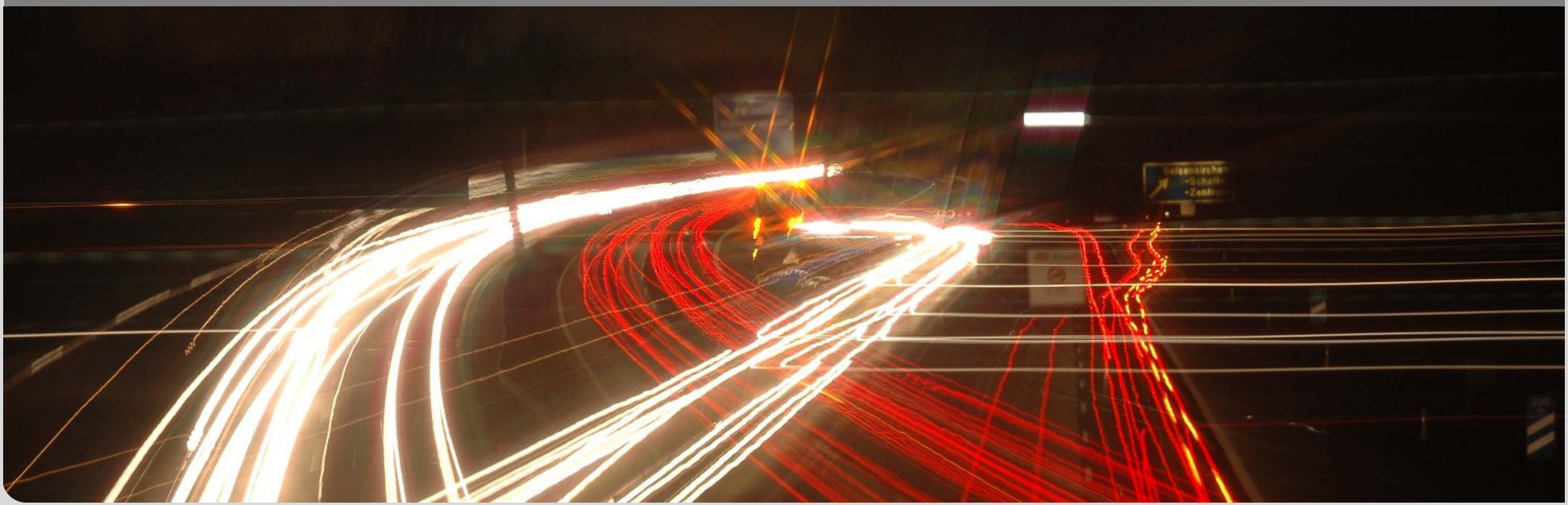
Karlsruher Institut für Technologie

Algorithmen für Routenplanung

15. Vorlesung, Sommersemester 2012

Daniel Delling | 20. Juni 2012

MICROSOFT RESEARCH SILICON VALLEY



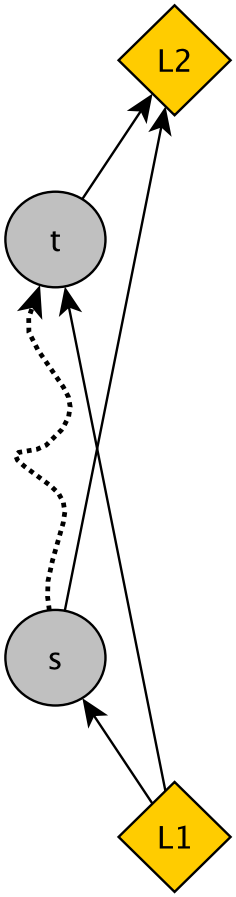
Letztes Mal

Karlsruher Institut für Technologie

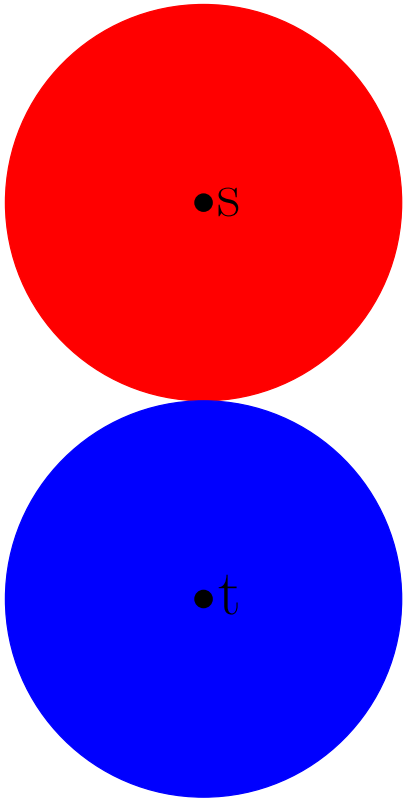
Einführung zeitabhängige Routenplanung

Heute: Beschleunigungstechniken

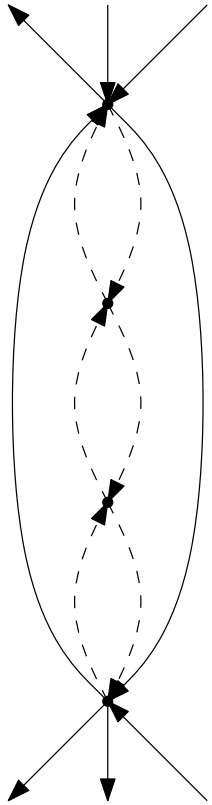
Landmarken



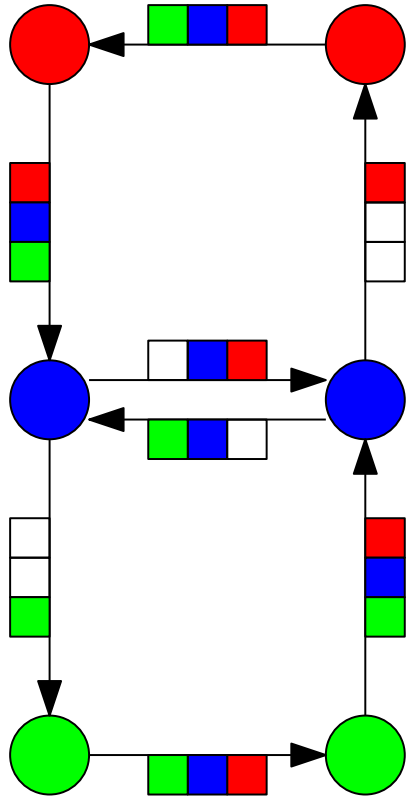
Bidirektionale Suche



Kontraktion



Arc-Flags



Landmarken

Vorberechnung:

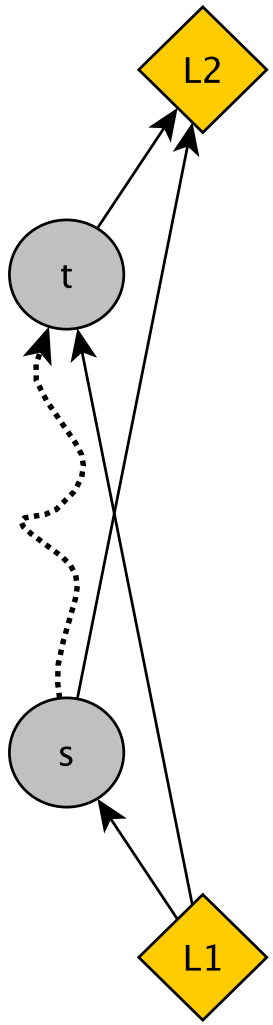
- wähle eine Hand voll (≈ 16) Knoten als Landmarken
- berechne Abstände von und zu allen Landmarken

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 reduzierte Kosten größergleich 0 sind

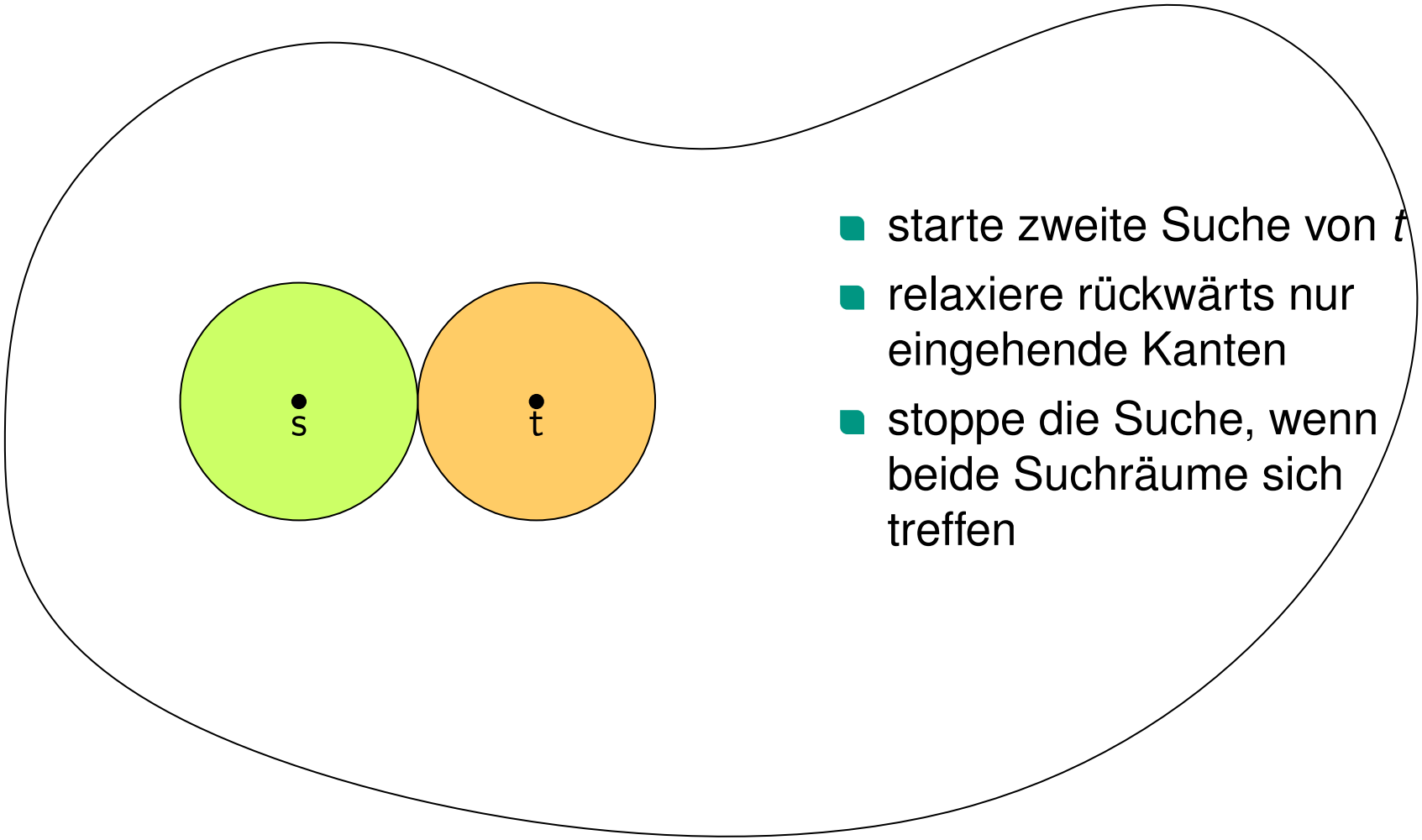
$$\text{len}_\pi(u,v) = \text{len}(u,v) - \pi(u) + \pi(v) \overset{!}{\geq} 0$$

- durch Erhöhen der Kantengewichte wird dies nicht verletzt

Somit:

- Definiere lowerbound-Graph $\underline{G} = (V, E, \underline{\text{len}})$ mit $\underline{\text{len}} := \min \text{len}$
- Vorberechnung auf lowerbound-Graph
- korrekt aber eventuell langsamere Anfragezeiten

Bidirektionale Suche



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

Anpassung

Zeitanfragen:

- Ankunft unbekannt $\Rightarrow$ Rückwärtsuche?
- Rückwärtssuche nur zum Einschränken der Vorwärtssuche benutzen
- je nach Beschleunigungstechnik verschieden $\rightsquigarrow$ später

Profilanfragen:

- Anfrage zu allen Startzeitpunkten
- somit Rückwärtsuche kein Problem
- μ temporäre Abstandsfunktion
- breche ab, wenn $\overline{\mu} \leq \text{minKey}(\overrightarrow{Q}) + \text{minKey}(\overleftarrow{Q})$

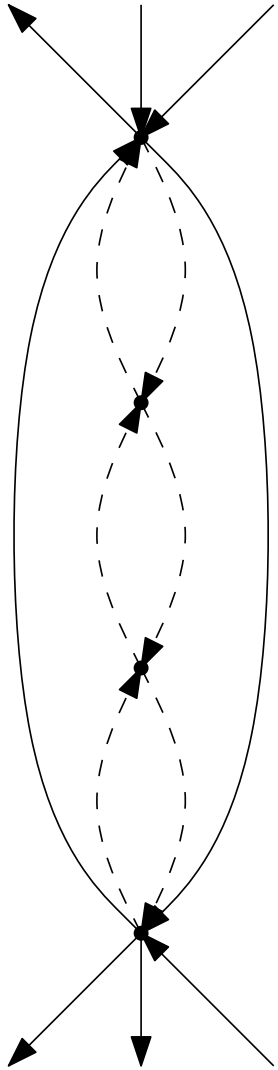
Kontraktion

Knoten-Reduktion:

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

Zeugensuche:

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



Anpassung Zeugensuche

Zeitunabhängig:

- Kante (u, v) nicht nötig, wenn (u, v) nicht Teil des kürzesten Weges von u nach v ist, also $\text{len}(u, v) > d(u, v)$
- lokale Dijkstra-Suche von u

Zeitabhängig:

- Kante (u, v) nicht nötig, wenn (u, v) nicht Teil eines kürzesten Wege von u nach v ist, also $\text{len}(u, v) > d_*(u, v)$
- lokale Profilsuche
- Problem: deutlich langsamer

Idee:

- lösche zunächst Kanten (u, v) für die $\underline{\text{len}(u, v)} > \overline{d_*(u, v)}$ gilt
- danach lokale Profilsuche

Korridorsuche

Idee:

- führe zunächst zwei Dijkstra-Suchen mit $\underline{\text{len}}$ und $\overline{\text{len}}$ durch
- relaxiere dann nur solche Kanten (u, v), für die $\underline{d(s,u)} + \underline{\text{len}(u,v)} \leq \overline{d(s,v)}$ gilt

Anmerkung:

- kann auch zur Beschleunigung einer s-t Profil-Suche genutzt werden

Approximation der Shortcuts

Problem:

- hoher Speicherbedarf der Shortcuts (Straße)

Ideen:

- Shortcuts on-the-fly entpacken und dann Gewicht des Pfades berechnen
- speichere Approximationen der Funktionen, führe dann Korridorsuche bei Query auf Originalgraphen durch
- durch speichern von Approximationen genauer

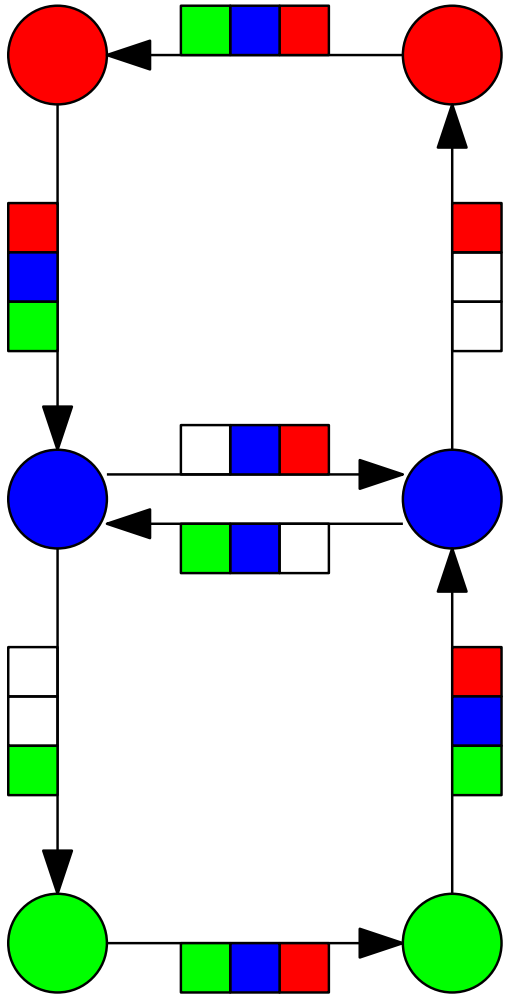
Arc-Flags

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



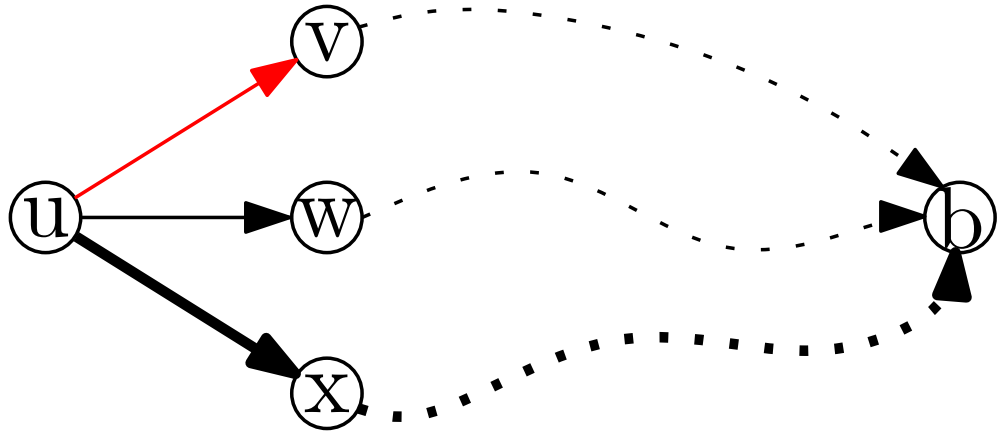
Anpassung

Idee:

- ändere Intuition einer gesetzten Flagge
- Konzept bleibt gleich: Eine Flagge pro Kante und Region
- setze Flagge wenn Kante mindestens ein mal am Tag “wichtig” ist

Anpassung:

- für alle Randknoten b und alle Knoten u:
- Berechne Abstandsfunktion $d_*(u, b)$
- setze Flagge wenn gilt $\text{len}(u, v) \oplus d_*(v, b) \not\succ d_*(u, b)$



Approximation Arc-Flags

Beobachtung:

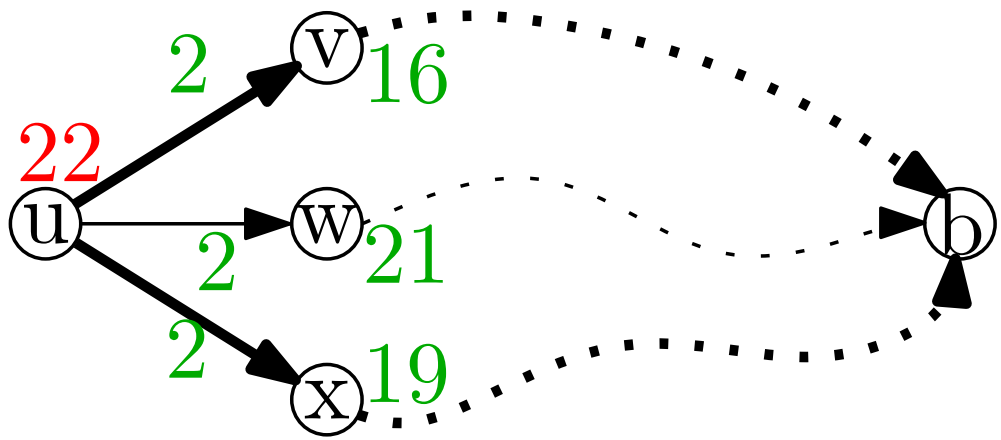
- viele Interpolationspunkte (Straße)
- Berechnung der Abstandsfunktionen ist sehr zeitintensiv
- Laufzeit stark abhängig von der Komplexität der Funktionen

Idee:

- benutze über- and Unterapproximation

$\Rightarrow$ schnellere Vorberechnung, langsamere Anfragen

$\Rightarrow$ aber immer noch korrekt



Heuristische Flaggen

Idee:

- führe von jedem Randknoten K Zeitanfragen aus
- mit fester Ankunftszeit
- setze Flagge, wenn Kante auf einem dem Bäume eine Baumkante ist

Beobachtungen:

- Flaggen eventuell nicht korrekt
- ein Pfad wird aber immer gefunden
- Fehlerrate?

Anpassung der Basismodule

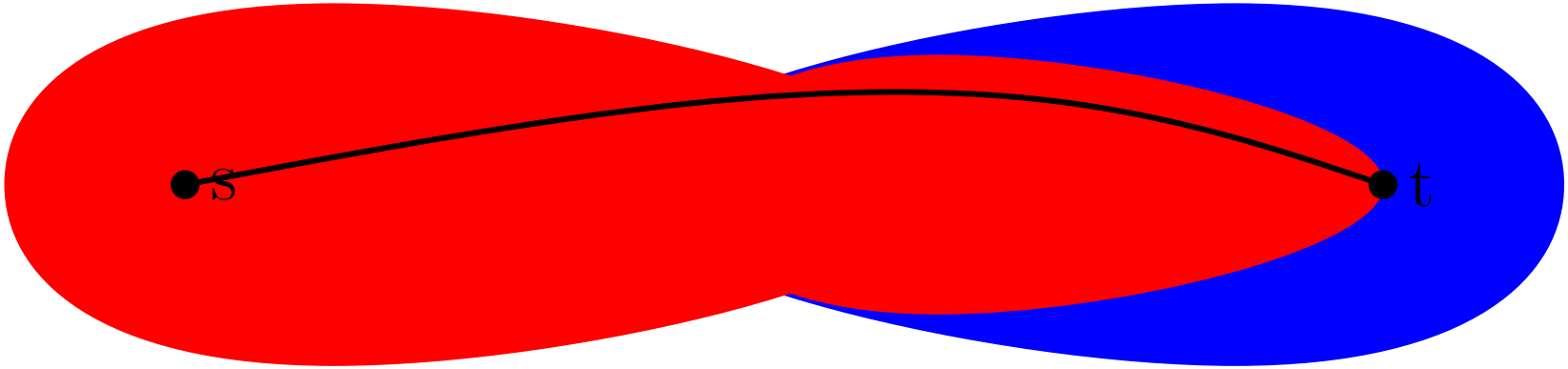
Basismodule:

0 Bidirektionale Suche
+ Landmarken
+ Kontraktion
+ Arc-Flags

Somit sind folgende Algorithmen gute Kandidaten

- ALT
- Core-ALT
- SHARC
- Contraction Hierarchies
- MLD

Bidirektionaler zeitabhängiger ALT



Idee - Drei Phasen:

1. Vorwärts zeitabhängig, Rückwärtssuche benutzt Minima der Funktionen. Fertig wenn Suchen sich treffen. Berechne zeitabhängige Distanz μ (durch Auswerten des gefundenen Weges).
2. Rückwärtssuche arbeitet weiter bis $\text{minKey}(\overleftarrow{Q}) > \mu$
3. Vorwärtssuche arbeitet weiter bis t abgearbeitet worden ist und besucht nur Knoten, die die Rückwärtssuche zuvor besucht hat

Approximation

Beobachtung:

- Phase 2 läuft recht lange weiter, bis $\text{minKey}(\overleftarrow{Q}) > \mu$ gilt
- insbesondere dann schlecht, wenn die lower bounds stark vom echten Wert abweichen

Approximation:

- breche Phase 2 bereits ab, wenn $\text{minKey}(\overleftarrow{Q}) \cdot K > \mu$ gilt
- dann ist der berechnete Weg eine K-Approximation des kürzesten Weges

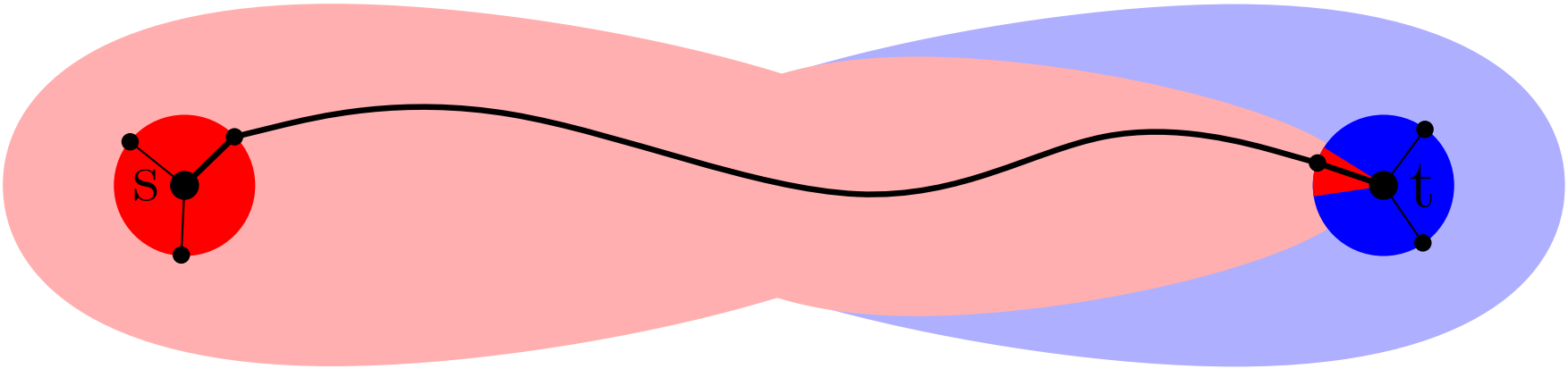
Experimente

scen.	algorithm	K	Error: relative rate	Error: relative av.	Error: relative max	Error: abs. max [s]	Query: #sett. nodes	Query: #rel. edges	Query: time [ms]
mid	uni-ALT	–	0.0%	0.000%	0.00%	0	200 236	239 112	147.20
	TDALT	1.00	0.0%	0.000%	0.00%	0	116 476	138 696	98.27
		1.15	12.4%	0.094%	14.32%	1 892	50 764	60 398	36.91
		1.50	12.5%	0.097%	27.59%	1 892	50 742	60 371	36.86
Sat	uni-ALT	–	0.0%	0.000%	0.00%	0	148 331	177 568	100.07
	TDALT	1.00	0.0%	0.000%	0.00%	0	63 717	76 001	47.41
		1.15	10.5%	0.088%	13.97%	2 613	50 042	59 607	36.00
		1.50	10.6%	0.089%	26.17%	2 613	50 036	59 600	35.63
Sun	uni-ALT	–	0.0%	0.000%	0.00%	0	142 631	170 670	92.79
	TDALT	1.00	0.0%	0.000%	0.00%	0	58 956	70 333	42.96
		1.15	10.4%	0.088%	14.28%	1 753	50 349	59 994	36.04
		1.50	10.5%	0.089%	32.08%	1 753	50 345	59 988	35.74

Core-ALT

Idee

- begrenze Beschleunigungstechnik auf kleinen Subgraphen (Kern)



Vorberechnung

- kontrahiere Graphen zu einem Kern
- Landmarken nur im Kern

Anfrage

- Initialphase: normaler Dijkstra
- benutze Landmarken nur im Kern
- zeitabhängig:
  - Rückwärtssuche ist zeitunabhängig
  - Vorwärtssuche darf alle Knoten der Rückwärtssuche besuchen

Experimente

		Preproc.		Error				Query		
		time	space	relative			abs.	#settled	#relaxed	time
scenario	K	[min]	[B/n]	rate	av.	max	max [s]	nodes	edges	[ms]
	1.00	9	50.3	0.0%	0.000%	0.00%	0	2 984	11 316	4.84
Monday	1.15	9	50.3	8.3%	0.051%	11.00%	1 618	1 588	5 303	1.84
	1.50	9	50.3	8.3%	0.052%	17.25%	1 618	1 587	5 301	1.84
	1.00	9	50.3	0.0%	0.000%	0.00%	0	3 190	12 255	5.36
midweek	1.15	9	50.3	8.2%	0.051%	13.84%	2 408	1 593	5 339	1.87
	1.50	9	50.3	8.2%	0.052%	13.84%	2 408	1 592	5 337	1.86
	1.00	8	44.9	0.0%	0.000%	0.00%	0	3 097	12 162	5.21
Friday	1.15	8	44.9	7.8%	0.052%	11.29%	2 348	1 579	5 376	1.82
	1.50	8	44.9	7.8%	0.054%	21.19%	2 348	1 579	5 374	1.82
	1.00	6	27.8	0.0%	0.000%	0.00%	0	1 856	7 188	2.42
Saturday	1.15	6	27.8	4.4%	0.031%	11.50%	1 913	1 539	5 542	1.71
	1.50	6	27.8	4.4%	0.031%	24.17%	1 913	1 539	5 541	1.71
	1.00	5	19.1	0.0%	0.000%	0.00%	0	1 773	6 712	2.13
Sunday	1.15	5	19.1	4.0%	0.029%	12.72%	1 400	1 551	5 541	1.68
	1.50	5	19.1	4.1%	0.029%	17.84%	1 400	1 550	5 540	1.68

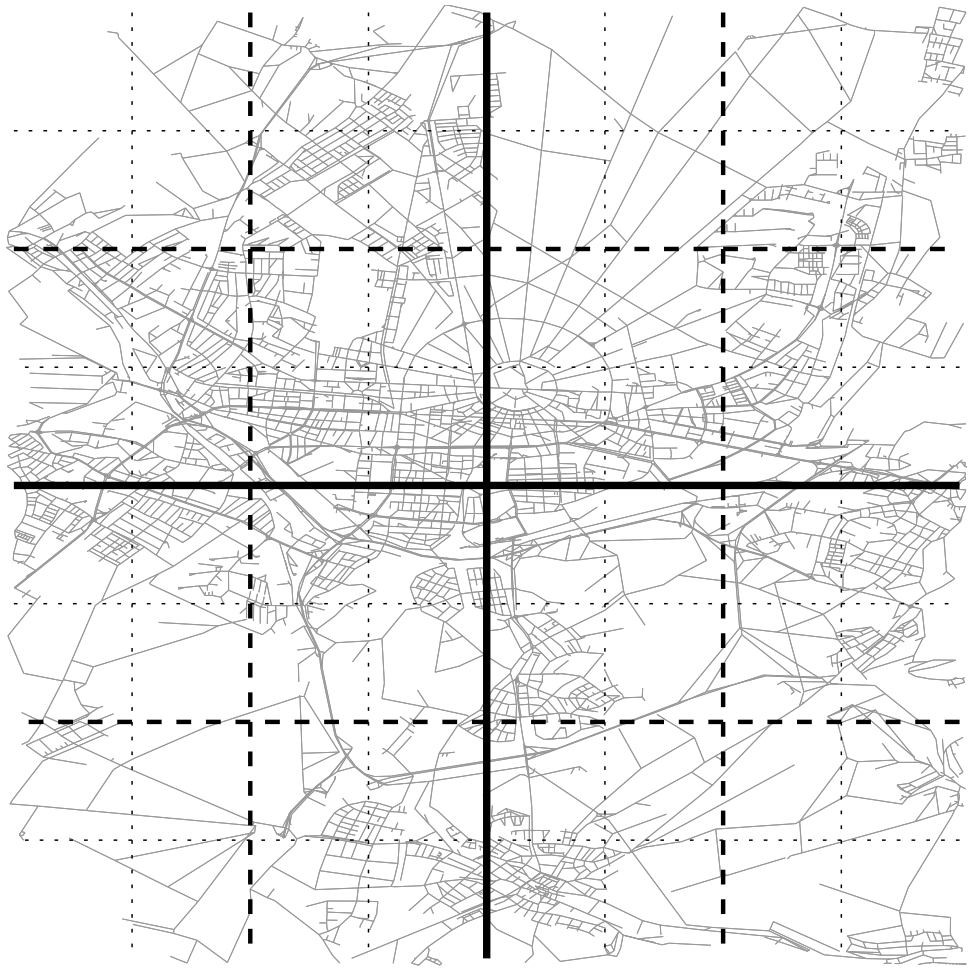
SHARC (Kontraktion, Arc-Flags)

Vorberechnung:

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

Anpassung:

- Kontraktion und Flaggen berechnung anpassen
- Verfeinerung durch (lokale) Profilsuchen



Experimente

		Preprocessing				Time-Queries	
scenario	algom	time [h:m]	space [B/n]	edge inc.	points inc.	time [ms]	speed up
Monday	eco	1:16	156.6	25.4%	366.8%	24.55	63
midweek	eco	1:16	154.9	25.4%	363.8%	25.06	60
Friday	eco	1:10	142.0	25.4%	358.0%	22.07	69
Saturday	eco	0:42	90.3	25.0%	283.6%	5.34	276
	agg	48:57	84.3	24.5%	264.4%	0.58	2 554
Sunday	eco	0:30	64.6	24.6%	215.8%	1.86	787
	agg	27:20	60.7	24.1%	202.6%	0.50	2 904
no traffic	static	0:06	13.5	23.9%	23.9%	0.30	4 075

Approximation

		Prepro		Error			Time-Queries	
scenario	algo	time [h:m]	space [B/n]	error -rate	max rel.	max abs.[s]	time [ms]	spd up
Monday	heu	3:30	138.2	0.46%	0.54%	39.3	0.69	2 253
midweek	heu	3:26	137.2	0.82%	0.61%	48.3	0.69	2 164
Friday	heu	3:14	125.2	0.50%	0.50%	50.3	0.64	2 358
Saturday	heu	2:13	80.4	0.18%	0.23%	16.9	0.51	2 887
Sunday	heu	1:48	58.8	0.09%	0.36%	14.9	0.46	3 163
no	static	0:06	13.5	0.00%	0.00%	0.0	0.30	4 075

Beobachtung:

- Fehler sehr gering
- hoher Speicherverbrauch

Profilsuchen

		Time-Queries		Profile-Queries			
traffic	var.	#del. mins	time [ms]	#del. mins	#re-ins.	time [ms]	profile /time
Monday	eco	19 136	24.55	19 768	402	51 122	2 082.6
	heu	810	0.69	1 071	24	1 008	1 460.9
midweek	eco	19 425	25.06	20 538	432	60 147	2 400.3
	heu	818	0.69	1 100	27	1 075	1 548.4
Friday	eco	17 412	22.07	19 530	346	52 780	2 391.9
	heu	769	0.64	1 049	21	832	1 293.2
Saturday	eco	5 284	5.34	5 495	44	3 330	624.0
	agg	721	0.58	865	9	134	232.5
	heu	666	0.51	798	8	98	191.9
Sunday	eco	2 142	1.86	2 294	12	536	288.1
	agg	670	0.50	781	5	57	113.5
	heu	635	0.46	738	5	45	97.9

Kompression

Gründe für Overhead:

- Regionsinformation
- Flaggen speichern
- Shortcuts (Einträge im Kantenarray)
- zusätzliche Interpolationspunkte

Space-Efficient SHARC:

- Arc-Flag Kompression
- Entfernung von Shortcuts
- entferne Interpolationspunkte und entpacke on-the-fly
- Overhead gleich wie zeitunabhängig (12 bytes pro Knoten)
- Queryzeiten von 2 ms
- aber eventuell nicht exakt

Contraction Hierarchies

Vorberechnung:

- benutze gleiche Knotenordnung
- kontrahiere zeitabhängig
- erzeugt Suchgraphen $G' = (V, \uparrow E \cup \downarrow E)$

Anfrage

- Rückwärts aufwärts mittels min-max Suche
- markiere alle Kanten (u, v) aus $\downarrow E$ mit $\underline{d(u, v)} + \underline{d(v, t)} \leq \overline{d(u, v)}$
- diese Menge sei $\downarrow E'$
- zeitabhängige Vorwärtsuche in $(V, \uparrow E \cup \downarrow E')$

Experimente

input	type of ordering	Contr. ordering [h:m]	Contr. const. [h:m]	Contr. space [B/n]	Queries time [ms]	Queries speed up
Monday	static min	0:05	0:20	1 035	1.19	1 240
	timed	1:47	0:14	750	1.19	1 244
midweek	static min	0:05	0:20	1 029	1.22	1 212
	timed	1:48	0:14	743	1.19	1 242
Friday	static min	0:05	0:16	856	1.11	1 381
	timed	1:30	0:12	620	1.13	1 362
Saturday	static min	0:05	0:08	391	0.81	1 763
	timed	0:52	0:08	282	1.09	1 313
Sunday	static min	0:05	0:06	248	0.71	1 980
	timed	0:38	0:07	177	1.07	1 321

Approximation

Idee:

- speicher jeden Shortcut als Approximation
- reduziert Speicher um bis zu Faktor 10

Query:

- min-max Suchen aufwärts in approximierter CH
- alle Knoten an denen Suchen sich treffen: Kandidaten C
- entpacke alle Shortcuts auf wegen s–C–t Pfaden
- erzeugt Subgraphen (Korridor)
- time-dependent Dijkstra im Korridor
- nicht viel langsamer (Faktor 2)
- ist exakt

Profilsuchen

Variante 1:

- normale Profilsuche in der CH
- langsam

Variante 2:

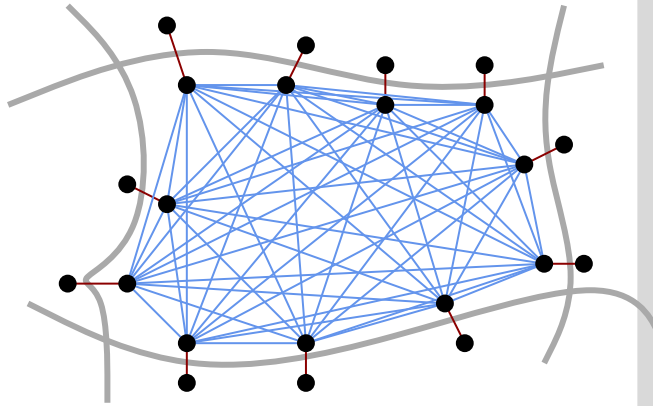
- normale Profilsuche im Korridor
- besser aber es geht noch besser

Variante 3:

- Kontraktion des Corridors
- Interpolationspunkte steigen langsamer an

$\Rightarrow$ ca. 30 ms

MLD



Beobachtungen

- Partitionierung metrik-unabhängig
- viele Shortcuts

work in progress

- unklar ob abspeichern aller Shortcuts sinnvoll
- hilft Approximation?

Ende

Literatur (Zeitabhängige Beschleunigungstechniken):

- Daniel Delling:
  Enginering and Augmenting Route Planning Algorithms
  Ph.D. Thesis, Universität Karlsruhe (TH), 2009.
- **Gernot Veit Batz, Robert Geisberger, Sabine Neubauer, Peter Sanders**
  Time-Dependent Contraction Hierarchies and Approximation
  In: *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*, 2010.
- **Edith Brunel, Daniel Delling, Andreas Gemsa, Dorothea Wagner**
  Space-Efficient SHARC-Routing
  In: *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*, May 2010.

Nächste Termine

Karlsruher Institut für Technologie

Montag, 25.6.2012

Montag, 2.7.2012
Montag, 9.7.2012
Montag, 16.7.2012