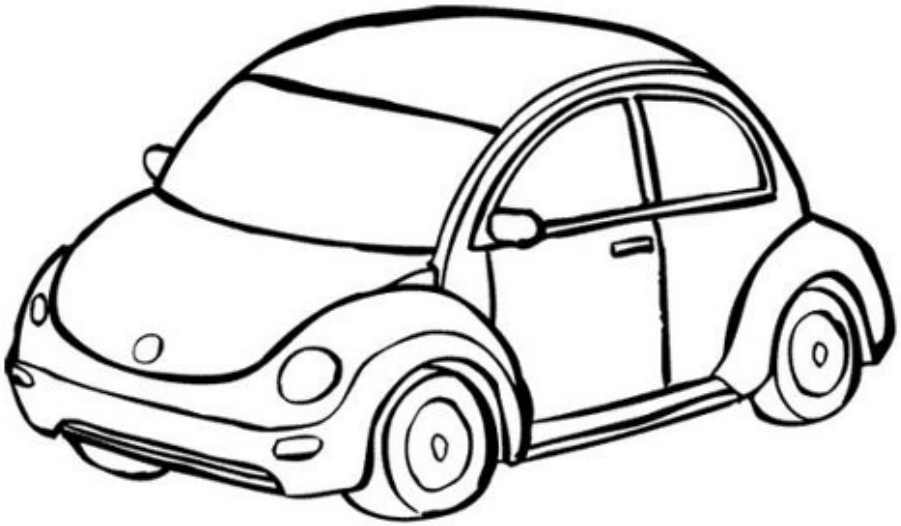
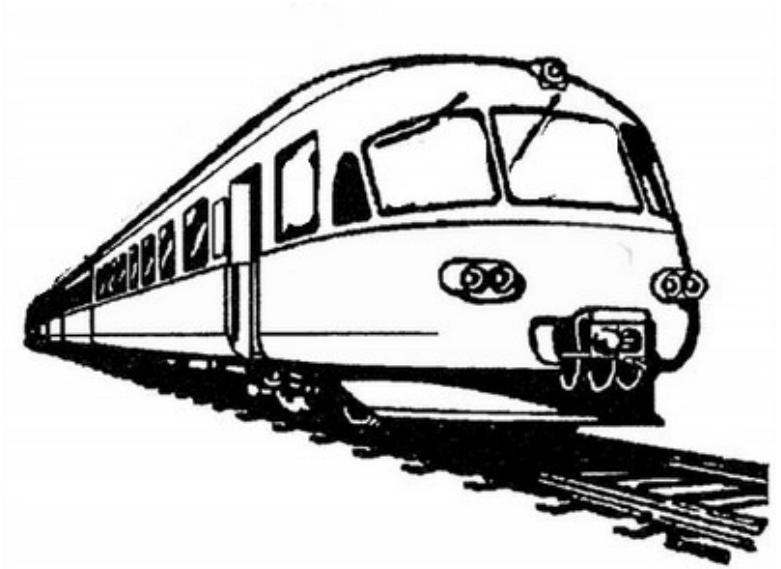
Karlsruher Institut für Technologie

# Fortgeschrittene Routenplanung in Transportnetzen

Advanced Route Planning in Transportation Networks

Dissertationsvortrag von Dipl.-Inform. Robert Geisberger





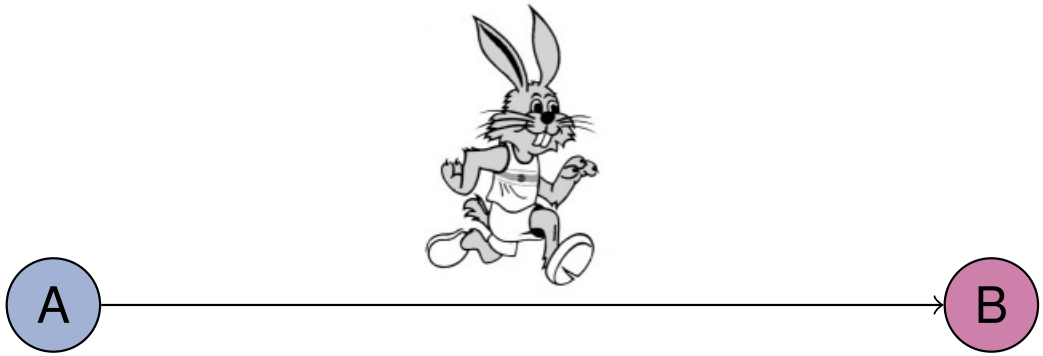
# Motivation

- **Routenplanungssysteme** sind allgegenwärtig
  - Autonavigation
  - Bus- und Bahnauskunft
  - Logistik
- Realität stellt **vielfältige Anforderungen** an Routenplanungssyteme:
  - Wahl von Zielfunktionen
    z.B. kürzeste vs. schnellste Route
  - Gesetzliche Bestimmungen
    z.B. Verbot von Gefahrguttransporten
  - Wahl der Transportmittel
    z.B. Auto oder Bahn
  - usw.


# Grundlegendes

## Routenplanung

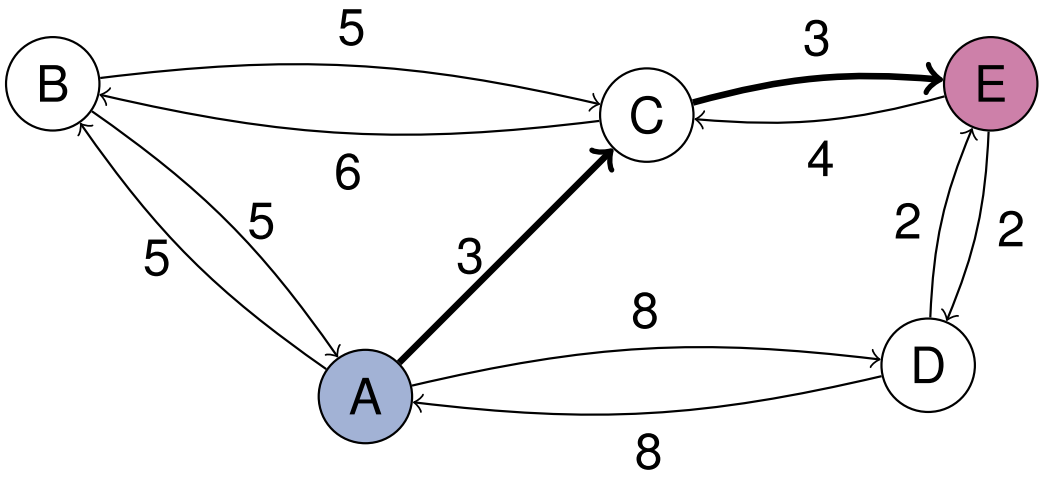
- **Problem:** Berechnung kürzester Wege auf gewichteten Graphen von Transportnetzen
- **Theorie:** Algorithmus von Dijkstra (oder Varianten)
- **Praxis:** Schnellere Algorithmen durch vorberechnete Daten



# Schwerpunkt bisheriger Forschung

## Einfache Routenplanung

- **Straßennetz**
- **Einfache Kantengewichte** $c : E \to \mathbb{R}_+$ (meist Reisezeit)
- **ein Start- und ein Zielknoten**



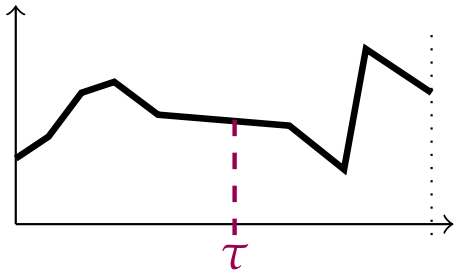

- Beschleunigungstechniken **sehr erfolgreich**
  - Berechnungszeit: Sekunden → Mikrosekunden
  - **nicht leicht** [Schultes 2008, Delling 2009]

# Bisherige Erweiterungen

## Auszug

**Erfolgreich:**

- **Dynamische** Routenplanung (Stau)
  [Schultes 2007]
- **Zeitabhängige** Routenplanung (Abfahrtszeit)
  [Delling 2008, Batz et al. 2009]
- Statische **Distanztabellen**
  [Knopp et al. 2007]



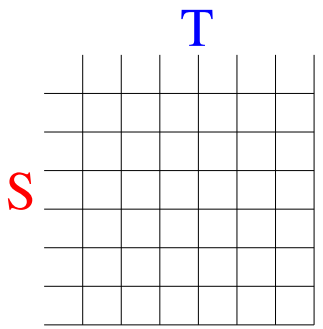

**Erfolgreich nur für eingeschränktes Szenario:**

- Öffentliche Verkehrsnetze

# Meine Arbeit

## Fortgeschrittene Routenplanung

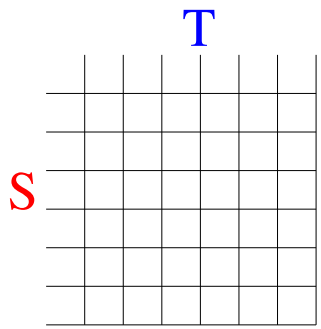
- Straßennetz
  → **Öffentliche Verkehrsnetz** (Bus, Bahn)
  - Routenplanung mit realistischen Umsteigezeiten
  - Vollständig realistische Routenplanung
- Einfaches Kantengewicht
  → **Flexibles Kantengewicht** auf Straßennetzen
  - Mehrere Kantengewichte, z.B. Reisezeit und Kosten
  - Kantenrestriktionen
- ein Start- und ein Zielknoten
  → **mehrere Start- und Zielknoten**
  - Vermittlung von Fahrgemeinschaften
  - Zeitabhängige Reisezeittabellen
  - Berechnung nächstgelegener Sonderziele

# Meine Arbeit

## Fortgeschrittene Routenplanung

- Straßennetz
  → **Öffentliche Verkehrsnetz** (Bus, Bahn)
  - Routenplanung mit realistischen Umsteigezeiten
  - Vollständig realistische Routenplanung
- Einfaches Kantengewicht
  → **Flexibles Kantengewicht** auf Straßennetzen
  - Mehrere Kantengewichte, z.B. Reisezeit und Kosten
  - Kantenrestriktionen
- ein Start- und ein Zielknoten
  → **mehrere Start- und Zielknoten**
  - Vermittlung von Fahrgemeinschaften
  - Zeitabhängige Reisezeittabellen
  - Berechnung nächstgelegener Sonderziele

# Fortgeschrittene Routenplanung

- Die meisten fortgeschrittenen Probleme lassen sich **nicht effizient** durch Algorithmen für einfache Probleme lösen
- **Neue Modelle, algorithmische Bauteile oder neue Algorithmen notwendig**

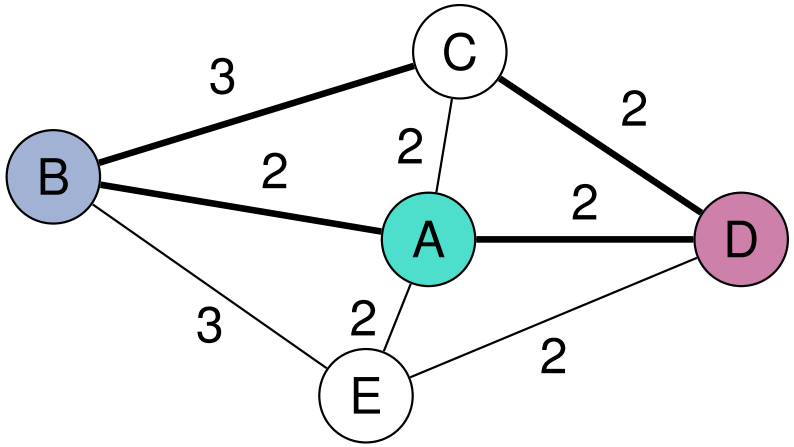
**Beispiel:** Bikriterielles Kantengewicht [ALENEX 2010]

- Zwei Einzelgewichte: $c^{(1)}$ und $c^{(2)}$ (z.B. Reisezeit und Kosten)
- Parameter $p$ aus diskretem Intervall $[L, U]$
- **Flexibles Kantengewicht** $c_p : c^{(1)} + p \cdot c^{(2)}$ kann **für jede Anfrage neu** gewählt werden

# Bikriterielles Kantengewicht

## Vorberechnung: Knotenkontraktion

- Entferne Knoten und füge Abkürzungskanten ein um die **Distanzen zwischen den verbleibenden Knoten zu erhalten**.
- Notwendig: Erhalte Distanzen für **jeden Parameterwert** $p \in [L, U]$
- **Standardvorgehen** Kontraktion von Knoten $v$:
  Vergleiche Distanzen zwischen **Nachbarn** mit und ohne Knoten $v$.
  **Erhöhte Distanz** $\Rightarrow$ **Abkürzungskante**



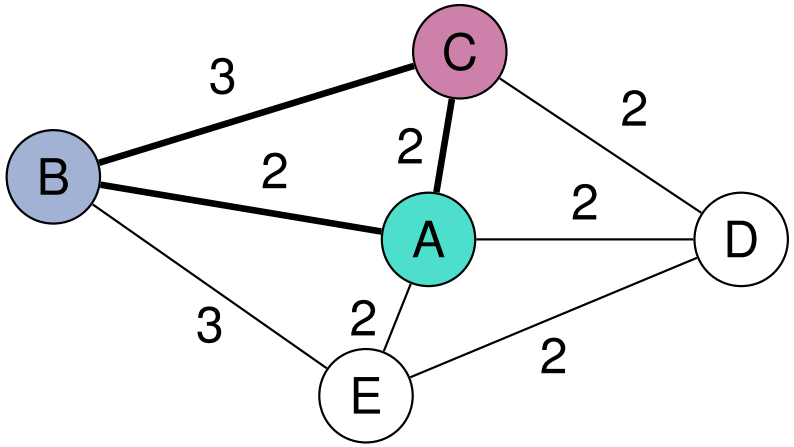

$5 > 4$
$\Rightarrow$ Abkürzungskante einfügen

- Problem: $|[L, U]|$ kann groß sein, Experimente: $|[L, U]| = 1024$

# Bikriterielles Kantengewicht

## Vorberechnung: Knotenkontraktion

- Entferne Knoten und füge Abkürzungskanten ein um die **Distanzen zwischen den verbleibenden Knoten zu erhalten**.
- Notwendig: Erhalte Distanzen für **jeden Parameterwert** $p \in [L, U]$
- **Standardvorgehen** Kontraktion von Knoten $v$:
  Vergleiche Distanzen zwischen **Nachbarn** mit und ohne Knoten $v$.
  **Erhöhte Distanz** $\Rightarrow$ **Abkürzungskante**




$3 \leq 4$
$\Rightarrow$ **Zeuge** verhindert Abkürzung

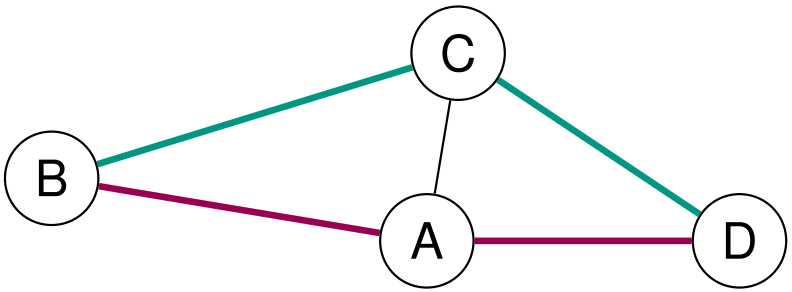
- Problem: $|[L, U]|$ kann groß sein, Experimente: $|[L, U]| = 1024$

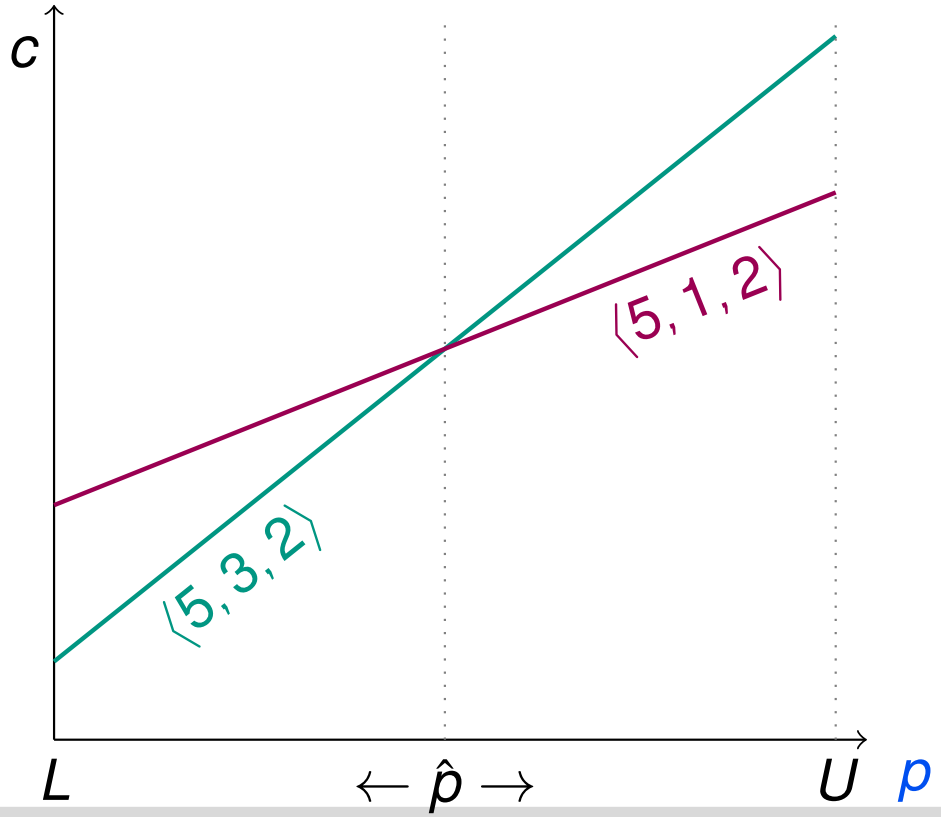
# Bikriterielles Kantengewicht

## Vorberechung: Verbesserte Zeugensuche

Beobachtung:

- Potentielle Abkürzung $A$: $c_p(A) = c^{(1)}(A) + p \cdot c^{(2)}(A)$
- Zeuge $Z$: $c_p(Z) = c^{(1)}(Z) + p \cdot c^{(2)}(Z)$
- Kosten von $A$ und $Z$ **lineare Funktionen** über $p$



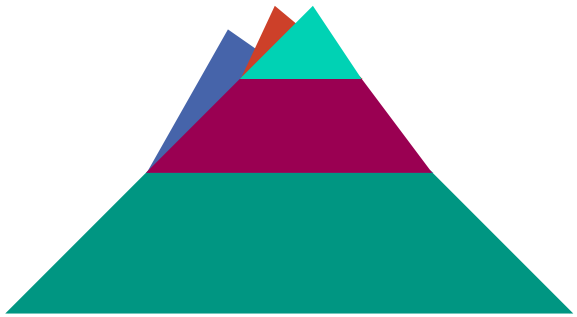



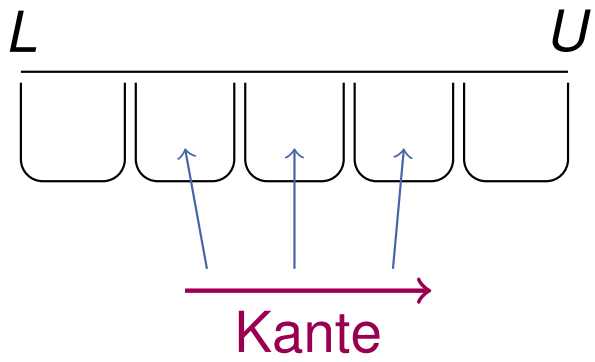

- Zeuge für **ganzes Intervall** gültig
- $\approx 2$ statt 1024 Zeugensuchen in der Praxis

# Bikriterielles Kantengewicht

## Vorteile eines Parameterintervalls

- Neu: **Linearkombination** zweier Gewichte mittels eines Parameters aus einem Intervall
- Voraussetzung für **effiziente Vorberechnung**.
- Weitere darauf aufbauende Techniken:
  - Kombination von Distanzen zu **Landmarken für A*** (ALT)
  - **Anpassung der Knotenordnung** auf Teilintervallen
  - **Notwendigkeit von Abkürzungen** eingeschränkt auf Intervall
  - **Bucket-Datenstruktur** für Kanten
- Anfrage auf Europa: 1.87 ms ($\approx 8\,000\times$ Beschleunigung)





# Öffentliche Verkehrsnetze

**Mindestanforderungen:**

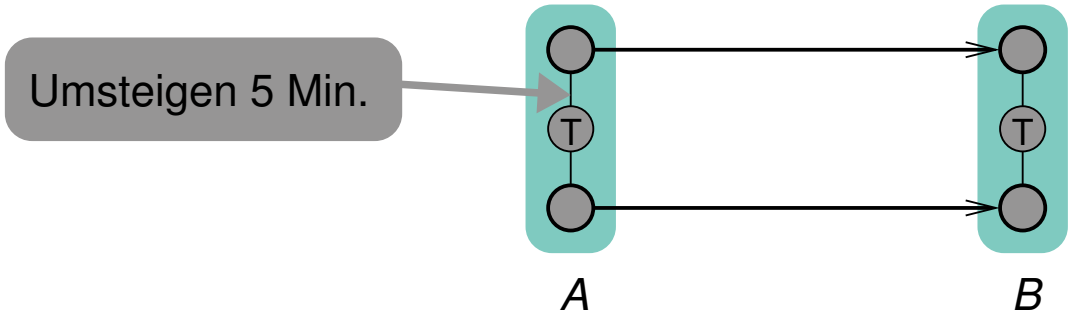
- **Zeitabhängigkeit** gegeben durch Fahrpläne
- **Zeitpuffer** beim Umsteigen

Weitere wichtige Eigenschaften für **vollständig realistisches Routing**:

- **Multikriterielle Optimierung** von Reisezeit, Umsteigehäufigkeit und weiteren Kosten
- **Fußwege** zwischen Stationen
- **Punkt-zu-Punkt Anfragen** statt Station-zu-Station Anfragen


# Öffentliche Verkehrsnetze

- Beschleunigungsalgorithmen für Straßennetze haben schon **Probleme beim Routing** mit Mindestanforderungen:
  - Sehr **hohe Vorberechungszeit**
  - **Geringe Beschleunigung** von Anfragen




- **Verbesserte Modellierung** [SEA 2010]:
  - Stationsgraphenmodell: **Ein Knoten pro Station**
  - Effiziente **Operationen auf Kantengewichten komplex**
  - $\approx 10\times$ Beschleunigung gegenüber bestehenden Verfahren

**Vollständig realistische Routenplanung:**

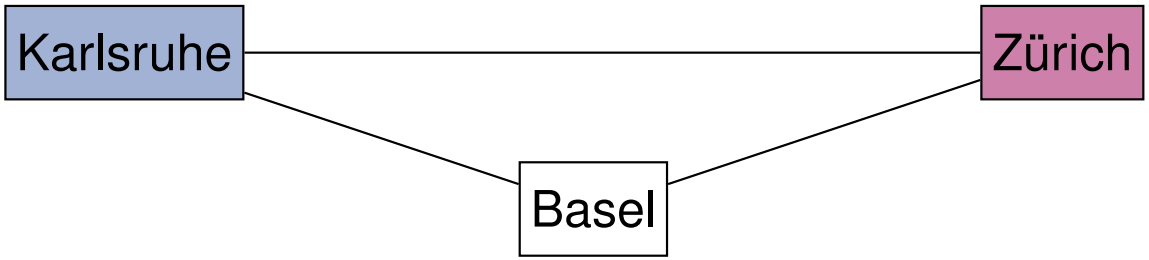
- Neuer **maßgeschneiderter Algorithmus**

# Öffentliche Verkehrsnetze

## Neuer Algorithmus [ESA 2010]

### Kernidee:

- **Aufteilung** der Berechnung optimaler Verbindungen
  1. Optimale **Umsteigemuster**
  2. Bewertung der Umsteigemuster mittels **Direktverbindungs-Anfragen**




### Hintergründe:

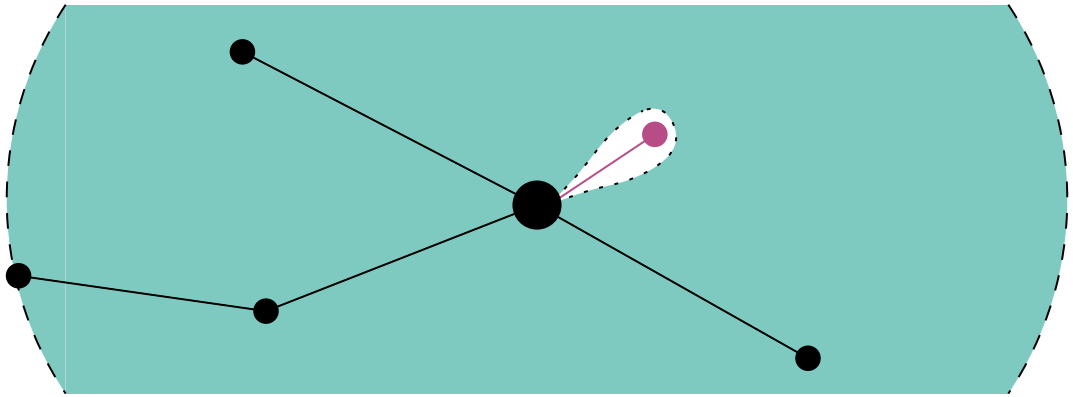
- Anzahl optimaler **Umsteigemuster gering**, kann vorberechnet werden
- Direktverbindungs-Anfragen können **effizient** beantwortet werden

# Umsteigemuster

## Vorberechnung

Schweiz, 20 594 Stationen

- **Alle optimale Verbindungen** zwischen allen Stationspaaren ausrechnen (parallelisierbar) 635 Std. / 18 562 MB
- **Wichtige Stationen** 586 Std. / 819 MB
  Globale Suchen: von wichtigen Stationen zu allen anderen
  Lokale Suchen: von unwichtigen nur bis wichtige Station

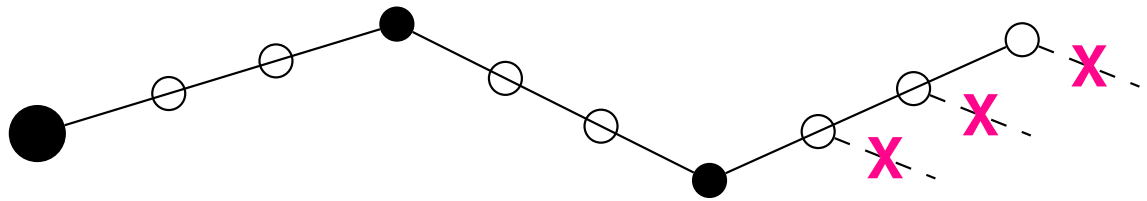


**Problem:** **Räumlich nahe** gelegene Orte sind **zeitlich weit entfernt**, wenn beispielsweise keine Verbindung über Nacht besteht.
Beispiel: Karlsruhe $\rightarrow$ Neupotz (22 km, 8 Std.)

# Umsteigemuster

## Vorberechnung

- **Wichtige Stationen** 586 Std. / 819 MB
- **Nur 2× Umsteigen** in lokaler Suche 88 Std. / 421 MB
  - 3 aus 10 000 Anfragen mit geringfügig suboptimalem Ergebnis
  - Fehler in den Eingabedaten sind ein deutlich größeres Problem




- Weitere **heuristische Einschränkungen** 61 Std. / 214 MB

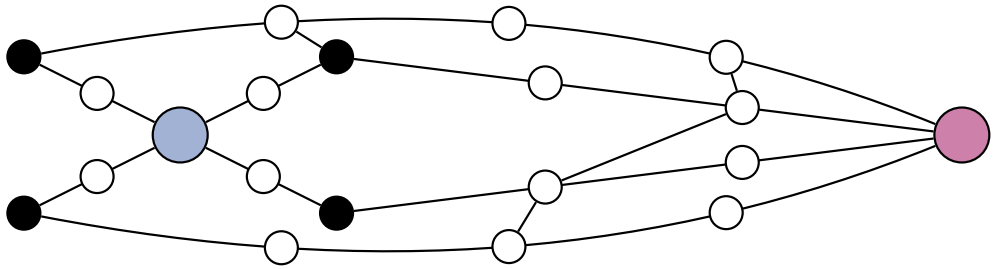
Für größere und schlecht strukturiertere Graphen ist **heuristische Berechnung notwendig**:

- Nordamerika, 338 133 Stationen 3 203 Std. / 14 000 MB
- Router im praktischen Einsatz auf **http://www.google.com/transit**

# Umsteigemuster

## Beantwortung von Anfragen

1. Erzeugen des Anfragegraphen aus Umsteigemustern
2. Zeitabhängige Suche auf Anfragegraphen



| **Graph** | Anfragezeit [ms] |
|---|---|
| Schweiz | 1 |
| Nordamerika | 12 |

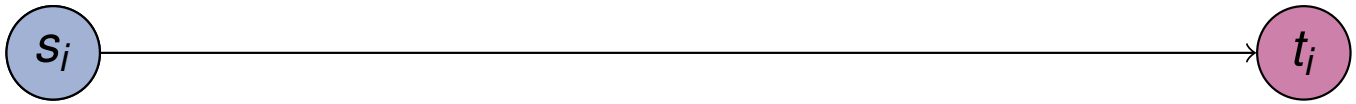
Bisherige Arbeiten [WEA 2008, ATMOS 2009]

- nur kleine, gut strukturierte Graphen ($\approx$ 9 000 Stationen, DB)
- Anfragezeit: einigen Hundert Millisekunden bis eine Sekunde

# Fahrgemeinschaften

## Problem mit mehreren Start- und Zielknoten

- Datenbank mit $k$ **Fahrern**. Fahrer $i$ fährt von $s_i$ nach $t_i$.
- **Beifahrer** sucht Mitfahrgelegenheit von $s'$ nach $t'$.
- Finde Mitfahrgelegenheit mit **geringstem Umweg** für Fahrer.
  - Bisherige Systeme meist **reine Datenbanken**
  - Start- und Zielpunkte müssen identisch oder nahe beieinander sein
  - Neuer Ansatz erst **durch einen schnellen Algorithmus möglich**, welcher die $2k + 1$ Distanzen effizient ausrechnen kann



# Fahrgemeinschaften

## Problem mit mehreren Start- und Zielknoten

- Datenbank mit $k$ **Fahrern**. Fahrer $i$ fährt von $s_i$ nach $t_i$.
- **Beifahrer** sucht Mitfahrgelegenheit von $s'$ nach $t'$.
- Finde Mitfahrgelegenheit mit **geringstem Umweg** für Fahrer.
  - Bisherige Systeme meist **reine Datenbanken**
  - Start- und Zielpunkte müssen identisch oder nahe beieinander sein
  - Neuer Ansatz erst **durch einen schnellen Algorithmus möglich**, welcher die $2k + 1$ Distanzen effizient ausrechnen kann

# Fahrgemeinschaften

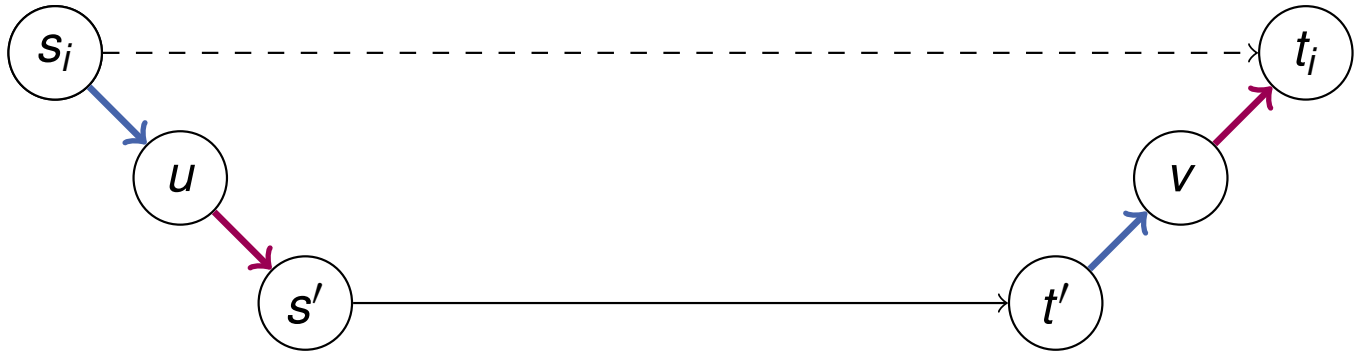
## Problem mit mehreren Start- und Zielknoten

- Datenbank mit $k$ **Fahrern**. Fahrer $i$ fährt von $s_i$ nach $t_i$.
- **Beifahrer** sucht Mitfahrgelegenheit von $s'$ nach $t'$.
- Finde Mitfahrgelegenheit mit **geringstem Umweg** für Fahrer.
  - Bisherige Systeme meist **reine Datenbanken**
  - Start- und Zielpunkte müssen identisch oder nahe beieinander sein
  - Neuer Ansatz erst **durch einen schnellen Algorithmus möglich**, welcher die $2k + 1$ Distanzen effizient ausrechnen kann

# Fahrgemeinschaften

## Effizienter Algorithmus [ATMOS 2010]

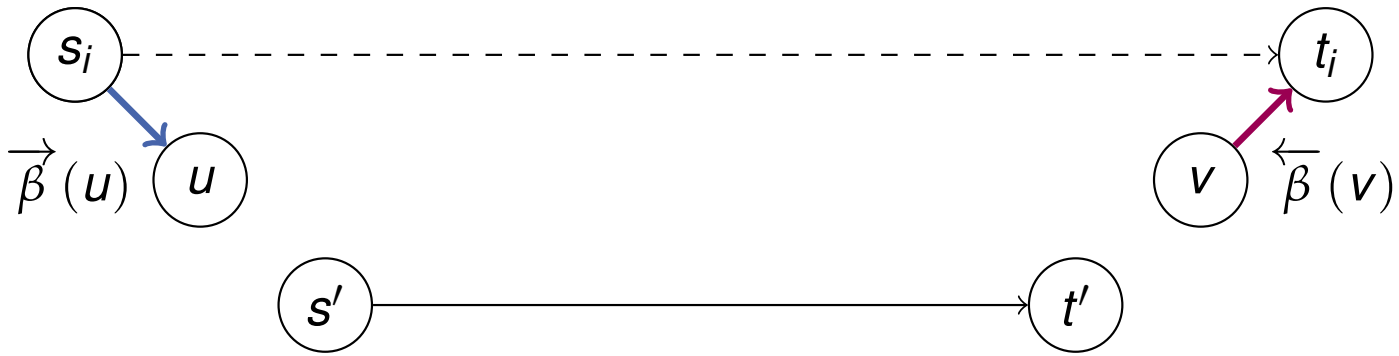
- Benutze **hierarchisches, nicht zielgerichtetes Verfahren**.
- Führe Vorwärtssuche von $s_i$ und Rückwärtssuche von $t_i$, $i = 1..k$ **nur einmal** aus und **speichere Suchräume in Buckets** (verwandt mit [Knopp et al. 07])
- Um Mitfahrgelegenheit zu finden, führe Rückwärtssuche von $s'$ und Vorwärtssuche von $t'$ aus und **durchlaufe Buckets**.
- Deutschland, $k = 100\,000$: 43.4 ms $\rightarrow$ 217 ns pro Distanz
  (Transit Node Routing: 1 900 ns)



# Fahrgemeinschaften

## Effizienter Algorithmus [ATMOS 2010]

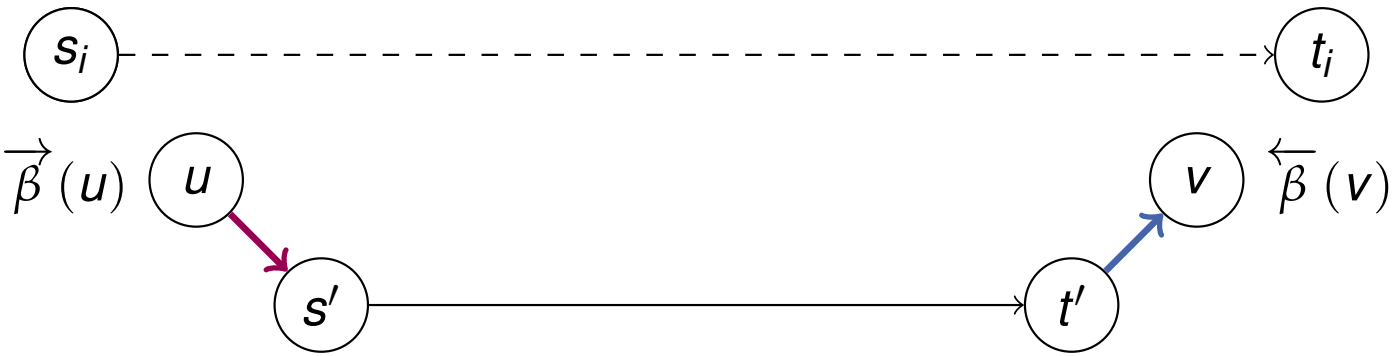
- Benutze **hierarchisches, nicht zielgerichtetes Verfahren**.
- Führe Vorwärtssuche von $s_i$ und Rückwärtssuche von $t_i$, $i = 1..k$ **nur einmal** aus und **speichere Suchräume in Buckets** (verwandt mit [Knopp et al. 07])
- Um Mitfahrgelegenheit zu finden, führe Rückwärtssuche von $s'$ und Vorwärtssuche von $t'$ aus und **durchlaufe Buckets**.
- Deutschland, $k = 100\,000$: 43.4 ms $\rightarrow$ 217 ns pro Distanz (Transit Node Routing: 1 900 ns)



# Fahrgemeinschaften

## Effizienter Algorithmus [ATMOS 2010]

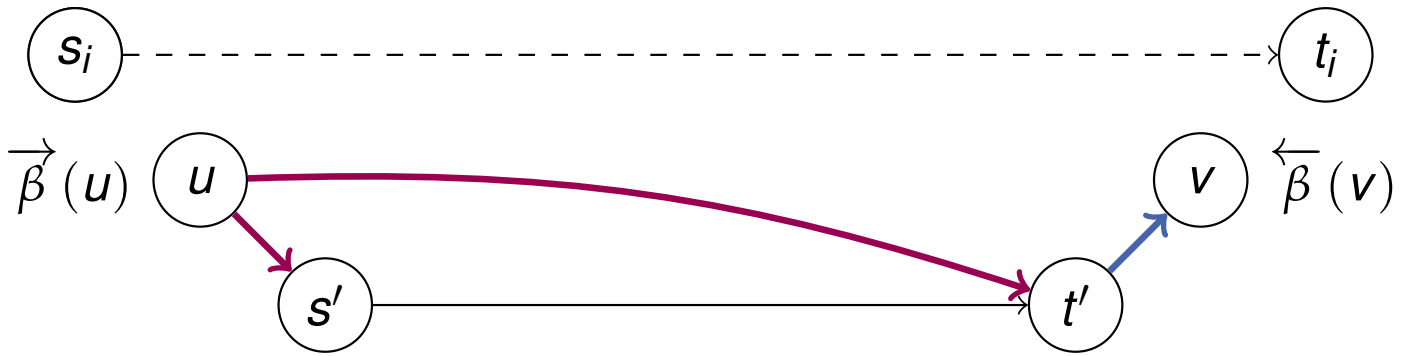
- Benutze **hierarchisches, nicht zielgerichtetes Verfahren**.
- Führe Vorwärtssuche von $s_i$ und Rückwärtssuche von $t_i$, $i = 1..k$ **nur einmal** aus und **speichere Suchräume in Buckets** (verwandt mit [Knopp et al. 07])
- Um Mitfahrgelegenheit zu finden, führe Rückwärtssuche von $s'$ und Vorwärtssuche von $t'$ aus und **durchlaufe Buckets**.
- Deutschland, $k = 100\,000$: 43.4 ms $\rightarrow$ 217 ns pro Distanz
  (Transit Node Routing: 1 900 ns)



# Fahrgemeinschaften

## Effizienter Algorithmus [ATMOS 2010]

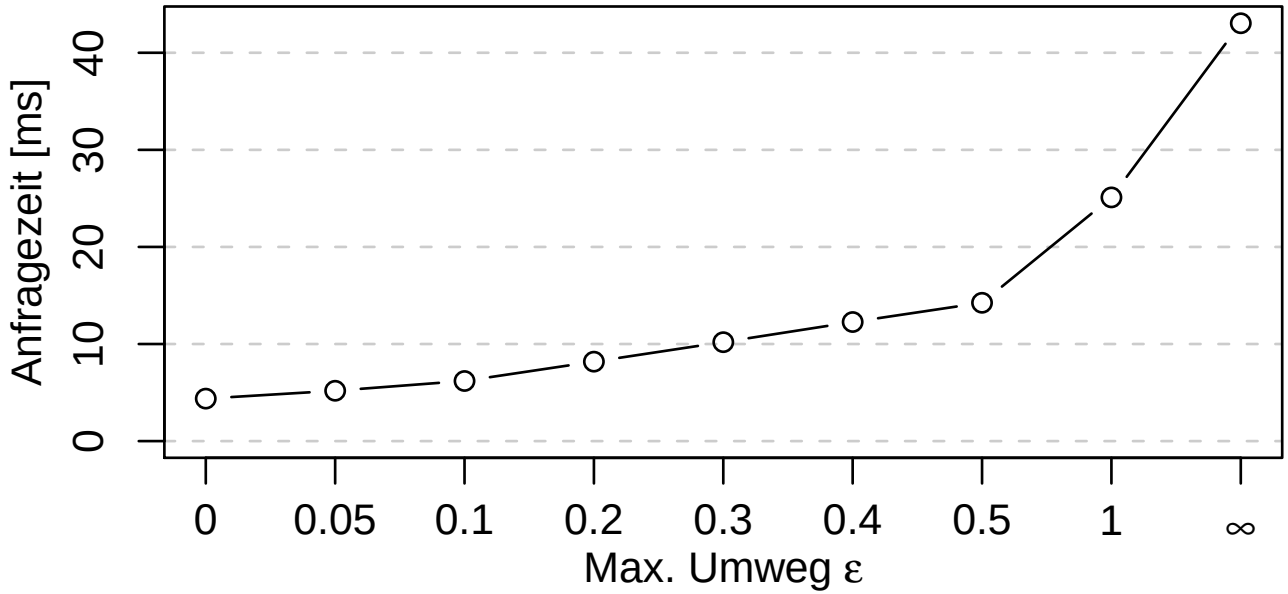
- Weitere Beschleunigung durch **Beschränkung des Umwegs**
- Umweg $\leq \varepsilon \cdot \mu(s', t')$
- Führe **zusätzliche Rückwärtssuche** von $t'$ aus, welche **möglicherweise** eine obere Schranke von $\mu(u, t')$ berechnet (symmetrisch für $\mu(s', v)$)
- Diese lässt den Umweg nach unten abschätzen.



# Fahrgemeinschaften

## Effizienter Algorithmus [ATMOS 2010]

- Weitere Beschleunigung durch **Beschränkung des Umwegs**
- Umweg $\leq \varepsilon \cdot \mu(s', t')$
- Führe **zusätzliche Rückwärtssuche** von $t'$ aus, welche **möglicherweise** eine obere Schranke von $\mu(u, t')$ berechnet (symmetrisch für $\mu(s', v)$)
- Diese lässt den Umweg nach unten abschätzen.



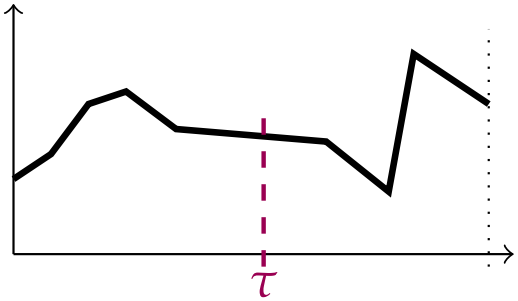
# Zusammenfassung

**Fortgeschrittene Routenplanung stellt neue algorithmische Herausforderungen:**

- **Flexible Anfragen** benötigen erweiterte Vorberechnung
  - Intelligente Zeugensuche notwendig
  - Kombination aus Knotenkontraktion und ALT beschleunigt Anfragen um 3–4 Größenordnungen
- Routing in **öffentlichen Verkehrsnetzen** profitiert von einem speziell entwickelten Algorithmus
  - Aufteilung der Berechnung in Umsteigemuster und Direktverbindungs-Anfragen
  - Erster effizienter Algorithmus für große, schlecht strukturierte Graphen
  - Vollständig realistisches Routing
- Berechnung vieler Distanzen mit sich **überschneidenden Start- oder Zielknoten** kann deutlich beschleunigt werden

# Ausblick

- **Weitere flexible Szenarien** berücksichtigen und kombinieren
  - **Zeitabhängige** Routenplanung
  - **Dynamische** Routenplanung




- **Umsteigemuster-Ansatz** erweitern, speziell Vorberechnung
- **Multimodale** Routenplanung (z.B. Auto + Bahn)

# Veröffentlichungen

1. **Route Planning with Flexible Objective Functions** mit M. Kobitzsch, P. Sanders. ALENEX 2010.
2. **Contraction of Timetable Networks with Realistic Transfers**. SEA 2010.
3. **Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns** mit H. Bast, E. Carlsson, A. Eigenwillig, C. Harrelson, V. Raychev, F. Viger. ESA 2010.
4. **Fast Detour Computation for Ride Sharing** mit D. Luxen, P. Sanders, S. Neubauer, L. Volker. ATMOS 2010.
5. **Engineering Time-Dependent Many-to-Many Shortest Paths Computation** mit P. Sanders. ATMOS 2010.
6. **Route Planning with Flexible Edge Restrictions** mit M. Rice, P. Sanders, V. Tsotras. Eingereicht bei JEA.