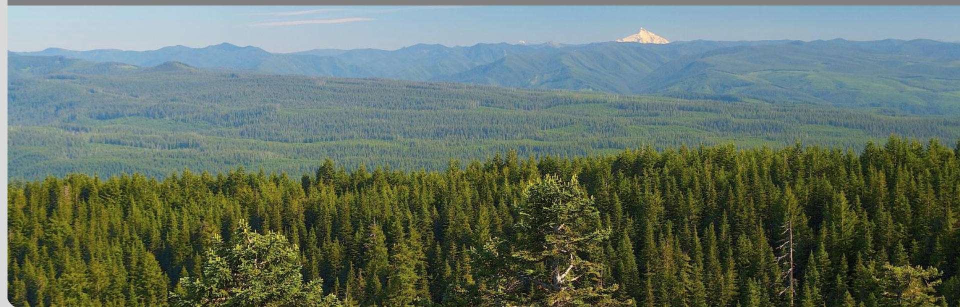
KIT
Karlsruher Institut für Technologie

Algorithmen für Routenplanung

11. Vorlesung, Sommersemester 2012

Daniel Delling | 6. Juni 2012

MICROSOFT RESEARCH SILICON VALLEY



Letztes Mal

Karlsruher Institut für Technologie

- Beschleunigung von one-to-all Anfragen
- PHAST anstatt Dijkstra
- Vorteil: Ausnutzung von moderner Hardware Architektur

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Anfrageszenarien

KIT
Karlsruher Institut für Technologie

Punkt-zu-Punkt

- zwei Punkte $\rightarrow$ kürzester Weg
- wird für Routenplanung benutzt
- Beschleunigungstechniken
- HubLabels 10Mx schneller

One-to-Many

- ein (variierender Knoten) und eine (feste) Menge $\rightarrow$ Distanz zu allen Knoten in der Menge
- wichtig für POI

One-to-All

- ein Knoten $\rightarrow$ Distanzen zu allen Knoten
- wird für Vorberechnung benutzt
- PHAST 500x schneller (auf GPU)
- nutzt Hardware aus

Many-to-Many

- zwei Mengen $\rightarrow$ Distanztabelle
- wichtig für Vehicle Routing

Das One-to-Many Problem

Karlsruher Institut für Technologie

Problem Definition:

- Eingabe: eine Knoten s und eine Menge T
- Ausgabe: Distanz von s zu allen $t \in T$
- Annahme: wir fixieren T und variieren s

offensichtliche Lösungen:

- Dijkstras Algorithmus (mit Stoppkriterium)
  - $\Rightarrow$ Performance stark abhängig von $|T|$ und Verteilung von T
- $|T|$ p2p Anfragen (mit HL)
  - $\Rightarrow$ Performance stark abhängig von $|T|$
- benutze PHAST (kein Stoppkriterium!)
  - $\Rightarrow$ Overkill (vor allem für kleine T)

Erste Ideen

Vorschläge?

Ungerichtete Bidir. Techniken

Definition:

- $\overrightarrow{\sigma}(s,t)$: Suchraum der Vorwärtssuche von s nach t
- $\overleftarrow{\sigma}(s,t)$ analog
- eine bidirektionale Suche ist Ziel-unabhängig, gdw.

$$\forall (s,t_1,t_2) \in V^3 : \overrightarrow{\sigma}(s,t_1) = \overrightarrow{\sigma}(s,t_2) \quad \text{und}$$
$$\forall (s_1,s_2,t) \in V^3 : \overleftarrow{\sigma}(s_1,t) = \overleftarrow{\sigma}(s_2,t)$$

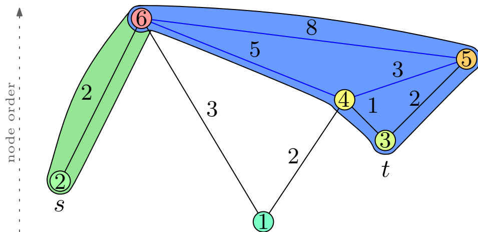
Beispiele:

- Bidirektionaler Dijkstra
- ohne Stoppkriterium, lass laufen bis Queues leer sind

Hierarchische Techniken

Beobachtung:

- suchen nur aufwärts
- sind nicht zielgerichtet



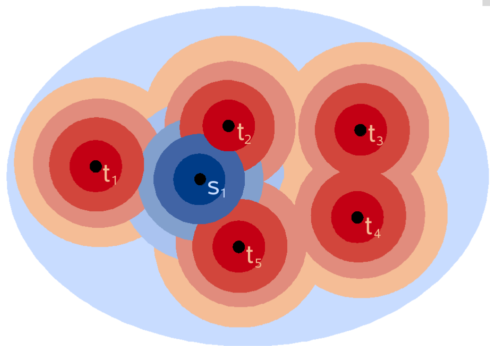
somit:

- Bidirektionaler Dijkstra
- Reach
- Contraction Hierarchies
- ohne Stoppkriterium, lass laufen bis Queues leer sind
- HL

Vorgehen

Idee:

- führe $|T|$ Rückwärtssuchen aus
- speicher für jedes besuchte u Abstände zu allen $t \in T$
- verwalte temporäres Distanzarray D_T
- führe Vorwärtssuche aus
- aktualisiere Einträge in D_T



Problem:

- verwalten der Suchräume?

Schneiden der Suchräume

während Rückwärtssuchen:

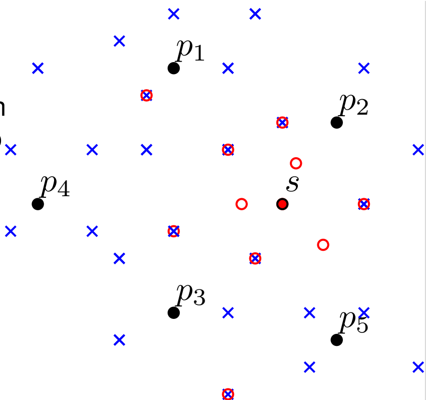
- breche nicht ab
- für jedes erreichte u:
  - füge Element $(u, t, d(u, t))$ in einen Vektor ein

Kompression:

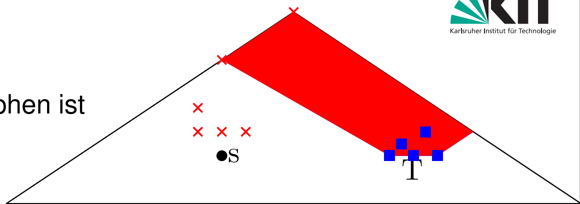
- sortiere Einträge nach u
- speichere für jeden Knoten u einen Bucket $\beta(u)$ mit allen $(t, d(u, t))$ ab
- mittels Adjazenzarray

während Vorwärtssuche:

- breche nicht ab
- für jedes erreichte u:
  - durchsuche Bucket $\beta(u)$
  - aktualisiere Distanzarray



RPHAST



Beobachtung:

- Sweep über den Graphen ist der Flaschenhals

Idee:

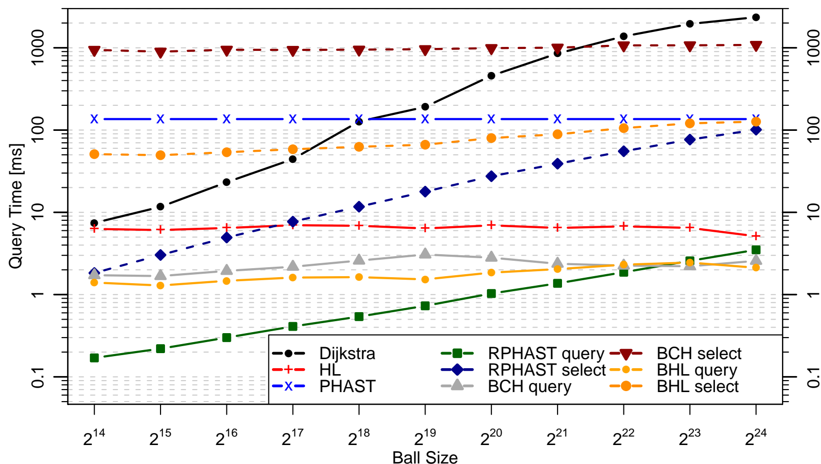
- extrahiere relevanten Teil des Graphen (Ziel Selektion)
- Aufwärtssuche im vollen Graphen
- Sweep auf extrahiertem Graphen

$\Rightarrow$

- Startknoten kann im ganzem Graphen liegen
- Grösse des extrahierten Graphen hängt von Verteilung und Anzahl T ab
- kann wie PHAST parallelisiert werden
- GPU implementation möglich

Experimente I

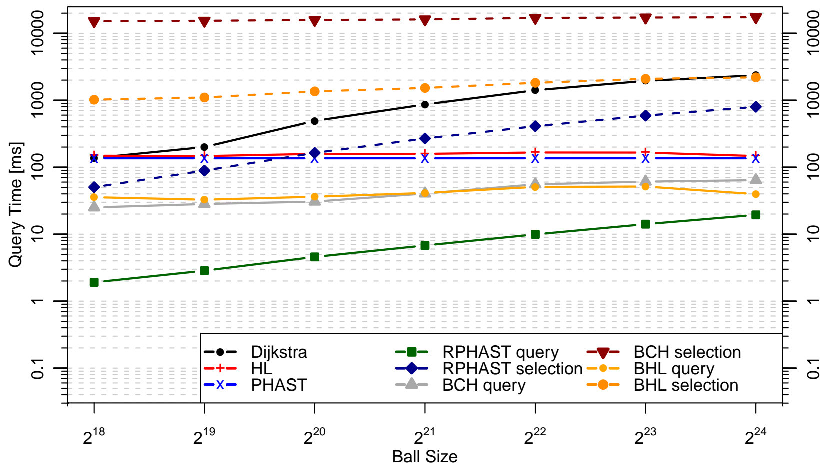
input: Westeuropa (18M Knoten), $|T| = 2^{14}$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente II

input: Westeuropa (18M Knoten), $|T| = 2^{18}$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

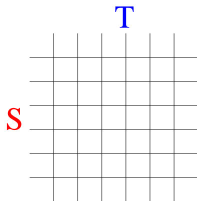
Many-to-Many Kürzeste Wege

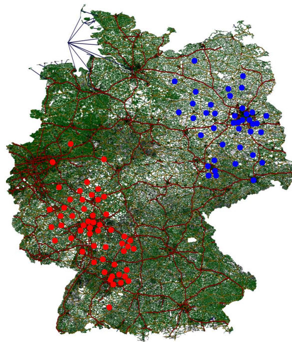
Gegeben:

- Graph
- Knotenmengen $S, T \in V$

Gesucht:

- Distanzmatrix D





Anwendungen:

- vehicle routing
- traveling salesman

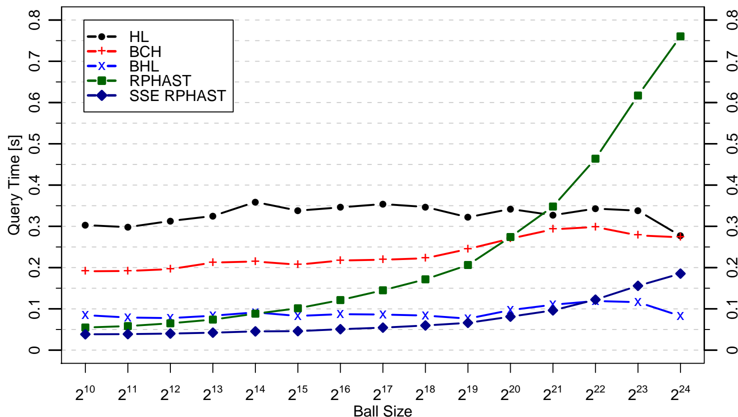
Lösung:

- $|S|$ one-to-many Anfragen
- speicher Distanzen in der Tabelle
- RPHAST kann multiples Setup (SSE) nutzen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente I

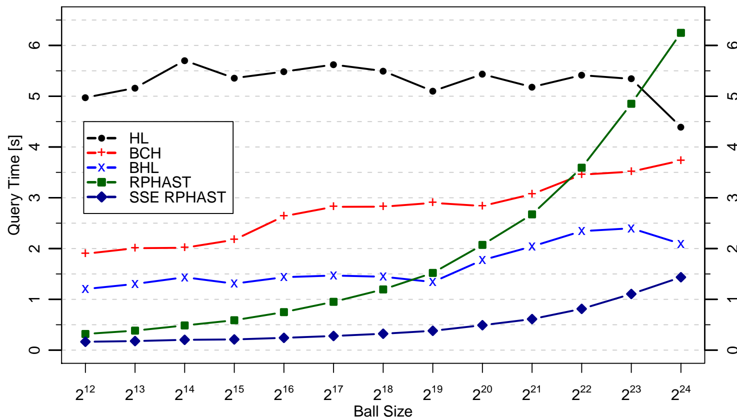
input: Westeuropa (18M Knoten), $|S| = |T| = 2^{10}$



Beobachtung: alle Techniken unter einer Sekunde

Experimente II

input: Westeuropa (18M Knoten), $|S| = |T| = 2^{12}$

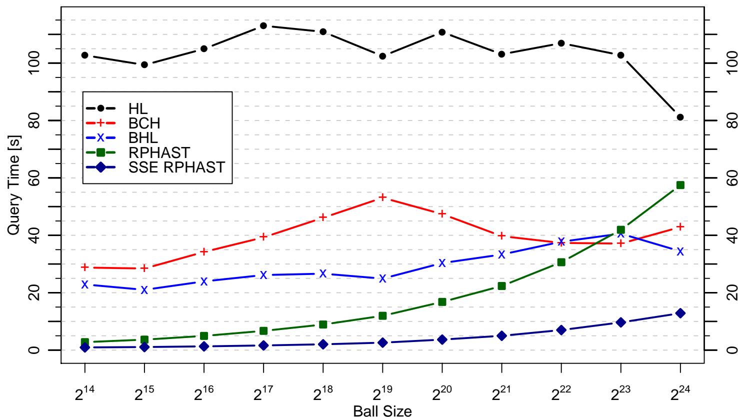


Beobachtung: SSE PHAST am schnellsten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente III

input: Westeuropa (18M Knoten), $|S| = |T| = 2^{14}$



Beobachtung: SSE PHAST am schnellsten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

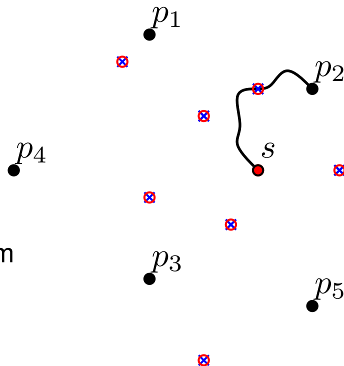
Nächster POI

Szenario:

- Zielknoten sind POIs
- finde *k* nächste POIs von einem Startknoten *s*

Lösung:

- wie one-to-many
- ordne die buckets pro Knoten auch nach aufsteigender Distanz
- in jedem Bucket müssen nur die *k* nächsten POIs durchsucht werden
- Laufzeit für POI Query nicht abhängig von Anzahl POIs im System
- Laufzeit: Suchraum · *k*



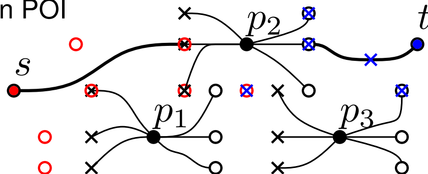
Bester Via Knoten

Szenario:

- Zielknoten sind POIs
- finde k best Via Knoten POIs von einem Startknoten s zu einem Startknoten t
- minimiere $\text{dist}(s, p) + \text{dist}(p, t)$ über alle POIs p

Lösung:

- Vorwärts- und Rückwartssuche von jedem POI
- speicher Kreuzprodukt der beiden Suchräume mit Distanz durch den POI
- Such von s und t: evaluiere jedes Paar
- Laufzeit: Suchraum$^2 \cdot k$



Aufgabenstellung

Karlsruher Institut für Technologie

Beobachtung:

- Queries sind schnell genug
- Visualisierung und Netzwerklatenz der Flaschenhals
- schwieriger und hoch optimierter Code

Können wir Geschwindigkeit gegen einfachere Bedienbarkeit eintauschen?

Idee:

- Implementier Routenplanung direkt in SQL
- auch die Erweiterungen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Location Services in SQL

Karlsruher Institut für Technologie

Vorteile:

- einfach zu nutzen
- Daten meist eh schon in SQL
- skalieren einfach (bestehende Datenbanksysteme, Cloud SQL)
- auch für Nicht-Routing-Experten zu nutzen
- External Memory Implementation “frei”

Nachteile:

- SQL viel langsamer als optimierter C++ Code
- keine aufwändigen Datenstrukturen (Graph, Priority Queue)
- Dijkstra-basierte Techniken sind keine Option

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ideensammlung

Karlsruher Institut für Technologie

Welcher Ansatz?

- keine Priority Queue?
- keine Graphdatenstruktur?

Idee: Hub Labeling

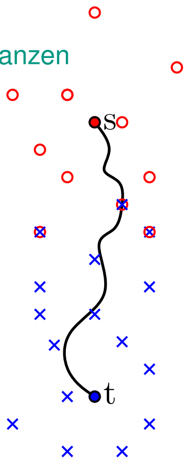
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Hub Labeling

KIT
Karlsruher Institut für Technologie

Vorberechung:

- für jeden Knoten u, berechne zwei Label $L_f(u), L_b(u)$
- ein Label ist eine Menge von Knoten (Hubs) und Distanzen
  - $\text{dist}(u, v)$ für jeden Hub $v \in L_f(u)$
  - $\text{dist}(v, u)$ für jeden Hub $v \in L_b(u)$
- die Label müssen die cover property einhalten:
  $\forall s, t$, $L_f(s) \cap L_b(t)$ überdeckt den kürzesten s–t Pfad



s–t Anfrage:

- finde Knoten $v \in L_f(s) \cap L_b(t) \ldots$
- $\ldots$ der $\text{dist}(s, v) + \text{dist}(v, t)$ minimiert

Beobachtungen:

- Label der Grösse ≈ 70 für Europe
- sehr schnelle Queryzeiten
- Set operationen

Speichern der Label

Idee:

- berechne Label mit C++ (wie bei HubLabels)
- aber speicher die Label direkt in der Datenbank
- ein Vorwärtslabel von Knoten v mit k Hubs:
  - erzeugt k Triples $(v, u, d(v, u))$ in Tabelle `forward`
- Rückwärtslabel genauso in `backward`
- ungefähr 1.35 Milliarden Zeilen pro Tabelle (ca. 19 GB pro Richtung)
- indiziere nach `node` (primary) und `hub` (secondary)

$L_f(1)$: | 1,0 | 4,1 | 5,2 | 7,3 |

$L_b(2)$: | 2,0 | 6,1 | 7,4 |

`forward`

node	hub	dist
1	1	0
1	4	1
1	5	2
1	7	3
2	14	2
⋮	⋮	⋮

`backward`

node	hub	dist
1	1	0
1	4	4
2	2	0
2	6	1
2	7	4
⋮	⋮	⋮

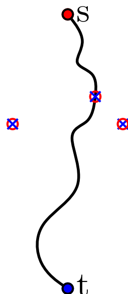
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

SQL Query

Algorithm 1: SQL_DIST

Input: source $s \in V$, target $t \in V$

```
1 SELECT
2       MIN(forward.dist+backward.dist)
3 FROM forward,backward
4 WHERE
5       forward.node = s AND
6       backward.node = t AND
7       forward.hub = backward.hub
```



Bemerkung:

- berechnet nur die Distanz

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pfadentpackung

Karlsruher Institut für Technologie

Idee:

- 2 Phasen
- speicher jeden Shortcut aus G^+ explizit (als Sequenz von Kanten IDs) in Tabelle `shortcuts`
- ca. 5 GB in Tabelle

Phase 1:

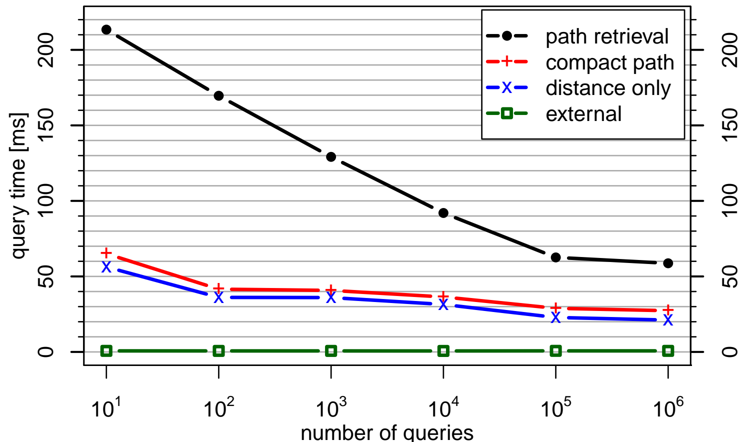
- erzeuge Pfad in G^+ durch Hubs auf dem Pfad
- erweitere Tabellen `forward` und `backward` um 2 Spalten: Parent und Shortcut
- erhöht Speicherverbrauch der Tabelle von 19 auf 32 GB

Phase 2:

- erzeuge Pfad in G durch matchen von G^+ mit `shortcuts`

Ergebnisse

Setup: MS SQL Server 2008 R2 mit Daten auf HDD, kalter Cache

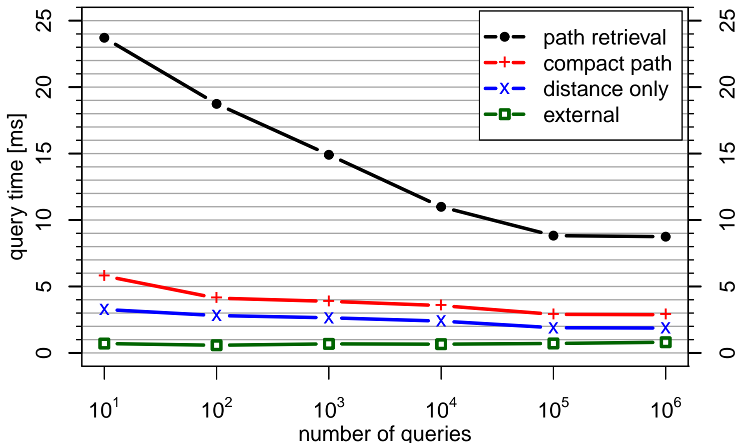


Beobachtung: Nicht schnell genug

Ergebnisse (SSD)

Karlsruher Institut für Technologie

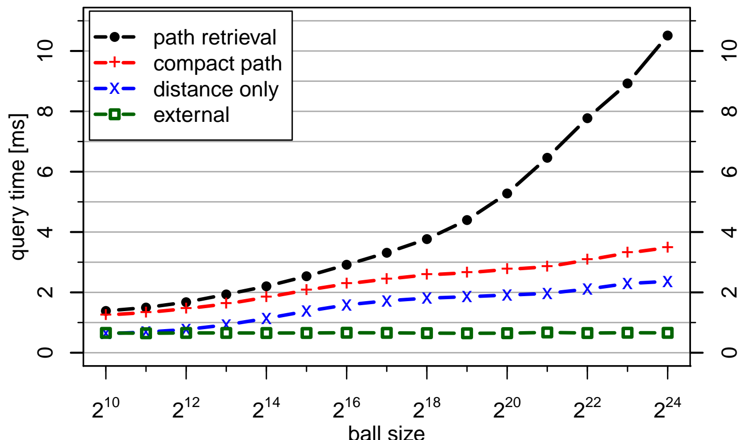
Setup: MS SQL Server 2008 R2 mit Daten auf SSD, kalter Cache



Beobachtung: SSD macht Queries schnell genug

Lokale Anfrage

Setup: Anfragen mit verschiedenem Rank, 10000 Anfragen, kalter Cache



Beobachtung: praxisrelevante Anfragen sehr schnell

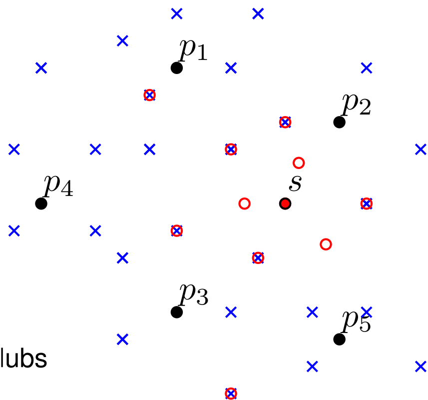
POI Anfragen in SQL

Idee:

- extrahiere die Rückwartslabel aus `backward`
- speicher sie in neuer Tabelle `poilab`
- indiziere nach `hub` und `dist`

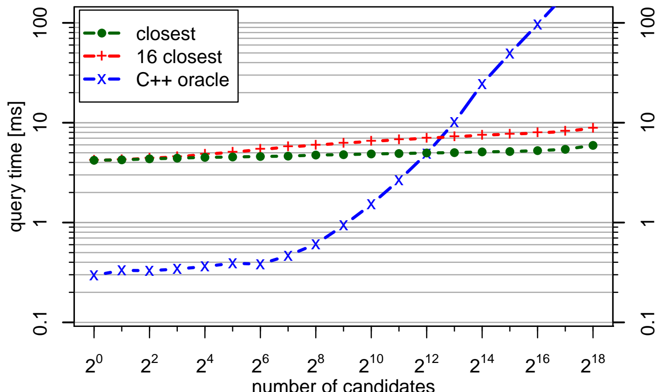
Query:

- iteriert über alle ausgehenden Hubs
- für jeden Hub werden nur die (k) nächsten POIs betrachtet
- Antwort: die k insgesamt nächsten POIs



Ergebnisse POI

Setup: verschiedene Anzahl POIs, zufällig gewählt



- externes Punkt-zu-Punkt Orakel: skaliert schlecht
- SQL Anfragen unabhängig von Anzahl POIs
- weitere Constraints einfach ("jetzt geöffnet")

Demo

Zusammenfassung

Karlsruher Institut für Technologie

- one-to-many shortest paths
- many-to-many
- POI Anfragen
- bester Via Knoten Anfragen
- Location Services in SQL

Ende

Karlsruher Institut für Technologie

Literatur:

- Sebastian Knopp, Peter Sanders, Dominik Schultes, Frank Schulz, and Dorothea Wagner:
  Computing Many-to-Many Shortest Paths Using Highway Hierarchies
  In: *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX'07)*, pages 36-45, 2007.
- Daniel Delling, Andrew V. Goldberg, Renato F. Werneck
  Faster Batched Shortest Paths in Road Networks
  In: *Proceedings of the 11th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'11)*, pages 52-63, 2011
- Ittai Abraham, Daniel Delling, Andrew V. Goldberg, Renato F. Werneck
  HLDB: Location-Based Services in Databases
  bald verfügbar

Nächste Termine

Karlsruher Institut für Technologie

Montag, 11.6.2012

Mittwoch, 13.6.2012
Montag, 18.6.2012
Mittwoch, 20.6.2012