

# Algorithmen für Routenplanung

21. Vorlesung, Sommersemester 2024

Adrian Feilhauer | 3. Juli 2024



- Die Prüfungstermine dieses Semester sind:
  - Montag 29.7.
  - Montag 26.8.
  - Mittwoch 18.9.
- Anmeldemodalität in den nächsten Tagen auf VL-Website

## Routenplanung in Transportnetzwerken:

- Dijkstras Algorithmus zu langsam in der Praxis
- ⇒ Unmengen an **Beschleunigungstechniken** [BDG<sup>+</sup>16]
- Sehr hohe Beschleunigung auf Straßennetzwerken
  - Moderate Beschleunigung auf Public Transit [BDGM09, BDW11, DDPW15]

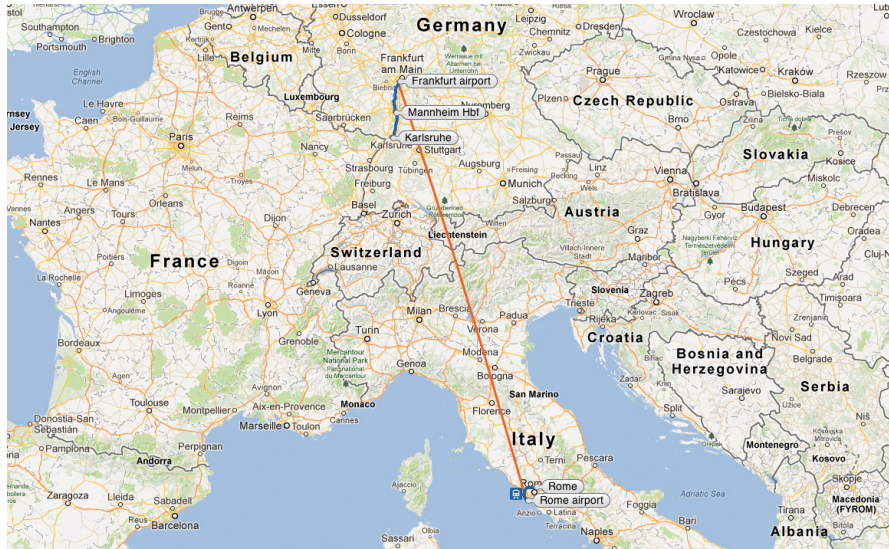
## Aber hauptsächlich unimodale Routenplanung:

- Beschränkt auf ein Transportnetzwerk
- Mit einer einzigen Art Zeitabhängigkeit, wenn überhaupt

## Realistische Szenarien sind multimodal:

- Zwischen beliebigen Orten
- In einem **gemischten Netzwerk** verschiedener Transportmodi

# Multimodale Routenplanung



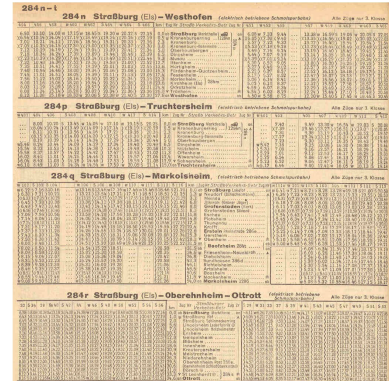
<http://rome2rio.com>

# Modellierung

# Problemstellung

Geg.: Transportnetzwerke, bestehend aus

- Privatauto/Taxi
- Bike-/Car-Sharing
- Laufen
- Fahrplanbasiertem öffentlichem Verkehr
- Flügen
- Verknüpfungen für Modalitätswechsel



The image displays four screenshots of German railway timetables, each for a different route starting from Straßburg (Els). The routes are:

- 284 n - l Straßburg (Els) - Westhofen**: Fehlfahrer betriebene Schienenverkehrsleistungen. Alle 21 Tage nur 3 Wochen.
- 284 p Straßburg (Els) - Truchtersheim**: Fehlfahrer betriebene Schienenverkehrsleistungen. Alle 21 Tage nur 3 Wochen.
- 284 q Straßburg (Els) - Markolsheim**: Fehlfahrer betriebene Schienenverkehrsleistungen. Alle 21 Tage nur 3 Wochen.
- 284 r Straßburg (Els) - Oberrhein - Ottrott**: Fehlfahrer betriebene Schienenverkehrsleistungen. Alle 21 Tage nur 3 Wochen.

Each screenshot shows a detailed grid of departure and arrival times for various train services, including regional and long-distance trains, with columns for dates and times.

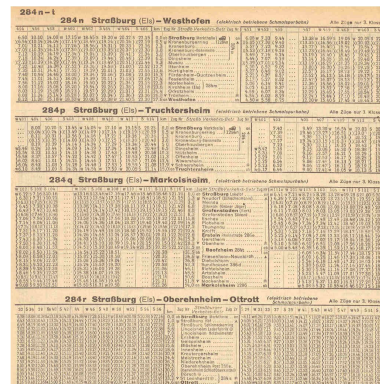
**Geg.:** Transportnetzwerke, bestehend aus

- Privatauto/Taxi
- Bike-/Car-Sharing
- Laufen
- Fahrplanbasiertem öffentlichem Verkehr
- Flügen
- Verknüpfungen für Modalitätswechsel

## Earliest Arrival Query:

Für Positionen  $s, t$  und Abfahrtszeit  $\tau_{\text{dep}}$ , berechne

- Journey nach  $t$ , die bei  $s$  nicht vor  $\tau_{\text{dep}}$  abfährt,
- und bei  $t$  so früh wie möglich ankommt.



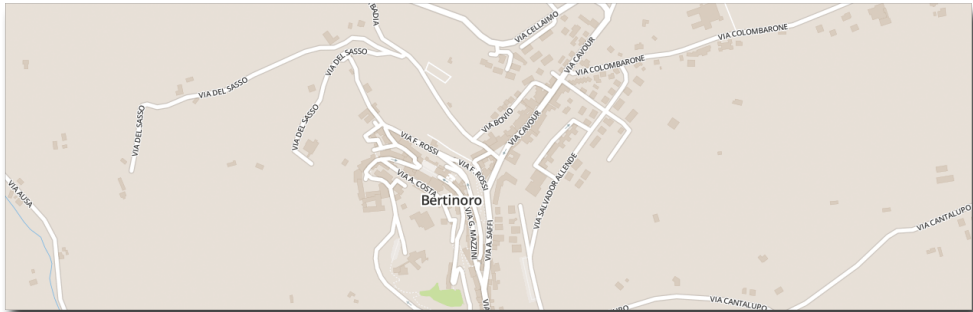
The image displays four screenshots of German railway timetables, each showing a grid of departure and arrival times for different stations. The routes are:

- 284 n-l**: Straßburg (Elz) – Weathofen
- 284 p**: Straßburg (Elz) – Truchtersheim
- 284 q**: Straßburg (Elz) – Markolsheim
- 284 r**: Straßburg (Elz) – Oberrhein – Ottrott

Each screenshot shows a detailed grid of times for various stations along the route, with columns for departure and arrival times and rows for specific stations.

## Einfaches Graphmodell:

- Knoten  $\hat{=}$  Kreuzungen
- (Gerichtete) Kanten  $\hat{=}$  Straßensegmente
- Für Auto, Laufen und Fahrradfahren (mit verschiedenen Kantengewichtsfunktionen)



Auch: Modelle mit Abbiegekosten [DGPW11], Zeitabhängigkeit [DW09]



## Zeitabhängiges Modell [BJ04, PSWZ08]:

- Trips werden in **Routen** eingeteilt
- Für jede Route an einem Stop: Ein **Routenknoten**
- Kantengewichte sind Funktionen



Auch: Zeitexpandiertes Modell [PSWZ08],  
nicht-graphbasierte Ansätze [DPW12a, DPSW13]

## Problem:

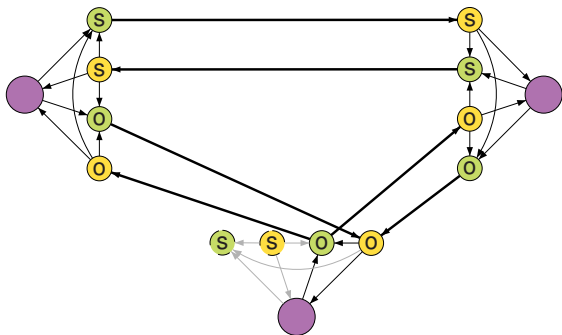
- Bisherige Modelle wurden für Zugverkehr erdacht
- Führen bei Flugnetzwerken zu viel zu großen Graphen

## Extramodell für Flugnetzwerke [DPWZ09]:

- Pro Flughafen: Terminal-Knoten
- Pro Flugallianz: Departure- und Arrival-Knoten

## Modelliert Prozeduren am Flughafen:

- Check-in-Dauer
- Check-out-Dauer
- Transfer-Dauer innerhalb
- ... und zwischen Flugallianzen



## Straßennetze:

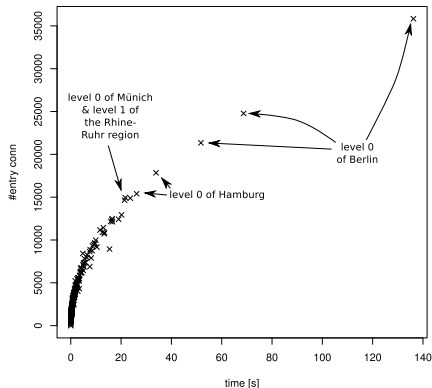
- Stark ausgeprägte Hierarchie (für Reisezeit)
- ↪ (lange) Kürzeste Wege konvergieren zur Autobahn
- Ausgenutzt in vielen Beschleunigungstechniken [ADGW11, ALS13]
- ↪ Punkt-zu-Punkt-Anfragen in Mikrosekunden oder weniger
- Kleine Separatoren ↪ ausgenutzt in CRP [DGRW11]

## Öffentliche Verkehrsnetze:

- Inhärent zeitabhängig (Abfahrten und Ankünfte)
- Eingeschränkte Fußwege für Transfers (vom Betreiber)
- Inhärent multikriteriell (Reisezeit vs. # Transfers)
- Weniger stark ausgeprägte Hierarchie
- Weniger gut separierbar [SW14]

## Öffentliche Verkehrsnetze:

- Inhärent zeitabhängig (Abfahrten und Ankünfte)
- Eingeschränkte Fußwege für Transfers (vom Betreiber)
- Inhärent multikriteriell (Reisezeit vs. # Transfers)
- Weniger stark ausgeprägte Hierarchie
- Weniger gut separierbar [SW14]



Viele Produktivsystem für „multimodale“ Routenplanung benutzen Heuristiken: Google Transit, bahn.de, ...

- Öffentlicher Verkehr (erinnere: vordefinierte Transferkanten!)
- + Eingeschränktes Laufen am Anfang und Ende (15 min Radii)  
(Oft mit Dijkstra berechnet)

Viele Produktivsystem für „multimodale“ Routenplanung benutzen Heuristiken: Google Transit, bahn.de, ...

- Öffentlicher Verkehr (erinnere: vordefinierte Transferkanten!)
- + Eingeschränktes Laufen am Anfang und Ende (15 min Radii)  
(Oft mit Dijkstra berechnet)
- Was ist mit 15:01 min Laufen?

Viele Produktivsystem für „multimodale“ Routenplanung benutzen Heuristiken: Google Transit, bahn.de, ...

- Öffentlicher Verkehr (erinnere: vordefinierte Transferkanten!)
- + Eingeschränktes Laufen am Anfang und Ende (15 min Radii)  
(Oft mit Dijkstra berechnet)
- Was ist mit 15:01 min Laufen?
- Auto und Flüge: 350 km? 351 km? 500 km?



Viele Produktivsystem für „multimodale“ Routenplanung benutzen Heuristiken: Google Transit, bahn.de, ...

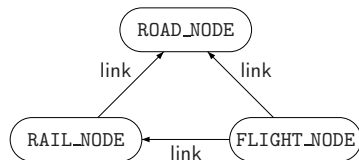
- Öffentlicher Verkehr (erinnere: vordefinierte Transferkanten!)
- + Eingeschränktes Laufen am Anfang und Ende (15 min Radii)  
(Oft mit Dijkstra berechnet)
- Was ist mit 15:01 min Laufen?
- Auto und Flüge: 350 km? 351 km? 500 km?

Wir wollen **beweisbar optimale** Journeys  
mit uneingeschränkter Fußweg- und Autonutzung.

# Kombination der Netzwerke

## Zwei Schritte:

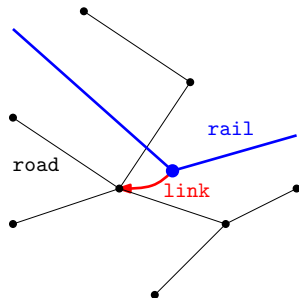
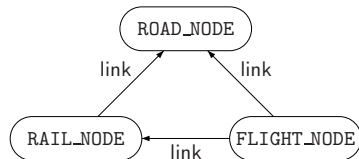
- Füge alle Graphen zusammen
- Verbinde sie durch Link-Kanten



# Kombination der Netzwerke

## Zwei Schritte:

- Füge alle Graphen zusammen
- Verbinde sie durch Link-Kanten

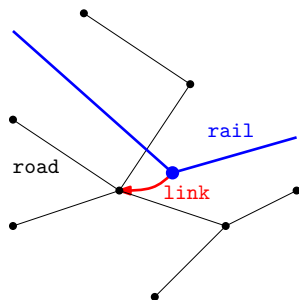
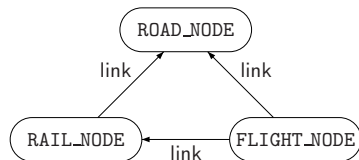


- Verbinde Knoten mit (geometrisch) nächstem Nachbar (oder besser: Indoor-Routing beachten)
- Labelle Kanten bzgl. ihres Transportmodus
- Resultierender Graph: groß, weniger deutliche Hierarchie

# Kombination der Netzwerke

## Zwei Schritte:

- Füge alle Graphen zusammen
- Verbinde sie durch Link-Kanten

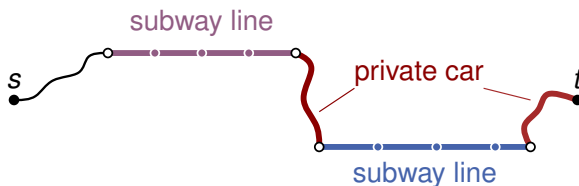


- Verbinde Knoten mit (geometrisch) nächstem Nachbar (oder besser: Indoor-Routing beachten)
- Labelle Kanten bzgl. ihres Transportmodus
- Resultierender Graph: groß, weniger deutliche Hierarchie

Dijkstras Algorithmus berechnet (schnellste) Journeys auf multimodalem Graphen.

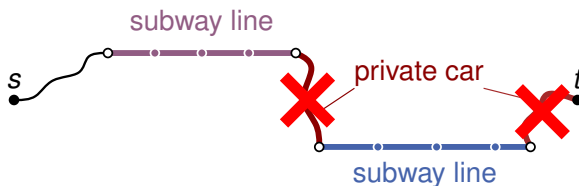
# Unerwünschte Modalitätswechsel

**Problem:** Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen



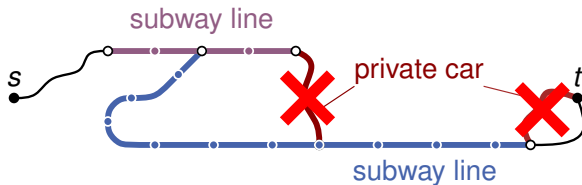
# Unerwünschte Modalitätswechsel

**Problem:** Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen



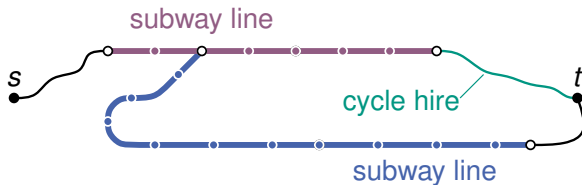
# Unerwünschte Modalitätswechsel

**Problem:** Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen



# Unerwünschte Modalitätswechsel

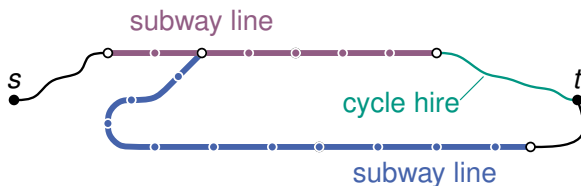
**Problem:** Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen





# Unerwünschte Modalitätswechsel

**Problem:** Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen

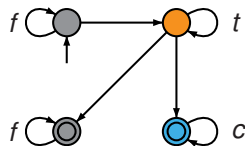


- Nicht alle modalen Folgen sind zulässig
- Verfügbare/gewünschte Modi hängen vom Nutzer ab

# Bekannte Lösung: LCSP

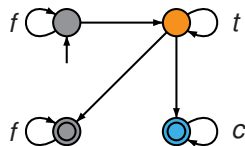
## Label Constrained Shortest Path Problem (LCSP):

- Definiere Alphabet  $\Sigma$  der Modi
- Kanten sind bzgl. Modus gelabelt
- Konkatination der Kantenlabels eines Pfads entspricht Wort  $w \in \Sigma^*$
- Erlaube nur Pfade mit  $w \in L$  für Sprache  $L \subset \Sigma^*$
- Lösbar in Polynomialzeit, wenn  $L$  reguläre Sprache [BJM00]



## Label Constrained Shortest Path Problem (LCSP):

- Definiere Alphabet  $\Sigma$  der Modi
- Kanten sind bzgl. Modus gelabelt
- Konkatination der Kantenlabels eines Pfads entspricht Wort  $w \in \Sigma^*$
- Erlaube nur Pfade mit  $w \in L$  für Sprache  $L \subset \Sigma^*$
- Lösbar in Polynomialzeit, wenn  $L$  reguläre Sprache [BJM00]



## Dijkstras Algorithmus für LCSP: [BJM00]

- Benutze Produktgraph  $G \times \mathcal{A}(L)$  mit endlichem Automaten  $\mathcal{A}(L)$
- Oder: Erlaube mehrere Distanzlabel pro Knoten (1 pro Zustand)
- Eher langsam ( $\approx 15$  s auf Europa-Instanz)

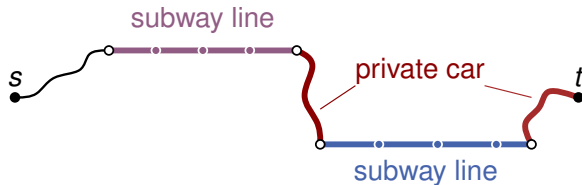
## Feste Constraints:

- Adaption von  $A^*$  und bidirektionaler Suche [BBH<sup>+</sup>09]
- Access-Node Routing (ANR) [DPW09]
- State-Dependent ALT (SDALT) [KLPC11]

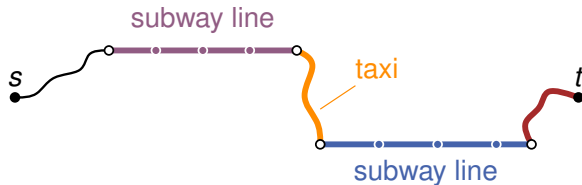
## Benutzerdefinierte Constraints:

- User-Constrained Contraction Hierarchies (UCCH) [DPW12b]

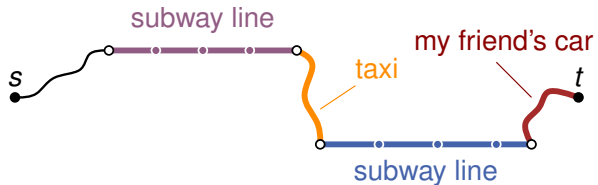
# Benutzerdefinierte Constraints

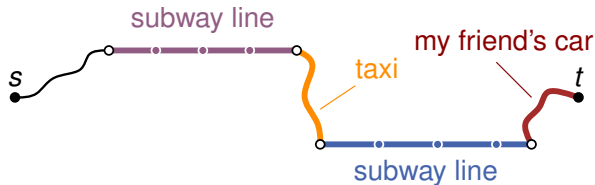


# Benutzerdefinierte Constraints



# Benutzerdefinierte Constraints





- Modale Restriktionen hängen vom **Benutzer** ab
- Automat als Eingabe zur **Anfrage**
- Beschleunigungstechnik benötigt **flexible Vorbereitung**



# Mode Sequence Constraints

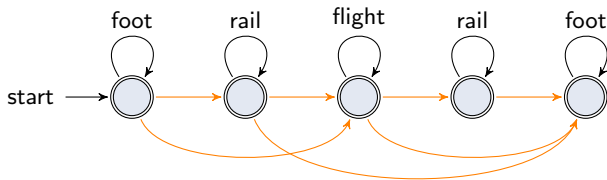
## Beobachtung:

Sinnvolle Constraints enthalten üblicherweise keine Einschränkungen innerhalb eines Teilnetzwerkes  
(nicht: „Benutze nur drei Straßenkanten“)

# Mode Sequence Constraints

## Beobachtung:

Sinnvolle Constraints enthalten üblicherweise keine Einschränkungen innerhalb eines Teilnetzwerkes  
(nicht: „Benutze nur drei Straßenkanten“)

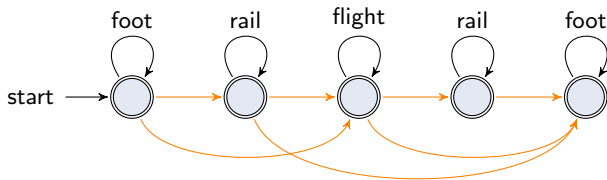


Idee: Betrachte **Mode Sequence (MS) Constraints**.

# Mode Sequence Constraints

## Beobachtung:

Sinnvolle Constraints enthalten üblicherweise keine Einschränkungen innerhalb eines Teilnetzwerkes (nicht: „Benutze nur drei Straßenkanten“)



Idee: Betrachte **Mode Sequence (MS) Constraints**.

## Ziel:

- Eine **gemeinsame** Vorberechnung für **jegliche MS**
- Mäßiger Platzverbrauch und schnelle Anfragen

# User-Constrained CH (UCCH) [DPW12b]

## Idee:

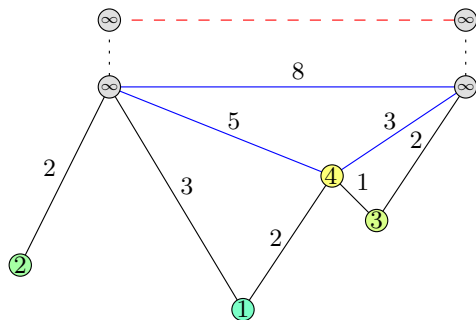
- Basierend auf Contraction Hierarchies [GSSV12]

## Idee:

- Basierend auf Contraction Hierarchies [GSSV12]
- Kontrahiere keine Knoten mit inzidenten Link-Kanten [DPW09]

## Vorbereitung:

- Rank von gelinkten Knoten ist  $\infty$
  - Shortcuts übergreifen keine Netzwerke
  - Zeugensuche beschränkt auf Teilnetz
- ⇒ Ein gemeinsamer Core,  
unabh. Teilkomponenten

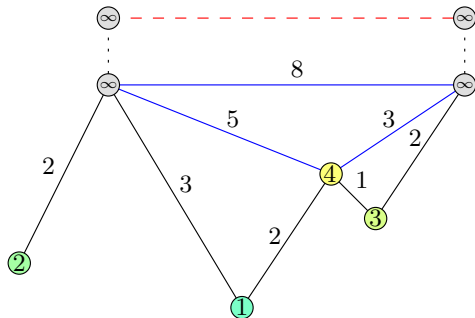


## Idee:

- Basierend auf Contraction Hierarchies [GSSV12]
- Kontrahiere keine Knoten mit inzidenten Link-Kanten [DPW09]

## Vorbereitung:

- Rank von gelinkten Knoten ist  $\infty$
  - Shortcuts übergreifen keine Netzwerke
  - Zeugensuche beschränkt auf Teilnetz
- ⇒ Ein gemeinsamer Core,  
unabh. Teilkomponenten



Vorbereitung unabhängig von Mode Sequence Constraints

## Zwei nahtlose Schritte:

1. Auf Straßenkomponente: CH-Anfrage [[GSSV12](#)]
2. Auf gemeinsamen Core: Multi-Source-Multi-Target LCSPD-Dijkstra [[BJM00](#)]

## Zwei nahtlose Schritte:

1. Auf Straßenkomponente: CH-Anfrage [\[GSSV12\]](#)
2. Auf gemeinsamen Core: Multi-Source-Multi-Target LCSPD-Dijkstra [\[BJM00\]](#)

## Verbesserungen:

- Knoten umsortieren: Core-Knoten nach vorne
- Knotengrad wegen unkontrahierter Core-Knoten recht hoch  
⇒ Nur partielle Kontraktion
- Keine Rückwärtssuche im Core



# Experimente

## Netzwerke:

Deutschland Straße (5 M Knoten, 12 M Kanten)

Deutschland Züge (6.8 k Stops, 0.5 M Connections)

	Preprocessing				Query (road only)			
	avg core degree	core nodes	shortcut edges	time [min]	settled nodes	relaxed edges	touched edges	time [ms]
UCCH	10	30 908	42.3 %	6	15 531	27 506	155 776	5.85
	15	16 003	43.1 %	7	8 090	16 844	121 631	3.11
	20	12 239	43.7 %	9	6 240	14 425	124 201	2.82
	25	10 635	44.2 %	10	5 465	13 687	135 151	2.80
	30	9 742	44.7 %	12	5 049	13 486	148 735	2.96
	35	9 171	45.1 %	14	4 794	13 598	163 376	3.15
	40	8 788	45.4 %	15	4 628	13 787	179 483	3.38
Part. CH	13	10 635	41.7 %	6	5 567	11 402	71 860	1.93
Part. CH	15	6 750	41.8 %	7	3 636	7 970	53 655	1.37
CH	—	0	41.8 %	9	677	1 290	11 434	0.25

## Netzwerke:

Europa & Nordamerika Straße (50 M Knoten, 125 M Kanten)

Europa Züge (31 k Stops, 1.6 M Connections)

Star Alliance Flüge (1 172 Stops, 28 k Connections)

		Preprocessing		Queries	
		Time [h:m]	Space [MiB]	Time [ms]	Speedup
flight	Dijkstra	—	—	33 862.00	1
	ANR	3:04	14 050	1.07	31 551
	UCCH	1:18	542	0.67	50 540
rail & flight	Dijkstra	—	—	35 261.00	1
	UCCH	1:27	558	70.52	500

Intel Xeon E5430, 2.66 GHz, 32 GiB RAM, 12 MiB L2 cache

## Bisher:

- Schnelle Berechnung **zulässiger** multimodaler Journeys
- Flexible Vorberechnung – auf statischen Straßennetzwerken
- Public Transit nur im Core  $\Rightarrow$  Dynamische Fahrpläne möglich

## Aber:

- Es ist nur *ein* kürzester Weg
- Gibt nicht die Fülle aller verfügbaren Optionen wieder
- Aktueller Straßenverkehr? Zeitabhängigkeit?
- Weniger ausgeprägte Hierarchie (im Vergleich zu unimodalen Straßennetzwerken)

## Bisher:

- Schnelle Berechnung **zulässiger** multimodaler Journeys
- Flexible Vorberechnung – auf statischen Straßennetzwerken
- Public Transit nur im Core  $\Rightarrow$  Dynamische Fahrpläne möglich

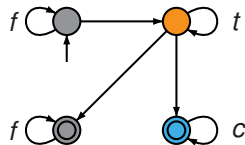
## Aber:

- Es ist nur *ein* kürzester Weg
- Gibt nicht die Fülle aller verfügbaren Optionen wieder
- Aktueller Straßenverkehr? Zeitabhängigkeit?
- Weniger ausgeprägte Hierarchie (im Vergleich zu unimodalen Straßennetzwerken)

# Problem gelöst?

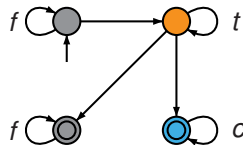
## Label Constrained Shortest Path Problem (LCSP):

- Definiere Alphabet über Transportmodi
- Graph mit Kantenlabels
- Gültiger Pfad muss Wort aus einer gegebenen regulären Sprache entsprechen



## Label Constrained Shortest Path Problem (LCSP):

- Definiere Alphabet über Transportmodi
- Graph mit Kantenlabels
- Gültiger Pfad muss Wort aus einer gegebenen regulären Sprache entsprechen

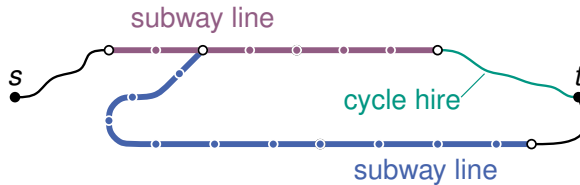


## Algorithmen für LCSP:

- Dijkstra auf Produktgraph mit endlichem Automaten
- Beschleunigungstechniken: ANR, SDALT
- Constraints als Query-Eingabe: UCCH

# Bisherige Ansätze

## Nachteile von LCSP:





# Bisherige Ansätze

Nachteile von LCSP:

$s$   
•

?

$t$   
•

## Nachteile von LCSP:



- Constraints müssen im Voraus spezifiziert werden
- Sind dem Nutzer ggf. nicht bewusst
- Nur eine Journey wird berechnet (keine Alternativen)

## Nachteile von LCSP:



- Constraints müssen im Voraus spezifiziert werden
- Sind dem Nutzer ggf. nicht bewusst
- Nur eine Journey wird berechnet (keine Alternativen)

**Ziel:** Berechne **sinnvolle Menge** multimodaler Journeys.

# Multikriterielle multimodale Journeys

**Ziel:** Berechne **multikriterielle** multimodale Pareto-Mengen.

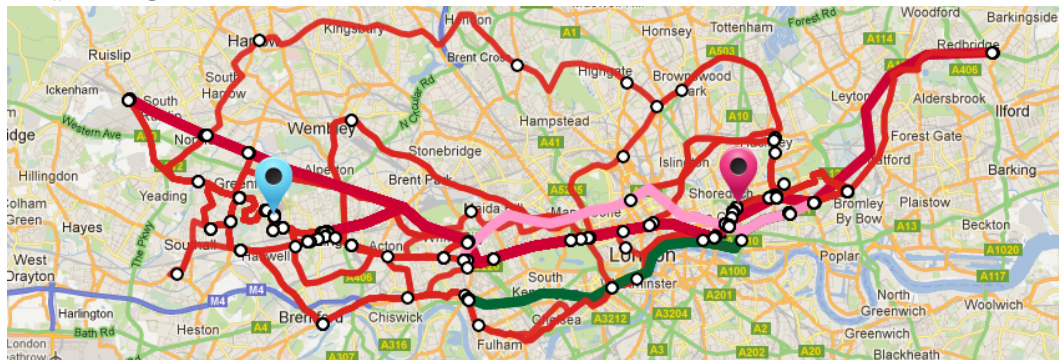
- Optimierte Ankunftszeit
- Vom Modus abhängige „Komfort“-Kriterien  
z.B. # Umstiege, Laufdauer, Taxikosten, etc.

# Multikriterielle multimodale Journeys

**Ziel:** Berechne **multikriterielle** multimodale Pareto-Mengen.

- Optimierte Ankunftszeit
- Vom Modus abhängige „Komfort“-Kriterien

z.B. # Umstiege, Laufdauer, Taxikosten, etc.



Criteria: Arrival time, # transfers, walking duration. Sixty-nine solutions.

**Bekanntes Problem:** Pareto-Menge stark wachsend in # Kriterien

## Definition (Dominanz, Pareto-Menge)

Journey  $J_1$  **dominiert** Journey  $J_2 \iff J_2$  in keinem Kriterium besser

**Pareto-Menge:** inklusionsmaximale Menge nichtdominierter Journeys

## Definition (Dominanz, Pareto-Menge)

Journey  $J_1$  **dominiert** Journey  $J_2 \iff J_2$  in keinem Kriterium besser

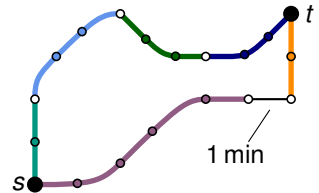
**Pareto-Menge:** inklusionsmaximale Menge nichtdominierter Journeys

### Irrelevante Lösungen:

Eine nichtdominierte Journey kann

- minimale Verbesserung für Kriterium  $A$  liefern,
- ist aber viel schlechter in Kriterium  $B$ .

Viele Kriterien  $\Rightarrow$  Kombinatorische Explosion



## Definition (Dominanz, Pareto-Menge)

Journey  $J_1$  **dominiert** Journey  $J_2 \iff J_2$  in keinem Kriterium besser

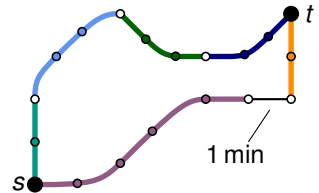
**Pareto-Menge:** inklusionsmaximale Menge nichtdominierter Journeys

### Irrelevante Lösungen:

Eine nichtdominierte Journey kann

- minimale Verbesserung für Kriterium  $A$  liefern,
- ist aber viel schlechter in Kriterium  $B$ .

Viele Kriterien  $\Rightarrow$  Kombinatorische Explosion



Wie identifiziert man **signifikante** Lösungen einer Pareto-Menge?



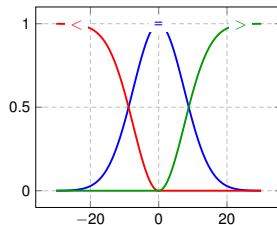
**Beobachtung:** Benutzer haben **unscharfe Vorstellung** bzgl. einiger Kriterien.

- $\pm 1$  Minute Laufen ist „ungefähr gleichwertig“,
- aber  $\pm 30$  Minuten machen einen Unterschied.

**Beobachtung:** Benutzer haben **unscharfe Vorstellung** bzgl. einiger Kriterien.

- $\pm 1$  Minute Laufen ist „ungefähr gleichwertig“,
- aber  $\pm 30$  Minuten machen einen Unterschied.

**Idee:** Relaxiere Dominanzbegriff mit **unscharfer Mengenlehre** (engl. fuzzy set theory).



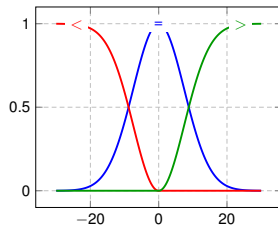
**Beobachtung:** Benutzer haben **unscharfe Vorstellung** bzgl. einiger Kriterien.

- $\pm 1$  Minute Laufen ist „ungefähr gleichwertig“,
- aber  $\pm 30$  Minuten machen einen Unterschied.

**Idee:** Relaxiere Dominanzbegriff mit **unscharfer Mengenlehre** (engl. fuzzy set theory).

- Fuzzy-Operatoren  $<$ ,  $>$  und  $=$
- Unterschiedliche Parametrisierung pro Kriterium

$\Rightarrow$  Grad der Dominanz  $d(J_1, J_2) \in [0, 1]$ .  
„Wie stark wird  $J_2$  von  $J_1$  dominiert?“



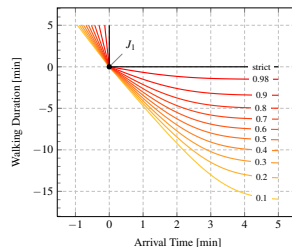
**Beobachtung:** Benutzer haben **unscharfe Vorstellung** bzgl. einiger Kriterien.

- $\pm 1$  Minute Laufen ist „ungefähr gleichwertig“,
- aber  $\pm 30$  Minuten machen einen Unterschied.

**Idee:** Relaxiere Dominanzbegriff mit **unscharfer Mengenlehre** (engl. fuzzy set theory).

- Fuzzy-Operatoren  $<$ ,  $>$  und  $=$
- Unterschiedliche Parametrisierung pro Kriterium

$\Rightarrow$  Grad der Dominanz  $d(J_1, J_2) \in [0, 1]$ .  
„Wie stark wird  $J_2$  von  $J_1$  dominiert?“



Journey  $J_1$  kann  $J_2$  z.B. 90 %-dominieren, auch wenn  $J_2$  1 min weniger Laufen bedeutet.

# Signifikante Journeys identifizieren

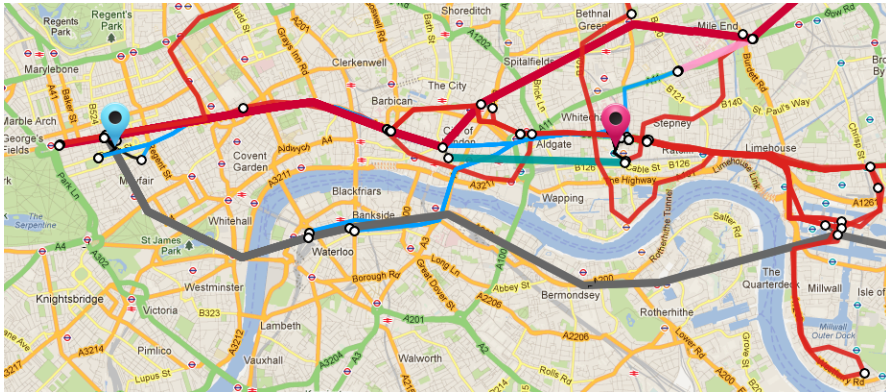
## Ansatz:

- Berechne volle (exakte) Pareto-Menge  $\{J_1, \dots, J_n\}$  mit multikriteriellem Algorithmus
- Bewerte jede Journey  $J$  mit  $1 - \max(d(J_1, J), \dots, d(J_n, J))$
- Dann gilt: Journeys mit höherer Bewertung sind wichtiger

# Signifikante Journeys identifizieren

## Ansatz:

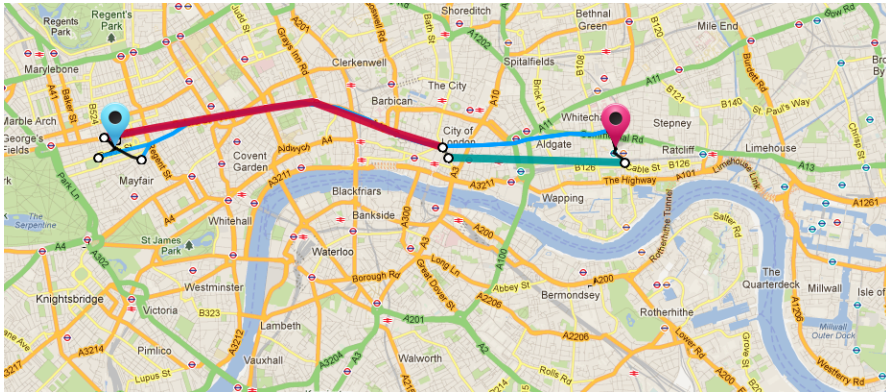
- Berechne volle (exakte) Pareto-Menge  $\{J_1, \dots, J_n\}$  mit multikriteriellem Algorithmus
- Bewerte jede Journey  $J$  mit  $1 - \max(d(J_1, J), \dots, d(J_n, J))$
- Dann gilt: Journeys mit höherer Bewertung sind wichtiger



# Signifikante Journeys identifizieren

## Ansatz:

- Berechne volle (exakte) Pareto-Menge  $\{J_1, \dots, J_n\}$  mit multikriteriellem Algorithmus
- Bewerte jede Journey  $J$  mit  $1 - \max(d(J_1, J), \dots, d(J_n, J))$
- Dann gilt: Journeys mit höherer Bewertung sind wichtiger



Three highest-scored journeys.

## Beobachtungen

- Modale Transfers nur an bestimmten Orten möglich
- Ein Trip immer mit einem Transportmodus



## Beobachtungen

- Modale Transfers nur an bestimmten Orten möglich
- Ein Trip immer mit einem Transportmodus

## Multimodal Multicriteria RAPTOR (MCR):

- Eine Runde pro Trip
- Verwalte Pareto-Mengen von Labels
  - an jedem Knoten
  - für jede Runde (Modus, Umstieg)
- In jeder Runde: Führe Subalgorithmus für jeden Transportmodus aus.
  - Public Transit: McRAPTOR
  - Laufen, Fahrrad, Taxi: MC-Dijkstra & UCCH
- Lies Labels aus Runde  $i - 1$  ein, schreibe nach Runde  $i$

# Heuristische Ansätze

**Problem:** Queries zu langsam (mehrere Sekunden)

**Problem:** Queries zu langsam (mehrere Sekunden)

Viele unwichtige Journeys  $\Rightarrow$  am besten gar nicht erst berechnen

**Problem:** Queries zu langsam (mehrere Sekunden)

Viele unwichtige Journeys  $\Rightarrow$  am besten gar nicht erst berechnen

**Relaxiere Dominanz:**

- **MCR-hf:** Fuzzy-Dominanz zwischen Labels während Algorithmus
- **MCR-hb:** Strikte Dominanz mit diskretisierten Kriterien („Buckets“)

**Problem:** Queries zu langsam (mehrere Sekunden)

Viele unwichtige Journeys  $\Rightarrow$  am besten gar nicht erst berechnen

## Relaxiere Dominanz:

- **MCR-hf:** Fuzzy-Dominanz zwischen Labels während Algorithmus
- **MCR-hb:** Strikte Dominanz mit diskretisierten Kriterien („Buckets“)

## Laufen einschränken:

- **MCR-tx-ry:** Max.  $x$  Minuten Laufen zwischen Trips und  $y$  Minuten an Start/Ziel

**Problem:** Queries zu langsam (mehrere Sekunden)

Viele unwichtige Journeys  $\Rightarrow$  am besten gar nicht erst berechnen

## Relaxiere Dominanz:

- **MCR-hf:** Fuzzy-Dominanz zwischen Labels während Algorithmus
- **MCR-hb:** Strikte Dominanz mit diskretisierten Kriterien („Buckets“)

## Laufen einschränken:

- **MCR-tx-ry:** Max.  $x$  Minuten Laufen zwischen Trips und  $y$  Minuten an Start/Ziel

## Reduziere Anzahl Kriterien:

- **MR-x:** Jeweils  $x$  Minuten Laufen zählen als Trip

**Motivation:** Service benutzt Algorithmus  $\mathcal{A}$ , präsentiert dem Nutzer Top- $k$ -Journeys aus  $P_{\mathcal{A}}$ .

**Motivation:** Service benutzt Algorithmus  $\mathcal{A}$ , präsentiert dem Nutzer Top- $k$ -Journeys aus  $P_{\mathcal{A}}$ .

## Ähnlichkeit von Journeys:

- Benutze Fuzzy  $=$ -Operator für jedes Kriterium
- (Gesamt-) Ähnlichkeit: minimale Ähnlichkeit aller Kriterien



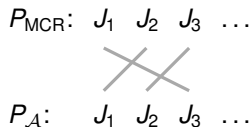
**Motivation:** Service benutzt Algorithmus  $\mathcal{A}$ , präsentiert dem Nutzer Top- $k$ -Journeys aus  $P_{\mathcal{A}}$ .

## Ähnlichkeit von Journeys:

- Benutze Fuzzy  $=$ -Operator für jedes Kriterium
- (Gesamt-) Ähnlichkeit: minimale Ähnlichkeit aller Kriterien

## Qualität der Lösungen:

- Betrachte  $P_{\text{MCR}}$  als „*Ground Truth*“
- Behalte nur Top- $k$ -Journeys in  $P_{\mathcal{A}}, P_{\text{MCR}}$
- Berechne maximales Matching zwischen  $P_{\mathcal{A}}$  und  $P_{\text{MCR}}$  bzgl. Ähnlichkeit



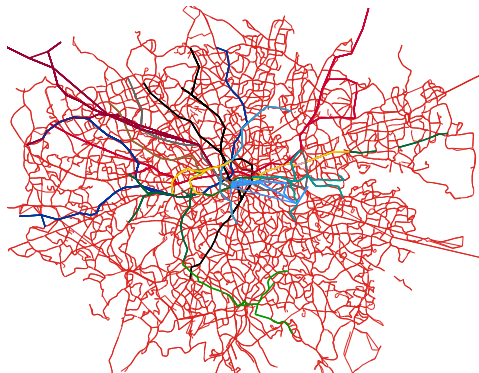
Ausgabe: Durchschnittliche Ähnlichkeit der gematchten Journeys

## Komplettes öffentliches Nahverkehrsnetz:

- inkl. Tube, Bus, DLR, Tram, ...
- 20 k Stops, 2.2 k Routen
- über 5 M Connections pro Tag
- 564 Fahrradleihstationen

## Straßen- und Fußgängernetzwerk:

- jeweils  $\approx 260$  k Knoten  
(27 k unkontrahiert)
- und 1.4 M Kanten



**Kriterien:** Ankunftszeit, # Transfers, Laufdauer

Algorithm	# Rnd.	# Jn.	Time [ms]	Quality-6	
				Avg.	Sd.
MCR	13.8	29.1	1 438.7	100 %	0 %
MCR-hf	15.6	10.9	699.4	89 %	11 %
MCR-hb	10.2	9.0	456.7	91 %	10 %
MCR-t10-r15	10.7	13.2	885.0	30 %	31 %
MR-10	20.0	4.3	39.4	45 %	29 %

One core of Intel Xeon E5-2670, 2.6 GHz, 64 GiB DDR3-1600 RAM






**Kriterien:** Ankunftszeit, # Transfers, Laufdauer, Kosten





Algorithm	Wlk.	# Rnd.	# Jn.	Time [ms]	Quality-6 Avg.	Sd.
MCR	●	16.3	1 666.0	1 960 234.0	100 %	0 %
MCR-hf	●	17.1	35.2	6 451.6	92 %	6 %
MCR-hb	●	9.9	27.6	2 807.7	92 %	6 %
MCR-hb	○	9.0	11.6	996.4	74 %	12 %

One core of Intel Xeon E5-2670, 2.6 GHz, 64 GiB DDR3-1600 RAM

Wlk. ○: Kombiniert Laufdauer mit Kosten in ein Kriterium

- Ansatz zu multikriterieller multimodaler Routenplanung
- Optimierte Ankunftszeit sowie Komfortkriterien
- Unscharfe Mengenlehre hilft, wichtige Lösungen zu identifizieren
- Heuristiken finden schnell Lösungen mit guter Qualität
- ... sofern Komfortkriterien nicht verworfen werden

-  Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck.  
A Hub-Based Labeling Algorithm for Shortest Paths on Road Networks.  
In Pardalos and Rebennack [PR11], pages 230–241.
-  *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*. SIAM, 2012.
-  Julian Arz, Dennis Luxen, and Peter Sanders.  
Transit Node Routing Reconsidered.  
In SEA'13 [SEA13], pages 55–66.
-  *Proceedings of the 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'09)*, OpenAccess Series in Informatics (OASISs), 2009.
-  Chris Barrett, Keith Bisset, Martin Holzer, Goran Konjevod, Madhav V. Marathe, and Dorothea Wagner.  
Engineering Label-Constrained Shortest-Path Algorithms.  
In Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson, editors, *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, volume 74 of *DIMACS Book*, pages 309–319. American Mathematical Society, 2009.

-  Hannah Bast, Daniel Delling, Andrew V. Goldberg, Matthias Müller–Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck.  
Route Planning in Transportation Networks.  
Technical Report abs/1504.05140, ArXiv e-prints, 2016.
-  Annabell Berger, Daniel Delling, Andreas Gebhardt, and Matthias Müller–Hannemann.  
Accelerating Time-Dependent Multi-Criteria Timetable Information is Harder Than Expected.  
In ATMOS'09 [ATM09].
-  Reinhard Bauer, Daniel Delling, and Dorothea Wagner.  
Experimental Study on Speed-Up Techniques for Timetable Information Systems.  
*Networks*, 57(1):38–52, January 2011.
-  Gerth Brodal and Riko Jacob.  
Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries.  
In *Proceedings of the 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'03)*, volume 92 of *Electronic Notes in Theoretical Computer Science*, pages 3–15, 2004.



Chris Barrett, Riko Jacob, and Madhav V. Marathe.  
Formal-Language-Constrained Path Problems.  
*SIAM Journal on Computing*, 30(3):809–837, 2000.



Daniel Delling, Julian Dibbelt, Thomas Pajor, Dorothea Wagner, and Renato F. Werneck.  
Computing Multimodal Journeys in Practice.  
In SEA'13 [SEA13], pages 260–271.




Daniel Delling, Julian Dibbelt, Thomas Pajor, and Renato F. Werneck.  
Public Transit Labeling.  
In *Proceedings of the 14th International Symposium on Experimental Algorithms (SEA'15)*, Lecture Notes in Computer Science, pages 273–285. Springer, 2015.




Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck.  
Customizable Route Planning.  
In Pardalos and Rebennack [PR11], pages 376–387.




 Daniel Delling, Andrew V. Goldberg, Ilya Razenshteyn, and Renato F. Werneck.  
Graph Partitioning with Natural Cuts.


In *25th International Parallel and Distributed Processing Symposium (IPDPS'11)*, pages 1135–1146. IEEE Computer Society, 2011.

 Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner.  
Intriguingly Simple and Fast Transit Routing.

In SEA'13 [SEA13], pages 43–54.

 Daniel Delling, Thomas Pajor, and Dorothea Wagner.  
Accelerating Multi-Modal Route Planning by Access-Nodes.

In Amos Fiat and Peter Sanders, editors, *Proceedings of the 17th Annual European Symposium on Algorithms (ESA'09)*, volume 5757 of *Lecture Notes in Computer Science*, pages 587–598. Springer, September 2009.

 Daniel Delling, Thomas Pajor, and Renato F. Werneck.  
Round-Based Public Transit Routing.

In ALENEX'12 [ALE12], pages 130–140.



Julian Dibbelt, Thomas Pajor, and Dorothea Wagner.  
User-Constrained Multi-Modal Route Planning.  
In ALENEX'12 [ALE12], pages 118–129.







Daniel Delling, Thomas Pajor, Dorothea Wagner, and Christos Zaroliagis.  
Efficient Route Planning in Flight Networks.  
In ATMOS'09 [ATM09].



Daniel Delling and Dorothea Wagner.  
Time-Dependent Route Planning.  
In Ravindra K. Ahuja, Rolf H. Möhring, and Christos Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 207–230. Springer, 2009.



Marco Farina and Paolo Amato.  
A Fuzzy Definition of “Optimality” for Many-Criteria Optimization Problems.  
*IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 34(3):315–326, 2004.

-  Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter.  
Exact Routing in Large Road Networks Using Contraction Hierarchies.  
*Transportation Science*, 46(3):388–404, August 2012.
-  Dominik Kirchler, Leo Liberti, Thomas Pajor, and Roberto Wolfler Calvo.  
UniALT for Regular Language Constraint Shortest Paths on a Multi-Modal Transportation Network.  
In *Proceedings of the 11th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'11)*, volume 20 of *OpenAccess Series in Informatics (OASICs)*, pages 64–75, 2011.
-  Panos M. Pardalos and Steffen Rebennack, editors.  
*Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, volume 6630 of *Lecture Notes in Computer Science*. Springer, 2011.
-  Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis.  
Efficient Models for Timetable Information in Public Transportation Systems.  
*ACM Journal of Experimental Algorithmics*, 12(2.4):1–39, 2008.



*Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, volume 7933 of *Lecture Notes in Computer Science*. Springer, 2013.



Ben Strasser and Dorothea Wagner.

Connection Scan Accelerated.

In Catherine C. McGeoch and Ulrich Meyer, editors, *Proceedings of the 16th Meeting on Algorithm Engineering and Experiments (ALENEX'14)*, pages 125–137. SIAM, 2014.