\infty$	$\infty$	8:19	7:12	$\infty$
	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 13



9:26

$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	$\infty$	8:19	7:12	$\infty$	$\infty$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 13



9:26

...	...	...	...	...	...	...	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	9:26	8:19	7:12	$\infty$	$\infty$
...	...	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 13



...	...	...	...	...	...	...
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$
$k$	$\infty$	$\infty$	<b>9:26</b>	8:19	7:12	$\infty$
...	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 11



...	...	...	...	...	...	...
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$
$k$	$\infty$	$\infty$	9:26	8:19	7:12	$\infty$
...	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 11



...	...	...	...	...	...	...	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	<b>9:26</b>	8:19	7:12	$\infty$	$\infty$
...	...	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 3



...	...	...	...	...	...	...	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	9:26	8:19	7:12	$\infty$	$\infty$
...	...	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$



Some route

Current Trip: 3



	...	...	...	...	...	...
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$
$k$	$\infty$	$\infty$	<b>9:26</b>	8:19	7:12	$\infty$
	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 3



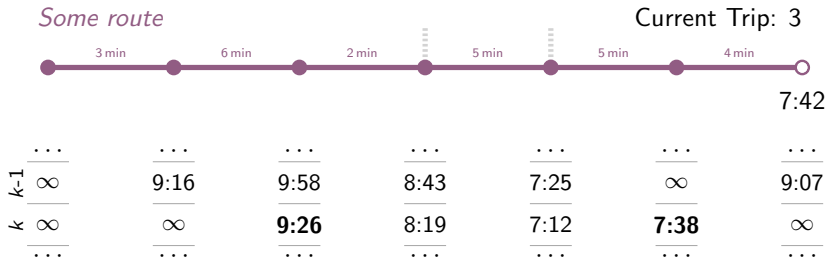
7:38

...	...	...	...	...	...	...
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$
$k$	$\infty$	$\infty$	<b>9:26</b>	8:19	7:12	<b>7:38</b>
...	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$



- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 3



7:42

...	...	...	...	...	...	...
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$
$k$	$\infty$	$\infty$	<b>9:26</b>	8:19	7:12	<b>7:38</b>
...	...	...	...	...	...	...

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 3



$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	<b>9:26</b>	8:19	7:12	<b>7:38</b>	<b>7:42</b>
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Some route

Current Trip: 3



$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
$k-1$	$\infty$	9:16	9:58	8:43	7:25	$\infty$	9:07
$k$	$\infty$	$\infty$	<b>9:26</b>	8:19	7:12	<b>7:38</b>	<b>7:42</b>
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

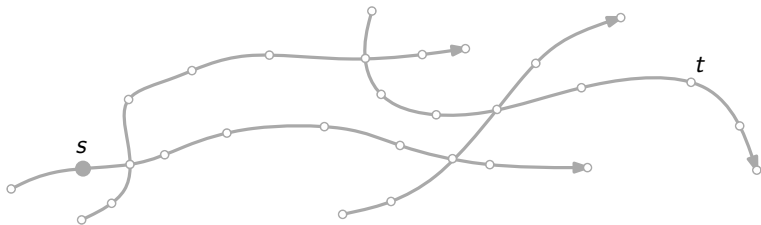
- Jeder Stop hat ein Label (Ankunftszeit) *pro Runde*
- Aktiver Trip entlang der Route wird stets verbessert

In Runde  $k$ :

- Update Labels von Runde  $k$  mit Labels aus Runde  $k - 1$

Dynamischer Programmierungsansatz

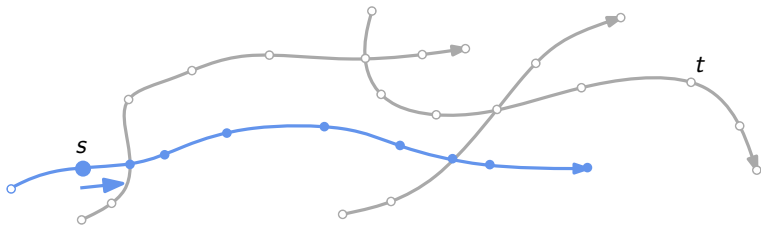
**Beobachtung:** Nicht alle Routen werden in jeder Runde erreicht.



## Markieren und Pruning:

- Route scannen: Markiere Stop, falls Ankunftszeit verbessert
- Nächste Runde: Nur Routen von markierten Stops scannen
- Scanne jede Route ab ihrem ersten markierten Stop

**Beobachtung:** Nicht alle Routen werden in jeder Runde erreicht.

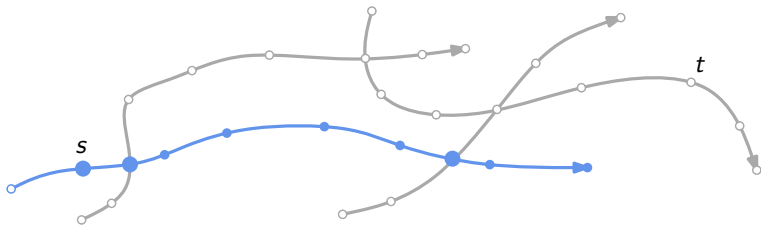


## Markieren und Pruning:

- Route scannen: Markiere Stop, falls Ankunftszeit verbessert
- Nächste Runde: Nur Routen von markierten Stops scannen
- Scanne jede Route ab ihrem ersten markierten Stop



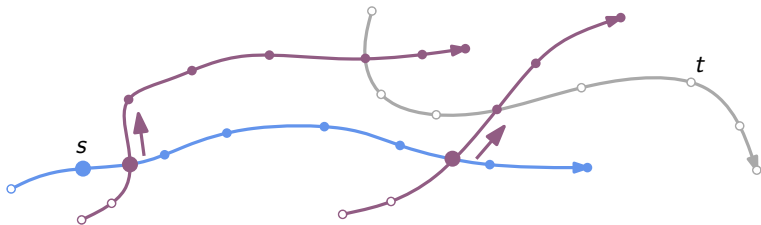
**Beobachtung:** Nicht alle Routen werden in jeder Runde erreicht.



## Markieren und Pruning:

- Route scannen: Markiere Stop, falls Ankunftszeit verbessert
- Nächste Runde: Nur Routen von markierten Stops scannen
- Scanne jede Route ab ihrem ersten markierten Stop

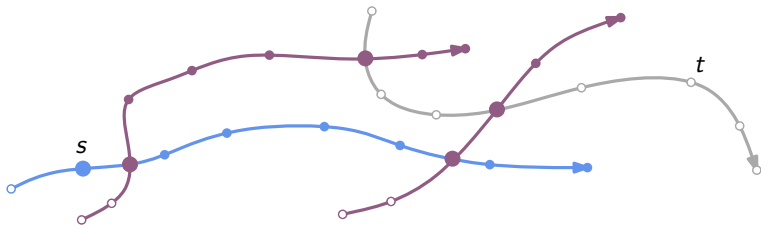
**Beobachtung:** Nicht alle Routen werden in jeder Runde erreicht.



## Markieren und Pruning:

- Route scannen: Markiere Stop, falls Ankunftszeit verbessert
- Nächste Runde: Nur Routen von markierten Stops scannen
- Scanne jede Route ab ihrem ersten markierten Stop

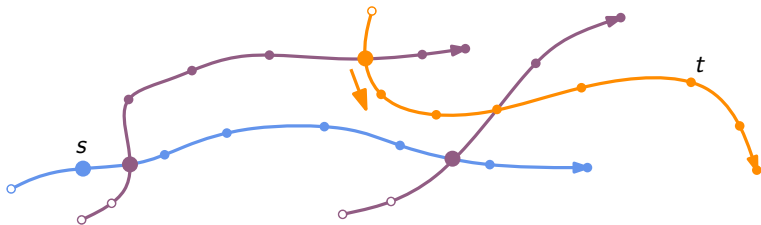
**Beobachtung:** Nicht alle Routen werden in jeder Runde erreicht.



## Markieren und Pruning:

- Route scannen: Markiere Stop, falls Ankunftszeit verbessert
- Nächste Runde: Nur Routen von markierten Stops scannen
- Scanne jede Route ab ihrem ersten markierten Stop

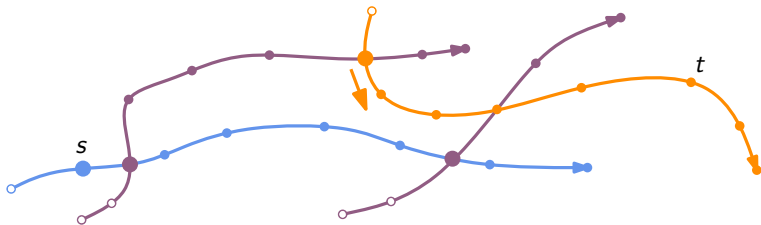
**Beobachtung:** Nicht alle Routen werden in jeder Runde erreicht.



**Markieren und Pruning:**

- Route scannen: Markiere Stop, falls Ankunftszeit verbessert
- Nächste Runde: Nur Routen von markierten Stops scannen
- Scanne jede Route ab ihrem ersten markierten Stop

**Beobachtung:** Nicht alle Routen werden in jeder Runde erreicht.

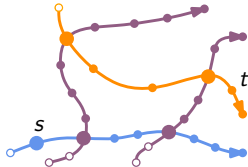


## Markieren und Pruning:

- Route scannen: Markiere Stop, falls Ankunftszeit verbessert
- Nächste Runde: Nur Routen von markierten Stops scannen
- Scanne jede Route ab ihrem ersten markierten Stop
- Markiere Stops nur, wenn sie die beste Ankunftszeit am Ziel verbessern

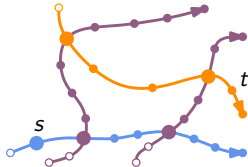
Für jede Runde  $k \leftarrow 1, 2, \dots$

- 1 Wähle erreichte Routen aus letzter Runde
- 2 Scanne diese Routen
- 3 Relaxiere Fußwege



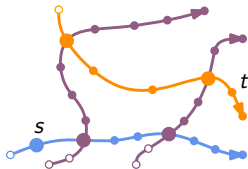
Für jede Runde  $k \leftarrow 1, 2, \dots$

- 1 Wähle erreichte Routen aus letzter Runde
- 2 Scanne diese Routen
- 3 Relaxiere Fußwege



Für jede Runde  $k \leftarrow 1, 2, \dots$

- 1 Wähle erreichte Routen aus letzter Runde
- 2 Scanne diese Routen
- 3 Relaxiere Fußwege

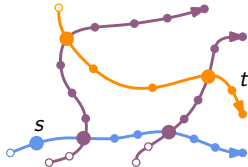




## Relaxiere Fußwege

Für jede Runde  $k \leftarrow 1, 2, \dots$

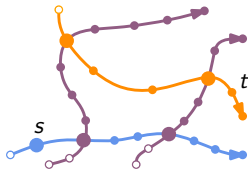
- 1 Wähle erreichte Routen aus letzter Runde
- 2 Scanne diese Routen
- 3 Relaxiere Fußwege



Relaxiere Fußwege

Für jede Runde  $k \leftarrow 1, 2, \dots$

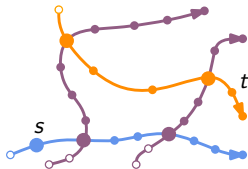
- 1 Wähle erreichte Routen aus letzter Runde
- 2 Scanne diese Routen
- 3 Relaxiere Fußwege



Relaxiere Fußwege

Für jede Runde  $k \leftarrow 1, 2, \dots$

- 1 Wähle erreichte Routen aus letzter Runde
- 2 Scanne diese Routen
- 3 Relaxiere Fußwege



Terminiere, wenn kein Stop markiert wurde.

**Beobachtung:** Routen werden in beliebiger Reihenfolge gescannt.

**Beobachtung:** Routen werden in beliebiger Reihenfolge gescannt.

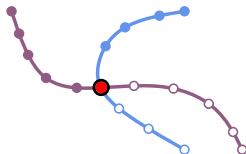
Verteile Routen auf verschiedene CPU-Kerne; scanne parallel.

**Beobachtung:** Routen werden in beliebiger Reihenfolge gescannt.

Verteile Routen auf verschiedene CPU-Kerne; scanne parallel.

## Vermeiden von Race-Conditions:

- Lock auf Schreiben von Labels (teuer)
- Synchronisiere Labels nach jeder Runde
- Sicherstellen, dass nur „unabhängige“ Routen gleichzeitig gescannt werden

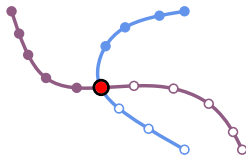


**Beobachtung:** Routen werden in beliebiger Reihenfolge gescannt.

Verteile Routen auf verschiedene CPU-Kerne; scanne parallel.

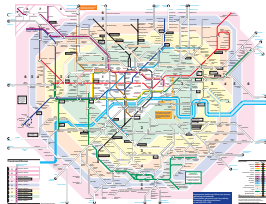
## Vermeiden von Race-Conditions:

- Lock auf Schreiben von Labels (teuer)
- Synchronisiere Labels nach jeder Runde
- Sicherstellen, dass nur „unabhängige“ Routen gleichzeitig gescannt werden (Reduktion auf Färbeproblem)



## Mögliche Erweiterungen:

- Profil-Anfragen (Intervallanfragen)  
Flexible Abfahrtszeiten
- Tarifzonen  
Längere Routen könnten billiger sein.
- Umstiegssicherheit  
Routen könnten knappe Umstiege haben.
- ...

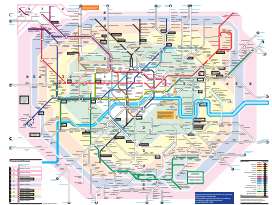


Performance hängt von Anzahl *nichtdominierter* Journeys ab.



## Mögliche Erweiterungen:

- Profil-Anfragen (Intervallanfragen)  
Flexible Abfahrtszeiten
- Tarifzonen  
Längere Routen könnten billiger sein.
- Umstiegssicherheit  
Routen könnten knappe Umstiege haben.
- ...



Performance hängt von Anzahl *nichtdominierter* Journeys ab.

# More Criteria: McRAPTOR

**Ziel:** Erweitern von RAPTOR auf zusätzliche Kriterien



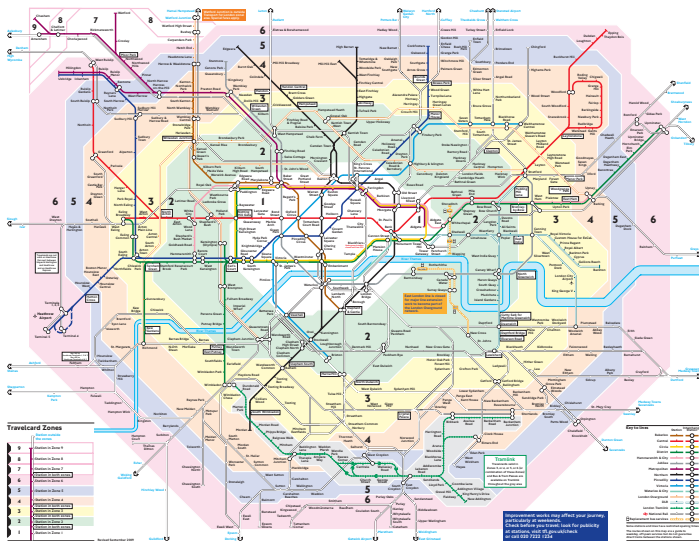
**Ziel:** Erweitern von RAPTOR auf zusätzliche Kriterien



**Ansatz:**

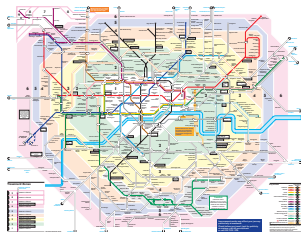
- Labels haben Wert für jedes zusätzliche Kriterium
- Mehrere nichtdominierte Labels pro Stop und Runde
- Mehrere aktive Trips beim Scannen von Routen
- Lösche dominierte Labels on-the-fly

# McRAPTOR Beispiel: Tarifzonen



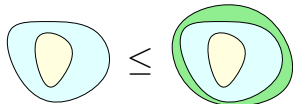
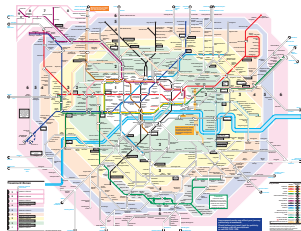
## Tarifzonen einbauen:

- Direkte Preise (£) nicht handhabbar
- ⇒ Berechne alle Kombinationen von Tarifzonen
- ... und filtere im Postprocessing



## Tarifzonen einbauen:

- Direkte Preise (£) nicht handhabbar
- ⇒ Berechne alle Kombinationen von Tarifzonen
- ... und filtere im Postprocessing



## Implementierung:

- Mengen von Tarifzonen als Kriterium
- Dominieren  $\hat{=}$  Teilmengenrelation
- Benutze Bits von `int64` für Mengen

# Profil-Anfragen: rRAPTOR

**Problem:** Finde alle optimalen Journeys, die in einem Zeitintervall  $\Delta$  abfahren.



# Profil-Anfragen: rRAPTOR

**Problem:** Finde alle optimalen Journeys, die in einem Zeitintervall  $\Delta$  abfahren.

- Lösbar mit McRAPTOR...
- ... mit Abfahrtszeit als Kriterium.





**Problem:** Finde alle optimalen Journeys, die in einem Zeitintervall  $\Delta$  abfahren.

- Lösbar mit McRAPTOR...
- ... mit Abfahrtszeit als Kriterium.



**Effizienterer Ansatz: rRAPTOR (Self-Pruning)**

- Sammle alle Abfahrten aus Intervall  $\Delta$  in Menge  $\mathcal{D}$
- Dann: RAPTOR für jedes  $\tau \in \mathcal{D}$  geordnet absteigend nach Zeit
- Reinitialisiere keine Labels zwischen den Aufrufen!

**Problem:** Finde alle optimalen Journeys, die in einem Zeitintervall  $\Delta$  abfahren.

- Lösbar mit McRAPTOR...
- ... mit Abfahrtszeit als Kriterium.



**Effizienterer Ansatz: rRAPTOR (Self-Pruning)**

- Sammle alle Abfahrten aus Intervall  $\Delta$  in Menge  $\mathcal{D}$
- Dann: RAPTOR für jedes  $\tau \in \mathcal{D}$  geordnet absteigend nach Zeit
- Reinitialisiere keine Labels zwischen den Aufrufen!

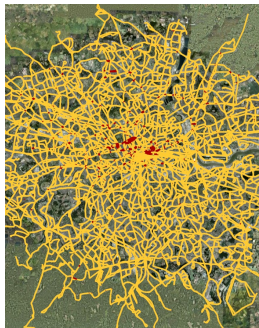
Prunt implizit Routen, die früher abfahren und später ankommen.

# Experimente



## Das vollständige Londoner Netzwerk

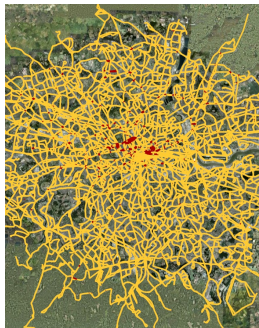
- Ein Dienstag
- Beinhaltet Tube, Bus, DLR, Tram, ...
- 20 843 Stops
- 2 225 Routen mit 133 011 Trips
- 5 132 672 Connections pro Tag



## Das vollständige Londoner Netzwerk

- Ein Dienstag
- Beinhaltet Tube, Bus, DLR, Tram, ...
- 20 843 Stops
- 2 225 Routen mit 133 011 Trips
- 5 132 672 Connections pro Tag

Experimente: 10 000 zufällige  $s-t$ -Anfragen



# Vergleich der Algorithmen

(Hardware: Intel Xeon X5680 mit 3.33 GHz und 96 GiB DDR3-1333 RAM)

Algorithm	Ar	R	Tr	Fz	Rounds	Journeys	[ms]
Dijkstra	●	○	○	○	—	0.9	14.2
RAPTOR	●	○	●	○	8.4	1.9	7.3
LD	●	○	●	○	—	1.9	44.5
MLC	●	○	●	○	—	1.9	67.2
McRAPTOR	●	○	●	●	10.8	9.0	107.4
MLC	●	○	●	●	—	9.0	399.5
McRAPTOR	●	●	●	○	9.5	16.3	259.8
rRAPTOR	●	●	●	○	138.5	16.3	87.0
SPCS	●	●	○	○	—	7.8	183.6

(Ar: Arrival Time, R: Range, Tr: Transfers, Fz: Fare Zones)

Algorithm	Ar	R	Tr	Fz	1 core [ms]	3 cores [ms]	6 cores [ms]	12 cores [ms]
RAPTOR	●	○	●	○	7.7	5.0	4.1	3.7
McRAPTOR	●	○	●	●	118.6	49.4	29.9	26.1
rRAPTOR	●	●	●	○	92.3	39.5	26.8	21.6
SPCS	●	●	○	○	183.6	69.1	44.9	38.9

(Ar: Arrival Time, R: Range, Tr: Transfers, Fz: Fare Zones)

Algorithm	Ar	R	Tr	Fz	1 core [ms]	3 cores [ms]	6 cores [ms]	12 cores [ms]
RAPTOR	●	○	●	○	7.7	5.0	4.1	3.7
McRAPTOR	●	○	●	●	118.6	49.4	29.9	26.1
rRAPTOR	●	●	●	○	92.3	39.5	26.8	21.6
SPCS	●	●	○	○	183.6	69.1	44.9	38.9

(Ar: Arrival Time, R: Range, Tr: Transfers, Fz: Fare Zones)

- Exzellente Speedups auf bis zu 6 Kernen
- RAPTOR immer  $\leq 30$  ms





Daniel Delling, Thomas Pajor, and Renato F. Werneck.

Round-Based Public Transit Routing.

In *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*, pages 130–140. SIAM, 2012.