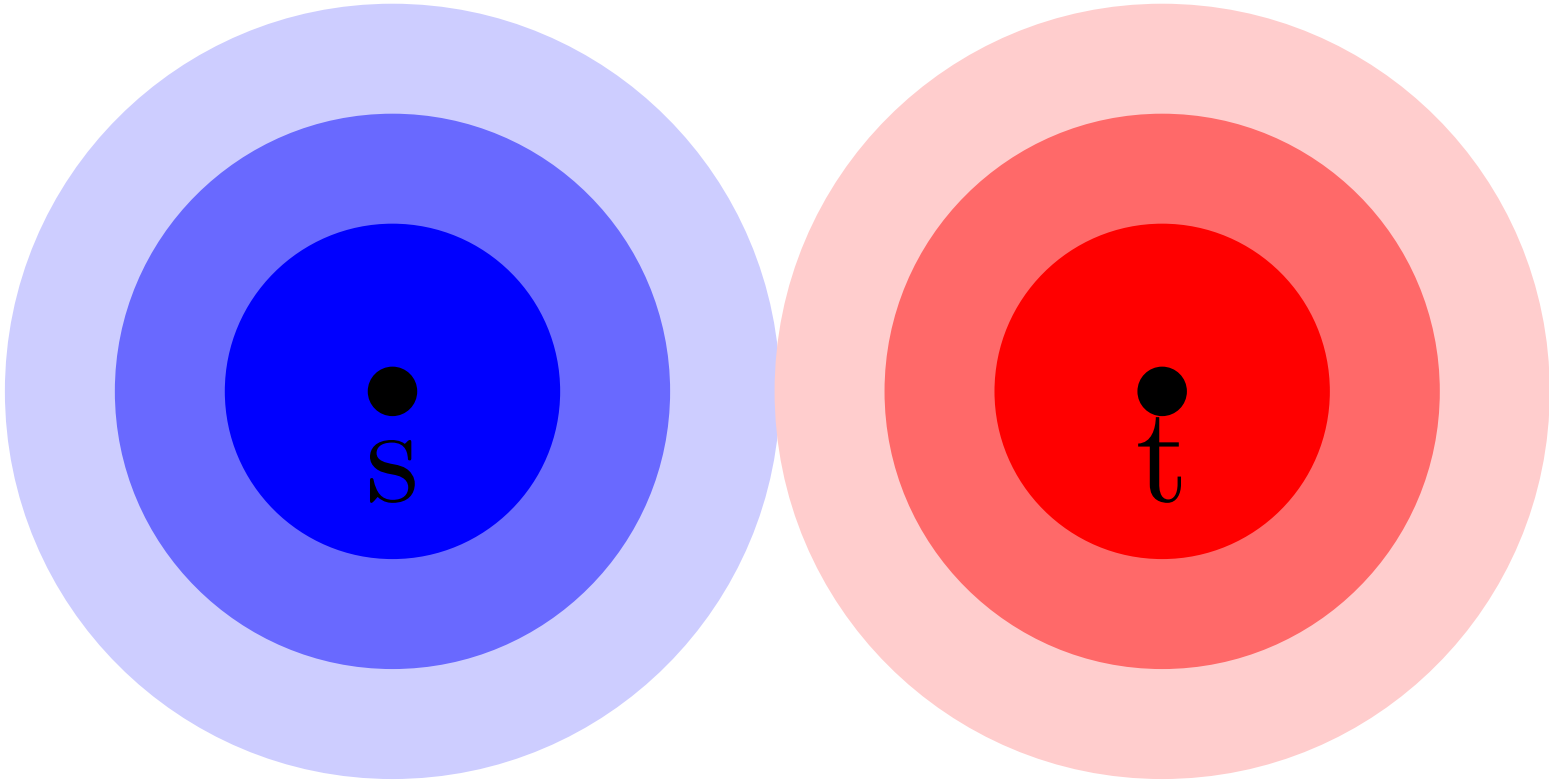
Algorithmen für Routenplanung – Vorlesung 5

Daniel Delling

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

Letztes Mal

- Reach
- RE/REAL



Ideensammlung

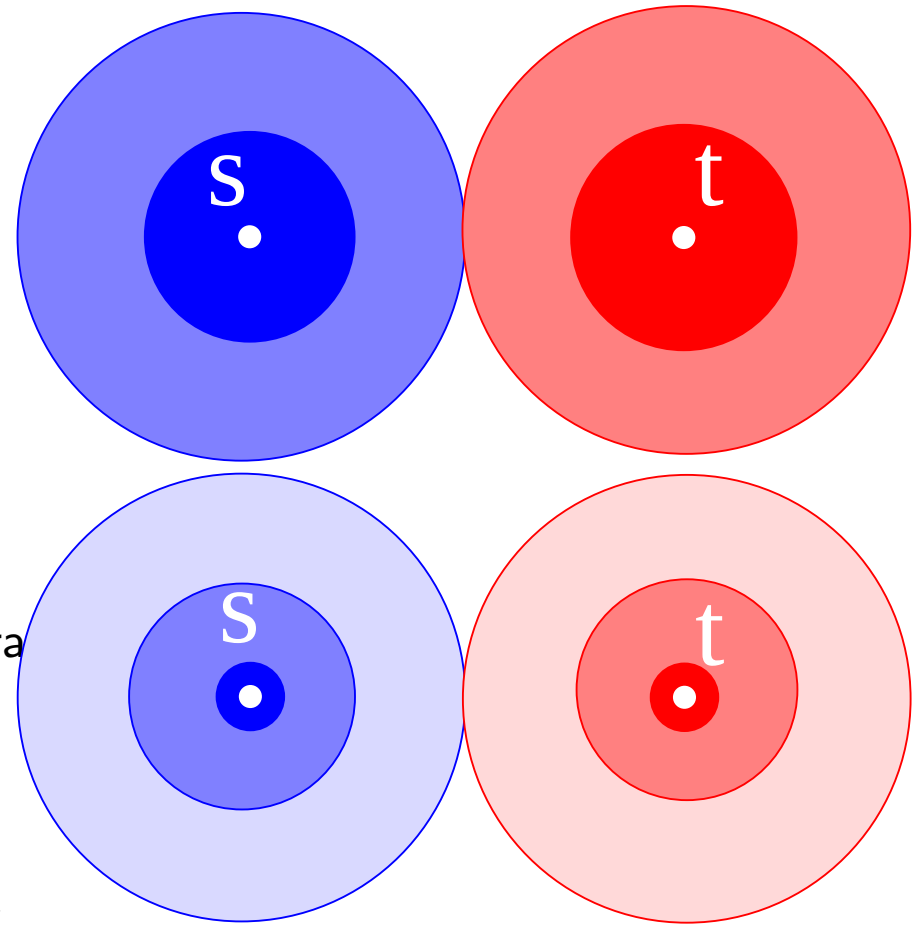
Wie Suche hierarchisch machen?

- identifiziere wichtige Knoten mit Zentralitätsmaß
- überspringe unwichtige Teile des Graphen

heute letzteres

Kommerzielle Systeme

- komplette Suche in lokalem Bereich
- suche in (dünnerem) Autobahn-Netzwerk
- iteriere $\rightsquigarrow$ highway hierarchy



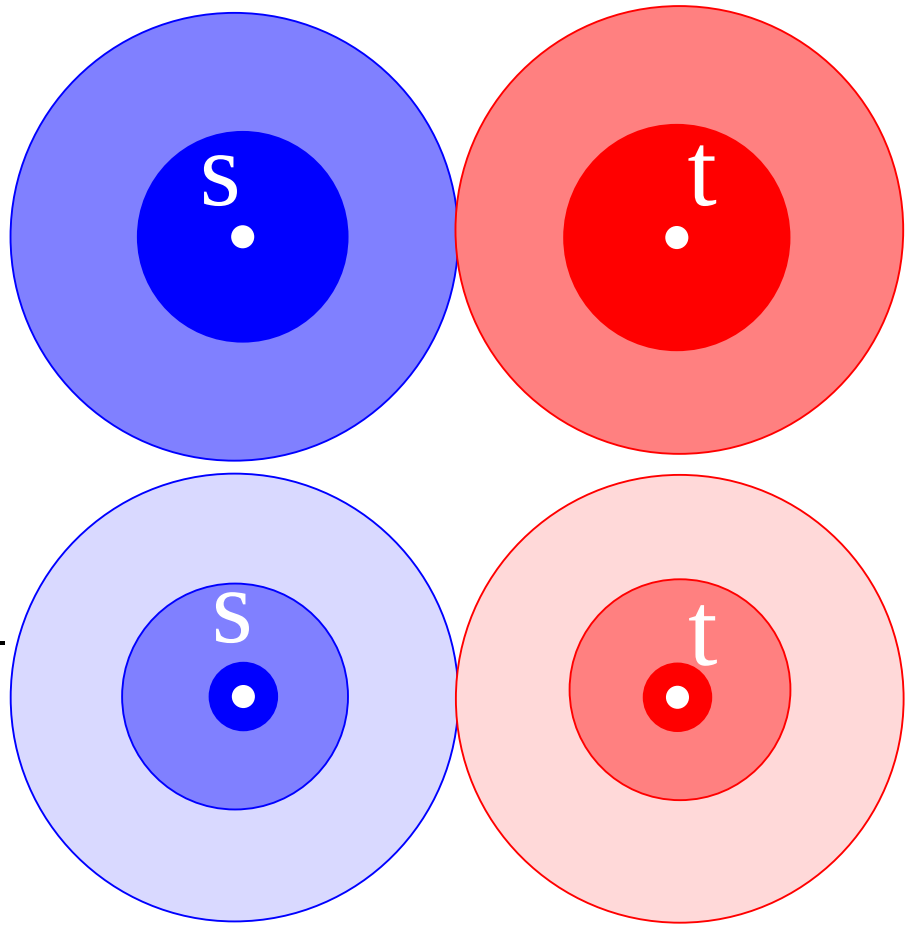
Definition des highway networks:

benutze Straßen-Kategorie (highway, federa highway, motorway,...) + Feinjustierung

- kompliziert einzustellen
- **Geschwindigkeit** $\Leftrightarrow$ **Genauigkeit**

Highway Hierarchies

- komplette Suche in lokalem Bereich
- suche in (dünnerem) Autobahn-Netzwerk
- iteriere $\rightsquigarrow$ highway hierarchy



Definition des highway networks:

minimales Netzwerk, dass kürzeste Wege erhält

- (voll) automatisch
- schneller als kommerzielle Systeme

Highway Hierarchies: Idee

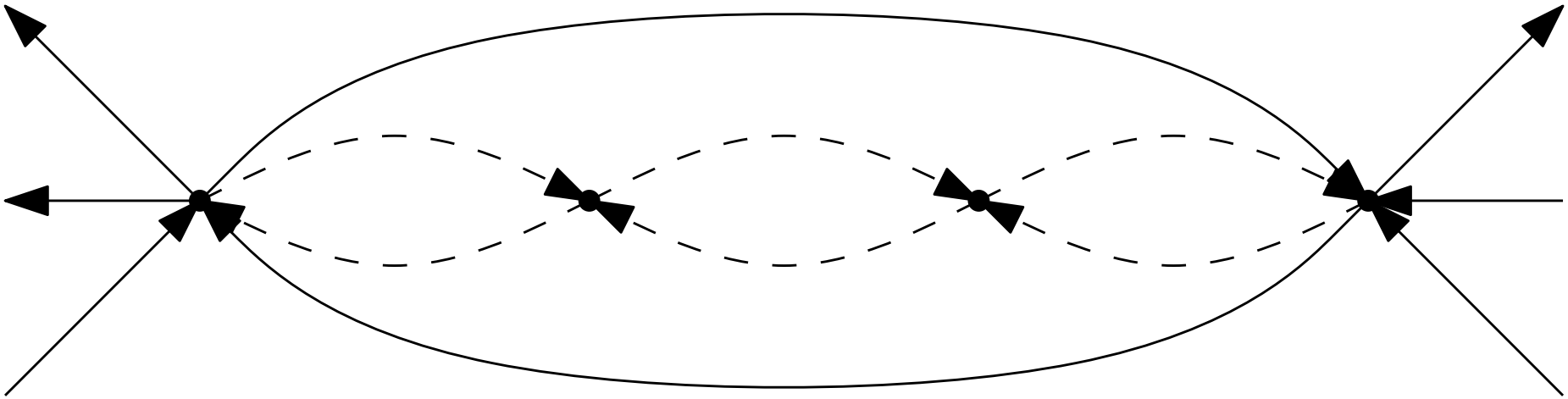
Vorberechnung:

- Knotenreduktion
- Kantenreduktion
- iterativ $\rightsquigarrow$ Hierarchie

Anfragen:

- bidirektional
- suche aufwärts

Knoten-Reduktion



Beobachtung:

- Knoten mit niedrigem Knotengrad unwichtig

$\Rightarrow$ kontrahiere Graphen

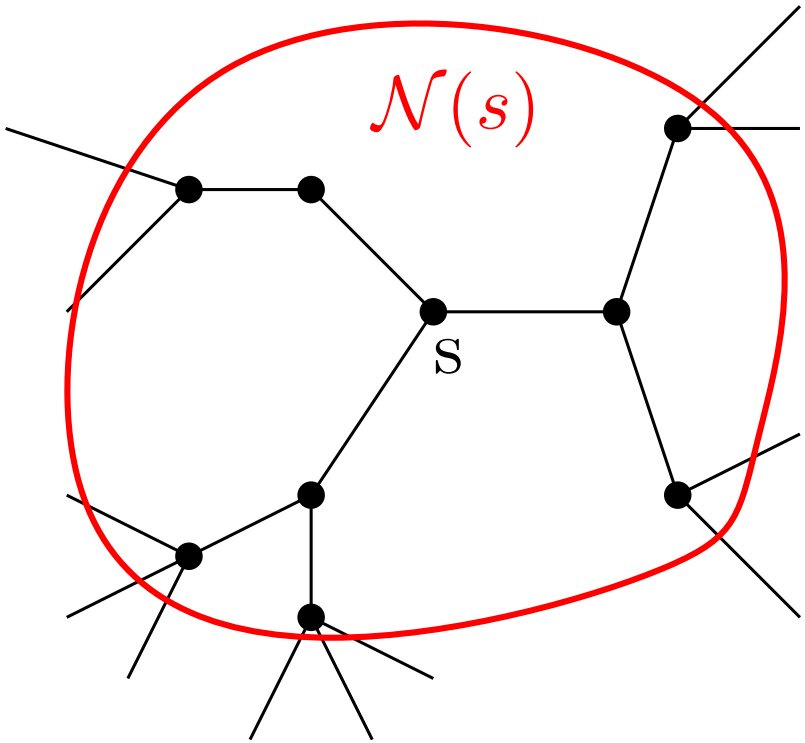
- entferne Knoten iterativ
- shortcuts um Abstände zwischen verbleibenden Knoten zu erhalten

Kanten-Reduktion I

Lokale Nachbarschaft:

- bestimme Nachbarschaftsradius $r(s)$
  - hier: Abstand zum H-nächsten Knoten für festes H
- Nachbarschaft von s:

$$\mathcal{N}(s) := \{v \in V \mid d(s,v) \leq r(s)\}$$



Kanten-Reduktion II

Highway-Kante:

- Kante (u, v) ist eine Autobahn-Kante, wenn es s und t gibt mit:
  - (u, v) ist auf kürzestem Weg von s nach t
  - $u \neq \mathcal{N}(s)$
  - $v \neq \mathcal{N}(s)$

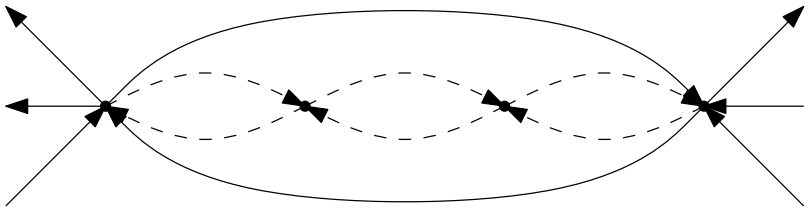
Idee:

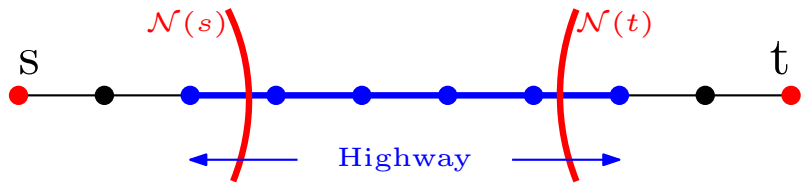
- entferne Kanten, die keine highway-Kanten sind

Vorberechnung

Vorberechnung:

- Knotenreduktion
  - entfernen von Knoten
  - einfügen von Shortcuts
- Kantenreduktion
  - entferne Autobahn-Kanten
- iterativ $\rightsquigarrow$ Hierarchie





Suchgraph:

- Graph mit Shortcuts
- Levelinformationen
- Radien

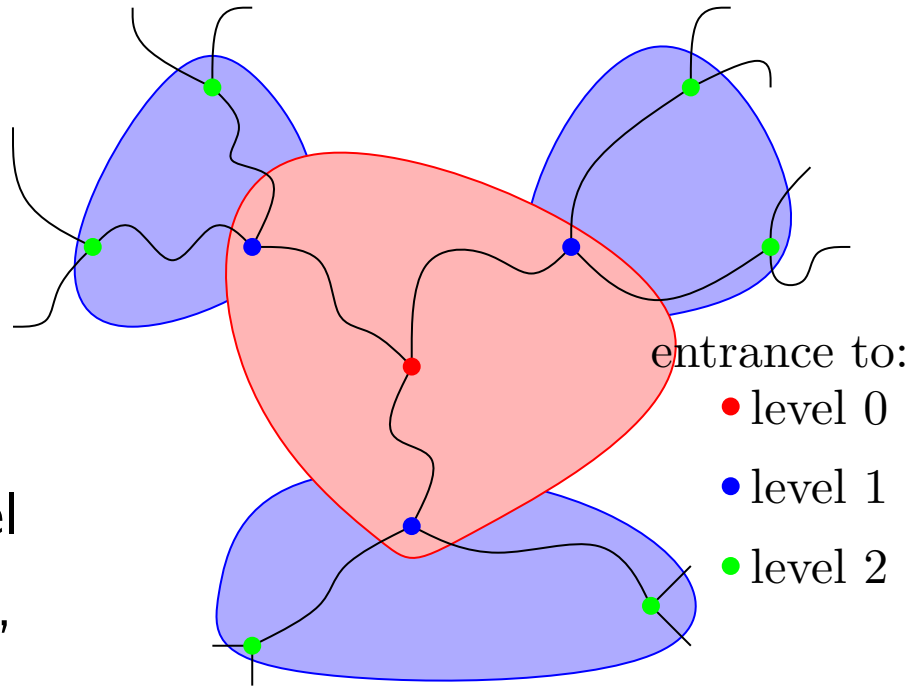
Anfragen

Query:

- bidirektionaler Dijkstra

Modifikation:

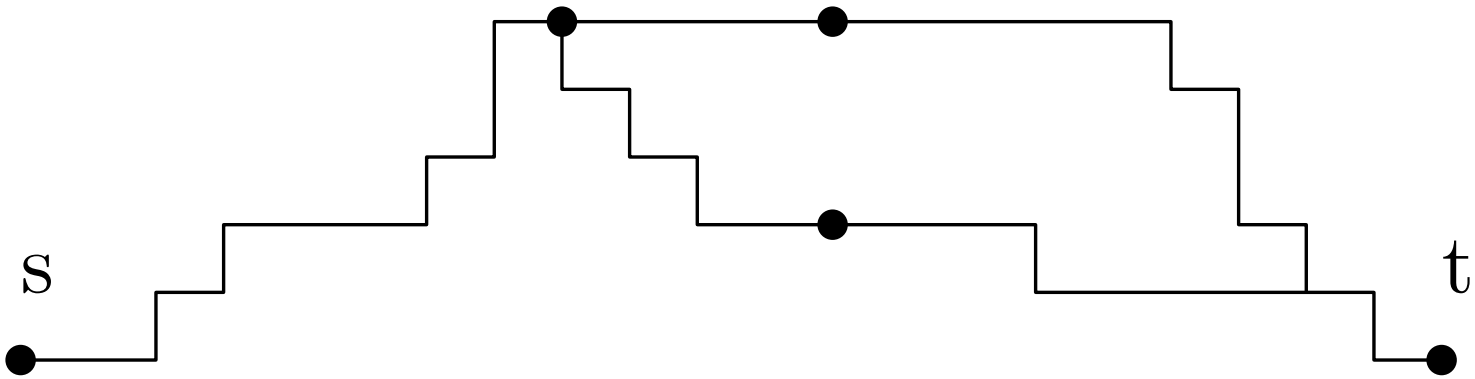
- verlasse nicht die Nachbarschaft des Eintrittspunkt in diesen Level *sondern:* wechsel in höheren Level
- relaxiere keine Kanten to Knoten, die in diesem Level durch Kontraktion entfernt worden sind



Stoppkriterium

Problem:

- bekanntes Stoppkriterium nicht korrekt
- Level eines kürzesten Weges kann sehr unausgeglichen sein



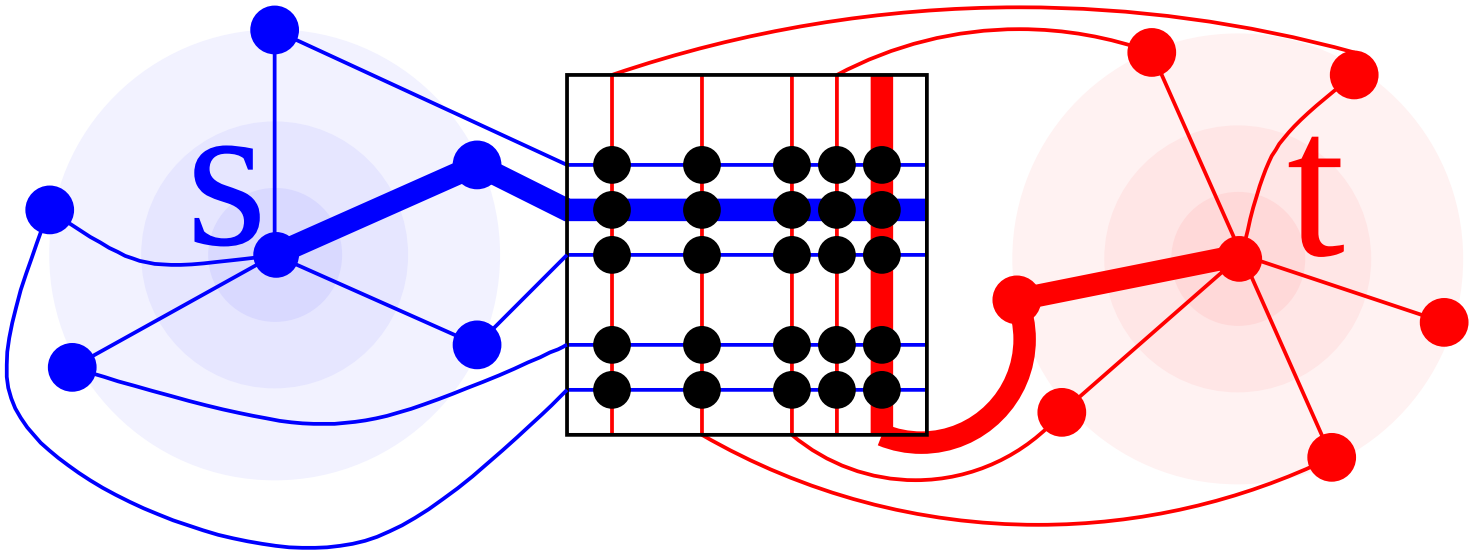
Lösung:

- sehr konservatives Stoppkriterium
- stoppe eine Suche wenn Queue leer oder $k_f \geq \mu$
- stoppe Anfrage, wenn beide Suchen fertig

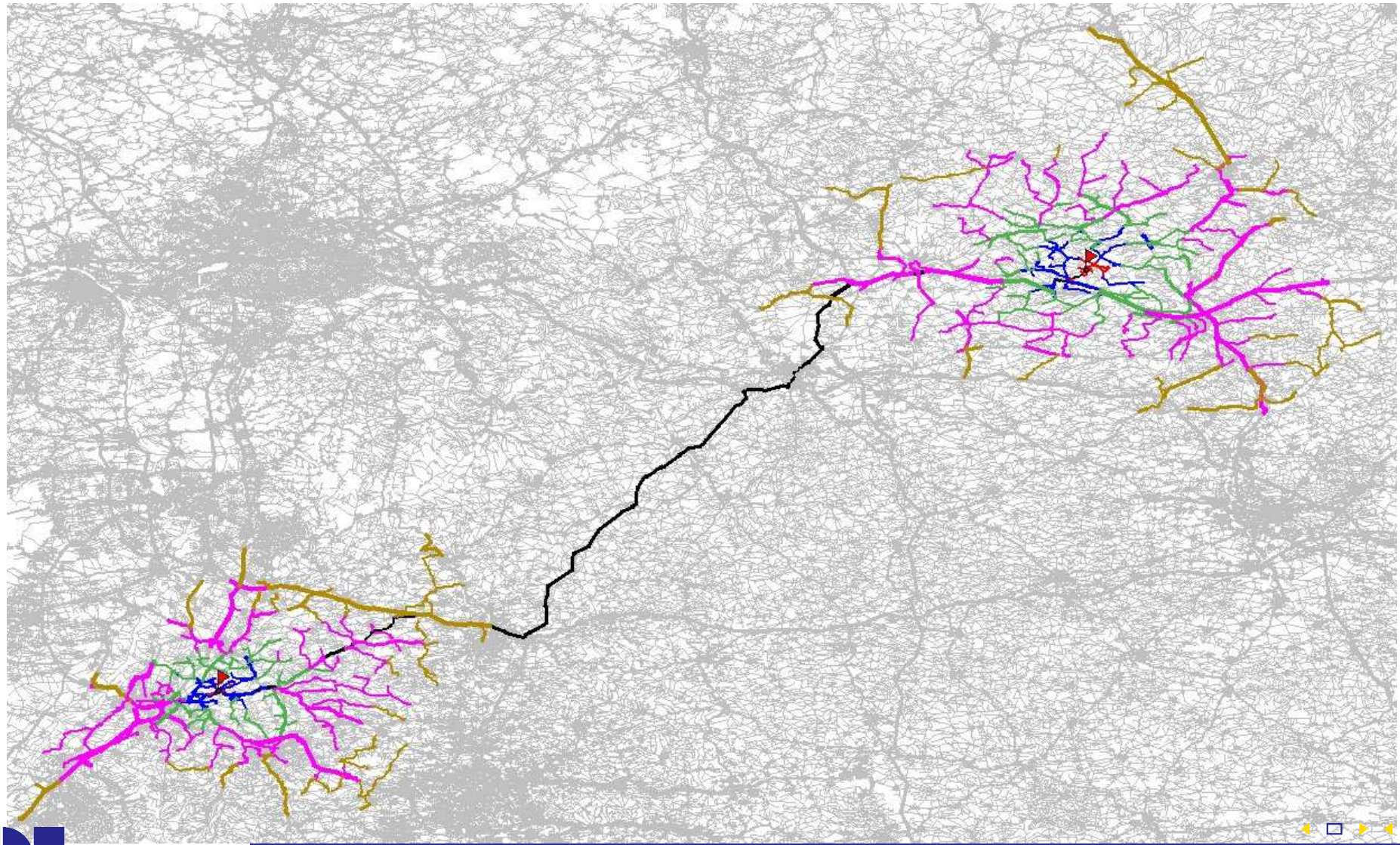
Optimierung

Idee:

- speichere Distanzmatrix für letzte Level
- suche aufwärts und abwärts bis Distanzmatrix
- dann table-lookups (ca. 55 pro Richtung)
- spart Durchsuchen der oberen Level



Beispiel Suchraum HH + Distanztabelle



Diskussion

Vorteile:

- schnelle Vorberechnung (rein lokale Operationen)
- war der erste Ansatz, der große Straßengraphen verarbeiten konnte
- angeblich (!) wird HH von Google-Maps benutzt

Nachteile:

- Identifikation der Highway-Kanten
- Stoppkriterium
- Korrektheitsbeweis ($>$ 5 Seiten)
- Datenstrukturen (für speichern des Radius und level-informationen)

$\Rightarrow$ schwierig zu implementieren

Highway-Node Routing

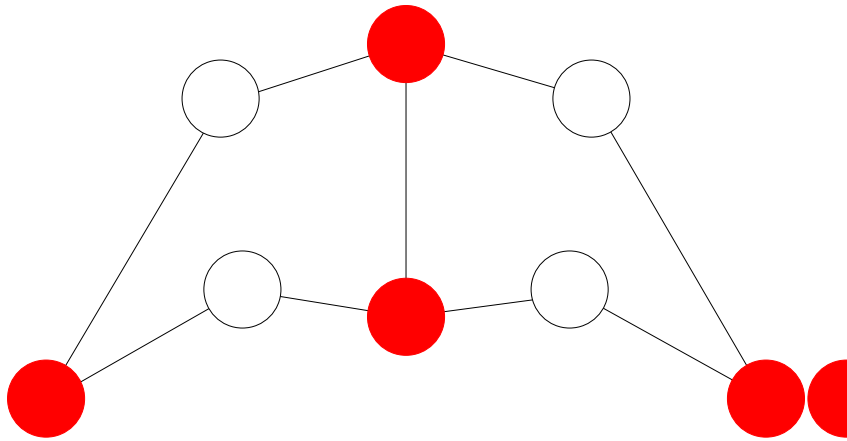
Idee:

- vereinfache Konzept der Highway Hierarchies
  - Vorberechnung
  - Anfragen

basiert auf:

- gegebenen Knotenleveln
- reiner Kontraktion
- deutlich einfacher Kantenreduktion
- Overlay-Graphen

Overlay-Graph



Gegeben:

- Graph $G = (V, E)$
- Knotenmenge $S \subseteq V$

Overlay-Graph:

- $G' = (S, E')$
- bestimme Kantenmenge E', so dass Abstände in S erhalten bleiben

Minimaler Overlay-Graph

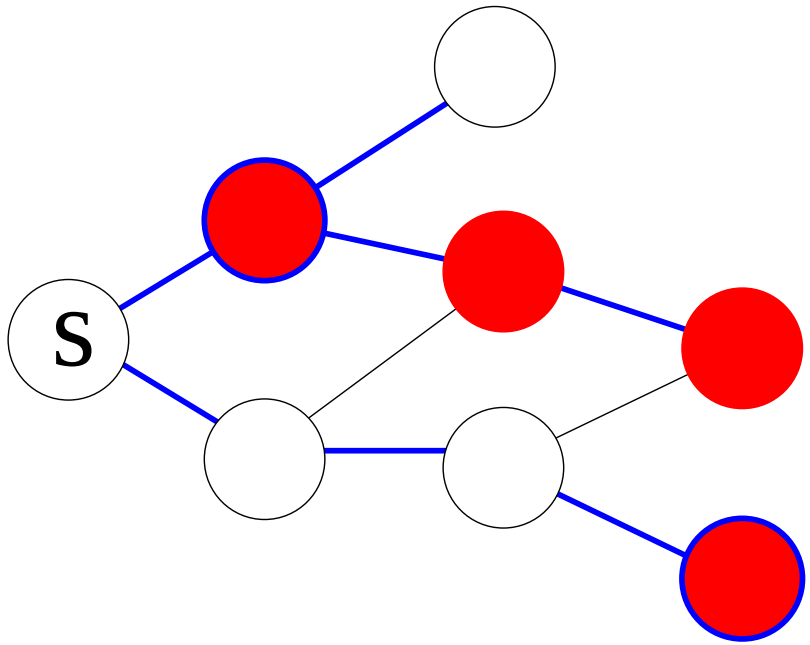
- bestimme E' mit

$$E' := \{(s,t) \in S \times S \mid \text{kein Knoten zwischen s und t gehört zu S}\}$$

Abgedeckende Knoten

Definitionen:

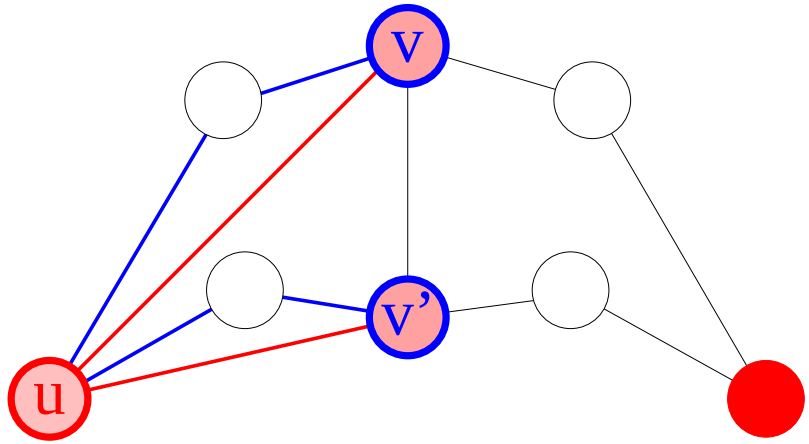
- abgedeckter Zweig: enthält Knoten $u \in S$
- abgedeckter Baum: alle Zweige abgedeckt
- abgedeckende Knoten: auf jedem Zweig der Knoten $u \in S$, der am nächsten an s ist



Konstruktion Overlay-Graph

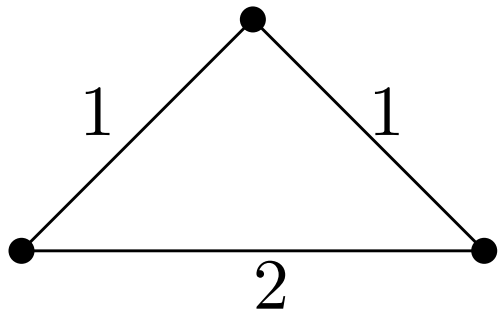
Vorgehen:

- von jedem Knoten $u \in S$:
- lokale Suche in G
- bis alle Zweige abgedeckt
- füge Kante von u zu abdeckenden Knoten ein



optional:

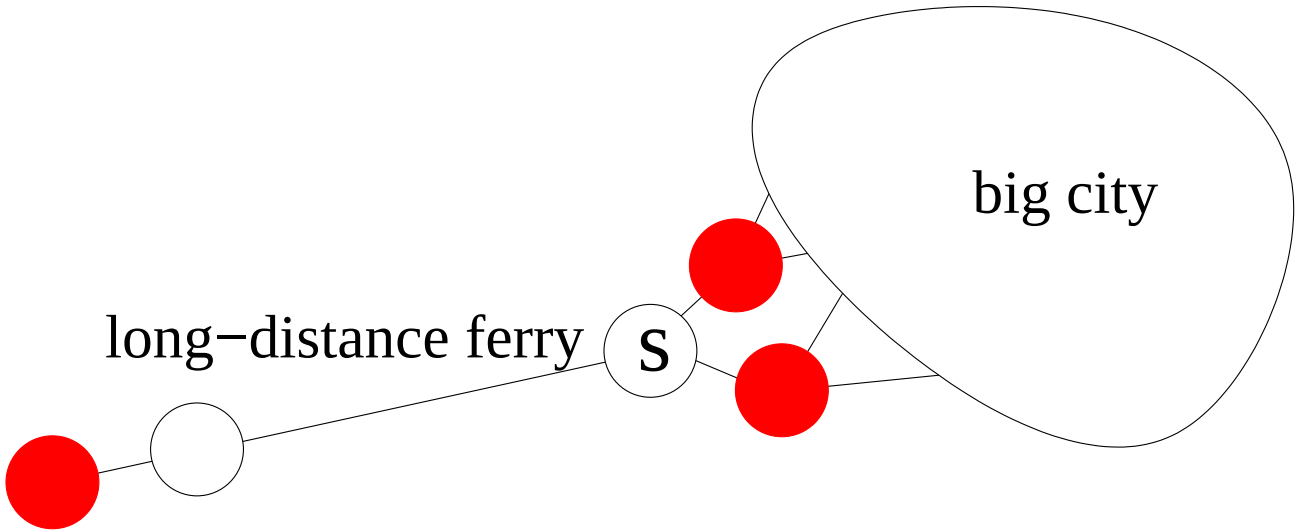
- suche in G' von jedem u bis alle Nachbarn v abgearbeitet
- bevorzuge hop-minimale Pfade
- checke ob u Vorgänger in v
- wenn nein, entferne (u, v)
- zusätzliche Kantenreduktion



Berechnung abdechender Knoten

Konservativer Ansatz:

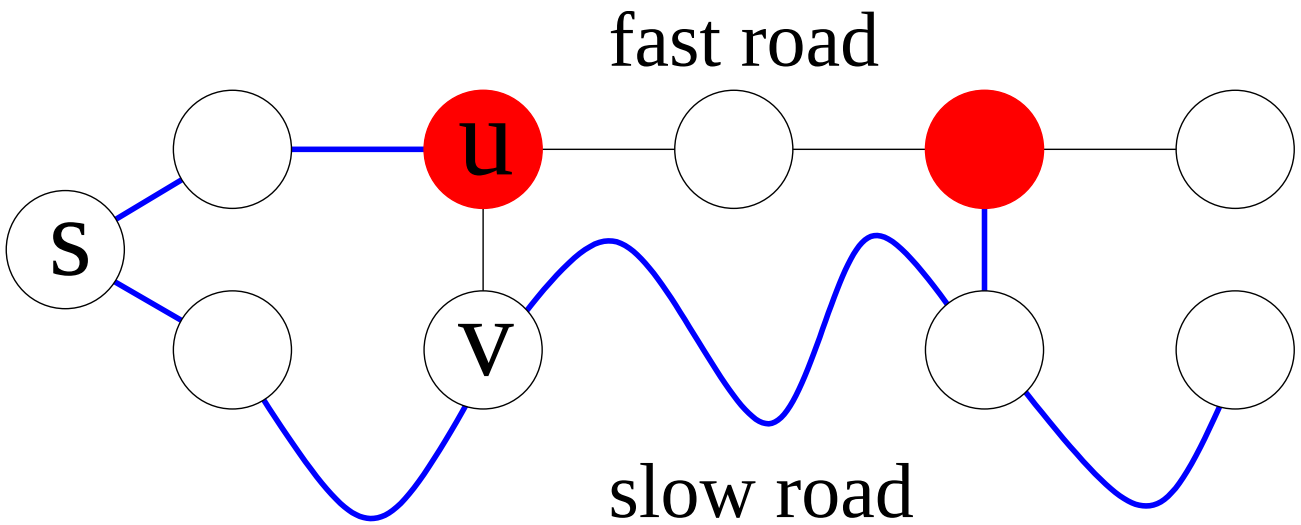
- breche Suche ab, wenn alle Zweige abgedeckt
- kann ineffizient sein



Berechnung abdechender Knoten

Aggressiver Ansatz:

- relaxiere keine Kanten auf abgedeckten Zweigen
- kann ineffizient sein



Berechnung abdechender Knoten

Kompromiss:

- Parameter p
- relaxiere keine Kanten auf Zweigen, die mehr als p Knoten aus S enthalten
- stoppe wenn alle Zweige abgedeckt
- $p = 1 \rightsquigarrow$ aggressive Variante
- $p = \infty \rightsquigarrow$ konservative Variante

Vorberechnung

- nehme Klassifikation einer Highway Hierarchy

$$S_1 \supseteq S_2 \supseteq S_3 \supseteq \cdots \supseteq S_L$$

- berechene multi-level Overlay-Graphen

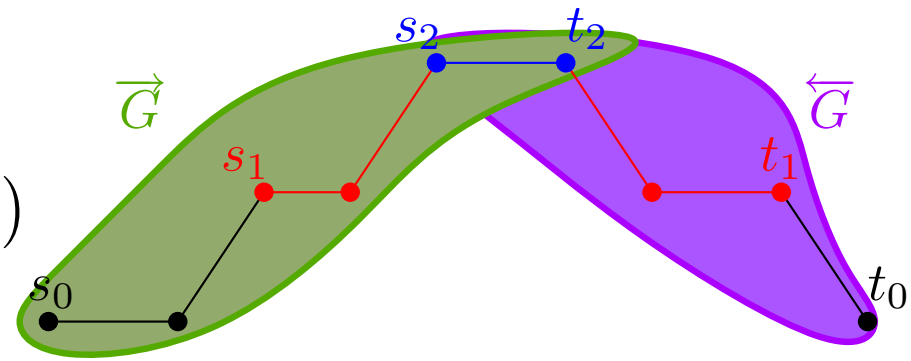
$$G_0 = G = (V, E), G_1 = (S_1, E_1), \ldots, G_L = (S_L, E_L)$$

- Suchgraph mit Shortcuts und level informationen

Anfragen

- Knotenlevel
  $l(u) := \max\{l \mid u \in S_l\}$
- Vorwärts-Suchgraph

$$\overrightarrow{G} := \left(V, \{(u,v) \mid (u,v) \in \bigcup_{i=l(u)}^{L} E_i\}\right)$$



- Rückwärts-Suchgraph

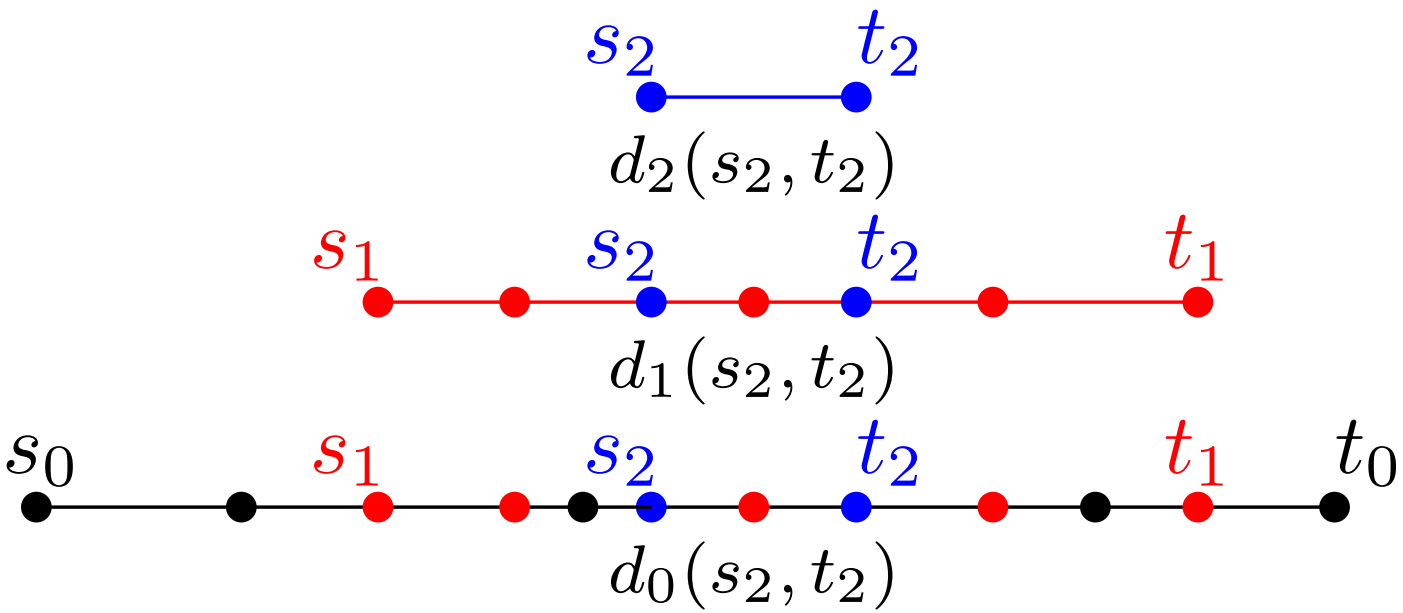
$$\overleftarrow{G} := \left(V, \{(u,v) \mid (v,u) \in \bigcup_{i=l(u)}^{L} E_i\}\right)$$

  - normale Suchen in $\overrightarrow{G}$ und $\overleftarrow{G}$ (mit konservativem Stoppkriterium
  - **Idee:** speicher nur Kanten, die aufwärts zeigen

Korrektheit

Idee:

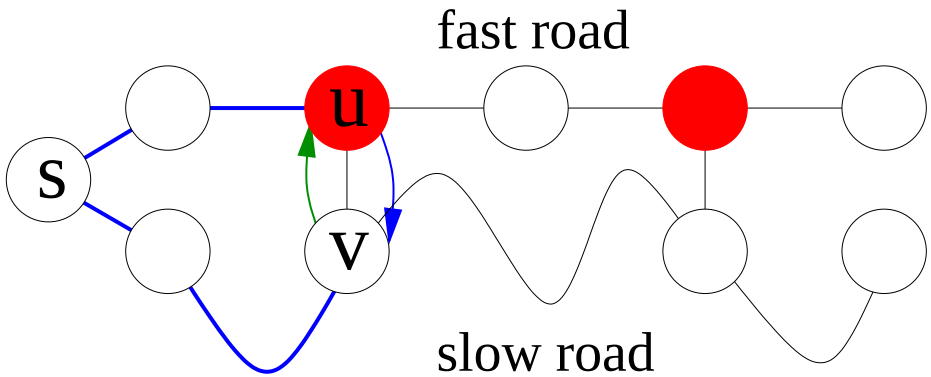
- Overlay-Graphen respektieren Abstände zwischen Knoten



Stall-On-Demand

Problem:

- unwichtige Äste werden während Anfrage abgearbeitet



Lösung:

- ein Knoten u kann anderen Knoten v stallen
  - wenn $d(s,u) + len(u,v) < d(s,v)$
  - Suche wird an v geprunt
- Knoten u' kann v auch wieder wecken
- stalling kann sogar propagiert werden (mit BFS)

Contraction Hierarchies

Nachteile Highway-Node Routing:

- basiert auf Highway-Hierarchies
- Pfad-Entpackung aufwendig

Contraction Hierarchies:

- n-level Hierarchie
- Hierarchie-Level durch Knotenordnung
- n Knoten- und Kanten-Reduktionen
- massive Vereinfachung gegenüber HNR

Idee

Contraction Hierarchies (CH):

- kontrahiere nur einen Knoten pro Knoten-Reduktion

genauer:

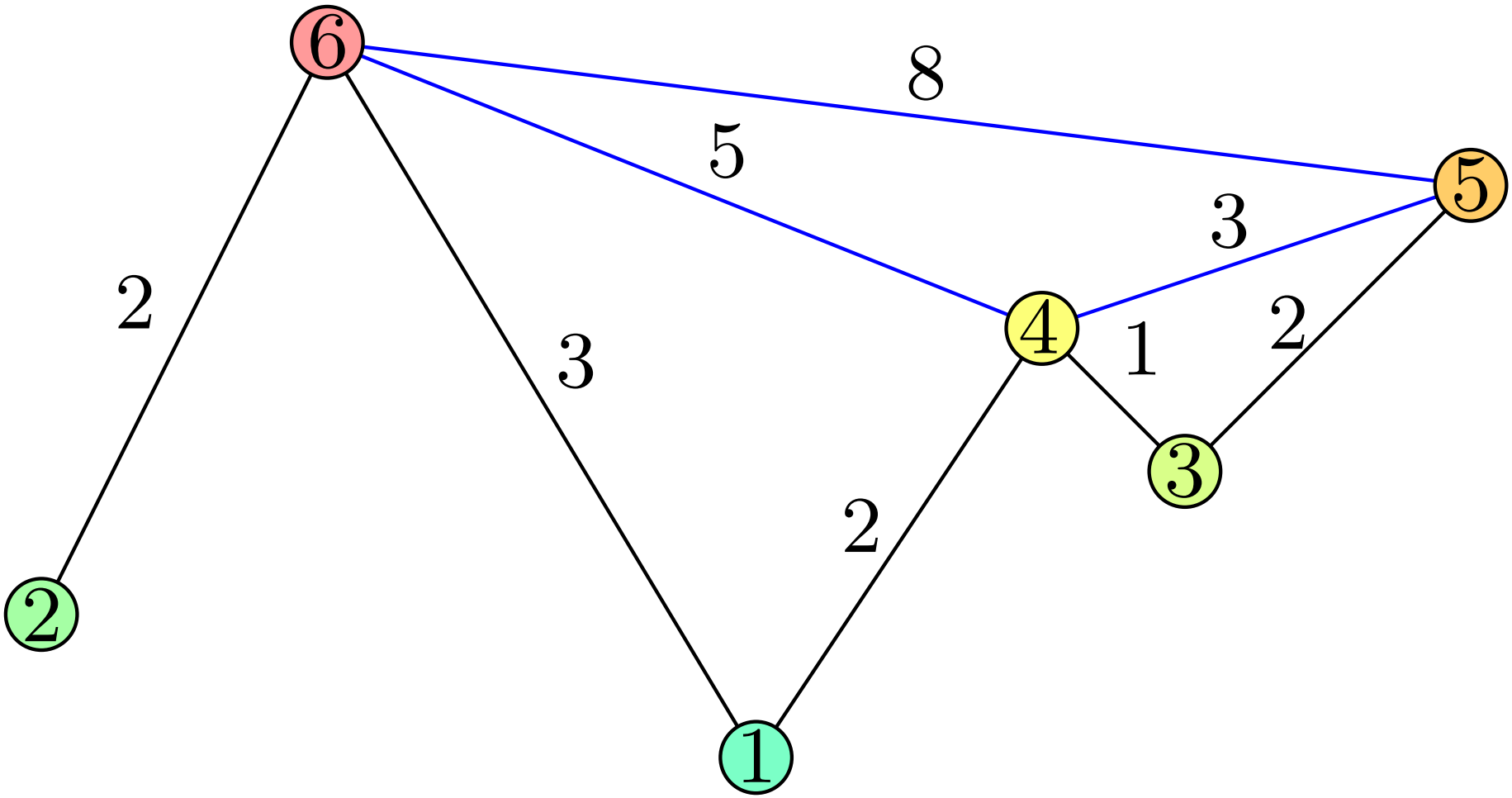
- ordne Knoten nach "Wichtigkeit"
- kontrahier Knoten in dieser Ordnung, Knoten v wird kontrahiert durch

---

foreach *pair* (u, v) *and* (v, w) *of edges* **do**
  if (u, v, w) *is a unique shortest path* **then**
    add shortcut (u, w) with weight $w(u, v, w)$

- Query relaxiert nur Kanten zu wichtigeren Knoten (ähnlich zu HNR)

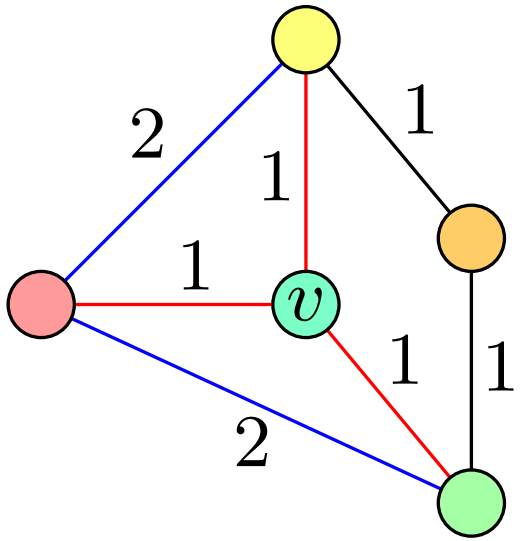
Beispiel: Konstruktion



Konstruktion

wie identifiziert man nötige Shortcuts?

- lokale Suchen von allen Knoten u der eingehenden Kanten (u, v)
- ignoriere Knoten v während der Suche
- füge shortcut (u, w) ein wenn $d(u, w) > w(u, v) + w(v, w)$



Optimierungen:

- limitiere Suchräume der lokalen Suchen
- limitiere hop-zahl der Suchen
- Spezialfälle: 1-hop-Suche, 2-hop-Suche

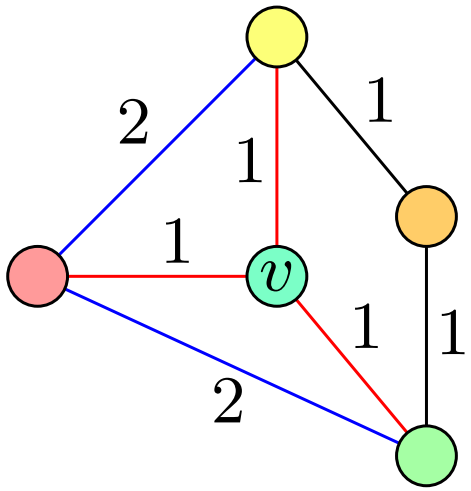
Knotenordnung

benutze priority queue, Knoten v wird gewichtet durch lineare Kombination:

- **Kanten-Differenz** #shortcuts – #Kanten inzident zu v
- **Uniformität** e.g. #entfernter Nachbarn
- **Kosten der Kontraktion** z.B. Suchraum während Kontraktion
- **Globale Zentralitäts-Maße**
- ...

Integrierte Konstruktion and Ordnung:

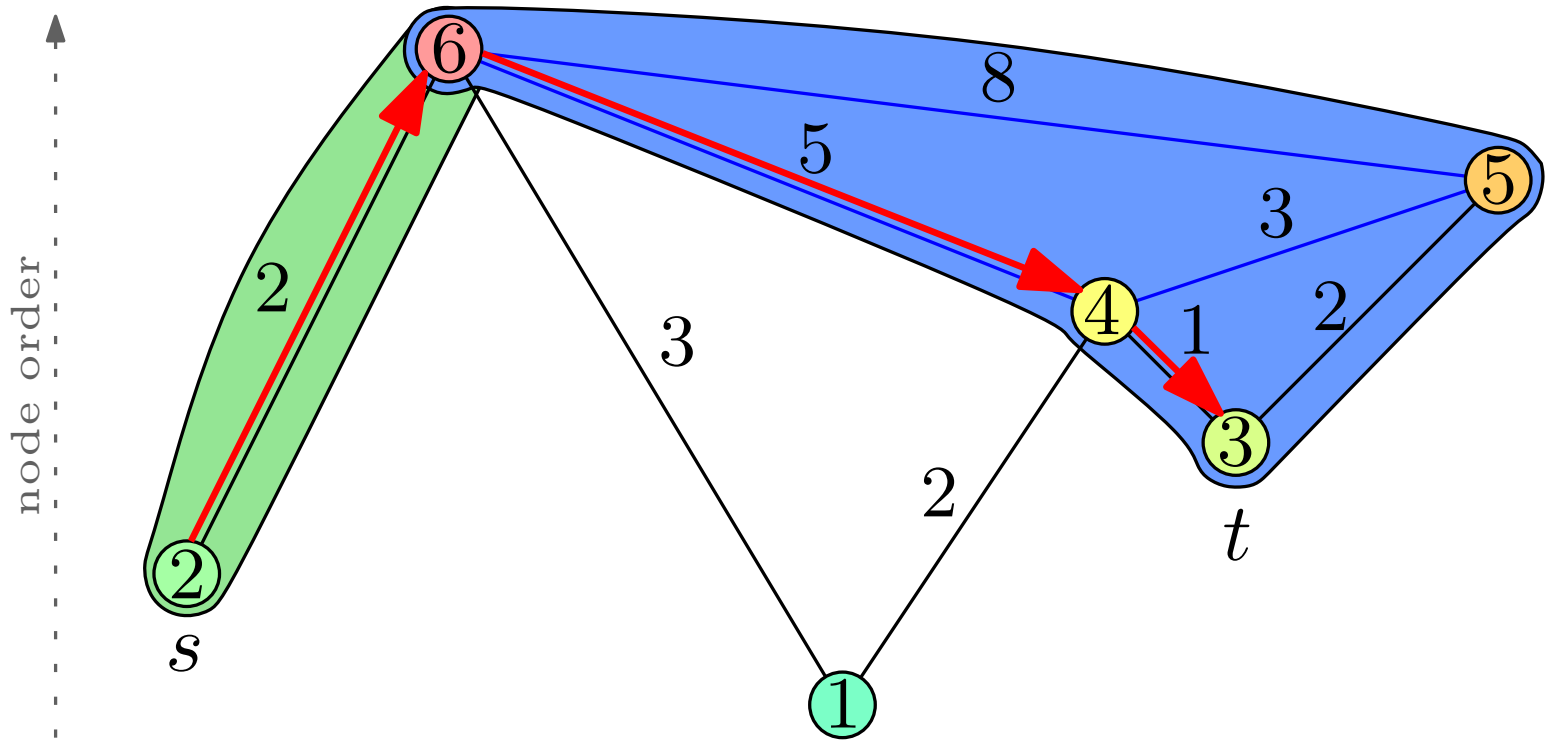
1. entferne Knoten v mit höchster Priorität
2. kontrahiere Knoten v
3. aktualisiere Prioritäten der anderen Knoten



$2 - 3 = -1$

Anfragen

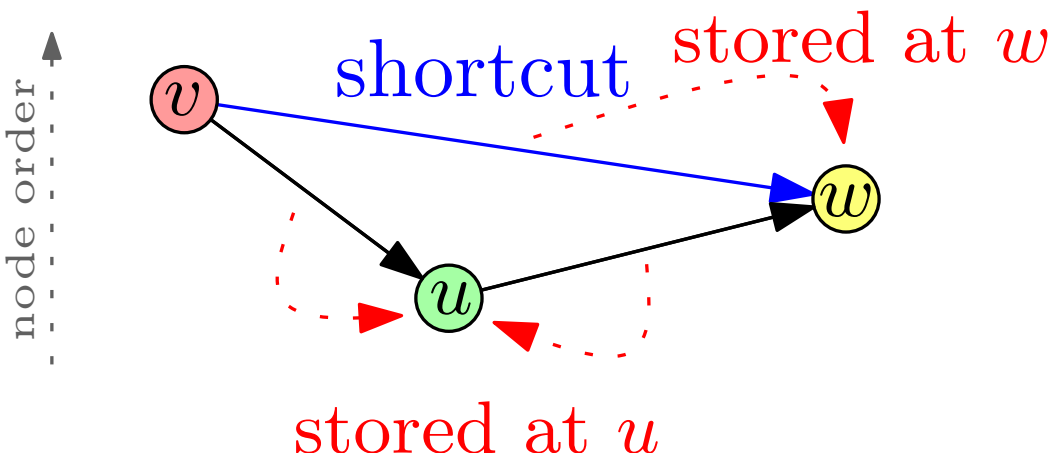
- modifizierter bidirektionaler Dijkstra algorithmus
- upward graph $G_\uparrow := (V, E_\uparrow)$ with $E_\uparrow := \{(u,v) \in E : u < v\}$
  downward graph $G_\downarrow := (V, E_\downarrow)$ with $E_\downarrow := \{(u,v) \in E : u > v\}$
- Vorwärts-Suche in $G_\uparrow$ and Rückwärtssuche in $G_\downarrow$



Datenstruktur

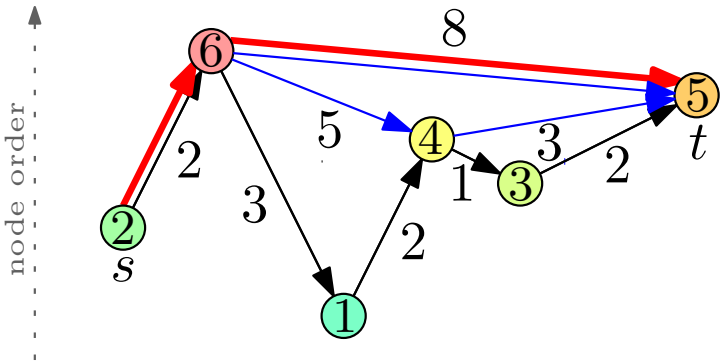
Suchgraph:

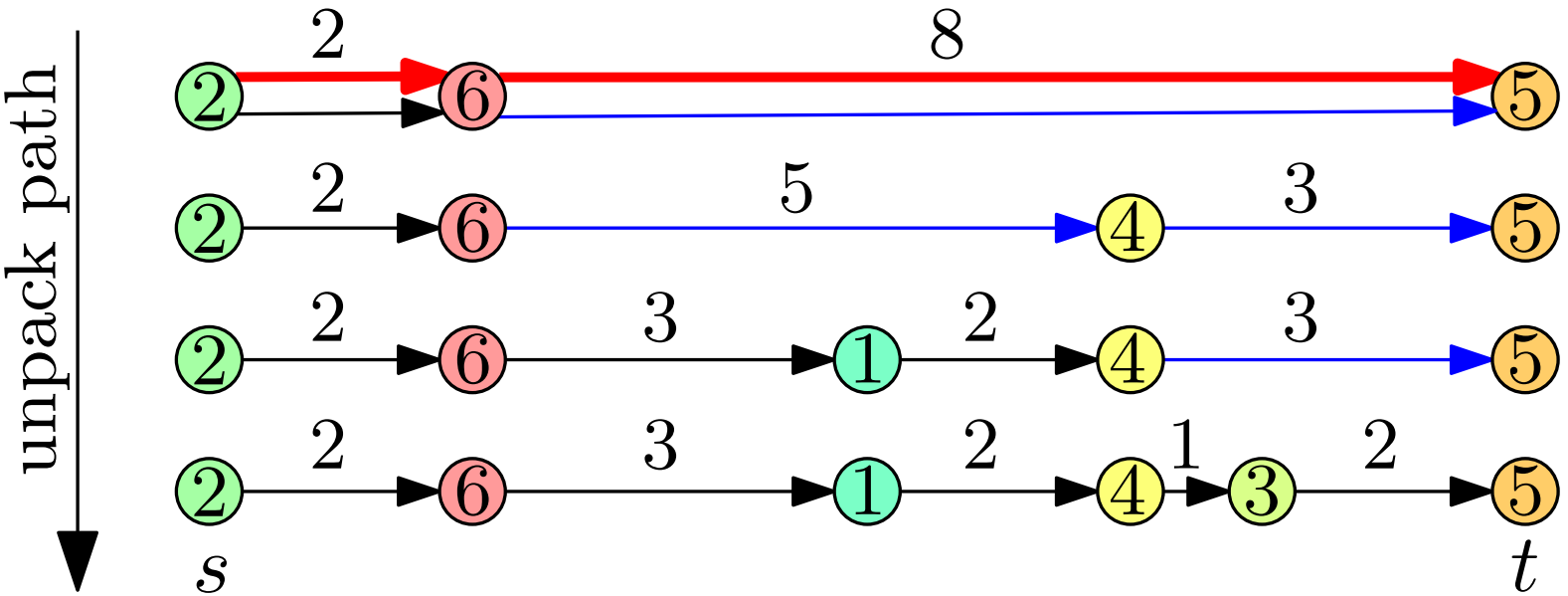
- normalerweise: speichere Kanten (v, w) in den Adjacenz-Arrays von v und w
- für die Suche reicht es aus, die Kante nur an den Knoten $\min\{v, w\}$ zu speichern
- durch ungerichtete Kanten negativer Speicherverbrauch möglich (!)



Ausgabe der Pfade

- für jeden Shortcut (u, w) eines Pfades (u, v, w), speichere Mittelknoten v an der Kante
- expandiere Pfade mittels Rekursion





Experimente

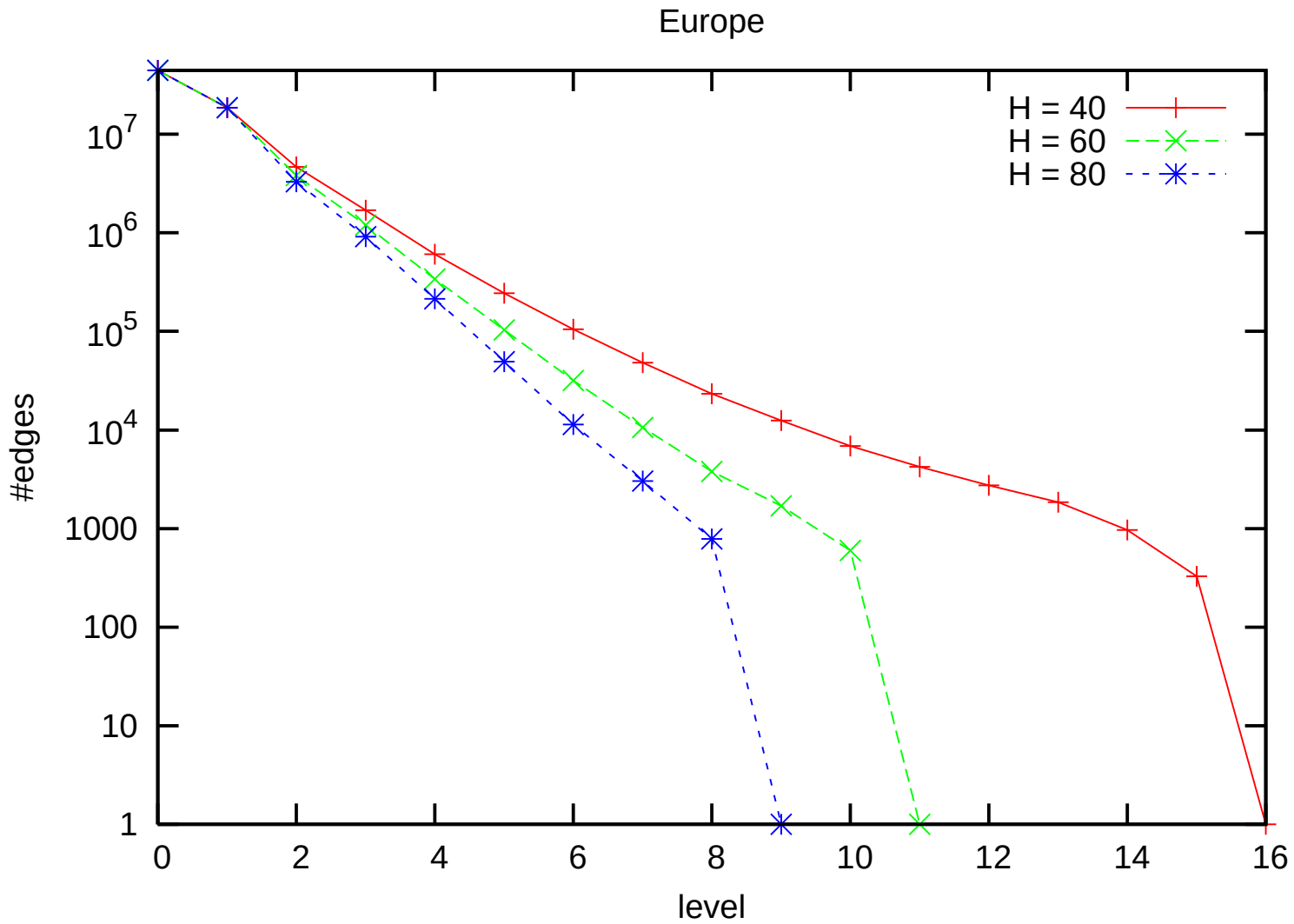
Eingaben:

- Straßennetzwerke
  - Europa: 18 Mio. Knoten, 42 Mio. Kanten
  - USA: 22 Mio. Knoten, 56 Mio. Kanten

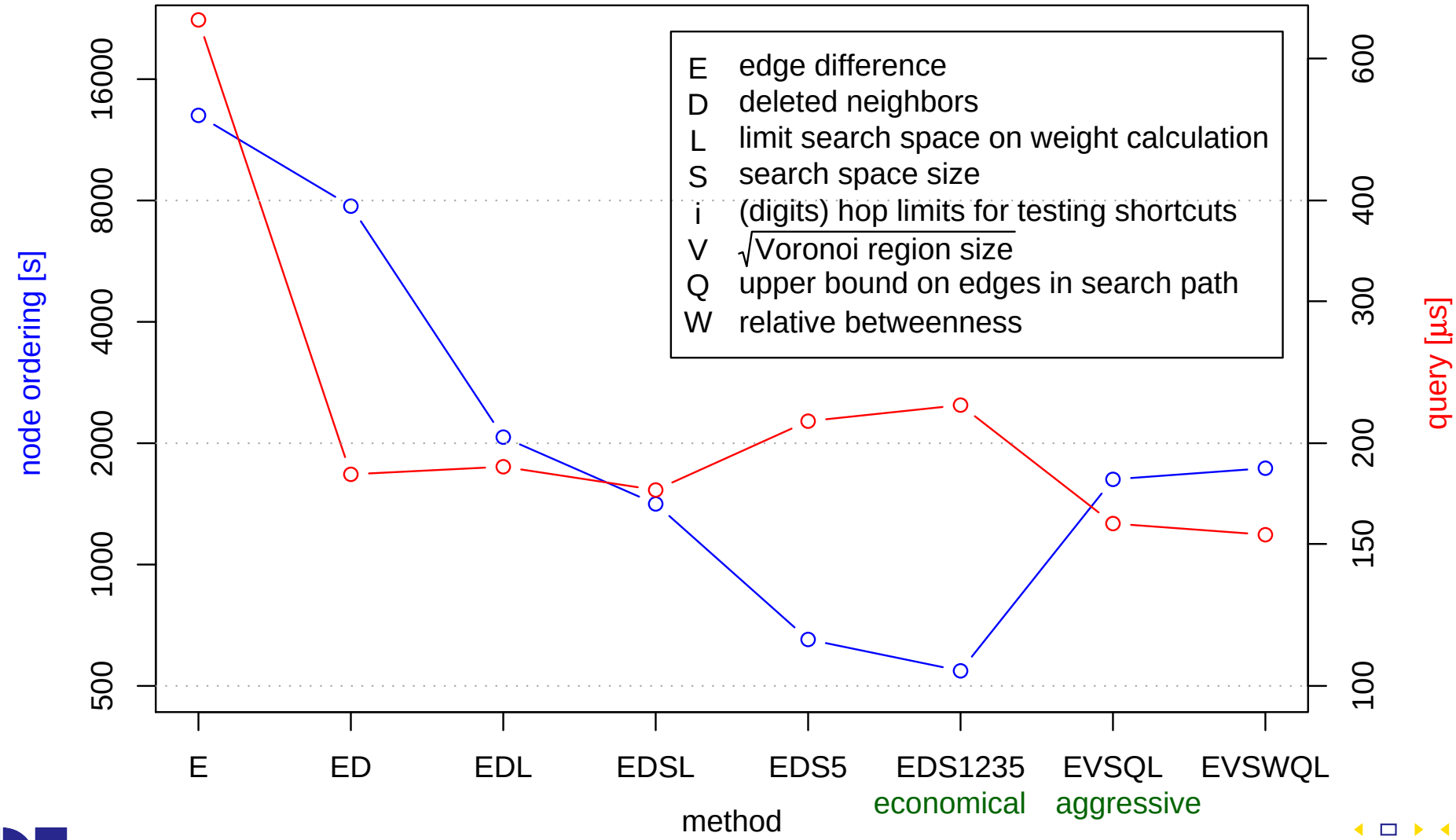
Evaluation:

- Vorberechnung in Minuten und zusätzliche Bytes pro Knoten
- durchschnittlicher Suchraum (#abgearbeitete Knoten) und Suchzeiten (in *ms*) von 10 000 Zufallsanfragen

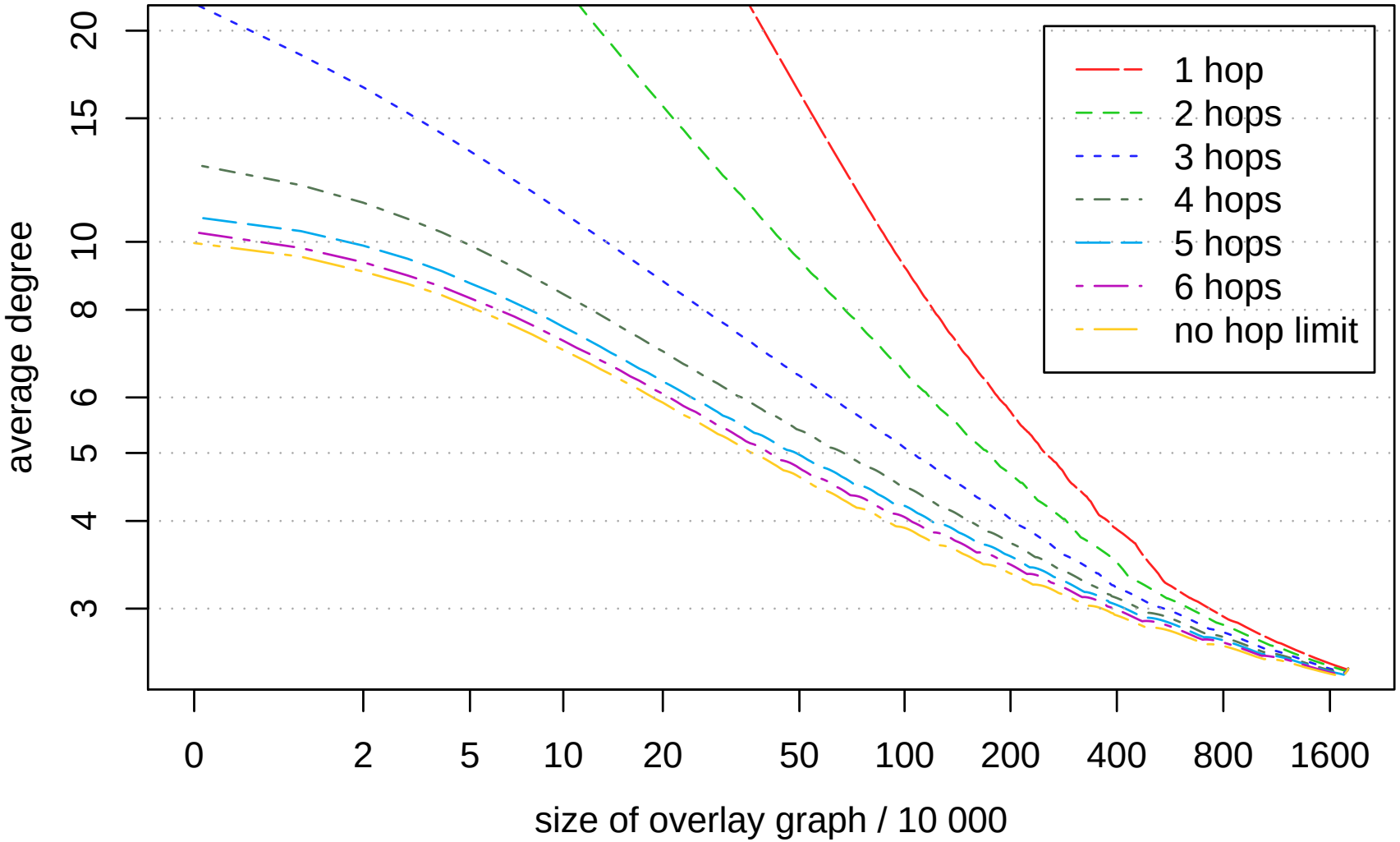
HH: Vorberechnung



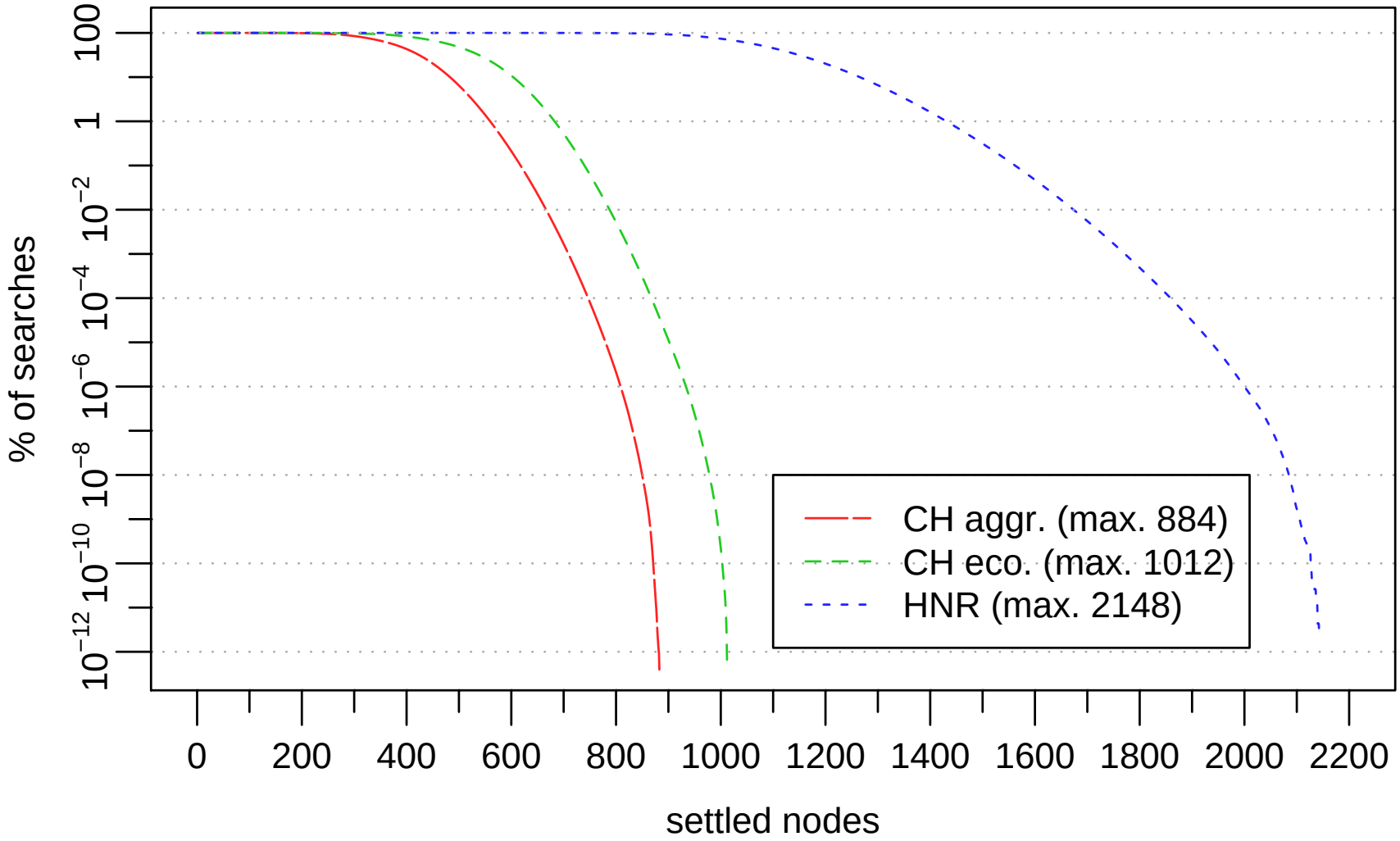
CH: Vorberechnung



CH: Knotengradentwicklung



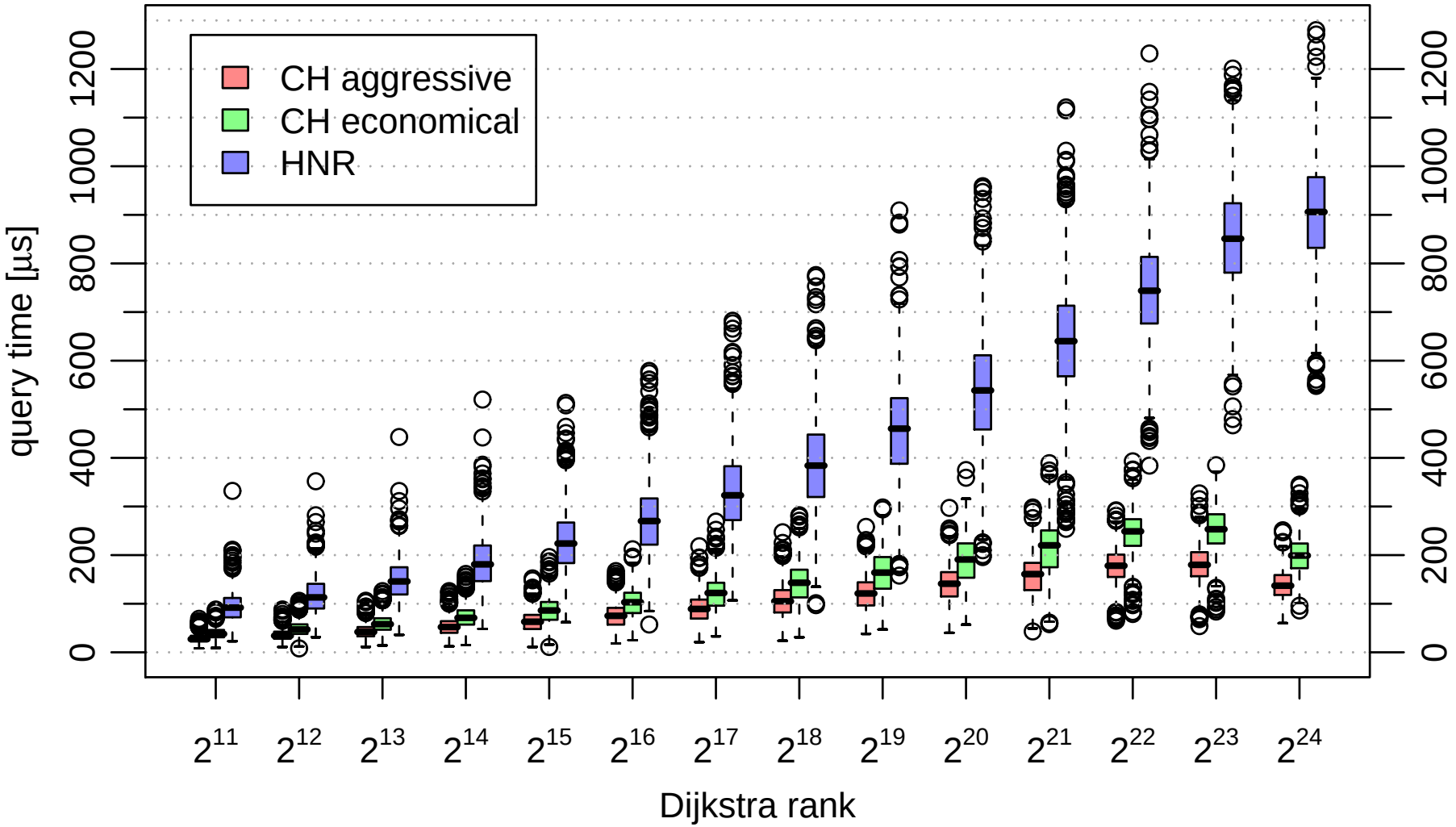
CH:Worst Case Costs



Übersicht: bisherige Techniken

	Vorberechnung		Anfrage		
	Zeit [h:m]	Platz [byte/n]	Such raum	Zeit [ms]	Beschl.
Dijkstra	0:00	0	9 114 385	5 591.6	1
ALT-16	1:25	128	74 669	53.6	104
Arc-Flags (128)	17:08	10	2 764	0.8	6 988
RE	1:22	13	4 643	3.5	1 597
REAL-(64,16)	2:20	35	679	1.1	5 037
HH	0:13	48	709	0.61	9 165
HNR	0:15	2.4	981	0.85	6 577
eco CH	0:10	0.6	459	0.22	25 413
agg CH	0:32	-3.0	359	0.15	37 273

Dijkstra Rank



Zusammenfassung Highway Hierarchies

- iteratives ausdünnen des Graphen
  - Knotenreduktion durch Kontraktion
  - Kantenreduktion durch entfernen von Nicht-Autobahn-Kanten
- Distanz-Tabelle auf oberen Levels
- schnelle Vorberechnung
- Korrektheitsbeweis schwierig
- aufwendige Datenstrukturen
- erster Ansatz, der Europa/USA effizient als Eingabe nutzen konnte

Zusammenfassung Highway-Node Routing

- Multi-Level Overlay-Graphen
- Overlay-Graph respektiert Abstände im Originalgraphen
- vereinfacht das Konzept von HH
  - einfache Datenstrukturen
  - einfacher Korrektheitsbeweis
- reduziert Speicherverbrauch gegenüber HH
- Problem: nutzt HH als ein Eingabe
- langsamer als HH

Zusammenfassung Contraction Hierarchies

- vereinfacht HNR nochmals
- n Hierarchielevel
  - Knotenreduktion durch Kontraktion
  - Kantenreduktion durch Zeugensuche
- speichere Kanten nur am unwichtigeren Knoten
- negativer Speicherverbrauch
- einfaches Konzept
- hohe Beschleunigung

Literatur

Highway Hierarchies:

- Sanders/Schultes 06,08

Highway-Node Routing:

- Schultes/Sanders 07

Contraction Hierarchies:

- Geisberger et al. 08,09

Anmerkung:

- wird auf der Homepage verlinkt