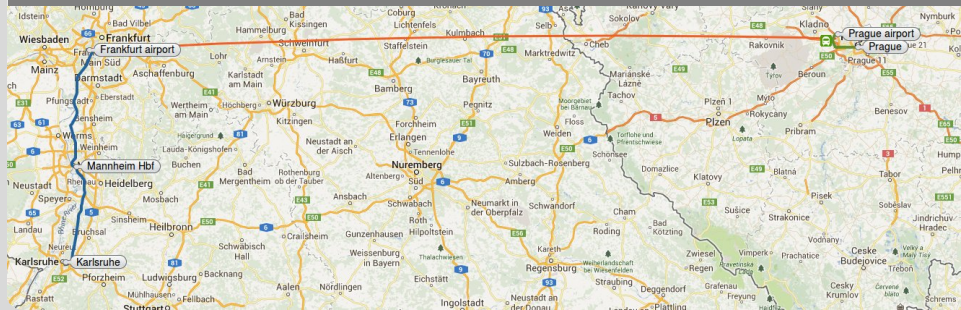


Algorithmen für Routenplanung

22. Vorlesung, Sommersemester 2021

Adrian Feilhauer | 30. Juni 2021

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



Routenplanung in Transportnetzwerken:

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

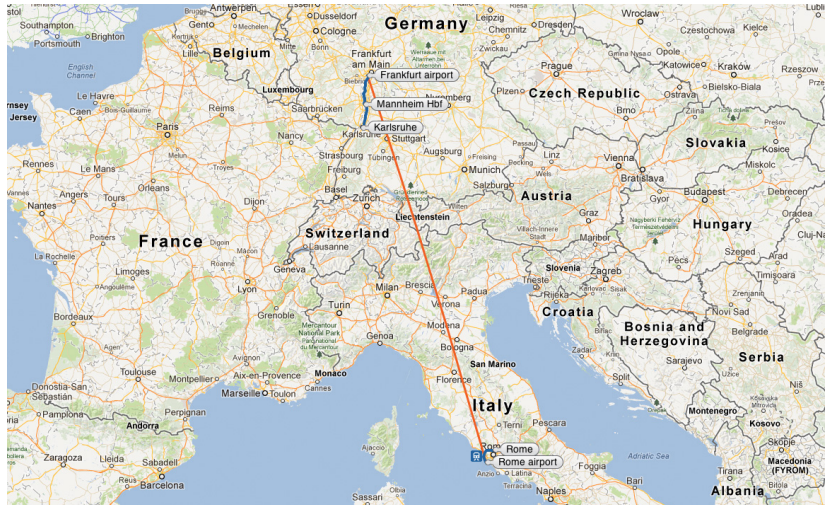
Aber hauptsächlich unimodale Routenplanung:

- 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://some2rio.com>

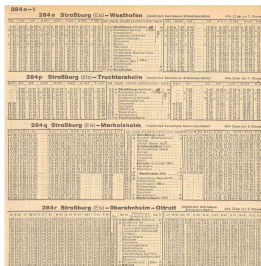
Modellierung



Problemstellung

Geg.: Transportnetzwerke, bestehend aus

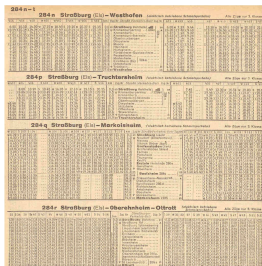
- Privatauto/Taxi
- Bike-/Car-Sharing
- Laufen
- Fahrplanbasiertem öffentlichem Verkehr
- Flügen
- Verknüpfungen für Modalitätswechsel



The image displays four stacked train timetables from the German railway system (DB) for the region around Ströburg. The tables are titled: 284 n Ströburg (Lk) - Westhofen, 284 p Ströburg (Lk) - Truchlarsheim, 284 q Ströburg (Lk) - Märktelshelm, and 284 r Ströburg (Lk) - Obereichenheim - Ottrill. Each table shows departure and arrival times for various destinations and train types.

Geg.: Transportnetzwerke, bestehend aus

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



The image shows four stacked train timetables from the Karlsruhe region, each with a yellow header and a grid of departure and arrival times. The titles of the timetables are: 284 n Stroßburg (St.) - Weiskofen, 284 p Stroßburg (St.) - Truchlarsheim, 284 q Stroßburg (St.) - Märktelshelm, and 284 r Stroßburg (St.) - Obereichenheim - Ottrill. Each timetable includes columns for station names, departure times, and arrival times.

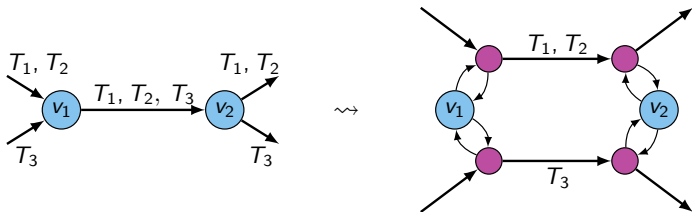
Earliest Arrival Query:

Für Positionen s, t und Abfahrtszeit τ_{dep} , berechne

- Journey(s) nach t , die bei s nicht vor τ_{dep} abfahren,
- und bei t so früh wie möglich ankommen.

Realistic time-dependent model [BJ04, PSWZ08]:

- Öffentlicher Verkehr
- Trips werden in **Routen** eingeteilt
- Für jede Route an einem Stop: Ein **Routenknoten**



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

Problem:

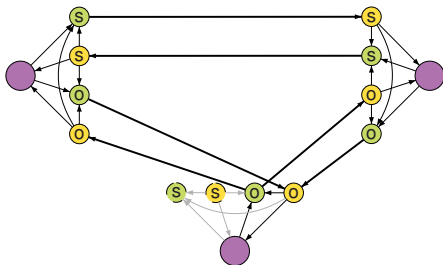
- Bisherige Modelle wurden für Zugverkehr erdacht
- Führen bei Flugnetzwerken zu viel zu großen Graphen

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



Straßennetzwerke:

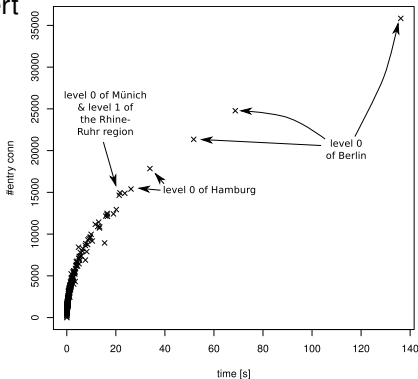
- Hierarchisch gut-strukturiert (für Reisezeit)
- ↪ (lange) Kürzeste Wege konvergieren zur Autobahn
- Kleine Highway Dimension [AFGW10, ADF⁺11]
- Ausgenutzt in vielen Beschleunigungstechniken [ADGW11, ALS13]
- ↪ Punkt-zu-Punkt-Anfragen in Mikrosekunden oder weniger
- Kleine Separatoren ↪ ausgenutzt in CRP [DGRW11]

Öffentliche Verkehrsnetze:

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

Öffentliche Verkehrsnetze:

- Inhärent zeitabhängig (Abfahrten und Ankünfte)
- Eingeschränkte Fußwege für Transfers (vom Betreiber)
- Inhärent multikriteriell (Reisezeit vs. # Transfers)
- Weniger hierarchisch gut-strukturiert
- 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, ...

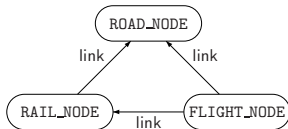
- Öffentlicher Verkehr (erinnere: vordefinierte Transferkanten!)
- + Eingeschränktes Laufen am Anfang und Ende (15 min Radii)
(Oft mit Dijkstra berechnet)
- Was ist mit 15:01 min Laufen?
- Auto und Flüge: 350 km? 351 km? 500 km?

Wir wollen **beweisbar optimale** Journeys
mit uneingeschränktem Fuß- und Autovorlauf.

Kombination der Netzwerke

Zwei Schritte:

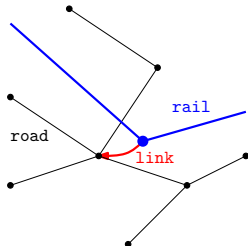
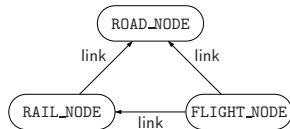
- Füge alle Graphen zusammen
- Verbinde sie durch Link-Kanten



Kombination der Netzwerke

Zwei Schritte:

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

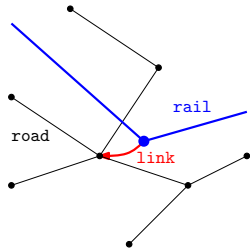
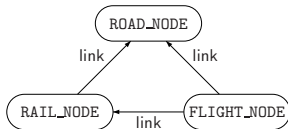


- Verbinde Knoten mit (geometrisch) nächstem Nachbar (oder besser: Indoor-Routing beachten)
- Labelle Kanten bzgl. ihres Transportmodus
- Resultierender Graph: groß, weniger deutliche Hierarchie

Kombination der Netzwerke

Zwei Schritte:

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

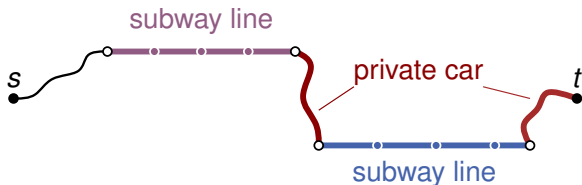


- Verbinde Knoten mit (geometrisch) nächstem Nachbar (oder besser: Indoor-Routing beachten)
- Labele Kanten bzgl. ihres Transportmodus
- Resultierender Graph: groß, weniger deutliche Hierarchie

Dijkstras Algorithmus berechnet (schnellste) Journeys auf multimodalem Graphen.

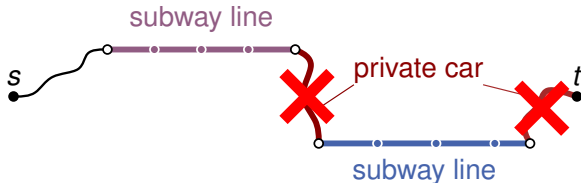
Unerwünschte Modalitätswechsel

Problem: Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen



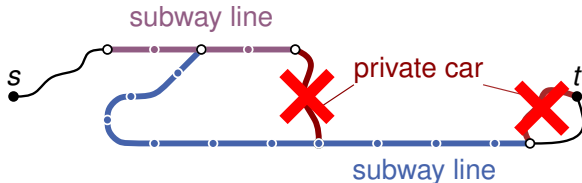
Unerwünschte Modalitätswechsel

Problem: Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen



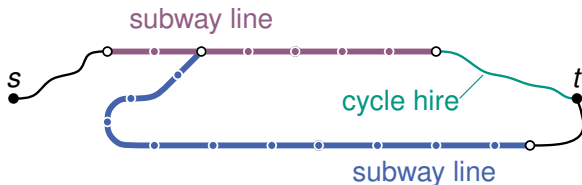
Unerwünschte Modalitätswechsel

Problem: Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen

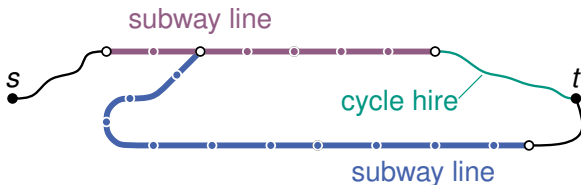


Unerwünschte Modalitätswechsel

Problem: Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen



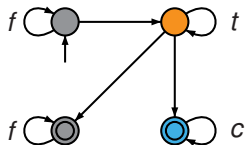
Problem: Kürzeste Wege haben evtl. merkwürdige Modalitätsfolgen



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

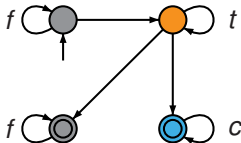
Label Constrained Shortest Path Problem (LCSPP):

- Definiere Alphabet Σ der Modi
- Kanten sind bzgl. Modus gelabelt
- Konkatenation 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 (LCSPP):

- Definiere Alphabet Σ 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 LCSPP: [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 (≈ 15 sec auf Europa-Instanz)

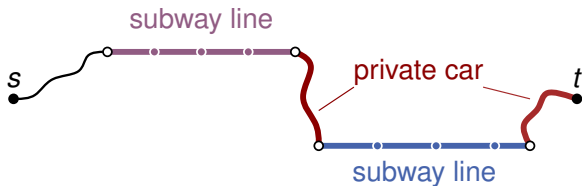
Feste Constraints:

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

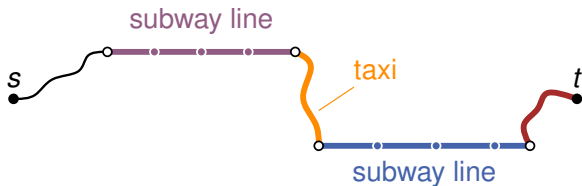
Benutzerdefinierte Constraints:

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

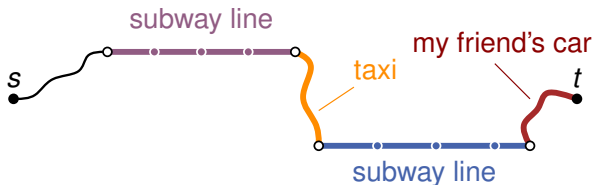
Benutzerdefinierte Constraints

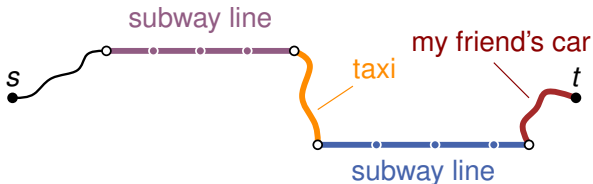


Benutzerdefinierte Constraints



Benutzerdefinierte Constraints





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

Mode Sequence Constraints

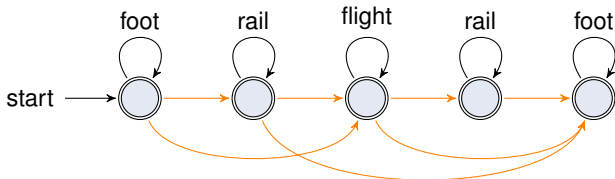
Beobachtung:

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")

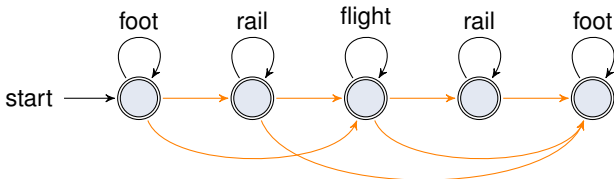


Idee: Betrachte **Mode Sequence** (MS) Constraints.

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:

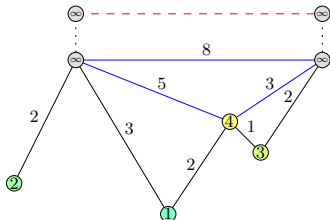
- Basierend auf Contraction Hierarchies [GSSV12]

Idee:

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

Vorbereitung:

- Setze Rank aller gelinkten Knoten auf ∞
 - Shortcuts übergreifen keine Netzwerke
 - Zeugensuche beschränkt auf Teilnetz
- ⇒ Ein gemeinsamer Core, unabh. Teilkomponenten

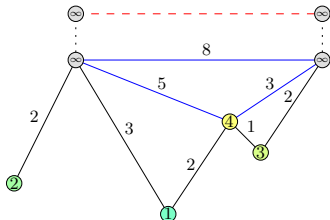


Idee:

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

Vorbereitung:

- Setze Rank aller gelinkten Knoten auf ∞
 - 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]

Lokale Anfragen automatisch identifiziert in Schritt 1

Zwei nahtlose Schritte:

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

Lokale Anfragen automatisch identifiziert in Schritt 1

Verbesserungen:

- Knoten umsortieren: Core-Knoten nach vorne
- Knotengrad wegen unkontrahierter Core-Knoten recht hoch
 - ⇒ Nur partielle Kontraktion
 - ⇒ Umsortieren der Kanten nach “ausgehend, gemischt, eingehend”
- Keine Rückwärtssuche im Core

Netzwerke:

Deutschland Straße (5 M Knoten, 12 M Kanten)

Deutschland Züge (6.8k 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
PCH	13	10 635	41.7 %	6	5 567	11 402	71 860	1.93
PCH	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	
Algorithm		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)
- \rightsquigarrow **Graph-Dekomposition**-basierte Techniken besser?

Bisher:

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

Aber:

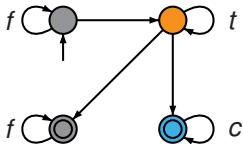
- Es ist nur *ein* kürzester Weg
 - Gibt nicht die Fülle aller verfügbaren Optionen wieder
 - Aktueller Straßenverkehr? Zeitabhängigkeit?
 - Weniger ausgeprägte Hierarchie (im Vergleich zu unimodalen Straßennetzwerken)
- \rightsquigarrow **Graph-Dekomposition**-basierte Techniken besser?

Problem gelöst?



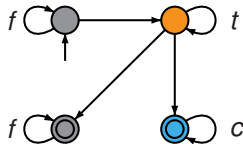
Label Constrained Shortest Path Problem (LCSP):

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



Label Constrained Shortest Path Problem (LCSP):

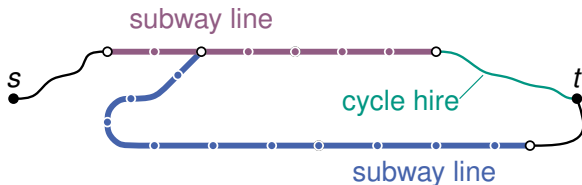
- Definiere Alphabet über Transportmodi
- Graph mit Kantenlabels
- Gültiger Pfad muss Wort aus einer gegebenen regulären Sprache entsprechen



Algorithmen für LCSP:

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

Nachteile von LCSPP:



Bisherige Ansätze

Nachteile von LCSPP:

s
•

?

t
•

Nachteile von LCSPP:

s

?

t

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

Nachteile von LCSPP:

s

?

t

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

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

Multikriterielle multimodale Journeys

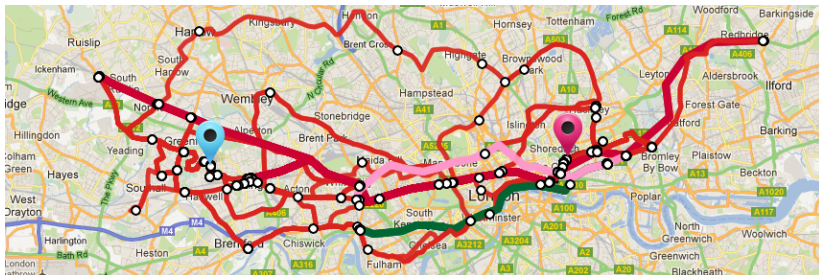
Ziel: Berechne **multikriterielle** multimodale Pareto-Mengen.

- 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)

Eine Journey J_1 **dominiert** eine Journey J_2 gdw. J_2 in allen Kriterien schlechter (oder gleich) ist.

Eine **Pareto-Menge** ist eine (inklusionsmaximale) Menge nichtdominierter Journeys.

Definition (Dominanz, Pareto-Menge)

Eine Journey J_1 **dominiert** eine Journey J_2 gdw. J_2 in allen Kriterien schlechter (oder gleich) ist.

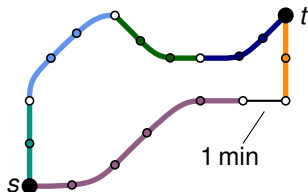
Eine **Pareto-Menge** ist eine (inklusionsmaximale) Menge nichtdominierter Journeys.

Irrelevante Lösungen:

Eine nichtdominierte Journey kann

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

Viele Kriterien \Rightarrow Kombinat. Explosion



Definition (Dominanz, Pareto-Menge)

Eine Journey J_1 **dominiert** eine Journey J_2 gdw. J_2 in allen Kriterien schlechter (oder gleich) ist.

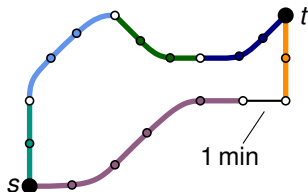
Eine **Pareto-Menge** ist eine (inklusionsmaximale) Menge nichtdominierter Journeys.

Irrelevante Lösungen:

Eine nichtdominierte Journey kann

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

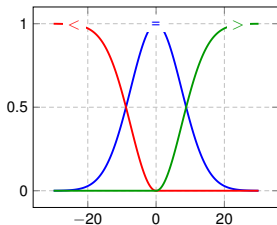
Viele Kriterien \Rightarrow Kombinat. Explosion



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

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

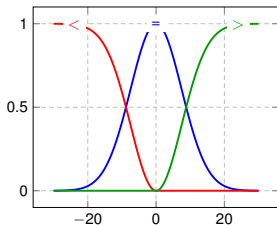
- ± 1 Minute Laufen ist “ungefähr gleichwertig”,
- aber ± 30 Minuten machen einen Unterschied aus.



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

- ± 1 Minute Laufen ist “ungefähr gleichwertig”,
- aber ± 30 Minuten machen einen Unterschied aus.

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

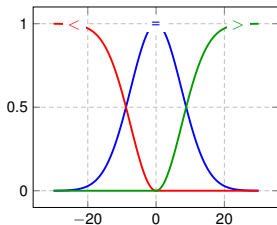


Beobachtung: Benutzer haben **unscharfe Vorstellung** bzgl. einiger Kriterien.

- ± 1 Minute Laufen ist “ungefähr gleichwertig”,
- aber ± 30 Minuten machen einen Unterschied aus.

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

- Fuzzy-Operatoren $<$, $>$ und $=$
 - Unterschiedliche Parametrisierung pro Kriterium
- ⇒ *Grad* der Dominanz $d(J_1, J_2) \in [0, 1]$.
“Wie stark wird J_2 von J_1 dominiert?”

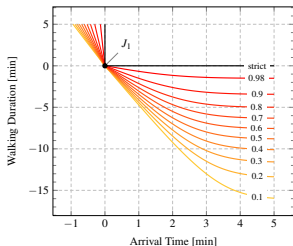


Beobachtung: Benutzer haben **unscharfe Vorstellung** bzgl. einiger Kriterien.

- ± 1 Minute Laufen ist “ungefähr gleichwertig”,
- aber ± 30 Minuten machen einen Unterschied aus.

Idee: Relaxiere Dominanzbegriff mit **unscharfer Mengenlehre** (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

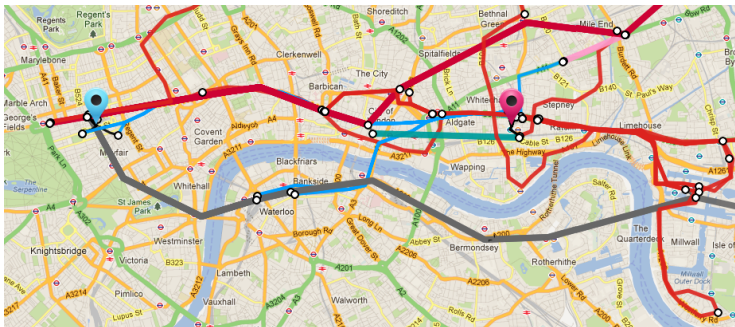
Ansatz:

- 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:

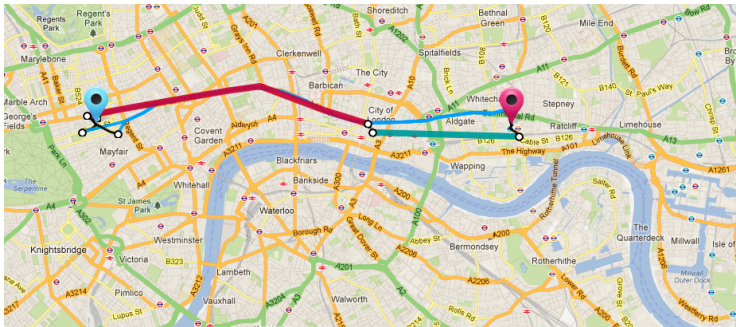
- Berechne volle (exakte) Pareto-Menge $\{J_1, \dots, J_n\}$ mit multikriteriellem Algorithmus
- Bewerte jede Journey J mit $1 - \max(d(J_1, J), \dots, d(J_n, J))$
- Dann gilt: Journeys mit höherer Bewertung sind wichtiger



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 mit 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)

Heuristische Ansätze

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

Qualität der Heuristiken

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

Motivation: Service benutzt Alg. \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

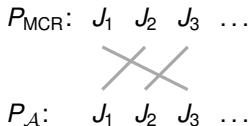
Motivation: Service benutzt Alg. \mathcal{A} , präsentiert dem Nutzer Top- k -Journeys aus $P_{\mathcal{A}}$.

Ähnlichkeit von Journeys:

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

Qualität der Lösungen:

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



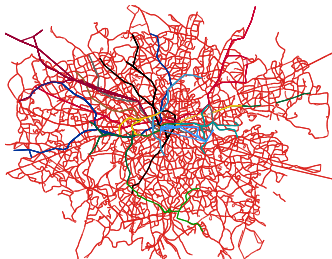
Ausgabe: Durchschnittliche Ähnlichkeit der gematchten Journeys

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 ≈ 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	Wik.	# 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

Wik. ○: Kombiniert Laufdauer mit Kosten in ein Kriterium

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



Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck.

VC-Dimension and Shortest Path Algorithms.

In *Proceedings of the 38th International Colloquium on Automata, Languages, and Programming (ICALP'11)*, volume 6755 of *Lecture Notes in Computer Science*, pages 690–699. Springer, 2011.



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.



Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck.

Highway Dimension, Shortest Paths, and Provably Efficient Algorithms.

In Moses Charikar, editor, *Proceedings of the 21st Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 782–793. SIAM, 2010.



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 (OASIs), 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 (OASISs), 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.