Algorithmen für Routenplanung

7. Sitzung, Sommersemester 2014

Prof. Christos Zaroliagis | 07. Mai 2014

INSTITUT FÜR THEORETISCHE INFORMATIK · ALGORITHMIK · PROF. DR. DOROTHEA WAGNER

Kürzeste Wege in Straßennetzwerken

Beschleunigungstechniken (Fortsetzung)

- Reach
- CRP

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Hierarchische Techniken

Bisher: bidirektionale Suche, zielgerichtete Suche (ALT, Arc-Flags)

Heute: Nutze “Hierarchie” im Graphen

in etwa: Seiten-, Haupt-, Land-, Bundesstraßen, Autobahnen

Ideensammlung:

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Hierarchische Techniken

Bisher: bidirektionale Suche, zielgerichtete Suche (ALT, Arc-Flags)

Heute: Nutze “Hierarchie” im Graphen

in etwa: Seiten-, Haupt-, Land-, Bundesstraßen, Autobahnen

Ideensammlung:

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

Motivation 1: Netzwerkanalyse

- Zentralitätsmaße bewerten Wichtigkeit von Knoten oder Kanten in einem Netzwerk
- Geläufige Beispiele
  - Google Page Rank
  - Erdös-Zahl

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zentralitätsmaße - Beispiele

- degree centrality

$$C_D(v) = \frac{\deg(v)}{n-1} \in [0,1]$$

- betweenness centrality

$$C_B(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}} \in [0,1]$$

wobei

- σ_{st} die Anzahl der kürzesten s-t-Wege ist (meistens 1)
- $\sigma_{st}(v)$ die Anzahl solcher Wege, die durch v gehen

Geeignetes Maß für Routenplanung?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Reach

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Reach

Zentralitätsmaß, das groß ist, falls ein Knoten in der Mitte eines langen kürzesten Weges liegt.

Vorberechnung:

- Zeichne um jeden Knoten u einen Kreis mit Radius $r(u)$, so dass:
  - für jedes Paar s, t gilt: wenn u auf kürzestem s–t-Weg liegt, ist entweder s oder t im Kreis.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Reach

Zentralitätsmaß, das groß ist, falls ein Knoten in der Mitte eines langen kürzesten Weges liegt.

Vorberechnung:

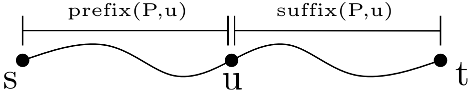
- Zeichne um jeden Knoten u einen Kreis mit Radius $r(u)$, so dass:
  - für jedes Paar s, t gilt: wenn u auf kürzestem s–t-Weg liegt, ist entweder s oder t im Kreis.

Strategie für Beschleunigungstechnik:

- Beachte Knoten u nicht, wenn s und t nicht im Kreis um u liegen.
- wie überprüfen?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

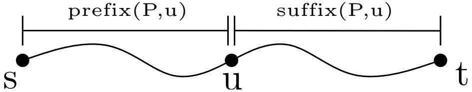
Das Zentralitätsmaß Reach



Definition:

- Sei $P = \langle s, \ldots, u, \ldots, t \rangle$ Pfad durch u
- dann Reach von u bezüglich P:

Das Zentralitätsmaß Reach

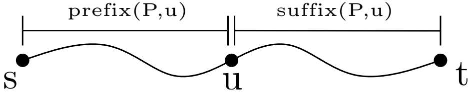


Definition:

- Sei $P = \langle s, \ldots, u, \ldots, t \rangle$ Pfad durch u
- dann Reach von u bezüglich P:

$$r_P(u) := \min\{\mathrm{len}(\mathit{prefix}(P,u)), \mathrm{len}(\mathit{suffix}(P,u))\}$$

Das Zentralitätsmaß Reach



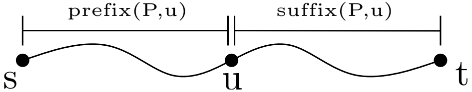
Definition:

- Sei $P = \langle s, \dots, u, \dots, t \rangle$ Pfad durch u
- dann Reach von u bezüglich P:

$$r_P(u) := \min\{\text{len}(\textit{prefix}(P, u)), \text{len}(\textit{suffix}(P, u))\}$$

- Reach von u:

Das Zentralitätsmaß Reach



Definition:

- Sei $P = \langle s, \ldots, u, \ldots, t \rangle$ Pfad durch u
- dann Reach von u bezüglich P:

$$r_P(u) := \min\{\text{len}(\mathit{prefix}(P, u)), \text{len}(\mathit{suffix}(P, u))\}$$

- Reach von u:
  Maximum der Reachwerte bezüglich aller kürzesten Pfade durch u:

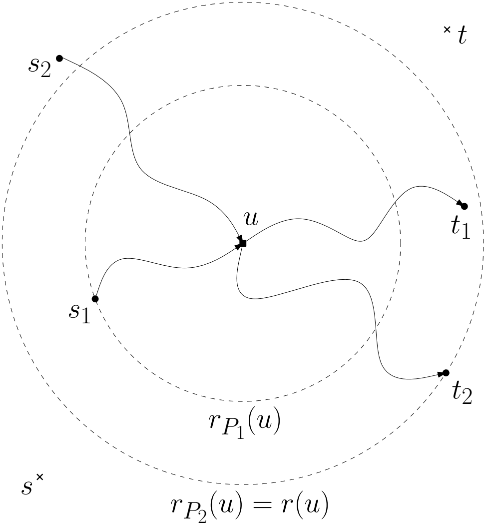
$$r(u) := \max\{r_P(u) \mid P \text{ kürzester Weg mit } u \in P\}$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Reach-Pruning

somit:

- Reach $r(u)$ von u gibt Suffix oder Prefix des längsten kürzesten Weges durch u
- wenn für u während Query $r(u) < d(s,u)$ und $r(u) < d(u,t)$ gilt, muss u nicht beachtet werden



Unidirectional Reach Dijkstra

ReachDijkstra($G = (V, E)$, s, t)

$d[s] = 0$
$Q.clear()$, $Q.add(s, 0)$
while $!Q.empty()$ **do**
  $u \leftarrow Q.deleteMin()$
  break if $u = t$; // Stoppkriterium
  if $r(u) < d[u]$ **and** $r(u) < d(u, t)$ **then continue**
  forall *edges* $e = (u, v) \in E$ **do**
    if $d[u] + \text{len}(e) < d[v]$ **then**
      $d[v] \leftarrow d[u] + \text{len}(e)$
      if $v \in Q$ **then** $Q.decreaseKey(v, d[v])$
      else $Q.insert(v, d[v])$

Problem

Problem?

Problem

Problem:

- Abfrage $r(u) < d(u, t)$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Problem

Problem:

- Abfrage $r(u) < d(u, t)$

- Im geometrischen Fall ist die Überprüfung einfach
- Auch möglich: benutze Landmarken
- Weiteres?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bidir. Self-Bounding Reach-Dijkstra

Idee: Bidirektionale Suche

$$r(u) < d(s,u) \quad \rightsquigarrow \quad r(u) < d_f[u]$$

$$r(u) < d(u,t) \quad \rightsquigarrow \quad r(u) < d_b[u]$$

Vorwärtssuche:

- ignoriere knoten u, wenn $r(u) < d_f[u]$ gilt
- überlasse den Check $r(u) < d(u,t)$ der Rückwärtssuche
- Rückwärtssuche analog
- ändere das Stoppkriterium

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bidir. Self-Bounding Reach-Dijkstra

Idee: Bidirektionale Suche

$$r(u) < d(s,u) \quad \rightsquigarrow \quad r(u) < d_f[u]$$

$$r(u) < d(u,t) \quad \rightsquigarrow \quad r(u) < d_b[u]$$

Vorwärtssuche:

- ignoriere knoten u, wenn $r(u) < d_f[u]$ gilt
- überlasse den Check $r(u) < d(u,t)$ der Rückwärtssuche
- Rückwärtssuche analog
- ändere das Stoppkriterium

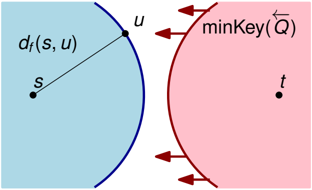
neues Stoppkriterium:

- stoppe eine Suchrichtung, wenn $\text{minKey}(Q) > \mu/2$ oder Queue leer
- stoppe Anfrage, wenn beide Suchrichtungen gestoppt haben
- Korrektheit: gute Fingerübung

Bidir. Dist.-Bounding Reach-Dijkstra

Idee (für Vorwärtssuche):

- wenn u von Rückwärtssuche noch nicht erreicht, ist minKey($\overleftarrow{Q}$) eine untere Schranke für $d(u, t)$
- wenn u von Rückwärtssuche abgearbeitet, ist $d(u, t)$ bekannt

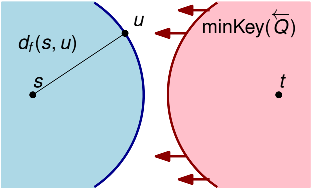


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bidir. Dist.-Bounding Reach-Dijkstra

Idee (für Vorwärtssuche):

- wenn u von Rückwärtssuche noch nicht erreicht, ist $\text{minKey}(\overleftarrow{Q})$ eine untere Schranke für $d(u,t)$
- wenn u von Rückwärtssuche abgearbeitet, ist $d(u,t)$ bekannt



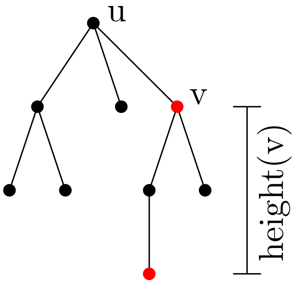
somit:

- ignoriere u, wenn $r(u) < d_f[u]$ und $r(u) < \min\{d_b[u], \text{minKey}(\overleftarrow{Q})\}$
- Stoppkriterium bleibt erhalten (also $\text{minKey}(\overrightarrow{Q}) + \text{minKey}(\overleftarrow{Q}) \geq \mu$)
- wenn als Alternierungsstrategie $\min\{\text{minKey}(\overrightarrow{Q}), \text{minKey}(\overleftarrow{Q})\}$ gewählt, gilt für Vorwärtssuche: $d_f[u] \leq \text{minKey}(\overleftarrow{Q})$

Vorberechnung: Erster Ansatz

Wie kann man Reach-Werte vorberechnen?

- initialisieren $r(u) = 0$ für alle Knoten
- für jeden Knoten u
  - konstruiere kürzeste Wege-Baum
  - höhe von Knoten v: Abstand von v zum am weitesten entfernten Nachfolger
  - für jeden Knoten v:
    $r(v) = \max\{r(v), \min\{d(u, v), \text{height}(v)\}\}$

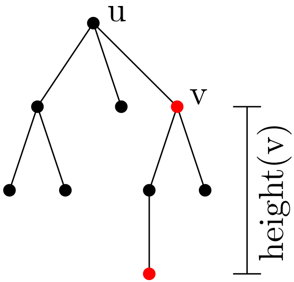


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Vorberechnung: Erster Ansatz

Wie kann man Reach-Werte vorberechnen?

- initialisieren $r(u) = 0$ für alle Knoten
- für jeden Knoten u
  - konstruiere kürzeste Wege-Baum
  - höhe von Knoten v: Abstand von v zum am weitesten entfernten Nachfolger
  - für jeden Knoten v: $r(v) = \max\{r(v), \min\{d(u, v), \text{height}(v)\}\}$



altes Problem:

- Vorberechnung basiert auf all-pair-shortest paths

Approximation

Beobachtung:

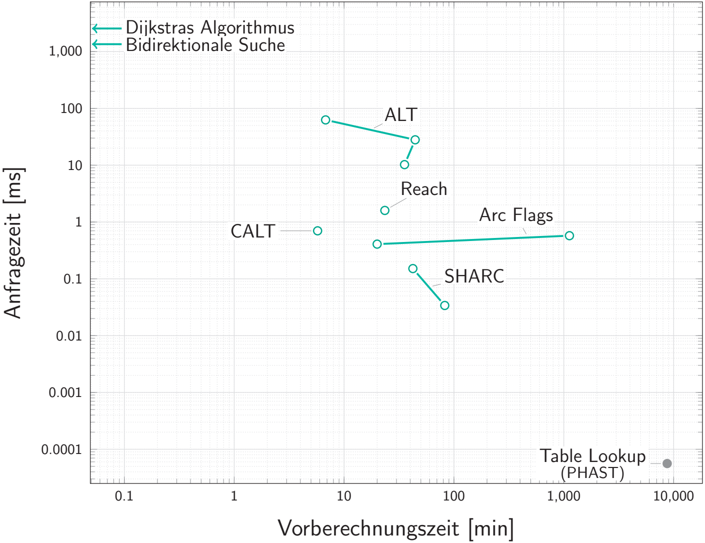
- es genügt, für jeden Knoten eine obere Schranke des Reach-Wertes zu haben

Problem:

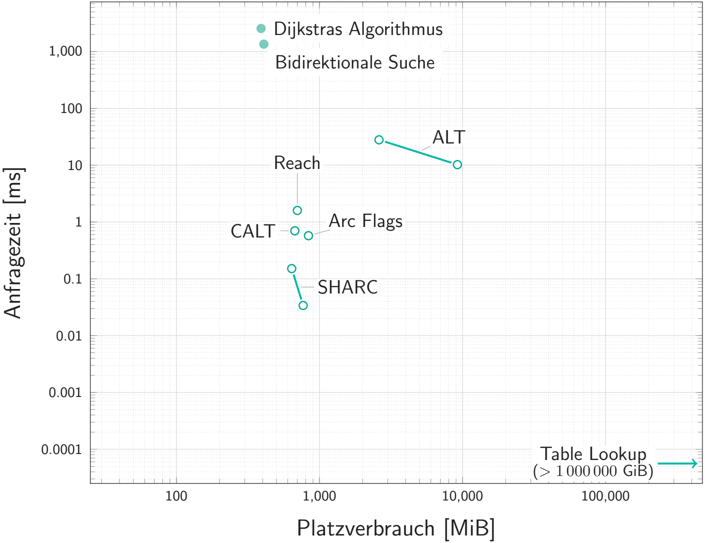
- untere Schranken einfach zu finden:
  - breche Konstruktion der Bäume einfach bei bestimmter Größe ab
- aber: untere Schranken sind unbrauchbar
- Berechnung von oberen Schranken deutlich schwieriger
- möglich, aber sehr aufwendig
- nicht (mehr) Bestandteil der Vorlesung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Übersicht bisherige Techniken



Übersicht bisherige Techniken



CRP

Hierarchische Techniken

Ideensammlung:

- Identifiziere wichtige Knoten mit Zentralitätsmaß
- Überspringe unwichtige Teile des Graphen

Hierarchische Techniken

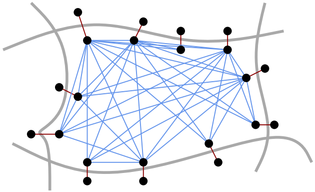
Ideensammlung:

- Identifiziere wichtige Knoten mit Zentralitätsmaß
- Überspringe unwichtige Teile des Graphen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

(Multi-Level) Overlays [SWZ02, HSW08]

Beobachtung: Viele (lange) Routen in Teilen identisch
Idee: Berechne Teillösungen vor

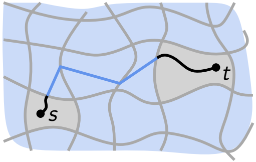


Overlay-Graph:

- Wähle wichtige Knoten (Separatoren, Pfadüberdeckung, heuristisch)
- Berechne Shortcut-Kanten:
  - Überspringen unwichtige Knoten
  - Erhalten Distanz

Anfragen:

- Multi-Level Dijkstra-Variante
- Ignoriert Kanten zu weniger wichtigen Knoten

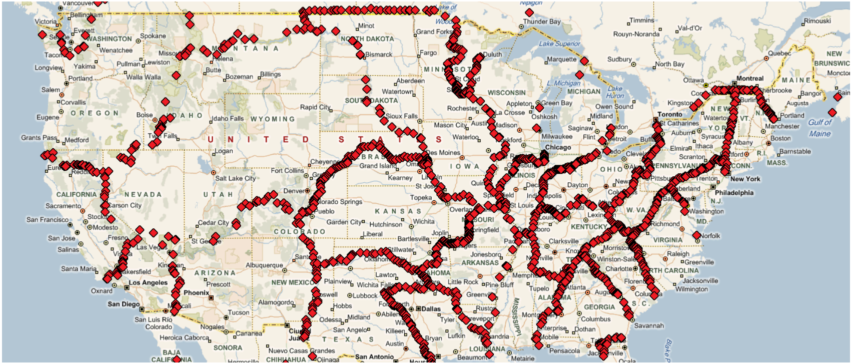


Analog: Hierarchie mehrerer Level immer wichtigerer Knoten

Partitionierung

Karlsruher Institut für Technologie

Beobachtung: Strassengraphen haben dünne, natürliche Schnitte



- Jeder Pfad durch eine Zelle betritt/verlässt die Zelle durch einen Randknoten

$\Rightarrow$ Minimiere # Schnittkanten mit Zellgrösse $\leq U$ (Eingabeparameter)

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

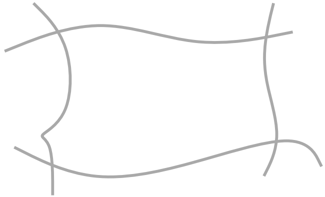
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

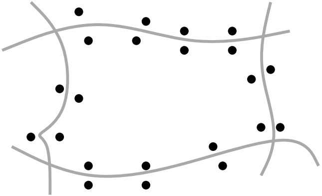
- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

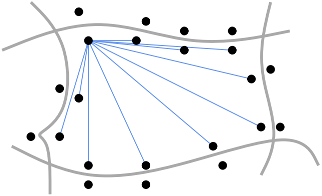
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



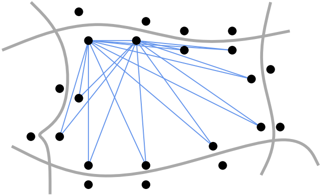
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

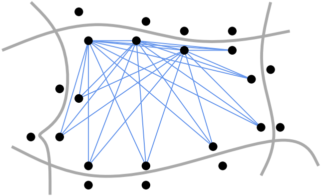
- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



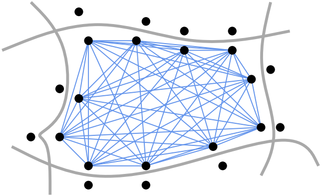
Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

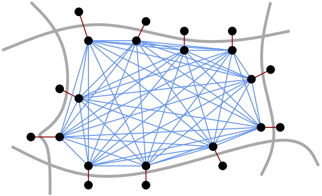
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

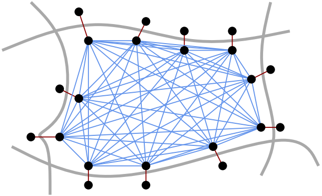
- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

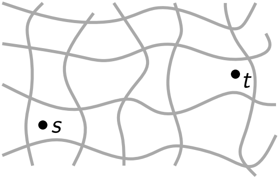
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Suchgraph:

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

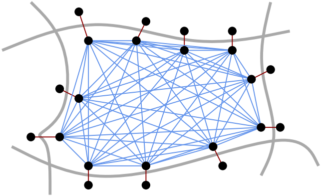
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

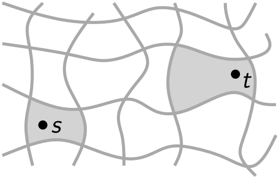
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

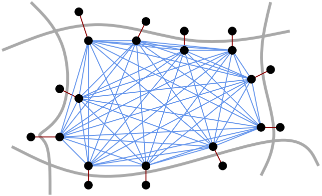
- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

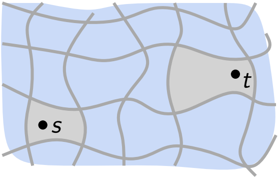
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

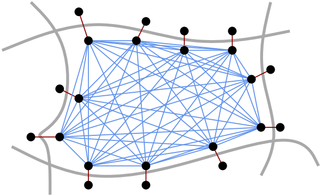
- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

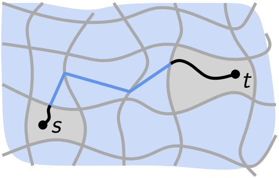
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

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

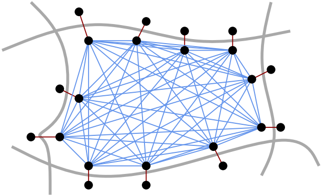
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

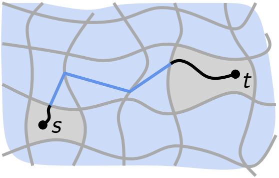
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





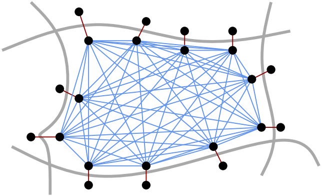
Suchgraph:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

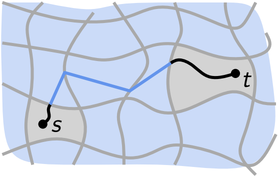
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Suchgraph:

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

Example

2^{15} Knoten pro Zelle, 626 Zellen $\Rightarrow$ 34 k Knoten im Overlaygraphen

Ergebnisse

Karlsruher Institut für Technologie

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ergebnisse

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Beispiel:

	Metric Customization		Queries	
metric	time [s]	space [MB]	scans	time [ms]
Travel time	20	10	45134	10
Distance	20	10	47127	11

(partition: $\leq 2^{14}$ nodes/cell)

West Europa (18 M nodes, 42 M edges)
Intel Core-i7 920 (four cores at 2.67 GHz)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ergebnisse

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Beispiel:

	Metric Customization		Queries	
metric	time [s]	space [MB]	scans	time [ms]
Travel time	20	10	45134	10
Distance	20	10	47127	11

(partition: $\leq 2^{14}$ nodes/cell)

West Europa (18 M nodes, 42 M edges)
Intel Core-i7 920 (four cores at 2.67 GHz)

Diskussion: Unterschied zu ArcFlags?

Ergebnisse

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Beispiel:

	Metric Customization		**Queries**	
metric	time [s]	space [MB]	scans	time [ms]
Travel time	20	10	45134	10
Distance	20	10	47127	11

(partition: $\leq 2^{14}$ nodes/cell)

West Europa (18 M nodes, 42 M edges)
Intel Core-i7 920 (four cores at 2.67 GHz)

Unterschied zu ArcFlags:
Vorberechnung außerhalb der Zelle vs. eingeschränkt auf Zelle

Diskussion

Verbesserungen?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Diskussion

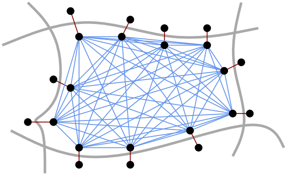
Verbesserungen?

- Ausdünnung der Graphen (weil Cliques unnötig viele Kanten haben?)
- Kombination mit zielgerichteter Suche
- Multi-Level Partitionierung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

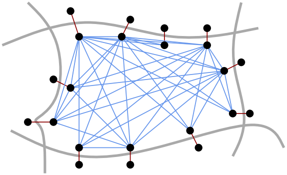
Ausdünnung des Overlaygraphen:



Naheliegendes

Ausdünnung des Overlaygraphen:

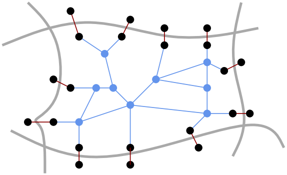
- entferne unnötige Kanten



Naheliegendes

Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)

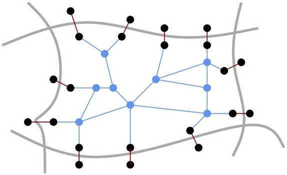


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)
- etwas schnellere Anfragen

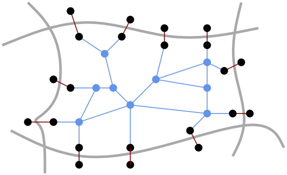


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)
- etwas schnellere Anfragen



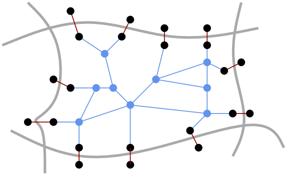
Kombination mit zielgerichteter Suche:

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

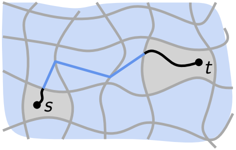
Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)
- etwas schnellere Anfragen



Kombination mit zielgerichteter Suche:

- nur auf (kleinem) Overlaygraphen
- ALT/Arc-Flags
- beschleunigt Anfragen, macht Vorberechnung und Queries komplizierter

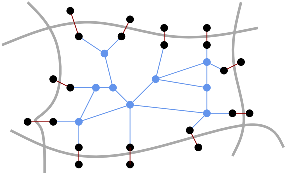


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

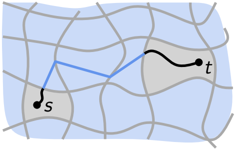
Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)
- etwas schnellere Anfragen



Kombination mit zielgerichteter Suche:

- nur auf (kleinem) Overlaygraphen
- ALT/Arc-Flags
- beschleunigt Anfragen, macht Vorberechnung und Queries komplizierter



Es geht besser (und einfacher!)

Multilevel-Overlay Graphs

Gegeben

- Eingabegraph $G = (V, E, \text{len})$
- Folge $V := V_0 \supseteq V_1 \supseteq \ldots \supseteq V_L$ von Teilmengen von V.

Berechne

- Folge $G_0 = (V_0, E_0, \text{len}_0), \ldots, G_L = (V_L, E_L, \text{len}_L)$ von Graphen, so dass Distanzen in G_i wie in G_0

Multi-Level Ansatz

Multi-Level Partition:

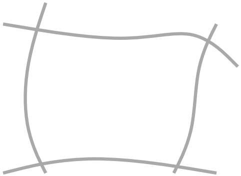
- benutze mehrere, geschachtelte Partitionen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

Multi-Level Partition:

- benutze mehrere, geschachtelte Partitionen



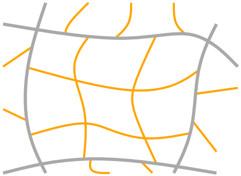
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

Karlsruher Institut für Technologie

Multi-Level Partition:

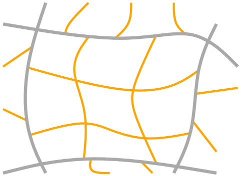
- benutze mehrere, geschachtelte Partitionen



Multi-Level Ansatz

Multi-Level Partition:

- benutze mehrere, geschachtelte Partitionen
- berechne Cliquen bottom-up
- benutze G_{i-1} um G_i zu bestimmen
- trade-off zwischen Platz und Suchzeiten

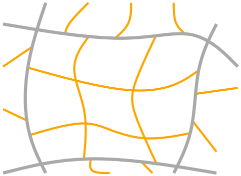


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

Multi-Level Partition:

- benutze mehrere, geschachtelte Partitionen
- berechne Cliquen bottom-up
- benutze G_{i-1} um G_i zu bestimmen
- trade-off zwischen Platz und Suchzeiten



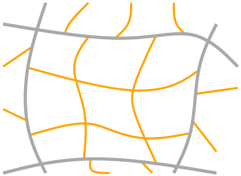
Suchgraph:

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

Multi-Level Partition:

- benutze mehrere, geschachtelte Partitionen
- berechne Cliquen bottom-up
- benutze G_{i-1} um G_i zu bestimmen
- trade-off zwischen Platz und Suchzeiten



Suchgraph:

- Overlay auf obersten Level (G_L) ...
- und alle Ziel- und Startzellen (auf jedem Level)
- (bidirektionaler) Dijkstra

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Phantom Level

Beobachtung: Viele Level $\Rightarrow$ schnellere Vorberechnung, mehr Platz

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Phantom Level

Beobachtung: Viele Level $\Rightarrow$ schnellere Vorberechnung, mehr Platz

- Benutze mehr Level für Vorberechnung (32 Knoten pro Zelle)
- Speichere die unteren Level aber nicht dauerhaft
- Schnellere Vorberechnung
- Werden aber nicht für Queries benutzt

Cell Size	Cust Time
$[2^{14}]$	20.0 s
$[2^{8} : 2^{14}]$	4.9 s

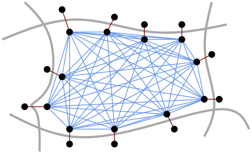
(metric: travel time)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Implementation

Repräsentation des Overlaygraphen:

- wirklich Graph-DS nötig?



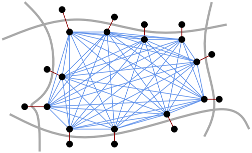
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Implementation

Repräsentation des Overlaygraphen:

- wirklich Graph-DS nötig?
- speicher Cliquen als Matrizen
- Beschleunigt Vorberechnung und Queries um einen Faktor 2

  Schneller als ausgedünnter Overlay!

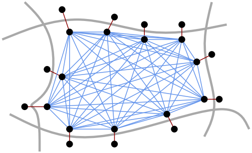


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Implementation

Repräsentation des Overlaygraphen:

- wirklich Graph-DS nötig?
- speicher Cliquen als Matrizen
- Beschleunigt Vorberechnung und Queries um einen Faktor 2
  Schneller als ausgedünnter Overlay!
- Entkoppelung von Metrik und Topology
- reduziert Platzverbrauch pro Metrik!



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

3-Phasen Algorithmus

1. Metrik-unabhängige Vorberechnung

- Partitionierung des Graphen
- Bauen der Topology des Overlay Graphen
- Linearer Platz für Partition, kann ein wenig dauern (nur einmal)
- PUNCH: 15-30 Minuten

3-Phasen Algorithmus

1. Metrik-unabhängige Vorberechnung

- Partitionierung des Graphen
- Bauen der Topology des Overlay Graphen
- Linearer Platz für Partition, kann ein wenig dauern (nur einmal)
- PUNCH: 15-30 Minuten

2. Metrik-abhängige Vorberechnung

- Berechnung der Gewicht der Matrix-Einträge
- Mit Hilfe von lokalen (hoch-parallelisierbaren) Dijkstra-Suchen
- Overhead pro Metrik klein und speicher-lokal (ein Array)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

3-Phasen Algorithmus

1. Metrik-unabhängige Vorberechnung

- Partitionierung des Graphen
- Bauen der Topology des Overlay Graphen
- Linearer Platz für Partition, kann ein wenig dauern (nur einmal)
- PUNCH: 15-30 Minuten

2. Metrik-abhängige Vorberechnung

- Berechnung der Gewicht der Matrix-Einträge
- Mit Hilfe von lokalen (hoch-parallelisierbaren) Dijkstra-Suchen
- Overhead pro Metrik klein und speicher-lokal (ein Array)

3. Queries

- Benutze Graph und beide Vorberechnungen
- (paralleler) bidirektionaler Dijkstra

Performance

	Algorithm	Customization time [s]	Customization space [MB]	Queries scans	Queries time [ms]
travel time	MLD-1 [2^{14}]	4.9	9.8	45134	5.67
	MLD-2 [$2^{12}:2^{18}$]	5.0	18.4	12722	1.79
	MLD-3 [$2^{10}:2^{15}:2^{20}$]	5.2	32.3	6074	0.91
	MLD-4 [$2^{8}:2^{12}:2^{16}:2^{20}$]	5.2	59.0	3897	0.71
distances	MLD-1 [2^{14}]	4.7	9.8	47127	6.19
	MLD-2 [$2^{12}:2^{18}$]	4.9	18.4	13114	1.85
	MLD-3 [$2^{10}:2^{15}:2^{20}$]	5.1	32.3	6315	1.01
	MLD-4 [$2^{8}:2^{12}:2^{16}:2^{20}$]	4.7	59.0	4102	0.77

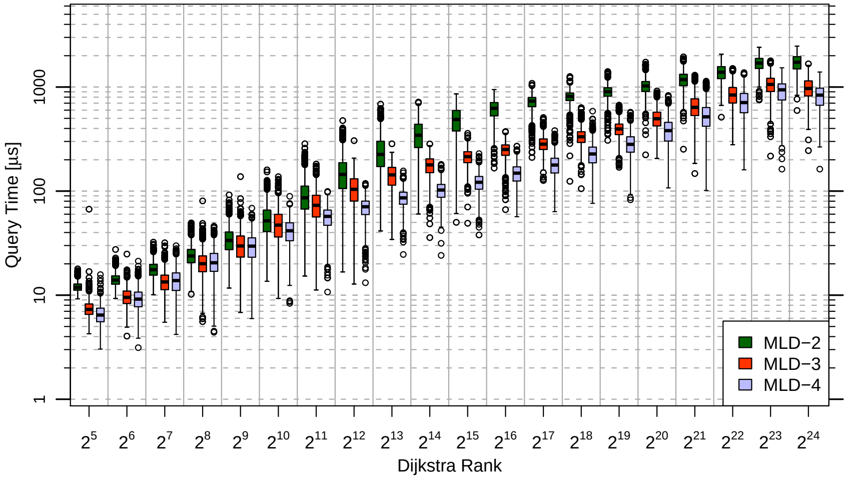
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Performance

	Algorithm	Customization time [s]	Customization space [MB]	Queries scans	Queries time [ms]
travel time	MLD-1 $[2^{14}]$	4.9	9.8	45134	5.67
	MLD-2 $[2^{12} : 2^{18}]$	5.0	18.4	12722	1.79
	MLD-3 $[2^{10} : 2^{15} : 2^{20}]$	5.2	32.3	6074	0.91
	MLD-4 $[2^{8} : 2^{12} : 2^{16} : 2^{20}]$	5.2	59.0	3897	0.71
distances	MLD-1 $[2^{14}]$	4.7	9.8	47127	6.19
	MLD-2 $[2^{12} : 2^{18}]$	4.9	18.4	13114	1.85
	MLD-3 $[2^{10} : 2^{15} : 2^{20}]$	5.1	32.3	6315	1.01
	MLD-4 $[2^{8} : 2^{12} : 2^{16} : 2^{20}]$	4.7	59.0	4102	0.77

MLD: fast customization / compact / real-time queries / robust

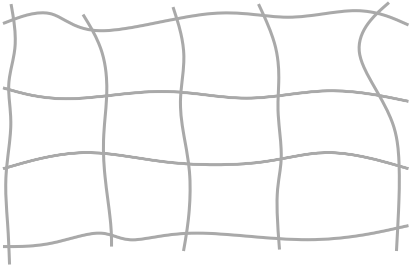
Local Queries



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entpacken der Shortcuts

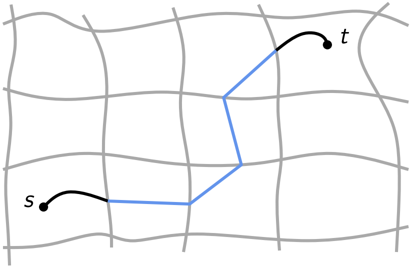
Von Shortcuts zu vollen Pfaden:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entpacken der Shortcuts

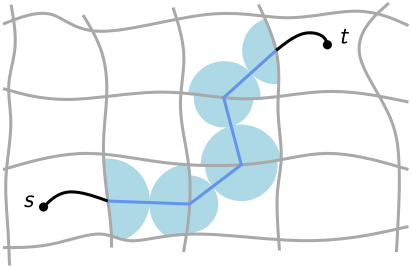
Von Shortcuts zu vollen Pfaden:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entpacken der Shortcuts

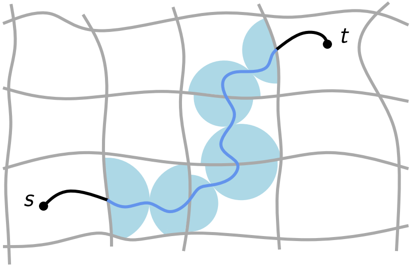
Von Shortcuts zu vollen Pfaden:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entpacken der Shortcuts

Von Shortcuts zu vollen Pfaden:

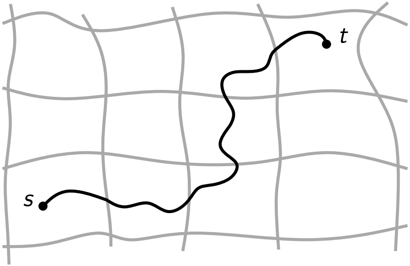


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entpacken der Shortcuts

Von Shortcuts zu vollen Pfaden:

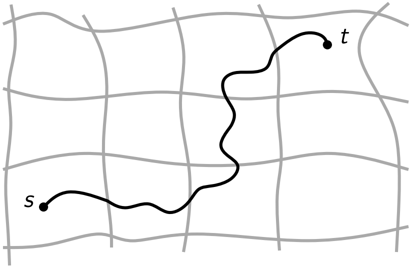
- bidirektionale Suche
- beschränkt auf die Zelle
- benutze untere Level rekursiv
- parallelisierbar
- benutze LRU-cache



Entpacken der Shortcuts

Von Shortcuts zu vollen Pfaden:

- bidirektionale Suche
- beschränkt auf die Zelle
- benutze untere Level rekursiv
- parallelisierbar
- benutze LRU-cache



Kaum zusätzlicher Speicher, 20-30% Zeit-Overhead.

Customizable Route Planning [DGPW11]

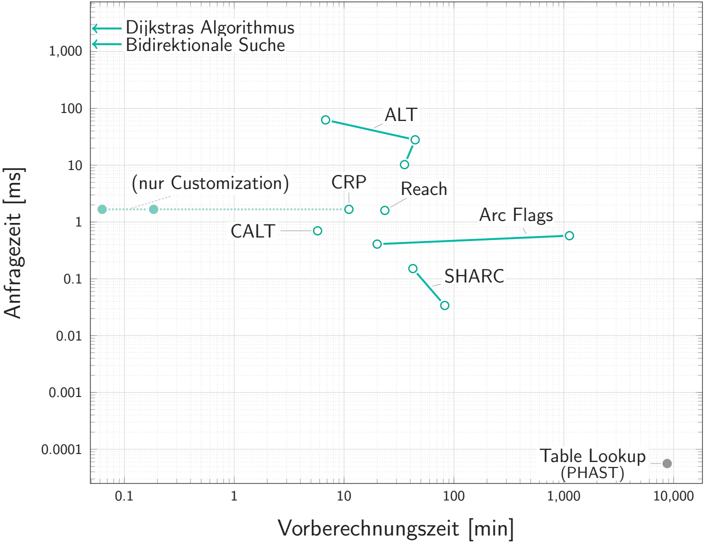
Eigenschaften:

- viele Metriken
- schnelle lokale and globale Updates
  Zelle in Millisekunden, gesamter Graph in Sekunden
- (hinreichend) schnelle Anfragen (1000$\times$ Dijkstra)
- Bing Maps Routing Engine, see `http://goo.gl/C7MsZf`

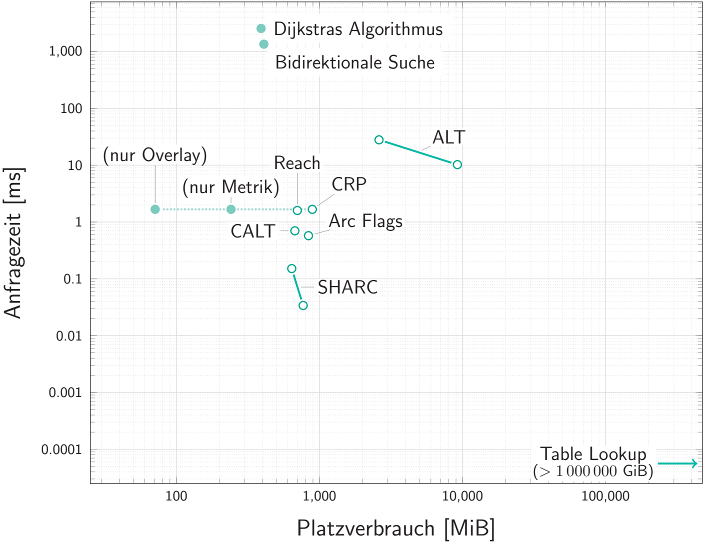
Gute Partition ist der Schlüssel!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Übersicht bisherige Techniken



Übersicht bisherige Techniken



Nächste Termine

Montag, 12.5.2014

Mittwoch, 14.5.2014

(Auf Deutsch)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Literatur I

Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck.
Customizable route planning.
In *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, volume 6630 of *Lecture Notes in Computer Science*, pages 376–387. Springer, 2011.

Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck.
Reach for A*: Shortest path algorithms with preprocessing.
In *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, volume 74 of *DIMACS Book*, pages 93–139. American Mathematical Society, 2009.

Ronald J. Gutman.
Reach-based routing: A new approach to shortest path algorithms optimized for road networks.
In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX'04)*, pages 100–111. SIAM, 2004.

Martin Holzer, Frank Schulz, and Dorothea Wagner.
Engineering multilevel overlay graphs for shortest-path queries.
ACM Journal of Experimental Algorithmics, 13(2.5):1–26, December 2008.

Literatur II

Frank Schulz, Dorothea Wagner, and Christos Zaroliagis.
Using multi-level graphs for timetable information in railway systems.
In *Proceedings of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02)*, volume 2409 of *Lecture Notes in Computer Science*, pages 43–59. Springer, 2002.