Routenfindung (3): Highway Hierarchies Star

SE Verkehrssimulation und Optimierung

Thomas Schüttler Frank Bicking

Institut für Informatik
Humboldt-Universität zu Berlin

10. Dezember 2008

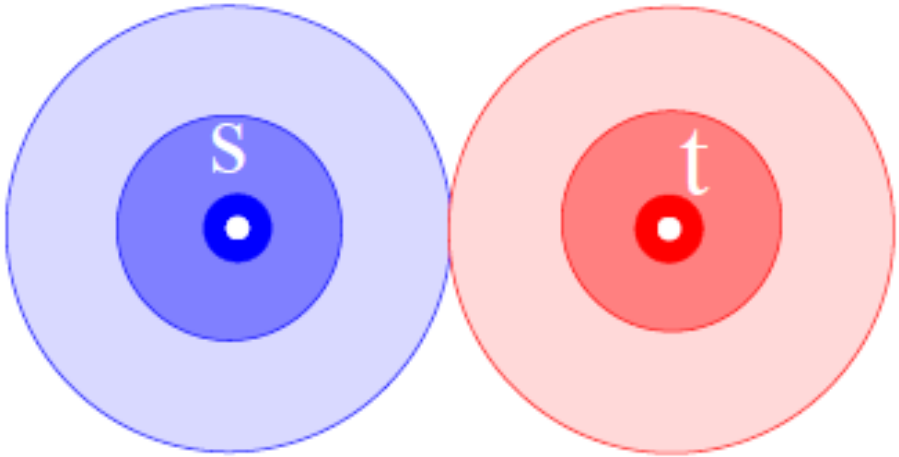
Gliederung

Gliederung

Problemstellung I

- Routenfindung in sehr großen Straßennetzen
  - statische Punkt-zu-Punkt Verbindungen
  - schnelle Vorberechnung
  - andere Algorithmen: schnellere Anfragen (Transit-Node) oder speicherfreundlicher (Highway-Node)

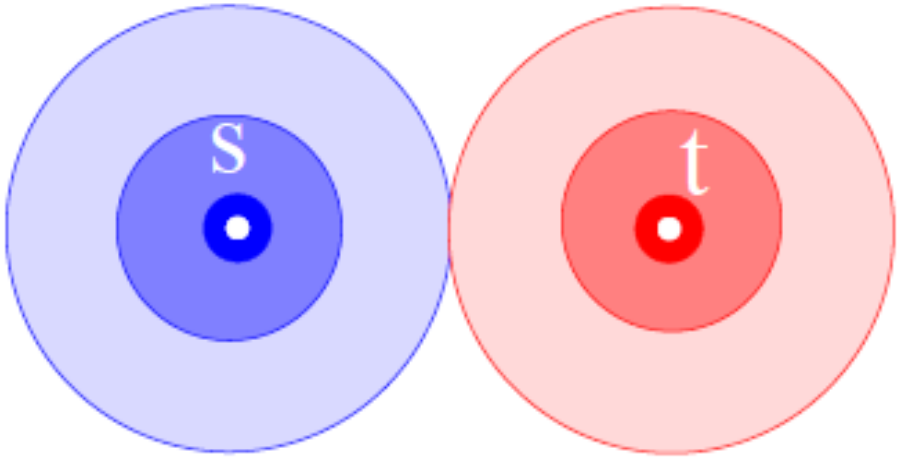
- bisher mit HH



Problemstellung I

- Routenfindung in sehr großen Straßennetzen
  - statische Punkt-zu-Punkt Verbindungen
  - schnelle Vorberechnung
  - andere Algorithmen: schnellere Anfragen (Transit-Node) oder speicherfreundlicher (Highway-Node)

- bisher mit HH



Problemstellung II

- bisher erreicht
  - exakter kürzester Pfad
  - akzeptable Vorberechnungszeiten
  - Anfragezeiten für mobile Navigation gut
- Ziele
  - zielorientierte Suche
  - effiziente Abbruchbedingungen
  - Speicher- und Zeitpotentiale

Problemstellung II

- bisher erreicht
  - exakter kürzester Pfad
  - akzeptable Vorberechnungszeiten
  - Anfragezeiten für mobile Navigation gut
- Ziele
  - zielorientierte Suche
  - effiziente Abbruchbedingungen
  - Speicher- und Zeitpotentiale

Problemstellung III

- kommerzielle Lösungen
  - 100 km vom Ziel: Autobahn
  - 20 km: Landstraße und Autobahn
  - noch näher: bidirektionale Suche über alle Straßen
- schnell, aber ungenau
- Suche mit HH*

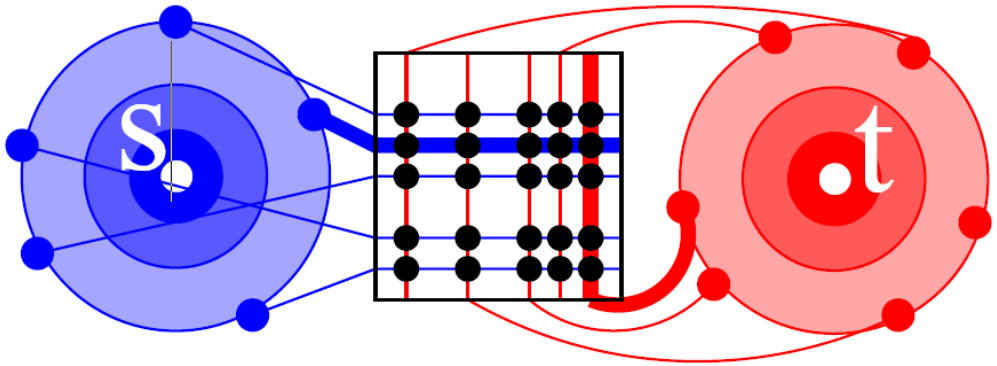
Problemstellung III

- kommerzielle Lösungen
  - 100 km vom Ziel: Autobahn
  - 20 km: Landstraße und Autobahn
  - noch näher: bidirektionale Suche über alle Straßen
- schnell, aber ungenau
- Suche mit HH*

Gliederung

Distanztabelle

- einfaches Verfahren
- Distanzen zwischen allen Knoten auf dem höchsten Kern
- Suche beendet, wenn alle Eintrittspunkte in höchsten Kern bestimmt
- Minimum aus allen Distanzen + Weglänge zu den Eintrittspunkten von Start und Ziel
- Speedup erst auf höchster Ebene



Prinzipien

- suche zum Ziel lenken
- Priorität der Knoten wird um Potential π erhöht
- äquivalent zu einem Graph mit reduzierten Kosten
- Kosten müssen positiv sein

Gliederung

zur Erinnerung: Algorithmus A*

- geschätzte Kosten $h(x)$ von x bis zum Zielknoten
- Heuristik darf nie überschätzen (geeignet: Luftlinie)

- Open List und Closed List (leer).
- Nimm Startknoten s in die Open List auf.
- Wiederhole solange Open List nicht leer:
  - Entnehme Knoten x mit geringster Priorität f_x aus Open List.
  - Wenn x Zielknoten, Terminiere mit Pfad gefunden.
  - Für alle Nachfolgerknoten v von x:
    - Wenn $v \in$ Closed List, Fortsetzung bei nächstem Knoten.
    - $f = \mathit{Kosten}(s,x) + \mathit{Kosten}(x,v) + h(v)$
    - Füge v zur Open List hinzu, wenn noch nicht enthalten oder besserer Weg gefunden.
  - Füge x zur Closed-Liste hinzu.

Gliederung

Landmarken

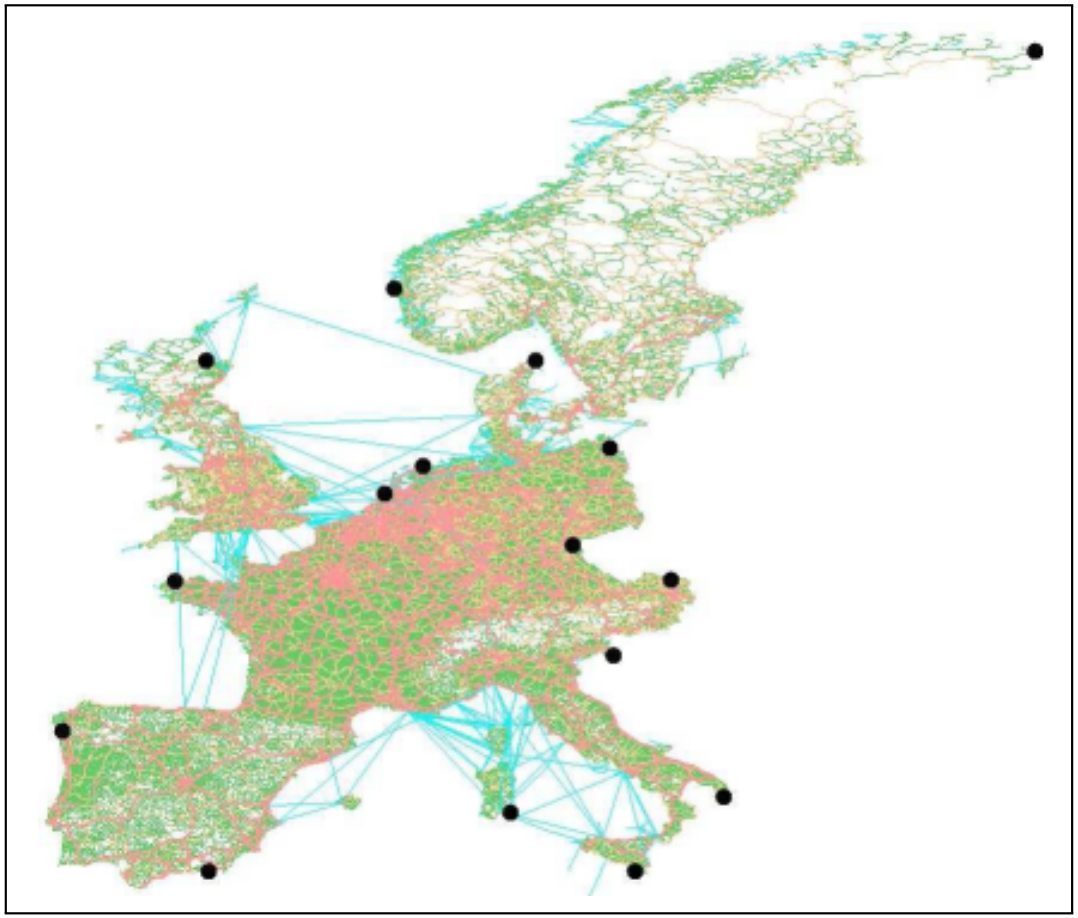


Abbildung: Landmarken

A* mit Landmarken I

- A* mit bidirektionaler Suche
- Konsistenz: $p_f = (\pi_f - \pi_r)/2 = -p_r$
- Menge von Landmarken $L \subseteq V$
- vorberechnete Abstände: $d(l, v), d(v, l)$ mit $v \in V, l \in L$

Dreiecksungleichungen

$$d(u, v) + d(v, l) \geq d(u, l)$$

$$d(l, u) + d(u, v) \geq d(l, v)$$

$\longrightarrow$ untere Schranke für $d(u, v)$

$$\underline{d}(u, v) = max_{l \in L} max\{d(u, l) - d(v, l), d(l, v) - d(l, u)\}$$

A* mit Landmarken I

- A* mit bidirektionaler Suche
- Konsistenz: $p_f = (\pi_f - \pi_r)/2 = -p_r$
- Menge von Landmarken $L \subseteq V$
- vorberechnete Abstände: $d(l, v), d(v, l)$ mit $v \in V, l \in L$

Dreiecksungleichungen

$$d(u, v) + d(v, l) \geq d(u, l)$$

$$d(l, u) + d(u, v) \geq d(l, v)$$

$\longrightarrow$ untere Schranke für $d(u, v)$

$$\underline{d}(u, v) = max_{l \in L} max\{d(u, l) - d(v, l), d(l, v) - d(l, u)\}$$

A* mit Landmarken I

- A* mit bidirektionaler Suche
- Konsistenz: $p_f = (\pi_f - \pi_r)/2 = -p_r$
- Menge von Landmarken $L \subseteq V$
- vorberechnete Abstände: $d(l, v), d(v, l)$ mit $v \in V, l \in L$

Dreiecksungleichungen

$$d(u, v) + d(v, l) \geq d(u, l)$$

$$d(l, u) + d(u, v) \geq d(l, v)$$

$\rightarrow$ untere Schranke für $d(u, v)$

$$\underline{d}(u, v) = max_{l \in L} max\{d(u, l) - d(v, l), d(l, v) - d(l, u)\}$$

A* mit Landmarken II

- ALT = A* + Landmarken + Triangulation
- Algorithmus: wie A*
- Auswahl von zwei Landmarken l_1 und l_2
  - Startknoten: l_1 mit größter unterer Schranke zum Zielknoten
  - und umgekehrt
- Schätzfunktion: aus Dreiecksungleichungen

- um Faktor 30 schneller gegenüber bidirektionalem Dijkstra
- stark von Qualität der Landmarken abhängig
- im Folgenden: Auswahlstrategien

A* mit Landmarken II

- ALT = A* + Landmarken + Triangulation
- Algorithmus: wie A*
- Auswahl von zwei Landmarken l_1 und l_2
  - Startknoten: l_1 mit größter unterer Schranke zum Zielknoten
  - und umgekehrt
- Schätzfunktion: aus Dreiecksungleichungen

- um Faktor 30 schneller gegenüber bidirektionalem Dijkstra
- stark von Qualität der Landmarken abhängig
- im Folgenden: Auswahlstrategien

Gliederung

Avoid

- Idee: Ermittle Bereiche, die wenig abgedeckt sind.

- Wähle zufälligen Knoten r als erste Landmarke.
- Wiederhole:
  - Bestimme Baum mit kürzesten Pfaden T_r ausgehend von r:
  - Ist r ein Blatt, füge r zu Landmarken hinzu und breche ab.
  - Für jeden Kindknoten:
    - Gewicht: $w_v = d(v, r) - \underline{d}(v, r)$
    - Größe: $s_v = w_v +$ Größe aller Kindknoten in T_r
    - Enthält T_v eine Landmarke, setze $s_v = 0$.
  - Setze r auf den Kindknoten mit maximaler Größe.

- Auswahl der Folgeknoten mit Wahrscheinlichkeit proportional zum Quadrat des Abstands zur nächsten Landmarke

Avoid

- Idee: Ermittle Bereiche, die wenig abgedeckt sind.

- Wähle zufälligen Knoten r als erste Landmarke.
- Wiederhole:
  - Bestimme Baum mit kürzesten Pfaden T_r ausgehend von r:
  - Ist r ein Blatt, füge r zu Landmarken hinzu und breche ab.
  - Für jeden Kindknoten:
    - Gewicht: $w_v = d(v, r) - \underline{d}(v, r)$
    - Größe: $s_v = w_v +$ Größe aller Kindknoten in T_r
    - Enthält T_v eine Landmarke, setze $s_v = 0$.
  - Setze r auf den Kindknoten mit maximaler Größe.

- Auswahl der Folgeknoten mit Wahrscheinlichkeit proportional zum Quadrat des Abstands zur nächsten Landmarke

MaxCover

- Nachteil von Avoid:
  - Resultate abhängig von zufällig ausgewähltem Startknoten.

MaxCover

- Idee: Berechne *n*-mal größere Menge von Landmarken mit Avoid.
- Ermittle geeignete Kandidaten durch lokale Suche.

- langsam: ca. 90 Minuten für 16 Landmarken

Core Landmarks

- MaxCover für Europa: ca. 75 Minuten
- Highway Hierarchie: ca. 15 Minuten

Idee: Nutze Highway Hierarchien

Reduziere mögliche Landmarken durch Highway Hierarchien.

- höherer Kern als Basis statt Originalgraph
- reduzierte Knotenzahl $\longrightarrow$ viel schnellere Berechnung
  - 16 MaxCover-Landmarken auf Level 3 in einer Minute
- Nachteile:
  - Highway Hierarchien tendieren zur Kartenmitte
  - Knoten am Rand wären dagegen geeignetere Landmarken
  - Landmarken in den unteren ebenen nicht nutzbar
  - ungerichtete Suche bis alle Eintrittspunkte in Kern bestimmt

Core Landmarks

- MaxCover für Europa: ca. 75 Minuten
- Highway Hierarchie: ca. 15 Minuten

Idee: Nutze Highway Hierarchien

Reduziere mögliche Landmarken durch Highway Hierarchien.

- höherer Kern als Basis statt Originalgraph
- reduzierte Knotenzahl $\longrightarrow$ viel schnellere Berechnung
  - 16 MaxCover-Landmarken auf Level 3 in einer Minute
- Nachteile:
  - Highway Hierarchien tendieren zur Kartenmitte
  - Knoten am Rand wären dagegen geeignetere Landmarken
  - Landmarken in den unteren ebenen nicht nutzbar
  - ungerichtete Suche bis alle Eintrittspunkte in Kern bestimmt

Core Landmarks

- MaxCover für Europa: ca. 75 Minuten
- Highway Hierarchie: ca. 15 Minuten

Idee: Nutze Highway Hierarchien

Reduziere mögliche Landmarken durch Highway Hierarchien.

- höherer Kern als Basis statt Originalgraph
- reduzierte Knotenzahl $\longrightarrow$ viel schnellere Berechnung
  - 16 MaxCover-Landmarken auf Level 3 in einer Minute
- Nachteile:
  - Highway Hierarchien tendieren zur Kartenmitte
  - Knoten am Rand wären dagegen geeignetere Landmarken
  - Landmarken in den unteren ebenen nicht nutzbar
  - ungerichtete Suche bis alle Eintrittspunkte in Kern bestimmt

Core Landmarks

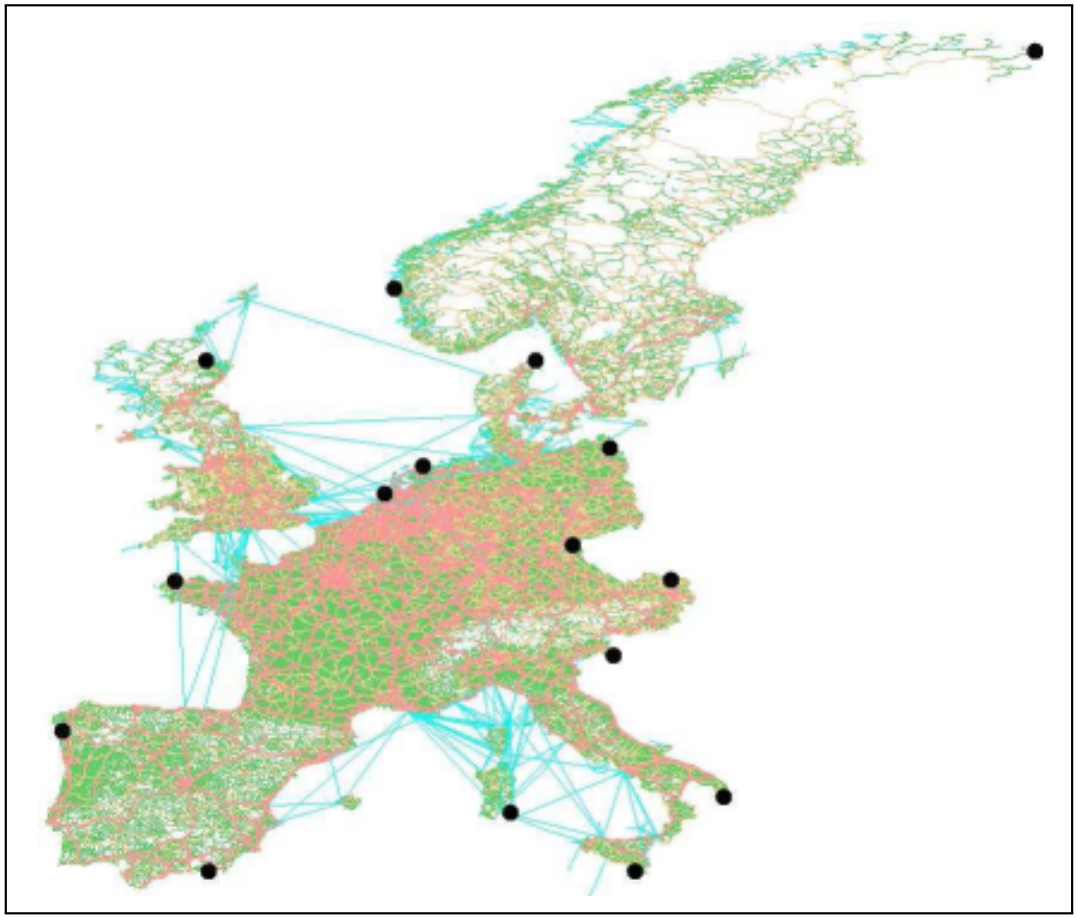


Abbildung: 16 core-1 Landmarken des westeuropäischen Straßennetzes

Gliederung

Highway Hierarchies *

- HH* kombiniert:
  - Highway Hierachien
  - bidirektionales A*
  - vorberechente Landmarken
  - Dreiecksungleichungen
  - vorberechnete Distanztabelle für höchsten Kern

Highway Hierarchies *

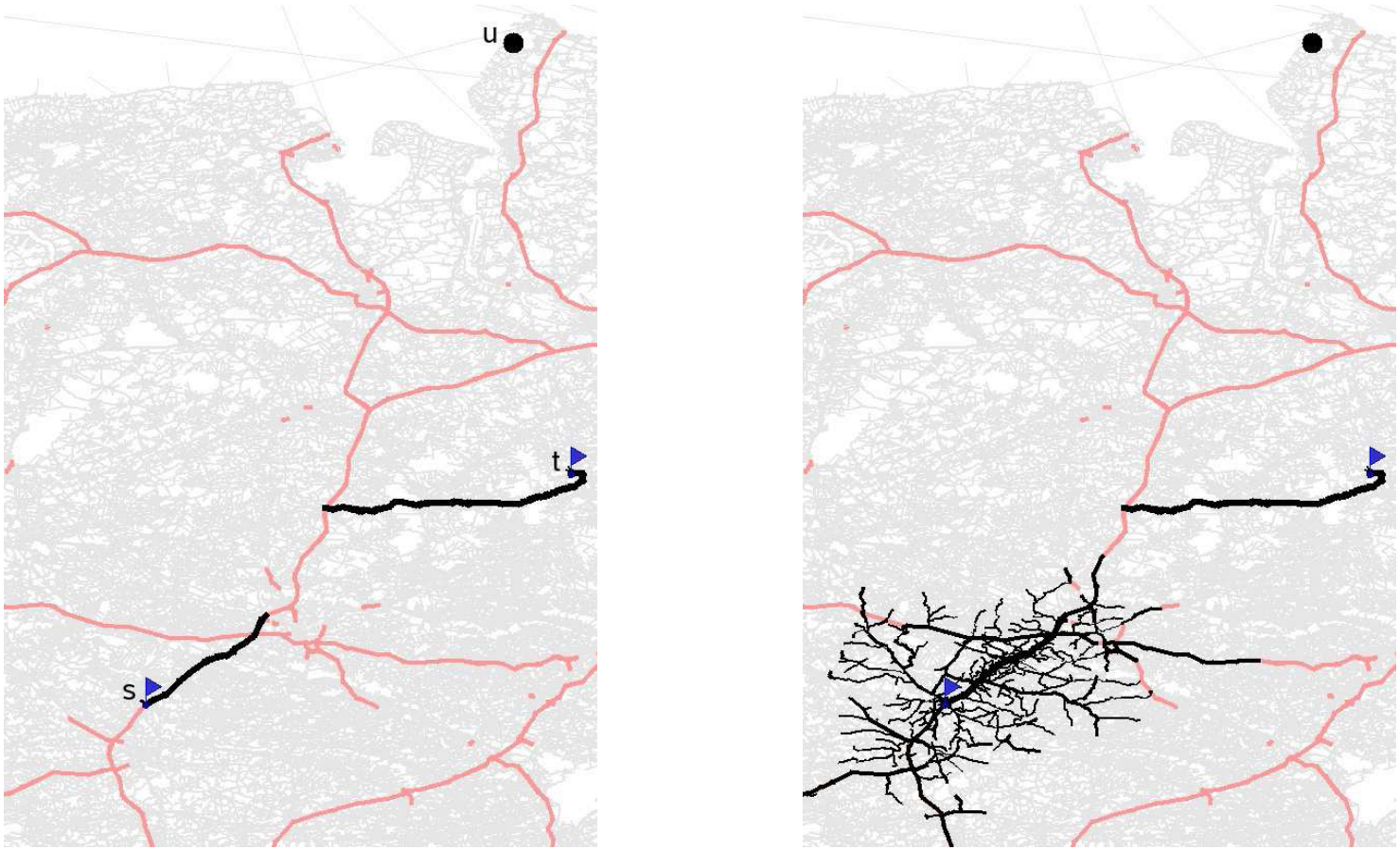


Abbildung: Suche mit Highway Hierachies *

Vorgehensweise bei der Suche I

- bidirektionale Suche
  - iterativ
  - bevorzuge jeweils die Richtung mit weniger Eintrittspunkten
  - wie HH: kein Abbruch bei Aufeinandertreffen der Suchbereiche

- Landmarken
  - Auswahl beim Start: nur eine für jede Richtung
  - zur Abschätzung der unteren Schranke für A*
  - wähle Landmarke mit der höchsten unteren Schranke für $d(s,t)$

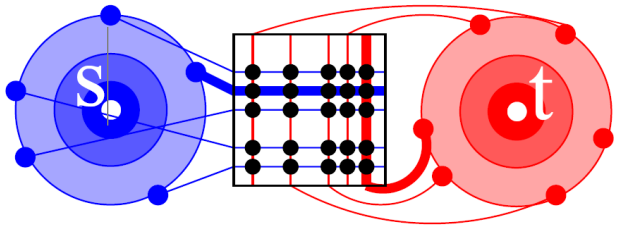
Vorgehensweise bei der Suche I

- bidirektionale Suche
  - iterativ
  - bevorzuge jeweils die Richtung mit weniger Eintrittspunkten
  - wie HH: kein Abbruch bei Aufeinandertreffen der Suchbereiche

- Landmarken
  - Auswahl beim Start: nur eine für jede Richtung
  - zur Abschätzung der unteren Schranke für A*
  - wähle Landmarke mit der höchsten unteren Schranke für $d(s,t)$

Vorgehensweise bei der Suche II

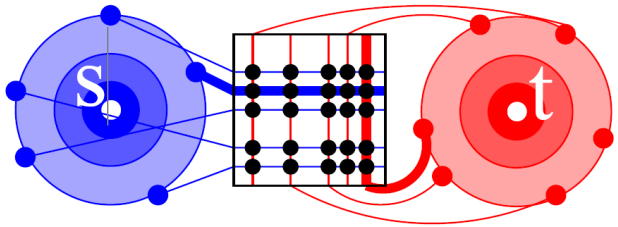
- einfacher Ansatz:
  - beginne mit ungerichteter Suche
  - bestimme alle Eintrittspunkte aus beiden Richtungen
  - ermittle minimalen Weg aus vorberechneter Distanztabelle
  - weiter mit zielgerichteter Suche im ersten Kern



- besser: bei Entdeckung eines Eintrittspunktes u:
  - betrachte alle Paare (u, v), v Eintrittspunkt aus Gegenrichtung
  - wenn möglich, verbessere vorläufigen kürzesten Pfad μ
  - veränderte Reihenfolge, gleicher Suchraum

Vorgehensweise bei der Suche II

- einfacher Ansatz:
  - beginne mit ungerichteter Suche
  - bestimme alle Eintrittspunkte aus beiden Richtungen
  - ermittle minimalen Weg aus vorberechneter Distanztabelle
  - weiter mit zielgerichteter Suche im ersten Kern



- besser: bei Entdeckung eines Eintrittspunktes u:
  - betrachte alle Paare (u, v), v Eintrittspunkt aus Gegenrichtung
  - wenn möglich, verbessere vorläufigen kürzesten Pfad μ
  - veränderte Reihenfolge, gleicher Suchraum

Vorgehensweise bei der Suche III

- Pruning
  - Pfad von s nach t
    $\longrightarrow$ obere Schranke μ für kürzesten Pfad.
  - $d(s,u) + \underline{d}(u,t) > \mu$
    $\longrightarrow$ u braucht nicht weiter verfolgt zu werden.
  - Identische untere Schranke für Pruning und Prioritäten:
    - Kriterium gilt für *alle* Knoten in der gleichen Prioritätswarteschlange.
    - Abbruchbedingung: Suche in diese Richtung kann gestoppt werden.

Vorgehensweise bei der Suche III

- Pruning
  - Pfad von s nach t
    $\rightarrow$ obere Schranke μ für kürzesten Pfad.
  - $d(s,u) + \underline{d}(u,t) > \mu$
    $\rightarrow$ u braucht nicht weiter verfolgt zu werden.
  - Identische untere Schranke für Pruning und Prioritäten:
    - Kriterium gilt für *alle* Knoten in der gleichen Prioritätswarteschlange.
    - Abbruchbedingung: Suche in diese Richtung kann gestoppt werden.

Vorgehensweise bei der Suche III

- Pruning
  - Pfad von s nach t
    $\longrightarrow$ obere Schranke μ für kürzesten Pfad.
  - $d(s,u) + \underline{d}(u,t) > \mu$
    $\longrightarrow$ u braucht nicht weiter verfolgt zu werden.
  - Identische untere Schranke für Pruning und Prioritäten:
    - Kriterium gilt für *alle* Knoten in der gleichen Prioritätswarteschlange.
    - Abbruchbedingung: Suche in diese Richtung kann gestoppt werden.

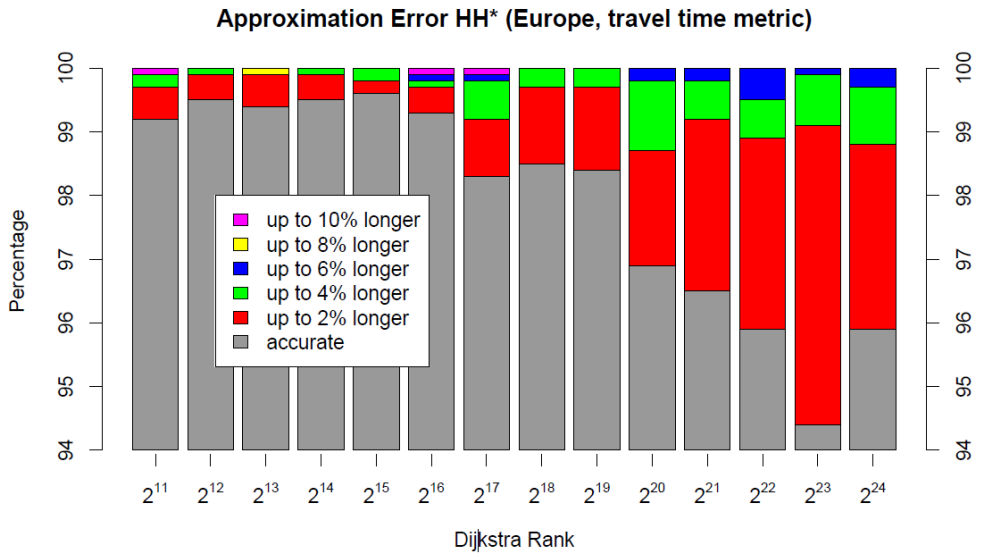
Gliederung

Näherungslösungen I

- einen Pfad finden geht sehr schnell
- Prüfen, das es der kürzeste ist dauert lange
- Optimalität vernachlässigen und neue Abbruchbedingung
- suchen nach einem Pfad der Länge P
  - $P \leq (1 + \varepsilon) \cdot OPT$
- deutlich verkürzte Suchzeiten
- Garantien werden eingehalten

Näherungslösungen II

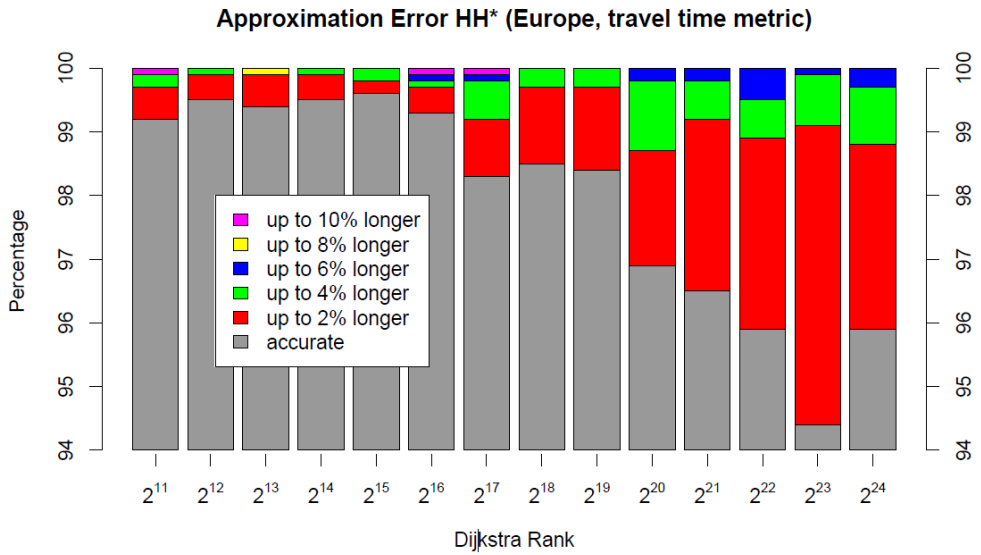
- Verteilung der Schätzungsfehler für Westeuropa bei Zeitmetrik
- garantierer maximaler Fehler $\varepsilon = 10\%$
- Dijkstra Rang ist die Anzahl der Knoten die beim normalen Dijkstra angefasst worden wären



- 94% liefern genaues Ergebnis
- totaler Fehler bei 0.056%

Näherungslösungen II

- Verteilung der Schätzungsfehler für Westeuropa bei Zeitmetrik
- garantierer maximaler Fehler $\varepsilon = 10\%$
- Dijkstra Rang ist die Anzahl der Knoten die beim normalen Dijkstra angefasst worden wären



- 94% liefern genaues Ergebnis
- totaler Fehler bei 0.056%

Gliederung

bessere obere Schranken

- bei der Suche kann man schon früher auf Knoten treffen, die in der Distanztabelle sind auch wenn man noch nicht in dem höchsten Kern ist
- diese Distanz könnte länger sein, als die man auf niederen Kernen erzielen kann
- deswegen nicht sofort als kürzester Pfad sondern nur obere Schranke
- aber: finden oberer Schranken funktioniert bereits sehr gut
- nutzbar vor allem für Näherungssuche
  - wenn Suche beendet, Prüfung der Knoten in den Prioritätswarteschlangen ob noch eine Verbesserung möglich

Gliederung

Limitierung der Komponentengröße

- Anzahl der Sprünge in einer Abkürzung begrenzen
- Komponenten der übersprungenen Knoten werden kleiner
- diese werden bei der Suche von Start bzw Ziel zum Eintrittspunkt benötigt
- zusätzlich positiver Einfluss auf worst-case Zeiten
  - große Komponenten in schwach besetzten Gebieten

Gliederung

Ergebnisse I

- Vergleich der unterschiedlichen HH-Arten
- ∅ - HH Pur, DT - HH mit Distanztabelle, ALT - HH mit ALT, both - HH*
- Wert in Klammern entspricht Näherungslösung
- disk-space inklusive Originalgraph

metric		Europe ∅	Europe DT	Europe ALT	Europe both	USA ∅	USA DT	USA ALT	USA both
time	preproc. time [min]	16	18	19	21	22	25	26	27
	total disk space [MB]	886	1 273	1 326	1 713	1 129	1 574	1 743	2 188
	#settled nodes	1 662	916	916	686 (176)	1 966	1 098	1 027	787 (162)
	query time [ms]	1.49	0.79	1.04	0.68 (0.21)	1.58	0.89	1.05	0.73 (0.21)
dist	preproc. time [min]	46	46	49	48	54	56	58	58
	total disk space [MB]	894	1 506	1 337	1 948	1 140	1 721	1 754	2 335
	#settled nodes	10 284	5 067	3 347	2 138 (177)	9 706	5 477	2 784	2 021 (169)
	query time [ms]	10.93	6.02	4.33	2.54 (0.30)	9.74	6.24	3.42	2.23 (0.33)

- Unterschiede zwischen Zeit- und Distanzmetrik
- USA: Distanz relativ zu Europa besser
- wesentliche Einsparung bei Näherungslösung
- kaum mehr Vorberechnung, aber mehr Platzbedarf

Ergebnisse I

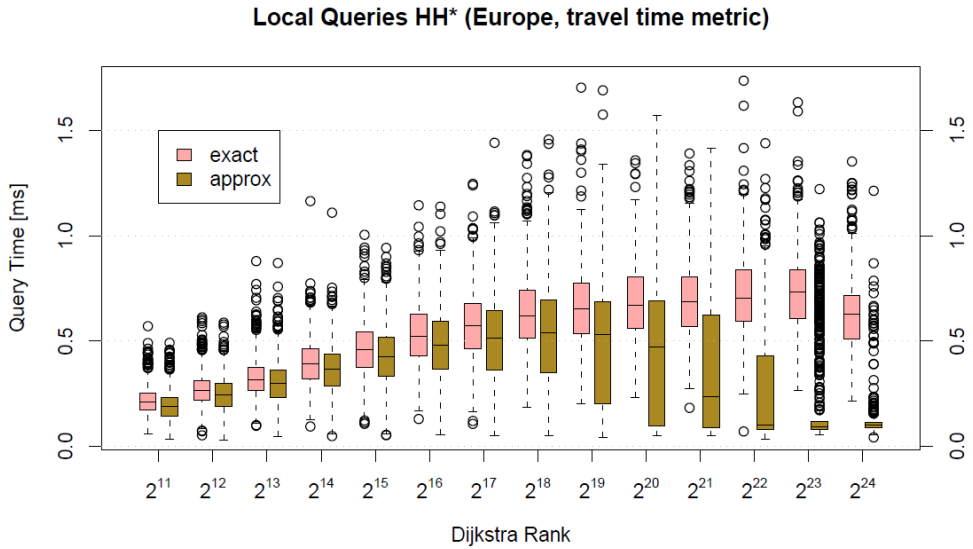
- Vergleich der unterschiedlichen HH-Arten
- ∅ - HH Pur, DT - HH mit Distanztabelle, ALT - HH mit ALT, both - HH*
- Wert in Klammern entspricht Näherungslösung
- disk-space inklusive Originalgraph

metric		Europe ∅	Europe DT	Europe ALT	Europe both	USA ∅	USA DT	USA ALT	USA both
time	preproc. time [min]	16	18	19	21	22	25	26	27
	total disk space [MB]	886	1 273	1 326	1 713	1 129	1 574	1 743	2 188
	#settled nodes	1 662	916	916	686 (176)	1 966	1 098	1 027	787 (162)
	query time [ms]	1.49	0.79	1.04	0.68 (0.21)	1.58	0.89	1.05	0.73 (0.21)
dist	preproc. time [min]	46	46	49	48	54	56	58	58
	total disk space [MB]	894	1 506	1 337	1 948	1 140	1 721	1 754	2 335
	#settled nodes	10 284	5 067	3 347	2 138 (177)	9 706	5 477	2 784	2 021 (169)
	query time [ms]	10.93	6.02	4.33	2.54 (0.30)	9.74	6.24	3.42	2.23 (0.33)

- Unterschiede zwischen Zeit- und Distanzmetrik
- USA: Distanz relativ zu Europa besser
- wesentliche Einsparung bei Näherungslösung
- kaum mehr Vorberechnung, aber mehr Platzbedarf

Ergebnisse II

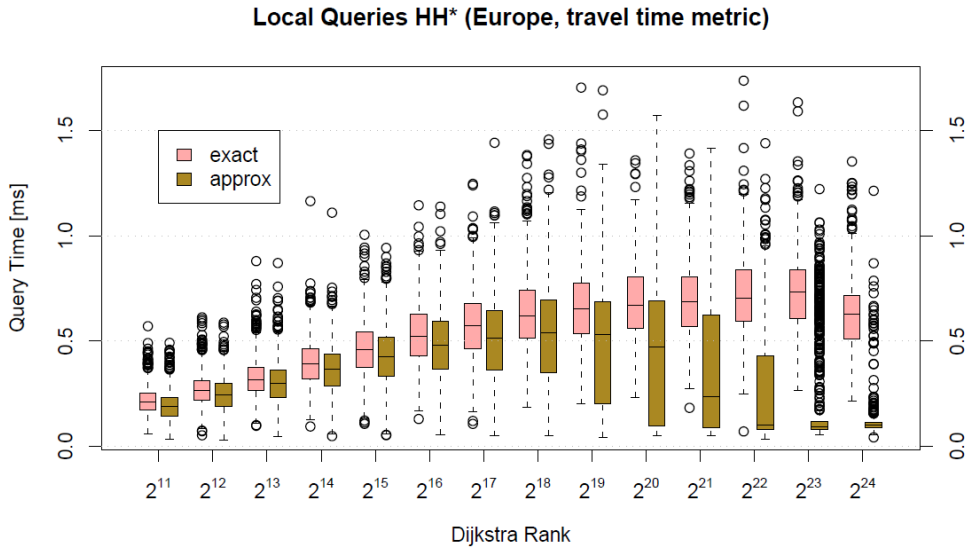
- Verteilung der Anfragezeiten für Westeuropa bei Zeitmetrik
- Vergleich: genaue Lösung und Näherungslösung



- lange Anfragen profitieren zunehmend von Näherungen
- lange Anfragen werden durch Distanztabelle beschränkt
- prozentual erlaubter Fehler führt zu früherem Abbrechen bei Näherung
- Varianz der Anfragezeiten zunehmend

Ergebnisse II

- Verteilung der Anfragezeiten für Westeuropa bei Zeitmetrik
- Vergleich: genaue Lösung und Näherungslösung



- lange Anfragen profitieren zunehmend von Näherungen
- lange Anfragen werden durch Distanztabelle beschränkt
- prozentual erlaubter Fehler führt zu früherem Abbrechen bei Näherung
- Varianz der Anfragezeiten zunehmend

Gliederung

Zusammenfassung

- Speedup von HH von ≈5000 gegenüber bidirektionalen Dijkstra
- HH* 2,2 mal schneller als HH
- mit Näherungslösung weitere 3,2 mal schneller
- Zeit- und Distanzmetriken möglich
- schnelle Vorberechnung
- Anfragezeiten und Speicherverbrauch praxistauglich für mobile Navigation
- für Verkehrssimulation eher nicht
- offene Probleme:
  - Kombination von Metriken: Zeit, Distanz, Verbrauch, Maut ...
  - lokale Updates des Straßennetzes: Staus, Baustellen ...

Zusammenfassung

- Speedup von HH von ≈5000 gegenüber bidirektionalen Dijkstra
- HH* 2,2 mal schneller als HH
- mit Näherungslösung weitere 3,2 mal schneller
- Zeit- und Distanzmetriken möglich
- schnelle Vorberechnung
- Anfragezeiten und Speicherverbrauch praxistauglich für mobile Navigation
- für Verkehrssimulation eher nicht
- offene Probleme:
  - Kombination von Metriken: Zeit, Distanz, Verbrauch, Maut ...
  - lokale Updates des Straßennetzes: Staus, Baustellen ...

Zusammenfassung

- Speedup von HH von ≈5000 gegenüber bidirektionalen Dijkstra
- HH* 2,2 mal schneller als HH
- mit Näherungslösung weitere 3,2 mal schneller
- Zeit- und Distanzmetriken möglich
- schnelle Vorberechnung
- Anfragezeiten und Speicherverbrauch praxistauglich für mobile Navigation
- für Verkehrssimulation eher nicht
- offene Probleme:
  - Kombination von Metriken: Zeit, Distanz, Verbrauch, Maut ...
  - lokale Updates des Straßennetzes: Staus, Baustellen ...