Karlsruher Institut für Technologie

Algorithmen für Routenplanung

13. Vorlesung, Sommersemester 2012

Daniel Delling | 13. Juni 2012

MICROSOFT RESEARCH SILICON VALLEY

Diskussion

Was fehlt noch für Einsatz in Produktion?

- Berücksichtigung von Staus
- Implementation auf Mobilgeräten
- Abbiegekosten/-verbote

Dynamische Szenarien

Szenario:

- Unfall auf einer Straße
- Reisezeit ändert sich auf dieser Straße
- berechne schnellsten Weg bezüglich der aktualisierten Reisezeiten

Herausforderung

Hauptproblem:

- Kantengewichte ändern sich
- Vorberechnung basiert auf ursprünglichen Kantengewichten
- komplette Vorberechnung für jeden Stau wenig sinnvoll

Serverszenario:

- identifiziere den Teil der beeinträchtigten Vorberechnung
- aktualisiere diesen Teil

mobiles Szenario:

- robuste Vorberechnung
- immer noch korrekt, wenn Kantengewichte sich erhöhen
- dadurch eventuell Anfragen langsamer
- oder passe Query an, so dass Vorberechnung nicht aktualisiert werrden muss

Landmarken [Goldberg, Harrelson 05]

Vorberechnung:

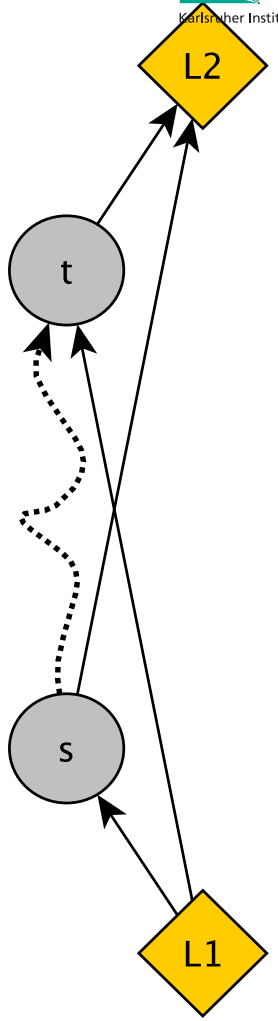
- wähle eine Hand voll (≈ 16) Knoten als Landmarken
- berechne Abstände von und zu allen Landmarken

Anfrage:

- benutze Landmarken und Dreiecksungleichung um eine untere Schranke für den Abstand zum Ziel zu bestimmen

$$\begin{aligned} d(s,t) &\geq d(L_1,t) - d(L_1,s) \\ d(s,t) &\geq d(s,L_2) - d(t,L_2) \end{aligned}$$

- verändert Reihenfolge der besuchten Knoten



Serverszenario

Beobachtung:

- Landmarkenwahl ist Heuristik, also nicht ändern
- Distanzen von und zu Landmarken sind nicht korrekt

Vorgehen:

- aktualisiere diese Distanzen
- benutze dynamischen Dijkstra Algorithmus um Distanzen zu aktualisieren
- dann Anfragen korrekt und so schnell wie bei kompletter neuer Vorberechnung

Mobiles Szenario

Beobachtung:

- Korrektheit von ALT basiert darauf, dass reduzierten Kantengewichte größer gleich 0 sind

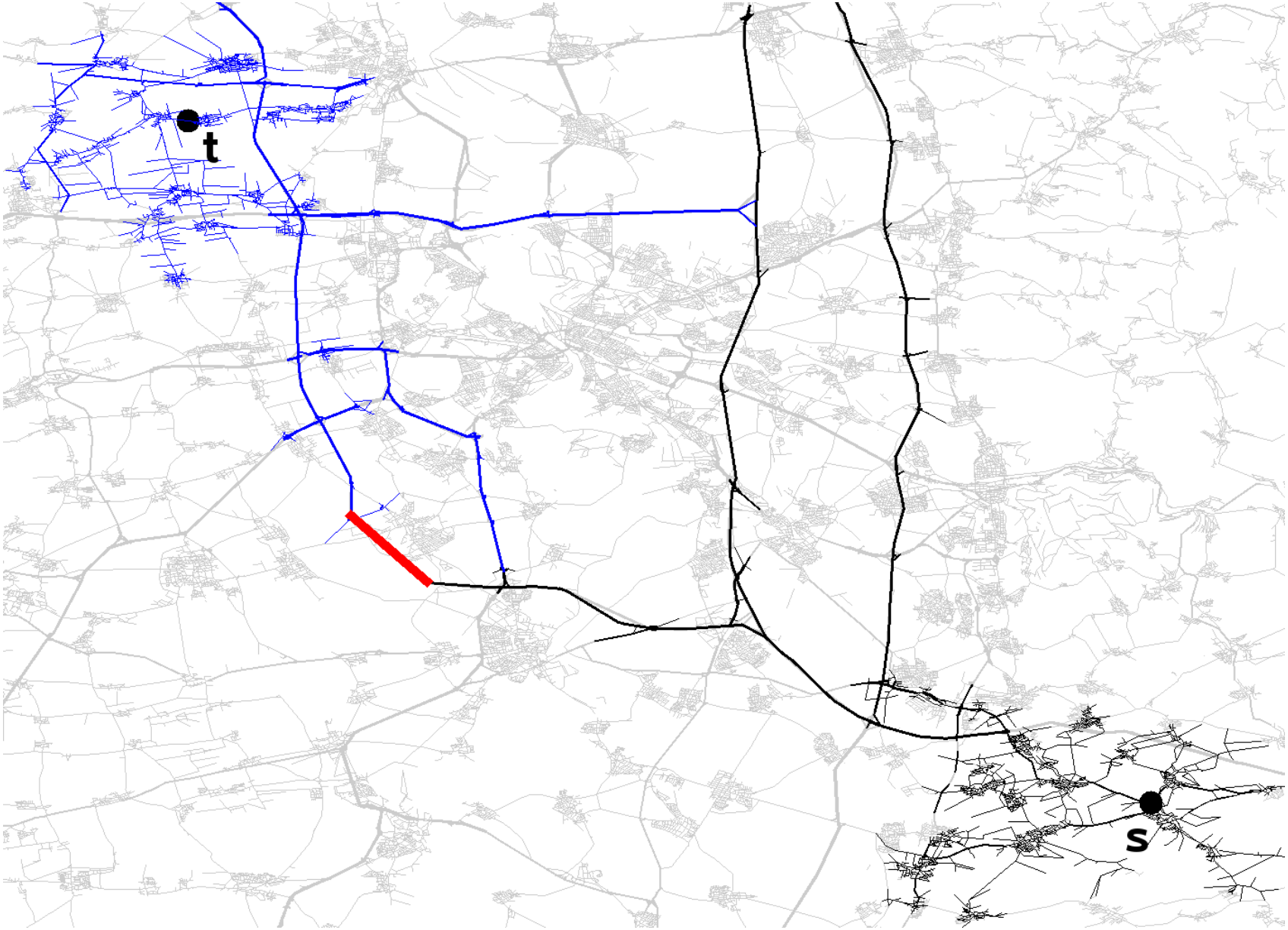
$$\text{len}_\pi(u, v) = \text{len}(u, v) - \pi(u) + \pi(v) \geq 0$$

- durch Erhöhen der Kantengewichte wird dies nicht verletzt
- durch Staus können Reisezeiten nicht unter Initialwert fallen

Idee:

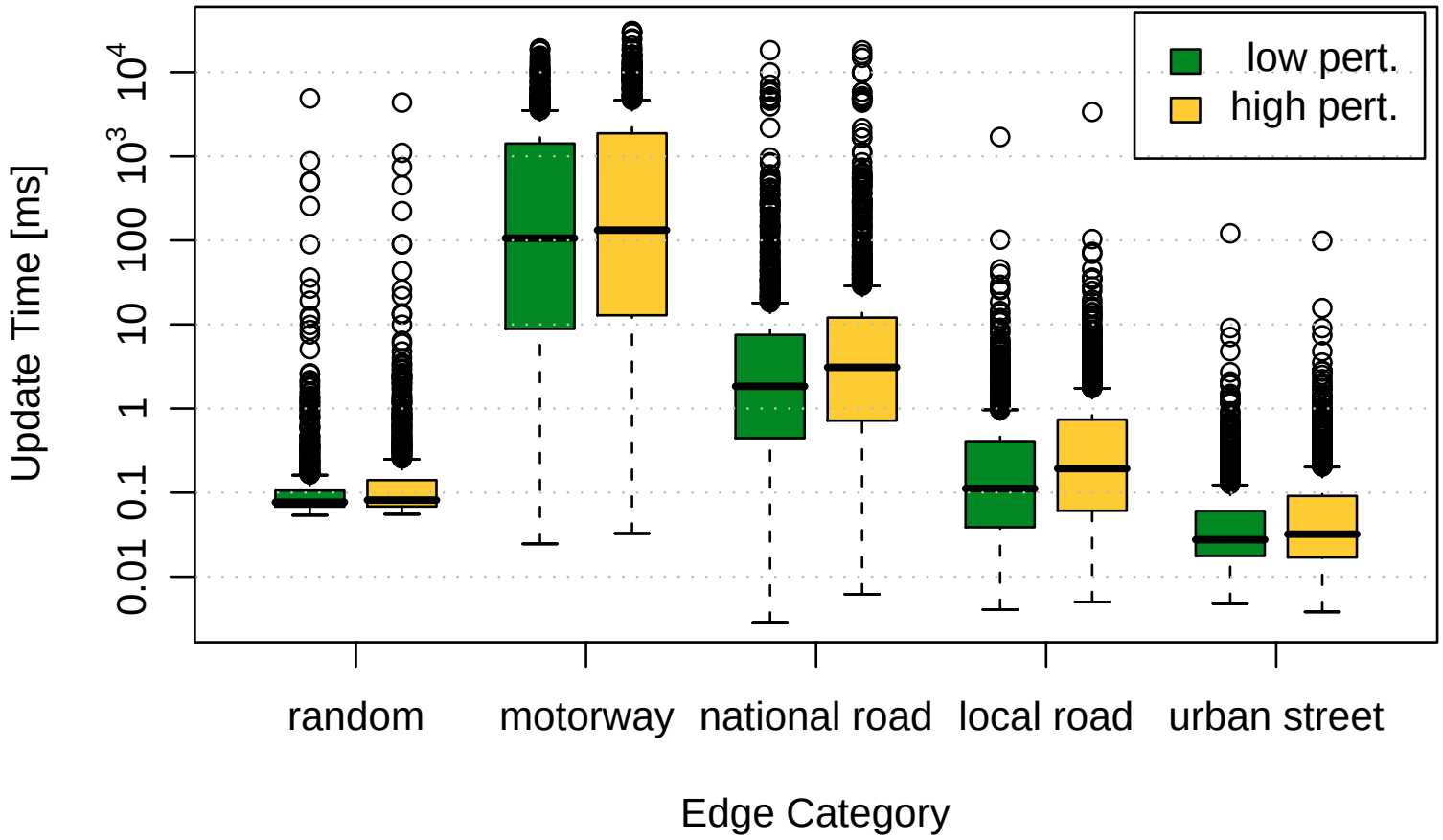
- Vorberechnung auf staufreiem Graphen
- Suchanfragen ohne Aktualisierung
- korrekt aber eventuell langsamere Anfragezeiten

Beispiel



Aktualisierung der Vorberechnung

Aktualisieren von 32 kürzeste-Wege Bäumen für 16 Landmarken:



No-update Variante: Typ des Updates

- Zufallsanfragen nach 1 000 updates (x2, x10 in Klammern)
- verschiedene Straßenkategorien

road type	affected queries	16 landmarks search space		32 landmarks search space	
All	7.5 %	74 700	(77 759)	41 044	(43 919)
urban	0.8 %	74 796	(74 859)	40 996	(41 120)
local	1.5 %	74 659	(74 669)	40 949	(40 995)
national	28.1 %	74 920	(75 777)	41 251	(42 279)
motorway	95.3 %	97 249	(265 472)	59 550	(224 268)

Beobachtung:

- nur Autobahnen haben Einfluß auf Suchraumgröße

No-update Variante: Anzahl Updates

- Zufallsanfragen nach Autobahn updates (x2, x10 in Klammern)
- verschiedene Anzahl Updates

updates	affected queries	16 landmarks search space		32 landmarks search space	
100	39.9 %	75 691	(91 610)	41 725	(56 349)
200	64.7 %	78 533	(107 084)	44 220	(69 906)
500	87.1 %	86 284	(165 022)	50 007	(124 712)
1 000	95.3 %	97 249	(265 472)	59 550	(224 268)
2 000	97.8 %	154 112	(572 961)	115 111	(531 801)
5 000	99.1 %	320 624	(1 286 317)	279 758	(1 247 628)
10 000	99.5 %	595 740	(2 048 455)	553 590	(1 991 297)

Beobachtung:

- $\leq$ 1000 gut verkraftbar

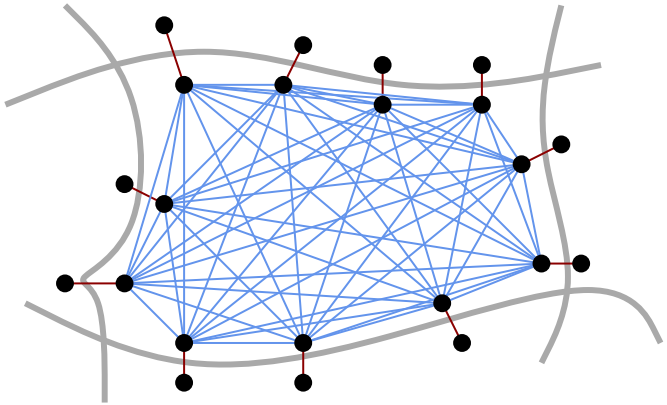
Customizable Route Planning

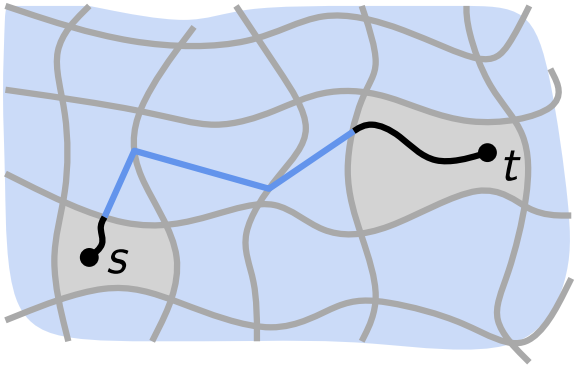
Idee:

- partitioniere Graphen
- Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

Optimierung: multiple Level

Update der Vorberechnung

Beobachtung:

- Shortcuts (Cliquenkanten) können invalidiert werden
- Shortcuts repräsentieren Pfade innerhalb der Zelle

Update:

- identifiziere Zelle in der die aktualisierte Kante liegt für jedes Level der Multi-Level Partition
- wiederhole (bottom-up) Preprocessing für diese Zellen
- dauert weniger als 10 ms

Mobiles Szenario

Beobachtung:

- Update invalidiert Zellen
- erhöht eventuell einige Gewichte
- restliche Shortcuts korrekt

Query:

- Suche muss bei invalidierten Zellen absteigen
- jeder Abstieg erhöht Suchraum um ca. 50%

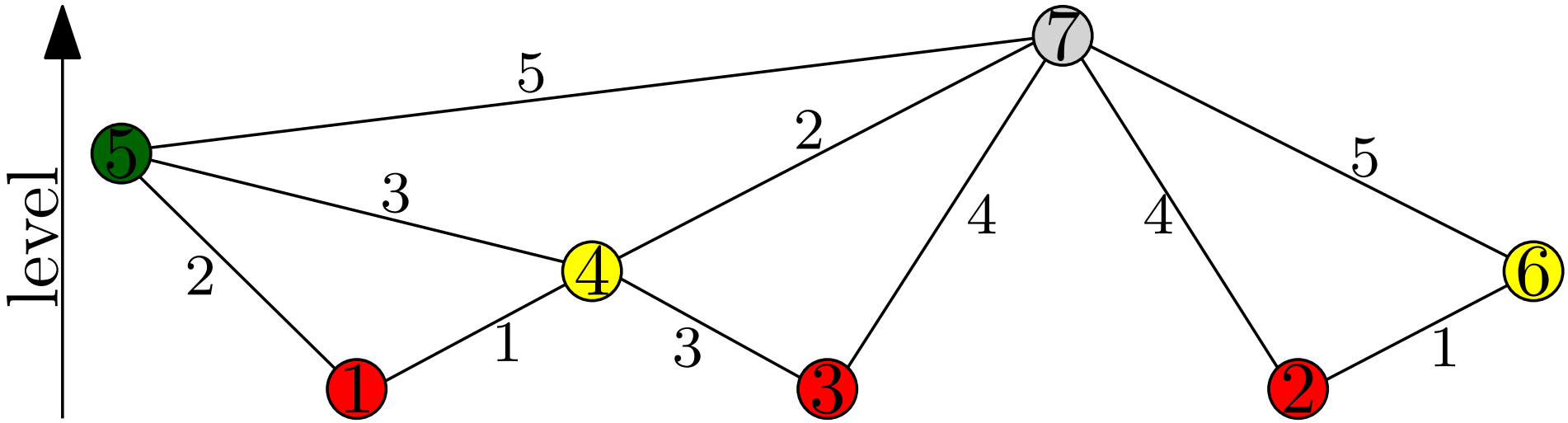
Optimierung:

- markiere invalidierte Zellen (aber kein Update)
- führe Anfrage aus
- wenn der Pfad nicht die aktualiesierte Kante enhält $\Rightarrow$ fertig
- sonst Suche mit Abstieg

Contraction Hierarchies

preprocessing:

- ordne Knoten nach Wichtigkeit
- bearbeite in der Reihenfolge
- füge Shortcuts hinzu
- Levelzuordnung (ca. 150 in Strassennetzwerken)



Update der Vorberechnung I

Beobachtung:

- Knotenordnung ist heuristik
- also behalte Ordnung bei
- geänderte Kante (u, v) kann zu Shortcuts beitragen
- aber auch Teil von Zeugensuchen gewesen sein
- Frage: welche Knoten müssen neu kontrahiert werden (mit Zeugensuche)

Update der Vorberechnung II

Idee:

- wiederhole Kontraktion für Knotenmenge U
- füge alle aufwärts erreichbaren Knoten von u, v zu U
- speicher für jede Kante e auf welchen Zeugensuchen sie benutzt wurde. Die zugehörigen Knoten speicher in A_e. Kann einfach während Vorberechnung gespeichert werden.
- füge alle aufwärts erreichbaren Knoten von A_e (und aller A_f für alle Shortcuts f, die e enthalten) zu U
- wiederhole Kontraktion für alle Knoten U bezüglich Originalordnung

Beobachtung:

- erhöht Speicherverbrauch von CH deutlich
- Updatezeit abhängig von Art der Kante

Mobiles Szenario

Beobachtung:

- Kontraktion von Knoten zu teuer

Idee:

- identiziere Knotenmenge U wie zuvor
- erlaube Abstieg an Kanten (u, v) mit $u, v \in U$
- iterativer Ansatz (wie bei CRP) auch hier möglich

Weitere Techniken

Arc-Flags:

- Aktualisierung aller Bäume nötig
- momentan keine vernünftige Update-Routine vorhanden

Transit-Node Routing

- schwierig
- Tabellen und Access-Nodes können sich ändern

HubLabels

- halte CH vor
- nach Update: aktualisierung alles Labels
- nicht getestet: nicht alle Labels müssen aktualisiert werden

Zusammenfassung

- Stau erhöht Kantengewicht temporär
- Anpassung Landmarken, CRP trivial
- Anpassung CH machbar
- sehr schwierig bei anderen Techniken

Anmerkungen

- es sollen nur “nahe” Staus beachtet werden
- viele Staus verhersehbar (nächste Vorlesungen)

Szenario

Routenplanung auf Mobilen Geräten

2 Varianten:

- server-basiert
  - Gerät sendet request und kriegt komplette Route übertragen
  - mit aktuellen Stauinformationen
  - Was, wenn der Fahrer sich verfährt?
  - Was passiert in Funklöchern
- direkt auf Gerät
  - wie Kartendaten aktualisieren?
  - Staus?
  - limitierte Rechenpower

Ein Kompromiss

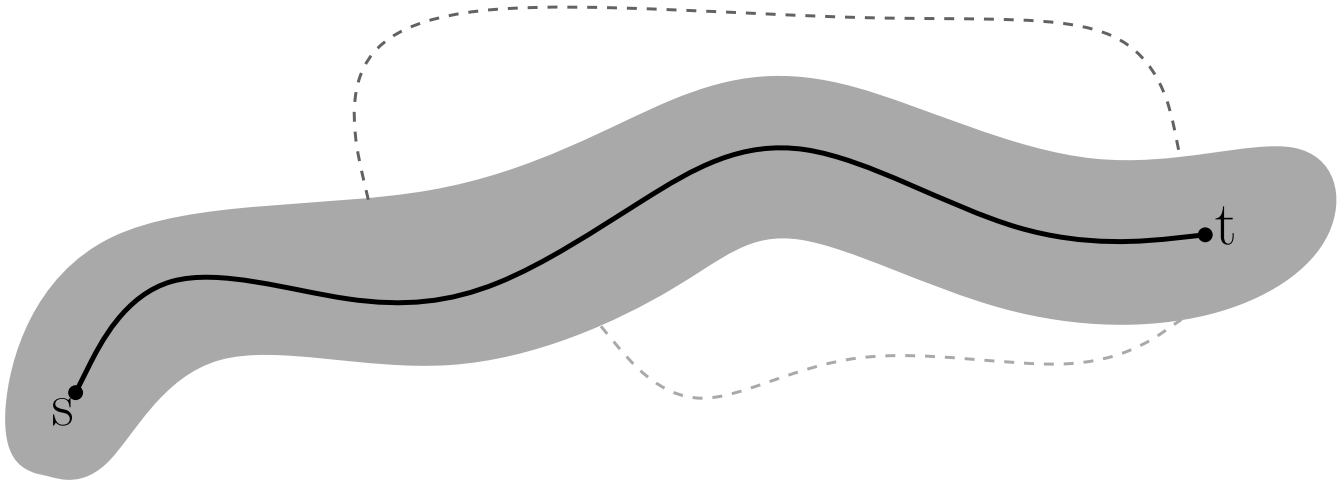
Idee:

- weiterhin server-basiert
- während ursprünglicher Anfrage haben wir Zugang zum Server
- aber sende mehr als die Route
- (Visualisierungsdaten auf dem Gerät)

Herausforderung:

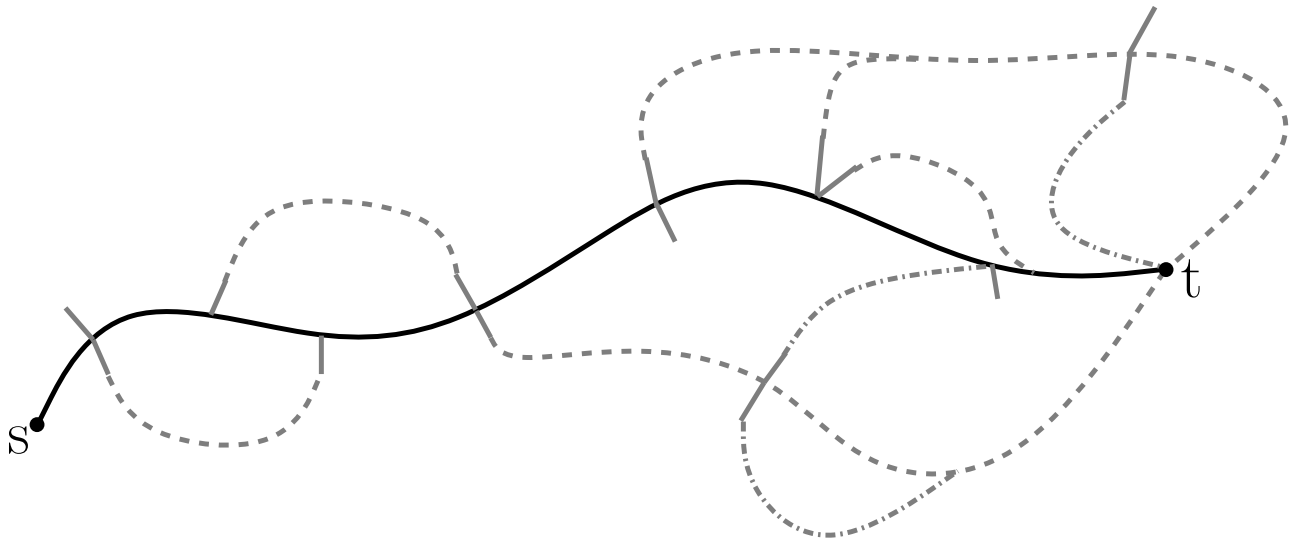
- tradeoff zwischen Robustheit und Menge an Information, die wir senden müssen
- tradeoff zwischen zwei Extrema
  - kürzesten Weg (wenig robust, klein)
  - voller kürzeste Wege Baum T (sehr robust, aber zuviele Daten)
- finde “sinnvollen” Teil von T?

Perimeter-Korridor



- wähle maximale Abweichzeit d
- berechne alle Knoten N, die maximal d weit weg vom kürzesten Weg sind
- Korridor: alle Knoten, die auf den kürzesten Wegen von N nach t liegen

Abbiege-Korridor



- wähle maximale Abweichen k
- starte von kürzestem Weg und füge ihn zu Korridor C hinzu
- wiederhole k mal
  - bestimme Nachbarknoten D zu C
  - füge alle kürzesten Wege von D zu t zu C hinzu

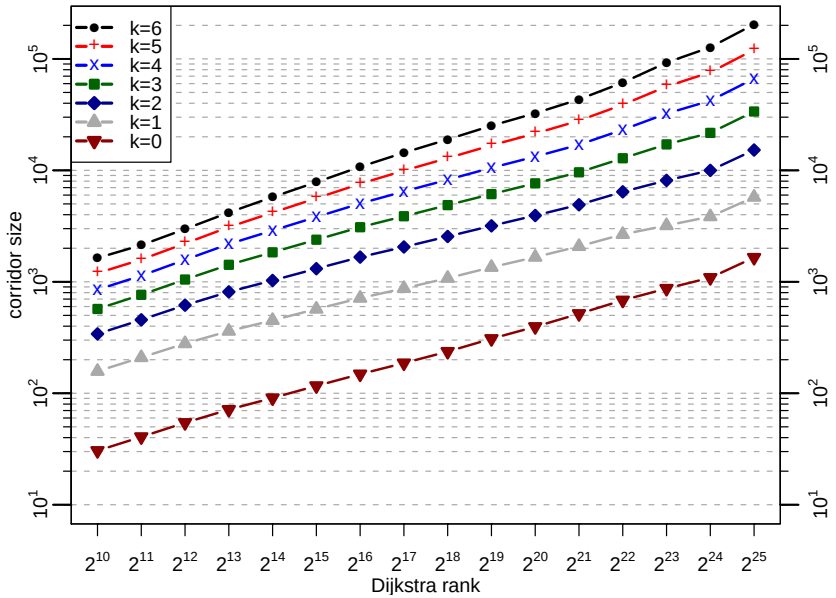
Evaluation

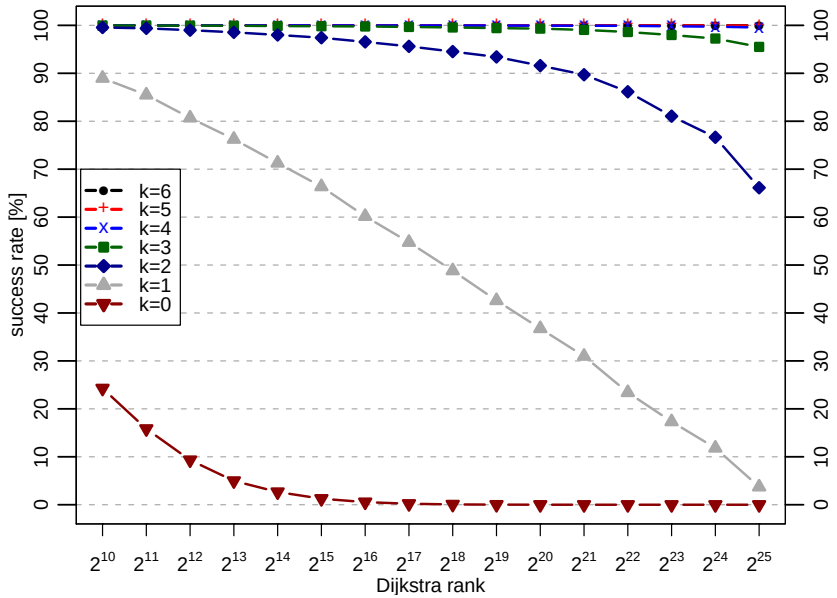
Setup:

- Europa
- Zufallsfahrer DD(x): macht an jeder Kreuzung eine falsche Abbiegung mit Wahrscheinlichkeit x

turn corridor			perimeter corridor		
	size	success [%]	τ	size	success [%]
k	$\lvert V \rvert$	DD(5%)	[s]	$\lvert V \rvert$	DD(5%)
0	1 351	0.0	0	1 351	0.0
1	4 835	10.3	10	6 223	7.0
2	12 204	73.2	20	7 689	8.7
3	25 892	96.8	30	9 570	10.6
4	50 271	99.7	50	14 284	15.0
5	88 742	100.0	100	31 547	28.2
6	148 370	100.0	300	209 513	79.6

Ergebnisse

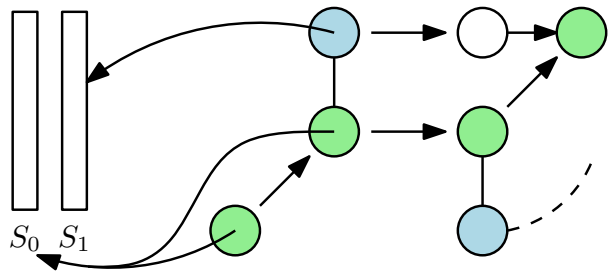




Beobachtungen

- Korridorgrösse verdoppelt sich nach jeder Iteration
- nahe 100% Erfolgsquote für $k = 3$

Generelles Vorgehen



- wir bauen k-turn Korridor C auf
- verwalte zwei Stacks (S_0, S_1)
- S_0 mit s initialisiert, S_1 leer
- führe $k + 1$ Iterationen durch
  - bearbeite alle Knoten in S_0:
    - nimm u vom Stack und füge u zu C hinzu
    - scanne alle ausgehenden Kanten (u, v) mit $v \neq C$
    - wenn v auf dem kürzesten u–t Weg, füge v zu S_0
    - sonst zu S_1
  - $S_0 = S_1$, $S_1 = \emptyset$

Beobachtungen:

- am Ende der Iteration i ist C der $i - 1$-turn Korridor
- und S_1 enthält die Nachbarknoten zu C

Berechnunsansätze

- benutze Dijkstra zum Bestimmen aller Distanzen
- PHAST
- benutze Punkt-zu-Punkt Anfragen um die kürzesten Wege von S_0 zu t am Anfang jeder Iteration zu bestimmen
- benutze One-To-Many Techniken zur Beschleunigung
  - Buckets
  - RPHAST Variante (TCC)

Ergebnisse

Setup:

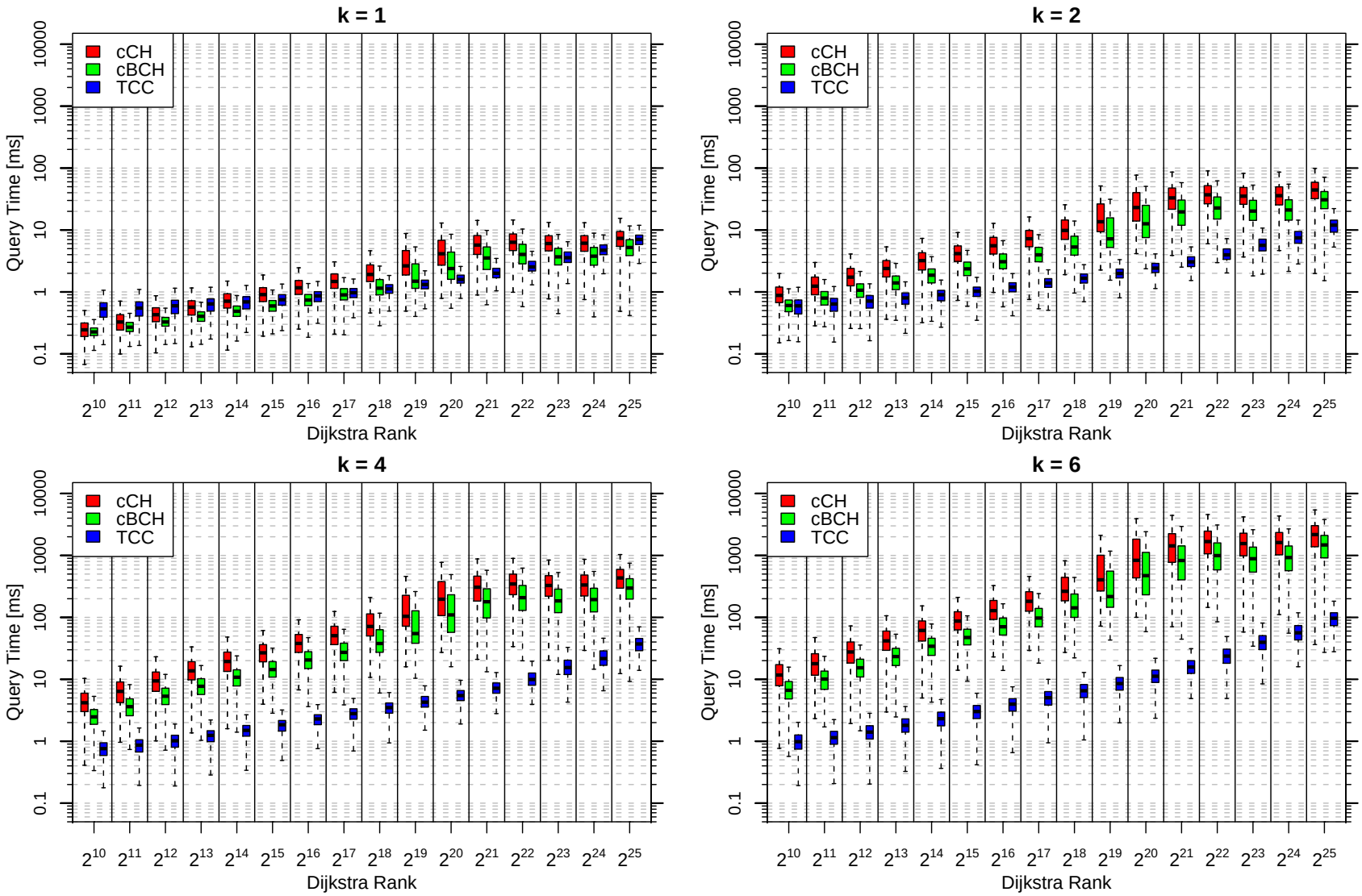
- Europa mit besserer Modellierung (nächstes Mal mehr)
- sequentielle Berechnung

k	cCH	cBCH	cPHAST	TCC
0	0.33	0.34	968.42	0.73
1	7.35	5.26	969.32	5.67
2	44.45	30.96	970.36	9.73
3	156.95	100.81	973.36	16.26
4	382.51	263.00	974.14	27.34
5	795.26	545.97	977.30	42.54
6	1643.34	1132.72	982.83	68.66

Beobachtung:

- TCC macht Berechnung schnell genug

Ergebnisse II



Zusammenfassung

- Korridor erlauben robuste mobile Routenplanung
- flexibler tradeoff zwischen Robustheit und Datenmenge
- können effizient berechnet werden

Literatur I

Dynamische Szenarien:

- Daniel Delling, Dorothea Wagner **Landmark-Based Routing in Dynamic Graphs**
  In: *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, 2007
- Robert Geisberger, Peter Sanders, Dominik Schultes, Christian Vetter
  Exact Routing in Large Road Networks Using Contraction Hierarchies
  In: *Transportation Science*, 2012
- Daniel Delling, Andrew V. Goldberg, Thomas Pajor, Renato Werneck
  Customizable Route Planning
  In: *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, 2011

Literatur II

Korridore

- Daniel Delling, Moritz Kobitzsch, Dennis Luxen, Renato F. Werneck
  Robust Mobile Route Planning with Limited Connectivity
  In: *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*, 2012

Nächste Termine

Karlsruher Institut für Technologie

Montag, 18.6.2012

Mittwoch, 20.6.2012
Montag, 25.6.2012