Algorithmen für Routenplanung – Vorlesung 11

Daniel Delling

Lehrstuhl für Algorithmik I
Institut für theoretische Informatik
Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Letztes Mal: Zeitabhängige Netzwerke

Szenario:

- Historische Daten für Verkehrssituation verfügbar
- Verkehrssituation vorhersagbar
- berechne schnellsten Weg bezüglich der erwarteten Verkehrssituation (zu einem gegebenen Startzeitpunkt)

Zeitabhängige Beschleunigungstechniken

Basismodule:

- 0 bidirektionale Suche
- \+ landmarken
- \+ Kontraktion (hoher Speicherverbrauch)
- \+ arc-flags
- – Table Look-ups

somit folgende Algorithmen gut in zeitabhängigen Szenarien verwendbar

- ≫ ALT
- ≫ Core-ALT
- ≫ SHARC
- ≫ Contraction Hierarchies

Heute: Multikriterielle Wege

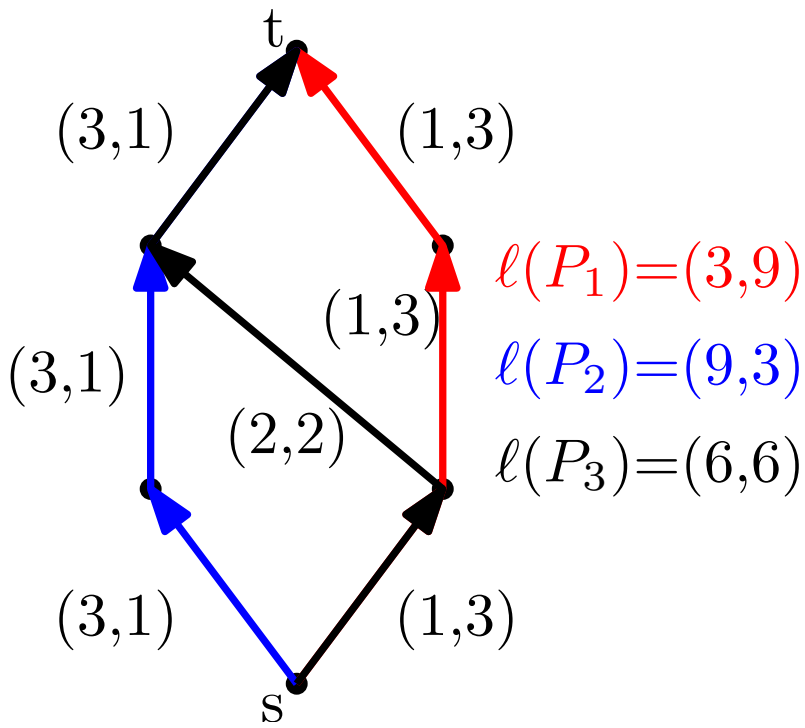
Szenario:

- schnellste Verbindung häufig nicht die Beste
- Anzahl Umstiege, Fahrtkosten, Distanz, etc.
- multikriterielle kürzeste Wege

Multi-Criteria Routes

Idee:

- hänge mehrere Gewichte an die Kanten (z.B.: Reisezeiten, Kosten)
- berechne alle Pareto-optimale Routen zwischen Punkten
  - eine Route ist Pareto-optimal wenn sie nicht von anderen Routen dominiert wird
  - eine Route dominiert eine andere, wenn sie besser in einer Metrik und mindestens so gut in allen anderen



Herausforderungen

- viele Routen zum Ziel

Multikriterieller Dijkstra

Vorgehen:

- verwalte Liste an Labeln an jedem Knoten
- füge Startknoten mit $(0, \ldots, 0)$ in PQ ein
- entferne u von der Queue (key: kleinster Eintrag im Label(?))
- für jede ausgehende Kante (u, v) erzeuge temporäres Label an v
- wenn Label nicht dominiert wird, füge Label zu $\texttt{list}(v)$
- stoppe wenn die Label an t all label in der PQ dominieren

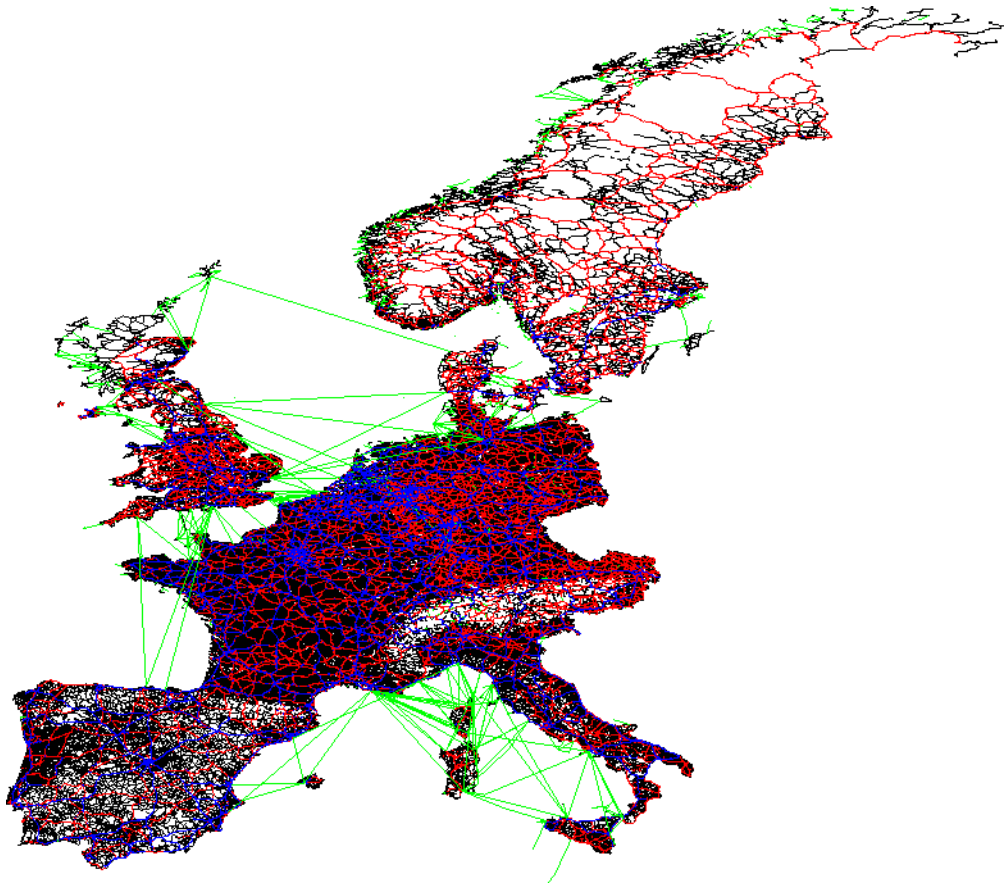
Anmerkungen:

- Knoten können mehr als einmal besucht werden
- Performance hängt massiv von Anzahl der Labels an den Knoten ab

Eingabe

Straßengraphen von:

- Luxemburg
- Karlsruhe
- Niederlande
- Europa



Metriken:

- Fahrzeiten für schnelles Auto
- Fahrzeiten für langsames Auto
- Kosten
- Distanzen
- Unit Metrik

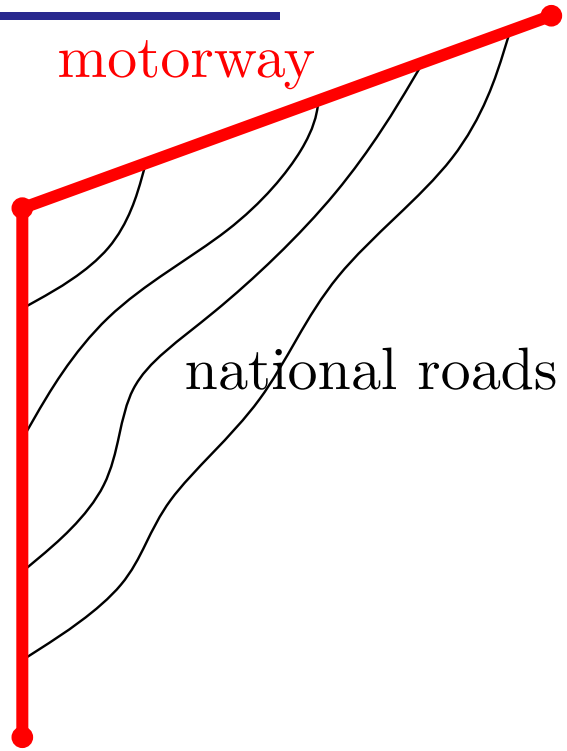
Ähnliche Metriken: Europa

metrics	target labels	#del. mins	time [ms]
fast car (fc)	1.0	442 124	156.44
slow car (sc)	1.0	452 635	151.68
fast truck (ft)	1.0	433 834	139.51
slow truck (st)	1.0	440 273	136.85
fc + st	2.2	1 039 110	843.48
fc + ft	2.0	947 042	698.21
fc + sc	1.2	604 750	369.31
sc + lt	1.9	876 998	577.05
sc + ft	1.7	784 459	474.77
ft + st	1.3	632 052	348.43
fc + sc +st	2.3	1 078 190	956.14
fc + sc + ft	2.0	940 815	751.16
sc + ft +st	1.9	880 236	640.47
fc + sc + ft + st	2.5	1 084 780	1016.39

Verschiedene Metriken

	Luxemburg			Karlsruhe		
metrics	target labels	#del. mins	time [ms]	target labels	#del. mins	time [ms]
fast car (fc)	1.0	15 469	2.89	1.0	39 001	8.2
slow truck (st)	1.0	15 384	2.80	1.0	38 117	7.1
costs	1.0	15 303	2.65	1.0	38 117	6.8
distances	1.0	15 299	2.49	1.0	39 356	7.3
unit	1.0	15 777	2.54	1.0	39 001	8.2
fc + st	2.0	30 026	8.70	1.9	77 778	28.7
fc + costs	29.6	402 232	1704.28	52.7	1 882 930	14909.5
fc + dist.	49.9	429 250	1585.23	99.4	2 475 650	30893.2
fc + unit	25.7	281 894	573.51	27.0	1 030 490	3209.9
costs + dist.	29.6	305 891	581.71	67.2	1 661 600	10815.1

Limitierung des Pareto-Sets



Problem:

- schnelles Auto und Kosten am interessantesten
- Anzahl Routen explodiert
- Europa nicht möglich (wie bei Profilsuchen)

Idee:

- setze Reisezeit als Hauptmetrik
- ändere Dominanzregeln
- erlaube nur solche Routen, die ϵ mal länger als die schnellste ist
- muss auch für Subrouten gelten

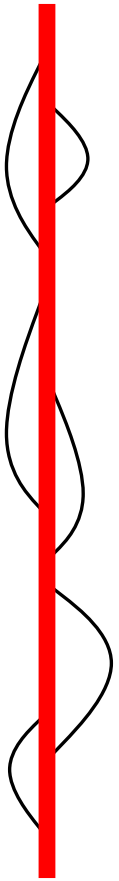
Limitierung des Pareto-Sets II

Problem:

- Ansatz klappt nur, wenn ϵ sehr klein
- uninterestante Routen (geringer Unterschied zum schnellsten Weg)

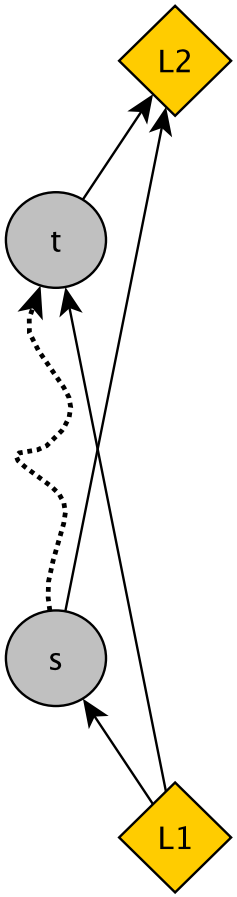
Idee:

- erlaube längere Routen (als schnellste) nur, wenn andere Metriken deutlich besser sind
- damit ϵ höher wählbar
- $\Rightarrow$ sinnvolle Teilmenge der Pareto-Routen

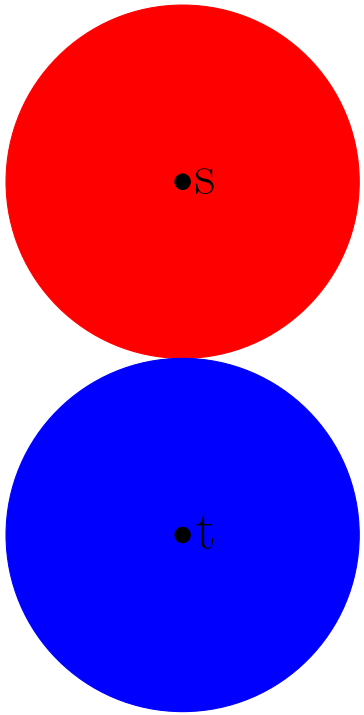


Multikriterielle Beschleunigungstechniken

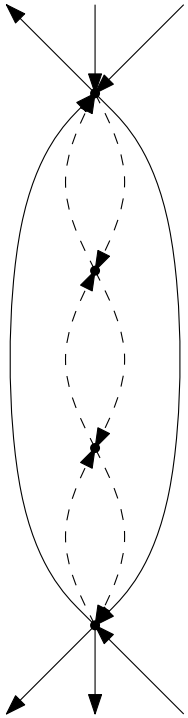
Landmarken



Bidirektionale Suche



Kontraktion



Arc-Flags

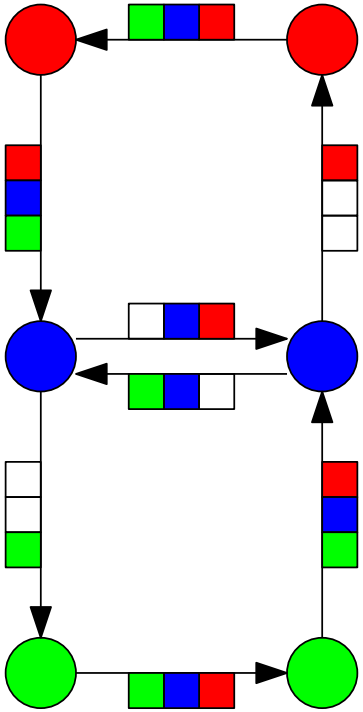
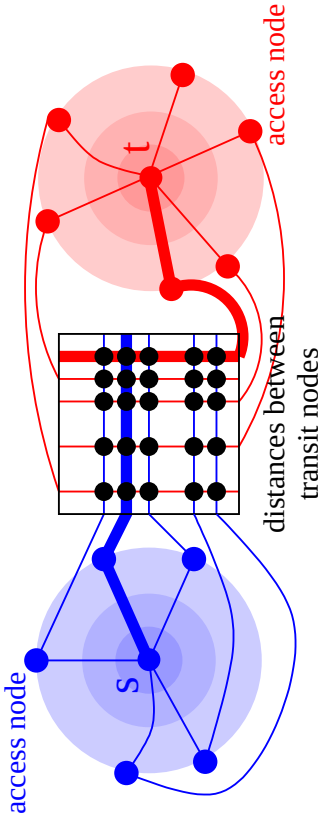


Table-Lookups



Landmarken

Vorberechnung:

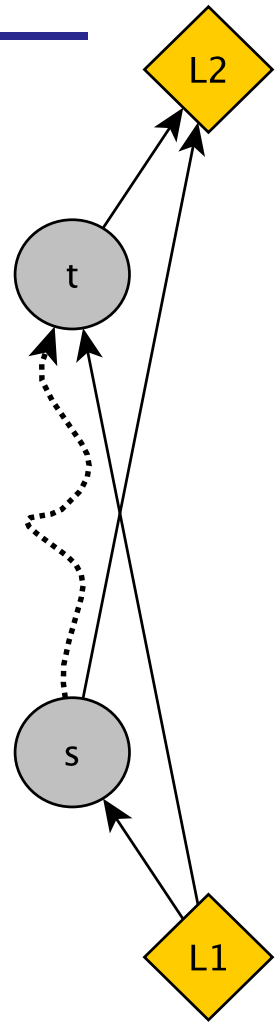
- wähle eine Hand voll (≈ 16) Knoten als Landmarken
- berechne Abstände von und zu allen Landmarken

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



Anpassung

Beobachtung:

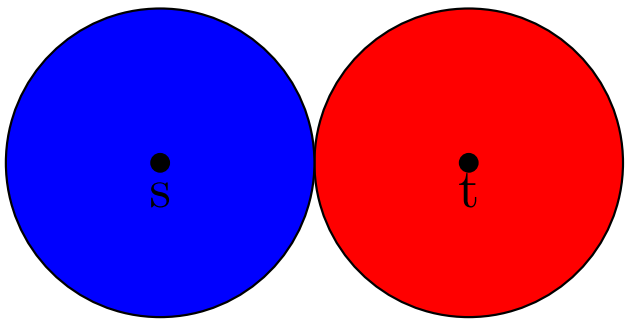
- Korrektheit von ALT basiert darauf, dass reduzierten Kantengewichte größer gleich 0 sind

$$len_\pi(u,v) = len(u,v) - \pi(u) + \pi(v) \geq 0$$

Idee:

- benutze nur eine Metrik zum berechnen der Distanzen
- Abbruchkriterium?

Bidirektionale Suche



- starte zweite Suche von t
- relaxiere rückwärts nur eingehende Kanten
- stoppe die Suche, wenn beide Suchräume sich treffen

Anpassung

Idee:

- rückwärtssuche kein Problem (solange zeitunabhängiges Netzwerk)
- zeitabhängig: Techniken vom letzten Mal

Offenes Problem:

- Abbruchkriterium?
- Kombination aller Elemente aus Vorwärts- und Rückwärtsqueue?

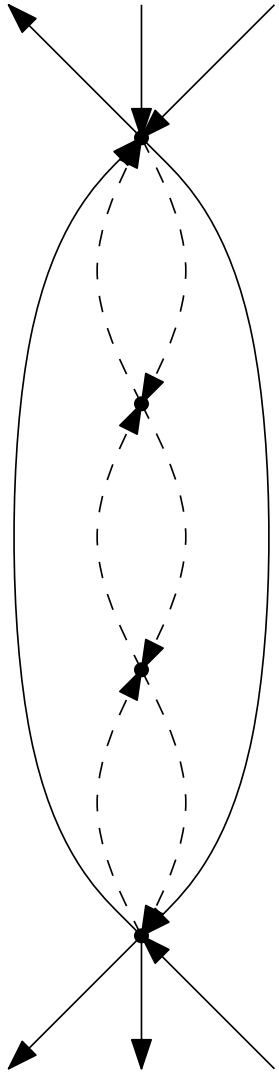
Kontraktion

Knoten-Reduktion:

- entferne diese Knoten iterativ
- füge neue Kanten (Abkürzungen) hinzu, um die Abstände zwischen verbleibenden Knoten zu erhalten

Kanten-Reduktion:

- behalte nur relevante Shortcuts
- lokale Suche während oder nach Knoten-reduktion



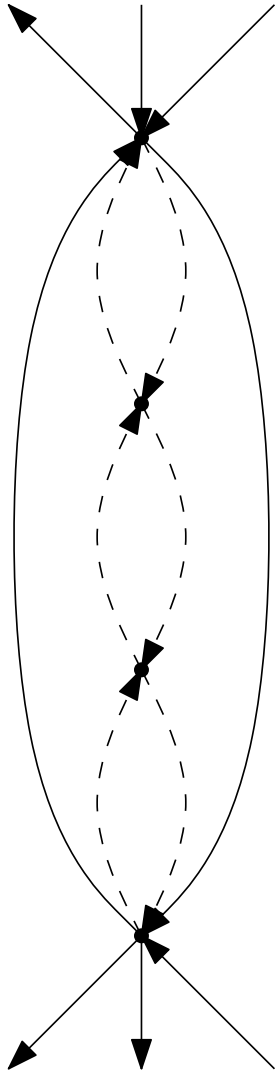
Anpassung Knoten-Reduktion

Beobachtung:

- Verfahren unabhängig von Metrik
- Shortcuts müssen dem Pfade entsprechen

Somit:

- Anpassung ohne Probleme



Anpassung Kanten-Reduktion

unikriteriell:

- lösche Kante (u, v), wenn (u, v) nicht Teil des kürzesten Weges von u nach v ist, also $len(u, v) < d(u, v)$
- lokale Dijkstra-Suche von u

multikriteriell:

- lösche Kante (u, v), wenn (u, v) nicht Teil eines Parteo-Weges von u nach v ist, als (u, v) dominiert wird von mindestens einem Weg
- lokale multi-kriterielle Suche
- Problem: Explosion der Anzahl der Routen
- Lösung: benutze Limitierungen des Pareto-Sets während Vorberechnung

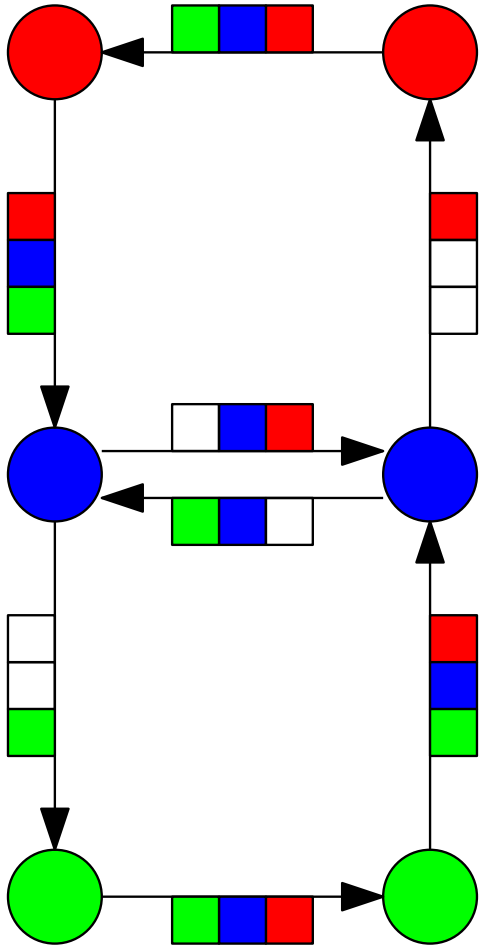
Arc-Flags

Idee:

- partitioniere den Graph in k Zellen
- hänge ein Label mit k Bits an jede Kante
- zeigt ob e wichtig für die Zielzelle ist
- modifizierter Dijkstra überspringt unwichtige Kanten

Beobachtung:

- Partition wird auf ungewichtetem Grahen durchgeführt
- Flaggen müssen allerdings aktualisiert werden



Anpassung

Idee:

- ändere Intuition einer gesetzten Flagge
- Konzept bleibt gleich: Eine Flagge pro Kante und Region
- setze Flagge
  - `multikriteriell`: wenn Kante für einen Pareto-Pfad "wichtig" ist

Anpassung:

- für alle Randknoten b und alle Knoten u:
- Berechne Pareto-Abstände $D(u, b)$
- setze Flagge wenn gilt (u, v) zugehörige Kante eines Pareto-Pfades ist

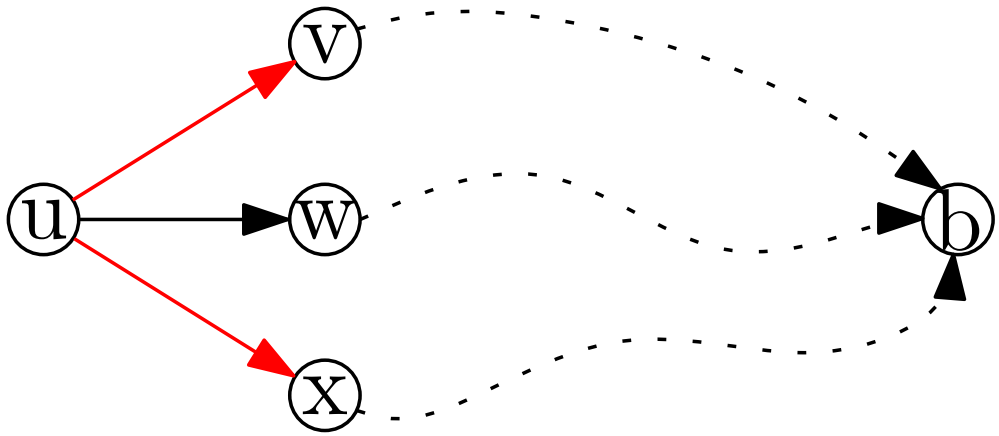
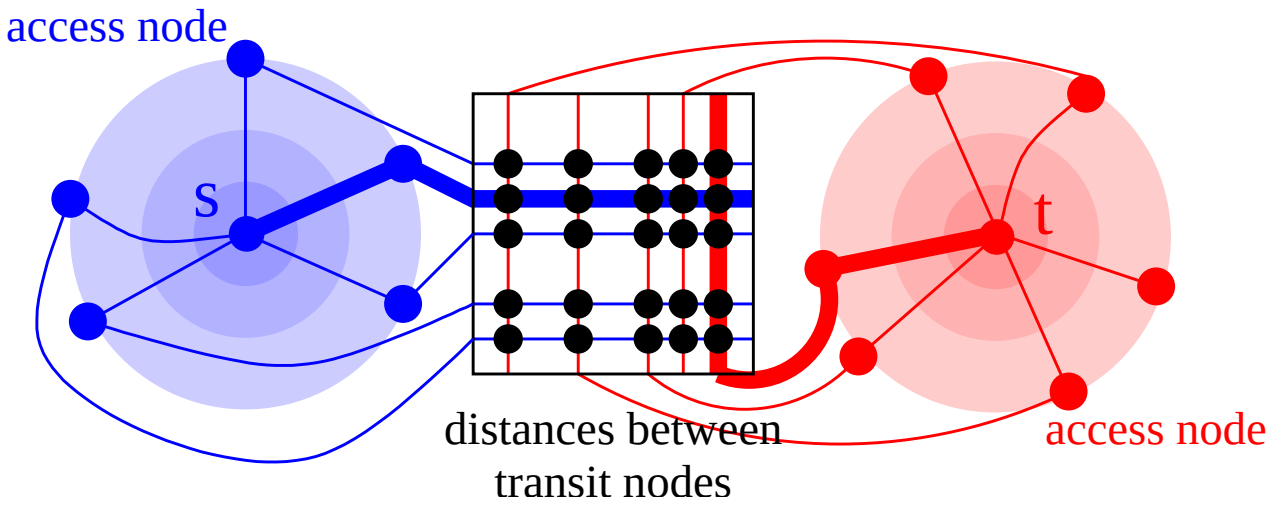


Table-Lookups

Idee:

- speichere Distanztabellen
- nur für "wichtige" Teile des Graphen
- Suchen laufen nur bis zur Tabelle
- harmoniert gut mir hierarchischen Techniken



Anpassung

Beobachtung:

- Distanz-Tabelle muss Pareto-Abstände abspeichern
- massiver Anstieg der Größe der Tabellen
- Pfadstruktur nicht mehr so gutmütig
- deutlich mehr access-nodes?

also:

- Speicherverbrauch deutlich zu groß?

Diskussion Anpassung der Basismodule

Basismodule:

- ? bidirektionale Suche
- ? landmarken
- \+ Kontraktion
- \+ arc-flags
- ? Table Look-ups

somit folgende Algorithmen gute Kandidaten

- ≫ SHARC

unklar:

- ≫ ALT
- ≫ Core-ALT
- ≫ Contraction Hierarchies

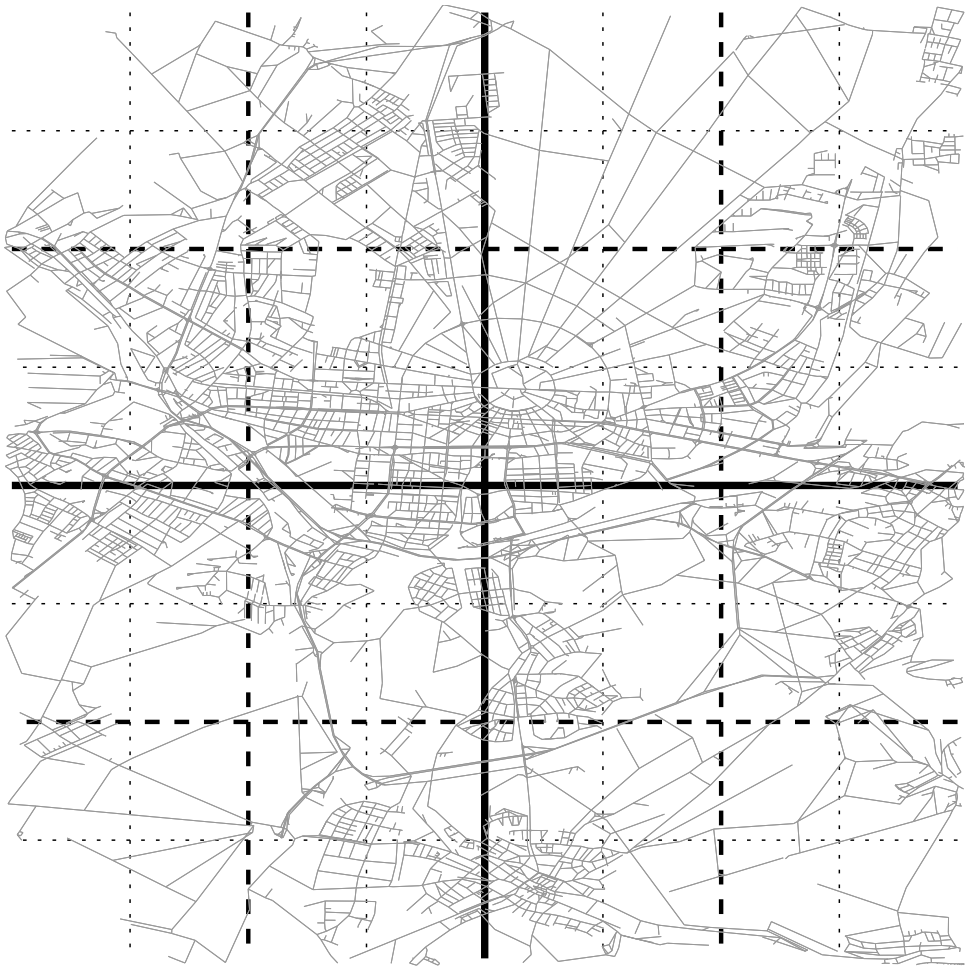
SHARC (Kontraktion, Arc-Flags)

Vorberechnung:

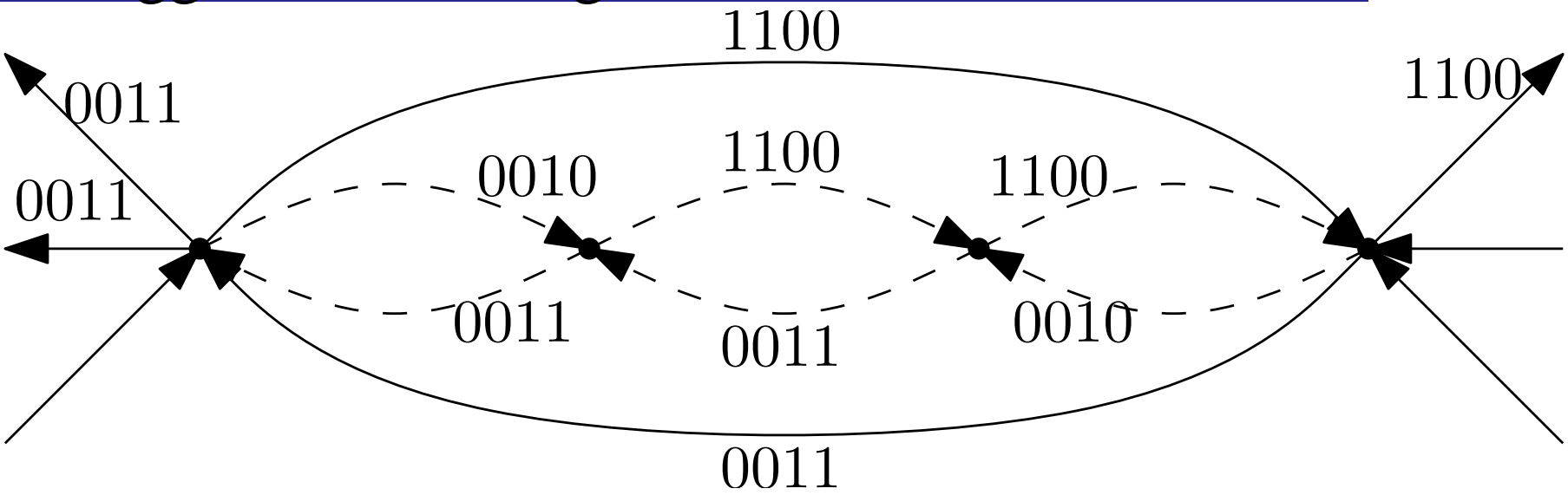
- Multi-Level-Partition
- iterativer Prozess:
  - kontrahiere Subgraphen
  - berechne Flaggen
- Flaggenverfeinerung

Anpassung:

- Kontraktion und Flaggen berechnung anpassen
- Verfeinerung?



Flaggenverfeinerung



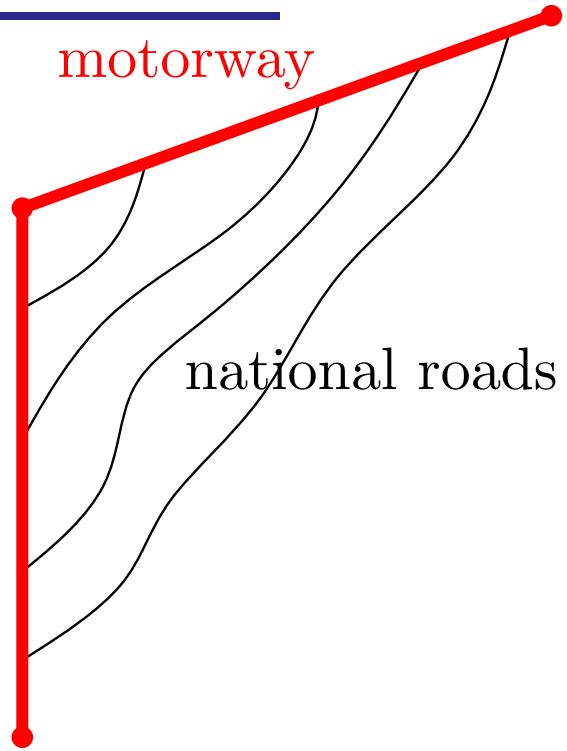
Vorgehen:

- verfeinere Flaggen
- propagiere Flaggen von wichtigen zu unwichtigen Kanten
- uni-kriteriell: mittels lokaler Suche
- multi-kriteriell: mittels lokaler multikriterieller Suchen
- auch hier wieder: effizient nur durch Limitierung des Pareto-Sets

Full Pareto-Set

	Luxemburg					Karlsruhe				
	Prepro	Query				Prepro	Query			
metrics	time [h:m]	target labels	#del. mins	time [ms]	spd up	time [h:m]	target labels	#del. mins	time [ms]	spd up
fast car (fc)	< 0:01	1.0	138	0.03	114	< 0:01	1.0	206	0.04	188
slow truck (st)	< 0:01	1.0	142	0.03	111	< 0:01	1.0	212	0.04	178
costs	< 0:01	1.0	151	0.03	96	< 0:01	1.0	244	0.05	129
distances	< 0:01	1.0	158	0.03	87	< 0:01	1.0	261	0.06	119
unit	< 0:01	1.0	149	0.03	96	< 0:01	1.0	238	0.05	147
fc + st	0:01	2.0	285	0.09	100	0:01	1.9	797	0.26	108
fc + costs	0:04	29.6	4 149	6.49	263	1:30	52.7	15 912	80.88	184
fc + dist.	0:14	49.9	8 348	20.21	78	3:58	99.4	31 279	202.15	153
fc + unit	0:06	25.7	4 923	5.13	112	0:17	27.0	11 319	16.04	200
costs + dist.	0:02	29.6	3 947	4.87	119	1:11	67.2	19 775	67.75	160

Limitierung des Pareto-Sets



Problem:

- schnelles Auto und Kosten am interessantesten
- Anzahl Routen explodiert
- Europa nicht möglich (wie bei Profilsuchen)

Idee:

- setze Reisezeit als Hauptmetrik
- ändere Dominanzregeln
- erlaube nur solche Routen, die ϵ mal länger als die schnellste ist
- muss auch für Subrouten gelten

Results (fast car + costs)

	The Netherlands				Europe			
	Prep	Query			Prep	Query		
ϵ	time [h:m]	target labels	#del. mins	time [ms]	time [h:m]	target labels	#del. mins	time [ms]
0.000	0:01	1.0	452	0.21	0:53	1.0	3 299	2.6
0.001	0:01	1.1	461	0.21	1:00	1.1	3 644	4.1
0.002	0:01	1.2	489	0.22	1:03	1.2	4 340	7.1
0.005	0:01	1.4	517	0.24	1:18	1.4	5 012	11.3
0.010	0:01	1.7	590	0.27	1:58	2.4	9 861	19.2
0.020	0:01	2.2	672	0.32	4:10	5.0	24 540	48.1
0.050	0:02	3.3	1 009	0.51	14:12	23.4	137 092	412.7
0.100	0:04	4.8	1 405	0.82	>24:00	–	–	–
0.200	0:09	7.2	2 225	1.67				
0.500	0:39	12.8	4 227	4.85				
1.000	3:44	20.0	12 481	26.85				
∞	>24:00	–	–	–				

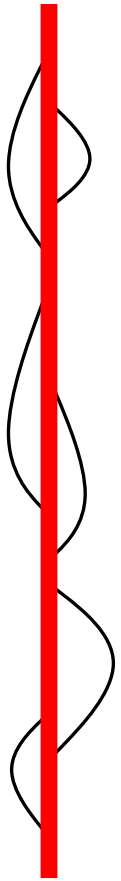
Limitierung des Pareto-Sets II

Problem:

- ≫ Ansatz klappt nur, wenn ϵ sehr klein
- ≫ uninterestante Routen (geringer Unterschied zum schnellsten Weg)

Idee:

- ≫ erlaube längere Routen (als schnellste) nur, wenn andere Metriken deutlich besser sind
- ≫ damit ϵ höher wählbar
- $\Rightarrow$ sinnvolle Teilmenge der Pareto-Routen



Results (fast car + costs)

	Prepro		Query		
	time	space	target	#del.	time
γ	[h:m]	[B/n]	labels	mins	[ms]
1.100	0:58	19.1	1.2	2 538	1.8
1.050	1:07	19.6	1.3	3 089	2.2
1.010	1:40	20.4	1.7	4 268	3.2
1.005	2:04	20.6	1.9	5 766	4.1
1.001	3:30	20.8	2.7	7 785	6.1
1.000	7:12	21.3	5.3	19 234	35.4
0.999	15:43	22.5	15.2	87 144	297.2
0.995	>24:00	–	–	–	–

Beobachtung:

- ≫ Europa möglich
- ≫ vernünftige Anzahl Routen
- ≫ gute Anfragenzeiten

Daniel Delling – Algorithmen für Routenplanung – Vorlesung 11
Lehrstuhl für Algorithmik I
Institut für theoretische Informatik


Eisenbahn

Eingabe:

- Deutscher Fern- und Regionalverkehr von 2008.
- 8817 stationen,
- 40034 Züge
- 392 Fußkanten
- 1 135 479 elementare Verbindungen

Ergebnisse

Query variant	average CPU time in s	average # pq-min operations	speed-up factor over base	
			CPU time	pq-min operations
base	6.14	122,441	1.00	1.00
base+lb	3.62	72,875	1.70	1.68
arc-flags	2.97	32,146	2.07	1.91
SHARC	2.30	40,503	2.67	3.02
SHARC+goal	1.46	30,291	4.21	4.04
greedy arc-flags	2.37	50,428	2.59	2.43
greedy SHARC	1.88	32,145	3.27	3.81
greedy SHARC+goal	1.18	24,117	5.19	5.08

Zusammenfassung Multikriterielle Beschleunigungst.

Basismodule:

- ? bidirektionale Suche
- ? landmarken
- \+ Kontraktion
- \+ arc-flags
- ? Table Look-ups

angepasst haben wir:

- ≫ SHARC
- ≫ aber nur effizient mit Einschränkung des Pareto-Sets

offen:

- ≫ andere Beschleunigungstechniken (?)
- ≫ anderes Pruning der Pareto-Pfade (?)
- ≫ Eisenbahnnetze (?)

Literatur

Multikriterielle Suche:

- Delling 09

Anmerkung:

- wird auf der Homepage verlinkt