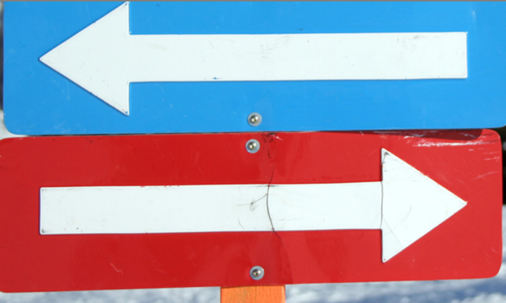
Karlsruher Institut für Technologie

Algorithmen für Routenplanung

8. Sitzung, Sommersemester 2013

Julian Dibbelt | 22. Mai 2013

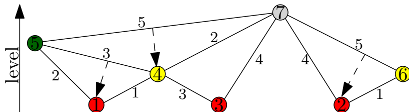
INSTITUT FÜR THEORETISCHE INFORMATIK · ALGORITHMIK · PROF. DR. DOROTHEA WAGNER



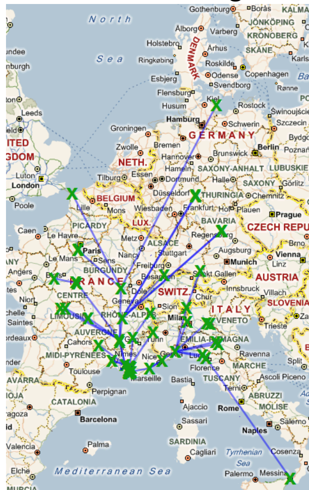
Letztes Mal

Karlsruher Institut für Technologie

Contraction Hierarchies

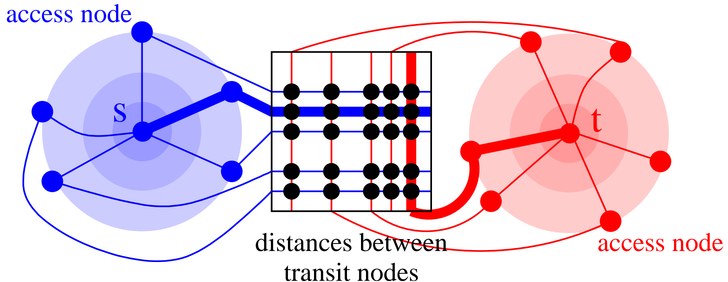


Hub-Labeling



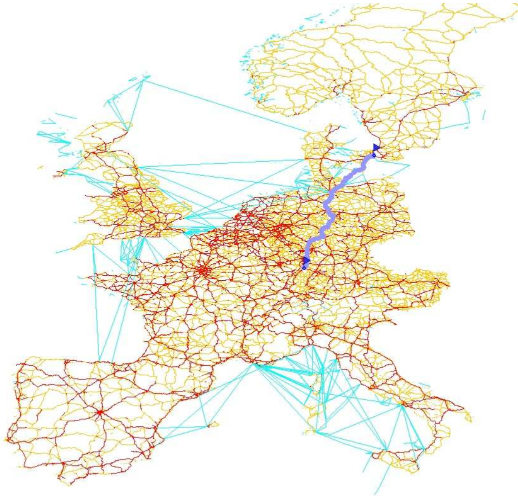
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing

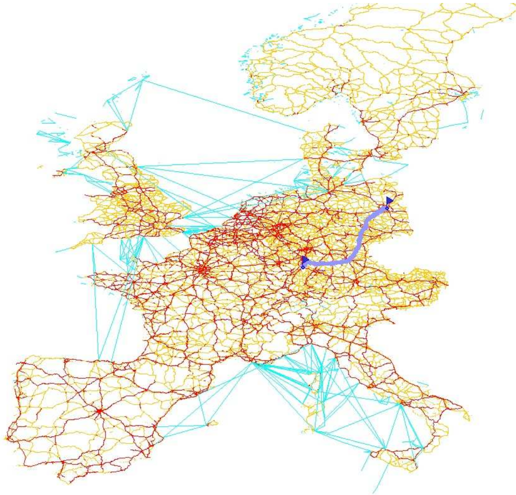


Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach…
Kopenhagen

Transit-Node Routing



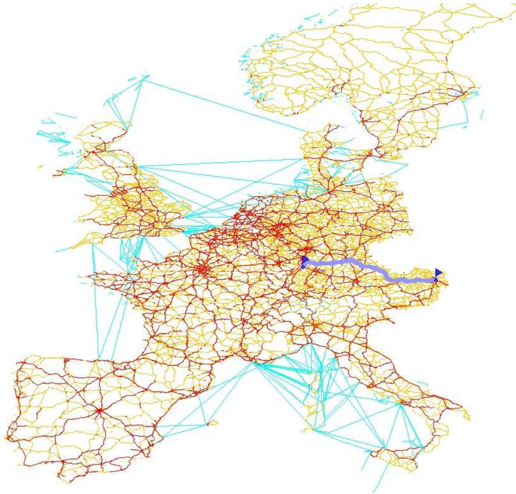
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach...
Berlin

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing



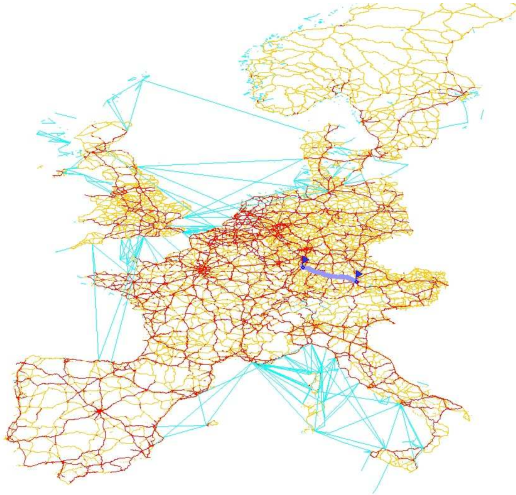
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach. . .
Wien

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing

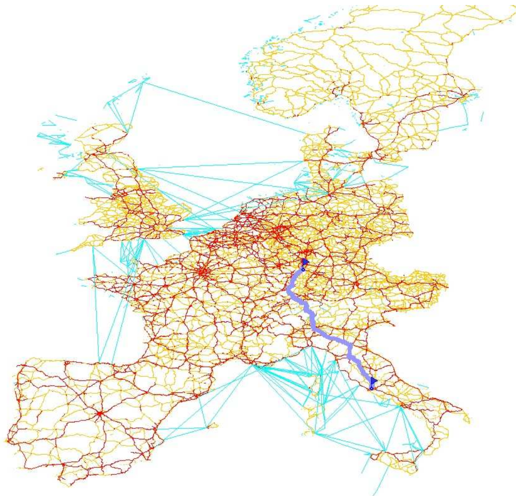


Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach. . .
München

Transit-Node Routing



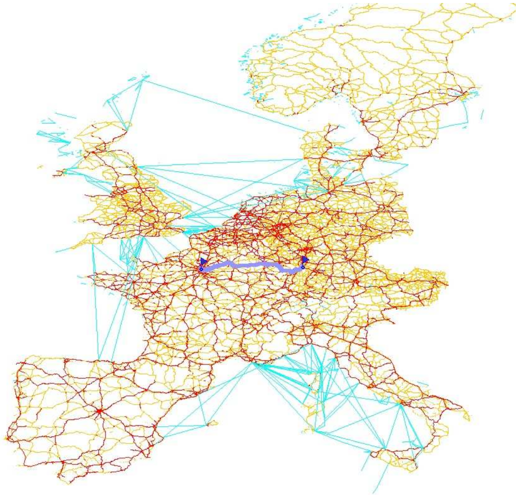
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach. . .
Rom

Transit-Node Routing

KIT
Karlsruher Institut für Technologie



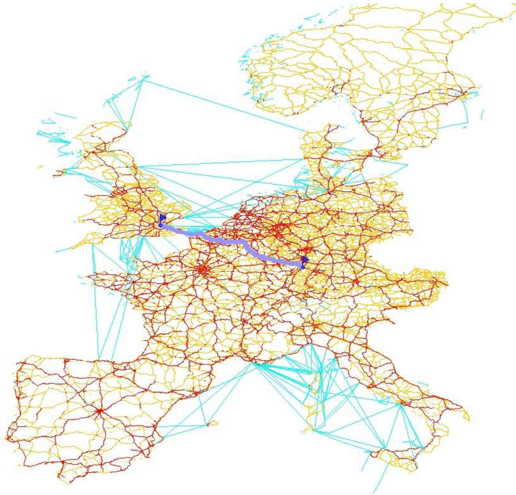
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach…
Paris

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing

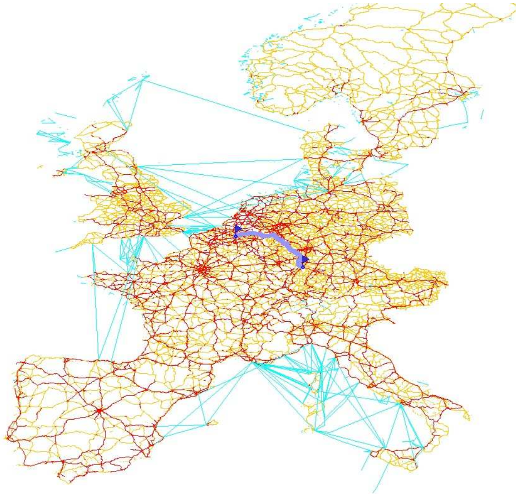


Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach...
London

Transit-Node Routing

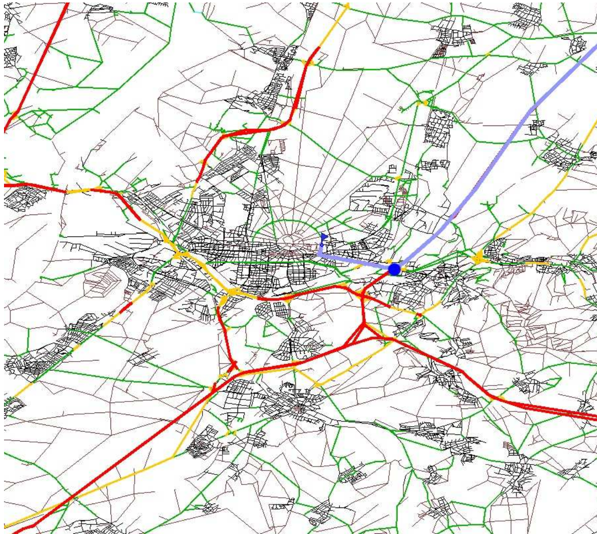


Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach...
Brüssel

Transit-Node Routing



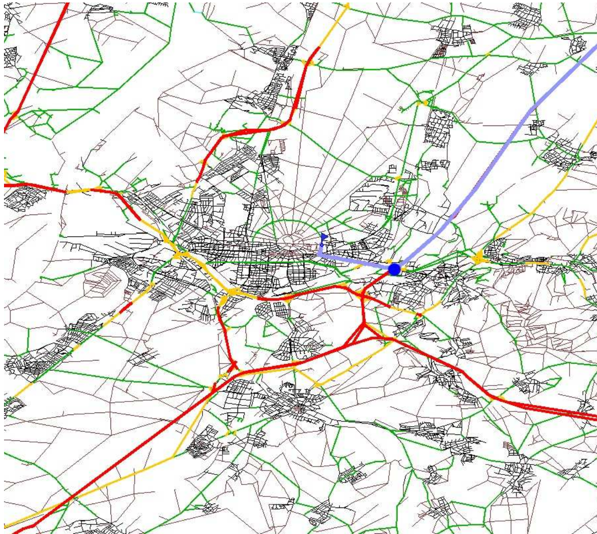
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach. . .
Kopenhagen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing



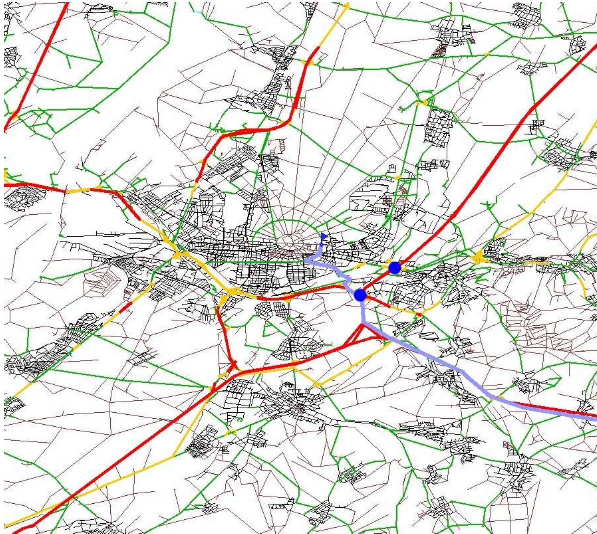
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach. . .
Berlin

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing



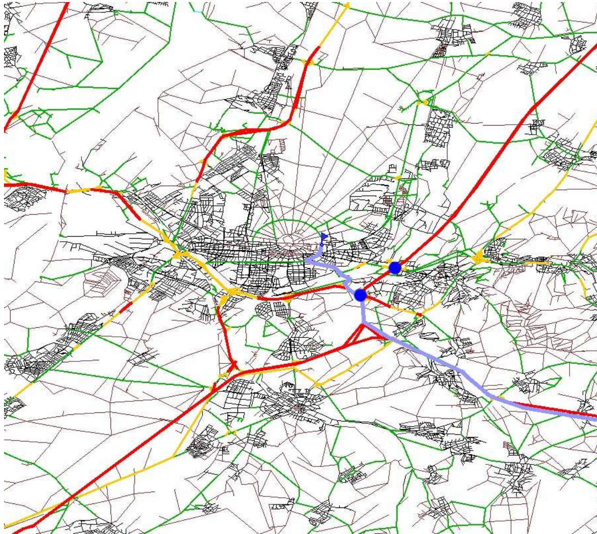
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach…
Wien

Transit-Node Routing

KIT
Karlsruher Institut für Technologie

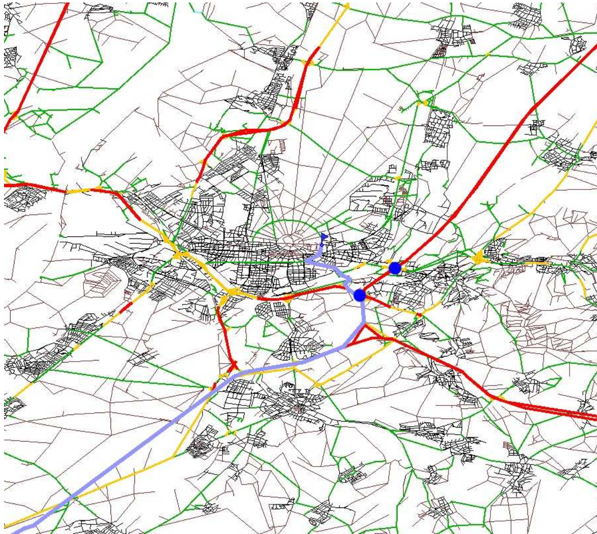


Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach. . .
München

Transit-Node Routing

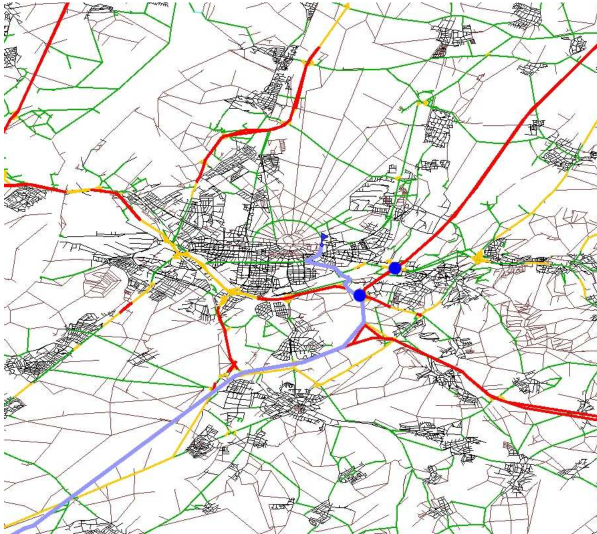


Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach. . .
Rom

Transit-Node Routing



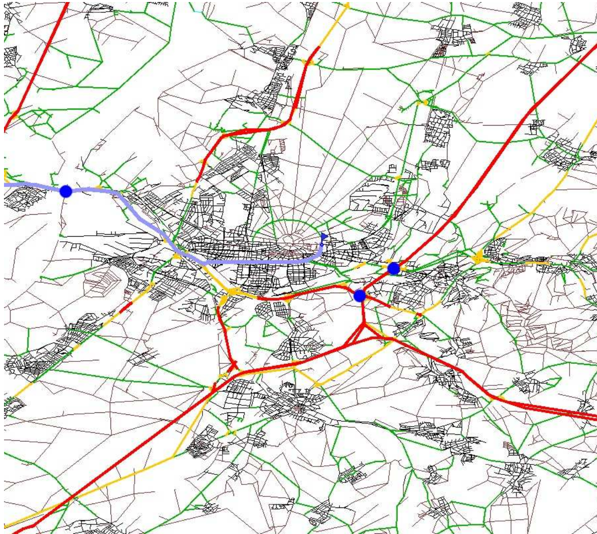
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach…
Paris

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing



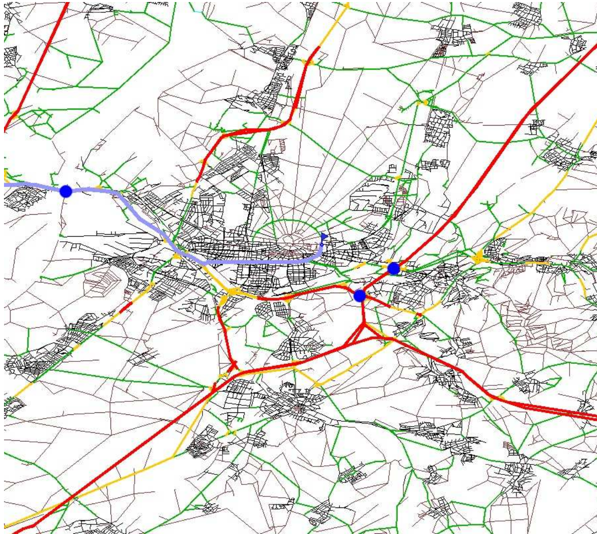
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach...
London

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Transit-Node Routing



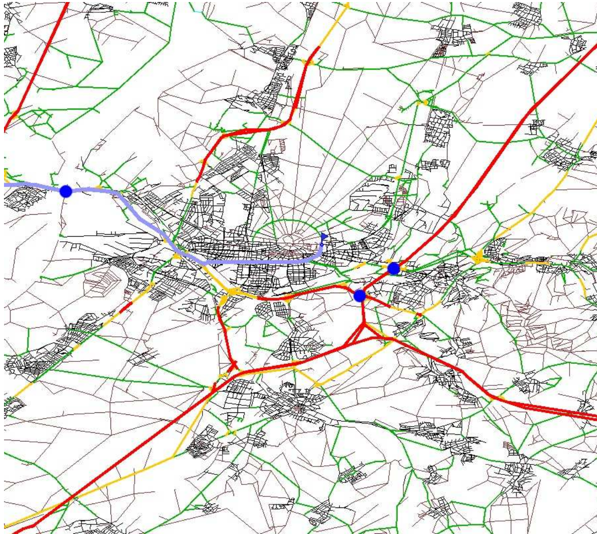
Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach...
Brüssel

Transit-Node Routing

Karlsruher Institut für Technologie



Beobachtung:

- wenn man weit weg fährt, fährt man immer an bestimmten Punkten vorbei
- hier: von Karlsruhe aus, an drei relevanten Stellen

Karlsruhe nach. . .

Abgrenzung Partition / Natural Cuts?
Wie ausnutzen?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

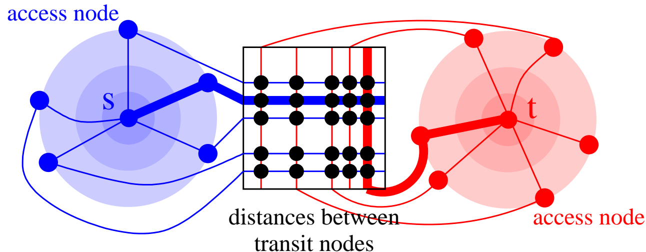
Transit-Node Routing

Idee:

- reduziere Anfragen auf Table-Lookups
- identifiziere “wichtige” Knoten
- vollständige Distanztabellen zwischen diesen Knoten

Probleme:

- Speicherverbrauch
- nahe Anfragen



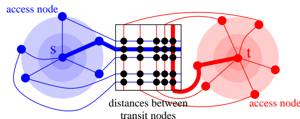
Generelles TNR-Framework

Karlsruher Institut für Technologie

- Transit nodes: $T \subseteq V$
- Forward/backward access mapping: $\overrightarrow{A} : V \to 2^T$, $\overleftarrow{A} : V \to 2^T$
  für jeden Knoten die für ihn wichtigen transit nodes, seine access nodes
- Vorberechnete Distanzen: D_T und d_A

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

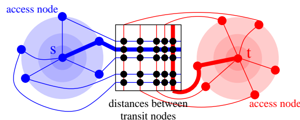
Generelles TNR-Framework



- Transit nodes: $T \subseteq V$
- Forward/backward access mapping: $\overrightarrow{A} : V \to 2^T$, $\overleftarrow{A} : V \to 2^T$
  für jeden Knoten die für ihn wichtigen transit nodes, seine access nodes
- Vorberechnete Distanzen: D_T und d_A

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

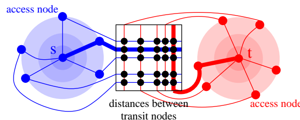
Generelles TNR-Framework



- Transit nodes: $T \subseteq V$
- Forward/backward access mapping: $\overrightarrow{A} : V \rightarrow 2^T$, $\overleftarrow{A} : V \rightarrow 2^T$
  für jeden Knoten die für ihn wichtigen transit nodes, seine access nodes
- Vorberechnete Distanzen: D_T und d_A
- $\text{dist}(s,t) \stackrel{?}{=} \min_{u \in \overrightarrow{A}(s), v \in \overleftarrow{A}(t)} \{d_A(s,u) + D_T(u,v) + d_A(v,t)\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Generelles TNR-Framework



- Transit nodes: $T \subseteq V$
- Forward/backward access mapping: $\overrightarrow{A} : V \to 2^T$, $\overleftarrow{A} : V \to 2^T$
  für jeden Knoten die für ihn wichtigen transit nodes, seine access nodes
- Vorberechnete Distanzen: D_T und d_A
- $\text{dist}(s,t) \overset{?}{=} \min_{u \in \overrightarrow{A}(s), v \in \overleftarrow{A}(t)} \{d_A(s,u) + D_T(u,v) + d_A(v,t)\}$

Berechnete Distanz nur für hinreichend weite Anfragen korrekt

- Locality filter: $L : V \times V \to \{\text{true}, \text{false}\}$
- true $\longrightarrow$ Fallback-Routine für lokale Anfragen
- Einseitige Fehler erlaubt (falsch positiv)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zutaten TNR-Verfahren

Also:

- Wie Transit-Nodes bestimmen?
- Wie Access-Nodes und deren Distanz bestimmen?
- Wie Distanztabelle zwischen Transit-Nodes berechnen?
- Welcher Locality-Filter?
- Wie lokale Anfragen berechnen?

Zutaten TNR-Verfahren

Karlsruher Institut für Technologie

Also:

- Wie Transit-Nodes bestimmen?
- Wie Access-Nodes und deren Distanz bestimmen?
- Wie Distanztabelle zwischen Transit-Nodes berechnen?
- Welcher Locality-Filter?
- Wie lokale Anfragen berechnen?

Ideen?

Grid-TNR [BMF06]

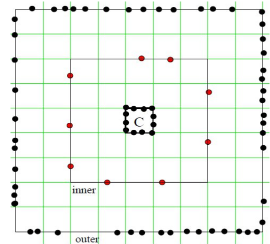
Karlsruher Institut für Technologie

Idee: Lege Locality-Filter fest, Rest folgt.

- Gleichmäßiges Gitter. LF: Abstand $\leq$ 4 Zellen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Grid-TNR [BMF06]

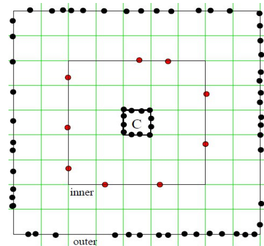


Idee: Lege Locality-Filter fest, Rest folgt.

- Gleichmäßiges Gitter. LF: Abstand $\leq$ 4 Zellen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

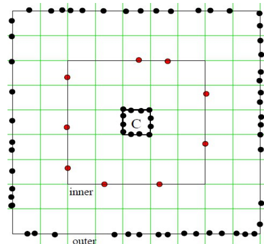
Grid-TNR [BMF06]



Idee: Lege Locality-Filter fest, Rest folgt.

- Gleichmäßiges Gitter. LF: Abstand $\leq$ 4 Zellen
- Zelle C, Inner: 5×5 Quadrat,
  Outer: 9×9 Quadrat

Grid-TNR [BMF06]



Idee: Lege Locality-Filter fest, Rest folgt.

- Gleichmäßiges Gitter. LF: Abstand $\leq$ 4 Zellen
- Zelle C, Inner: 5×5 Quadrat,
  Outer: 9×9 Quadrat
- Access-Nodes von C: Randknoten von Inner,
  die auf KW von C zu Outer-Randknoten liegen
- Transit-Nodes: Vereinigung aller Access-Nodes
- Distanzberechnung und lokale Anfragen: Dijkstra

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Highway-Hierarchies-TNR [SS08]

Karlsruher Institut für Technologie

Idee: Lege Transit-Nodes fest, alles bis auf Locality-Filter folgt.

- HH ist ein Vorgänger von CH (nicht Teil der VL)
- Transit-Nodes: Top-k Knoten der Hierarchie

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Highway-Hierarchies-TNR [SS08]

KIT
Karlsruher Institut für Technologie

Idee: Lege Transit-Nodes fest, alles bis auf Locality-Filter folgt.

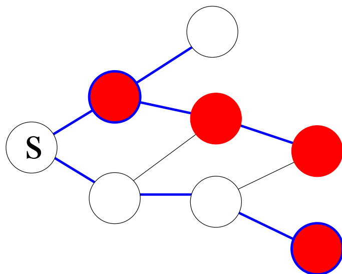
- HH ist ein Vorgänger von CH (nicht Teil der VL)
- Transit-Nodes: Top-k Knoten der Hierarchie
- Distanztabelle: Many-to-many [KSSSW07] (mehr in späterer VL)
- Access-Nodes von v: „erste“ Transit-Nodes auf KW-Baum von v
- Locality-Filter: geometrisch
- Lokale Anfragen: HH query
- Multi-Level
- Anmerkung: geht natürlich auch mit CH statt HH
  (hat keinen offiziellen Namen, hier „CH-TNR-geo“)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

HH-TNR: Access-Nodes berechnen

Vorgehen:

- lokale Suche, bis Transit-Knoten alle Zweige abdecken



HH-TNR: Access-Nodes berechnen

KIT
Karlsruher Institut für Technologie

Vorgehen:

- lokale Suche, bis Transit-Knoten alle Zweige abdecken

Beschleunigung:

- *prune* an Transit-Knoten (breche Zweig bei Transit-Knoten ab)
- dünne dann später mit Hilfe der Distanz-Tabelle wieder aus

S

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

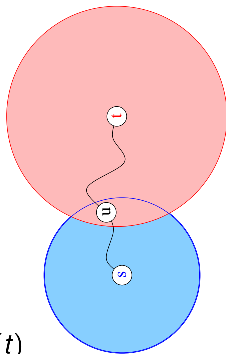
HH-TNR: Locality Filter

Vorgehen:

- speicher für jeden Knoten den (euklidischen) Abstand zum am weitesten entfernten lokalen Knoten ab, also $\overrightarrow{K}(u)$ und $\overleftarrow{K}(u)$
- geht während Vorberechnung
- locality filter schlägt zu, wenn

$$||s - t|| < \overrightarrow{K}(s) + \overleftarrow{K}(t)$$

- dadurch manchmal falscher Alarm



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level HH-TNR [SS08]

Gegeben:

- $L + 1$ Mengen von *transit nodes* $V =: T_0 \supseteq T_1 \supseteq \cdots \supseteq T_L$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level HH-TNR [SS08]

Karlsruher Institut für Technologie

Gegeben:

- $L + 1$ Mengen von *transit nodes* $V =: T_0 \supseteq T_1 \supseteq \cdots \supseteq T_L$

Betrachte für jeden Level ℓ mit $0 \leq \ell \leq L$:

- *access mapping* $\overrightarrow{A}_\ell : V \to 2^{T_\ell}$ mapt Knoten auf *access nodes*

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level HH-TNR [SS08]

KIT Karlsruher Institut für Technologie

Gegeben:

- $L + 1$ Mengen von *transit nodes* $V =: T_0 \supseteq T_1 \supseteq \cdots \supseteq T_L$

Betrachte für jeden Level ℓ mit $0 \leq \ell \leq L$:

- *access mapping* $\overrightarrow{A}_\ell : V \to 2^{T_\ell}$ mapt Knoten auf *access nodes*
- *locality filter* $L_\ell : V \times V \to \{\texttt{true}, \texttt{false}\}$ gibt an, ob $d(s, t)$ *nicht* mit Transit-Knoten der Level $\geq \ell$ berechnet werden kann

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level HH-TNR [SS08]

Karlsruher Institut für Technologie

Gegeben:

- $L + 1$ Mengen von *transit nodes* $V =: T_0 \supseteq T_1 \supseteq \cdots \supseteq T_L$

Betrachte für jeden Level ℓ mit $0 \leq \ell \leq L$:

- *access mapping* $\overrightarrow{A}_\ell : V \to 2^{T_\ell}$ mapt Knoten auf *access nodes*
- *locality filter* $L_\ell : V \times V \to \{\texttt{true}, \texttt{false}\}$ gibt an, ob $d(s,t)$ *nicht* mit Transit-Knoten der Level $\geq \ell$ berechnet werden kann
- *Distanztabelle* $D_\ell : T_\ell \times T_\ell \to \mathbb{R}_0^+ \cup \{\infty\}$ Abstände zwischen Transit-Knoten auf Level ℓ bis auf solche, die mit höheren Leveln berechnet werden können

Multi-Level HH-TNR [SS08]

Gegeben:

- $L + 1$ Mengen von *transit nodes* $V =: T_0 \supseteq T_1 \supseteq \cdots \supseteq T_L$

Betrachte für jeden Level ℓ mit $0 \leq \ell \leq L$:

- *access mapping* $\overrightarrow{A}_\ell : V \to 2^{T_\ell}$ mapt Knoten auf *access nodes*
- *locality filter* $L_\ell : V \times V \to \{\texttt{true}, \texttt{false}\}$ gibt an, ob $d(s,t)$ *nicht* mit Transit-Knoten der Level $\geq \ell$ berechnet werden kann
- *Distanztabelle* $D_\ell : T_\ell \times T_\ell \to \mathbb{R}_0^+ \cup \{\infty\}$ Abstände zwischen Transit-Knoten auf Level ℓ bis auf solche, die mit höheren Leveln berechnet werden können
- $d_\ell : V \times V \to \mathbb{R}_0^+ \cup \{\infty\}$ gibt Abstand zwischen zwei Knoten auf einem Level an, also Abstand zu access nodes und zwischen Transit-Knoten auf Level ℓ

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level HH-TNR [SS08]

KIT Karlsruher Institut für Technologie

Gegeben:

- $L + 1$ Mengen von *transit nodes* $V =: T_0 \supseteq T_1 \supseteq \cdots \supseteq T_L$

Betrachte für jeden Level ℓ mit $0 \leq \ell \leq L$:

- *access mapping* $\overrightarrow{A}_\ell : V \to 2^{T_\ell}$ mapt Knoten auf *access nodes*
- *locality filter* $L_\ell : V \times V \to \{\texttt{true}, \texttt{false}\}$ gibt an, ob $d(s,t)$ *nicht* mit Transit-Knoten der Level $\geq \ell$ berechnet werden kann
- *Distanztabelle* $D_\ell : T_\ell \times T_\ell \to \mathbb{R}_0^+ \cup \{\infty\}$ Abstände zwischen Transit-Knoten auf Level ℓ bis auf solche, die mit höheren Leveln berechnet werden können
- $d_\ell : V \times V \to \mathbb{R}_0^+ \cup \{\infty\}$ gibt Abstand zwischen zwei Knoten auf einem Level an, also Abstand zu access nodes und zwischen Transit-Knoten auf Level ℓ
- $d_{\geq \ell} : V \times V \to \mathbb{R}_0^+ \cup \{\infty\}$ gibt Abstand an, der mit Hilfe aller Level $\geq \ell$ berechnet werden kann

ML: Eigenschaften

Karlsruher Institut für Technologie

Bemerkungen:

- In der Distanztabelle werden nur relevante Informationen abgelegt

$$D_\ell(s,t) := \begin{cases} d_\ell(s,t) & \text{wenn} \quad d_\ell(s,t) < d_{\geq \ell+1}(s,t) \\ \infty & \text{sonst} \end{cases}$$

- d_ℓ gibt die Distanz in dem entsprechenden Level an

$$d_\ell(s,t) := \min_{\substack{u \in \overrightarrow{A}(s) \\ v \in \overleftarrow{A}(t)}} \{d(s,u) + D_\ell(u,v) + d(v,t)\}$$

- $d_{\geq \ell}$ gibt die Distanz aller Level $\geq \ell$ an

$$d_{\geq \ell}(s,t) := \min_{k \geq \ell} d_k(s,t)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

ML: TNR Anfragen

Karlsruher Institut für Technologie

TNR-Query(s,t)

```
1 d' = ∞
2 for ℓ = L down to 0 do
3     d' := min{d', d_ℓ(s, t)}
4     if ¬L_ℓ(s, t) then break
5 return d'
```

Bemerkungen:

- speichere in D_ℓ nur nicht-∞-Einträge (Hash-Tabelle)
- benutze HH/CH (oder bel. andere Technik) für Level 0 Anfragen

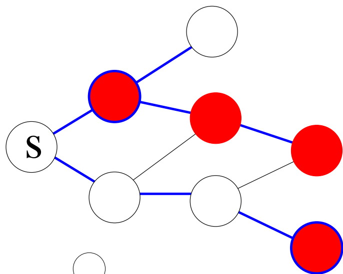
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

ML: Access-Nodes berechnen

KIT
Karlsruher Institut für Technologie

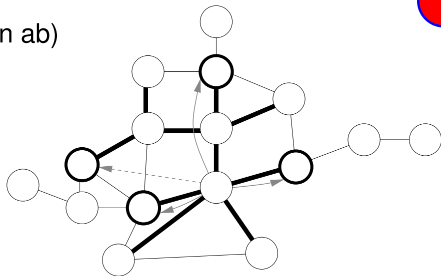
Vorgehen:

- lokale Suche, bis Transit-Knoten des Levels alle Zweige abdecken



Beschleunigung:

- *prune* an Transit-Knoten (breche Zweig bei Transit-Knoten ab)
- dünne dann später mit Hilfe der Distanz-Tabelle wieder aus



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

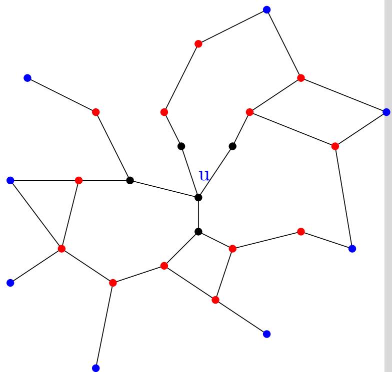
ML: “Runter”propagieren von ANs

Idee: Access-Nodes können propagiert werden

- Berechne $\overrightarrow{A}_\ell(v)$ für alle $v \in T_{\ell-1}$
- Berechne $\overrightarrow{A}_{\ell-1}$ für alle $u \in V$
- Schließe von $\ell - 1$ auf ℓ:

$$\overrightarrow{A}_\ell(u) := \bigcup_{v \in \overrightarrow{A}_{\ell-1}(u)} \overrightarrow{A}_\ell(v)$$

- kann auf mehrere Level verallgemeinert werden
  Berechne ANs für TNs “runter”,
  berechne ANs für “normale” Knoten “hoch”
- können mit Distanztabellen wieder ausgedünnt werden



ML: Distanztabellen

Karlsruher Institut für Technologie

Vorgehen:

- benutze Many-to-Many Ansatz (später mehr Details dazu!)

Problem:

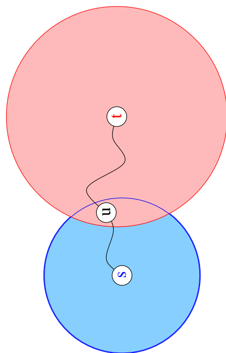
- wir speichern $d_\ell(u, v)$ in $D_\ell(u, v)$ nur, wenn $d_\ell(u, v) < d_{\ell+1}(u, v)$
- Anzahl Transit Knoten auf niederigem Level hoch $\leadsto$ ineffizient

Lösungen:

- gehe top-down vor, dann ist $d_{\ell+1}(u, v)$ verfügbar
- breche Suchen an hochleveligen Transit-Knoten ab

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

ML: Locality Filter



Vorgehen:

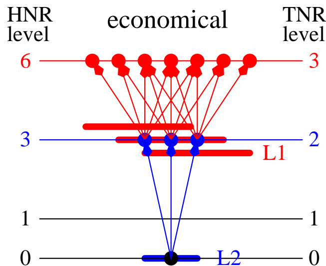
- speicher für jeden Knoten den (euklidischen) Abstand zum am weitesten entfernten lokalen Knoten ab, also $\overrightarrow{K}(u)$ und $\overleftarrow{K}(u)$
- für jeden Level, also $\overrightarrow{K}_\ell(u)$, $\overleftarrow{K}_\ell(u)$
- geht während Vorberechnung
- locality filter schlägt zu, wenn

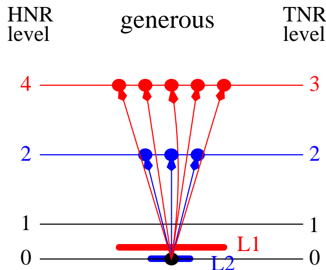
$$||s - t|| < \overrightarrow{K}_k(s) + \overleftarrow{K}_k(t) \quad \forall k \leq \ell$$

- dadurch manchmal falscher Alarm

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

ML HH-TNR: 2 Varianten





economical:

- speicher keine transit knoten für level 1, benutze CH/HH
- oberster locality filter auf level der transit Knoten

generous:

- mehr transit nodes
- locality filter auf unterstem Level
- höherer Platzverbrauch, schneller

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

TNR Reconsidered: CH-TNR

Karlsruher Institut für Technologie

Idee: TNR rein auf CH basiert. Keine geometrischen Zutaten!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

TNR Reconsidered: CH-TNR

Karlsruher Institut für Technologie

Idee: TNR rein auf CH basiert. Keine geometrischen Zutaten!

- Studienarbeit Julian Arz
- Konferenzbeitrag SEA'13, Juni 2013
- Vielen Dank an Julian und Dennis Luxen
  kann gut sein, dass wir diesen Teil erst nach der SEA online stellen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

TNR Reconsidered

Karlsruher Institut für Technologie

Ingredients

- Fallback routine: CH query
- Transit nodes: CH, top k nodes
- Distance table: [KSSSW07] (spätere VL)
- Access node mapping & distances: CH-search space exploration
- Locality filter: CH-search space exploration $+$ graph Voronoi
- Geometry: none

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimental evaluation

Karlsruher Institut für Technologie

Tabelle: Comparison of locality filter quality. Values for last three columns in [%]

Method	$\lvert T\rvert$	Local queries	$L =$ true	False Pos.
Grid-TNR	7 426	2.600	2.60	-
Grid-TNR	24 899	0.800	0.80	-
HH-TNR-eco	8 964	0.540	2.87	81.2
HH-TNR-gen	11 293	0.260	1.35	80.7
CH-TNR	10 000	0.148	0.41	64.3
CH-TNR(compr)	10 000	0.148	0.58	73.6
CH-TNR(compr)	24 000	0.048	0.17	72.1
CH-TNR(compr)	28 000	0.041	0.14	71.2

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimental evaluation

Karlsruher Institut für Technologie

Tabelle: Comparison Between Various Distance Oracles. AMD Opteron 270.
$\star$ scaled figures

Method	Preprocessing [hh:mm]	Overhead [byte]	Query [μs]	
CH	00:13	24	246	
Grid-TNR	≈20:00	21	63	
HH-TNR-eco	00:25	120	11	
HH-TNR-gen	01:15	247	4.30	
CH-TNR-geo-gen	00:46	193	3.30	
TNR+AF	03:49	321	1.90	
HL-0 local	00:35	1341	1.34	$\star$
HL-∞ global	≈120:00	1055	0.49	$\star$
CH-TNR	00:34	147	3.27	

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimental evaluation

Karlsruher Institut für Technologie

Tabelle: Comparison Between Various Distance Oracles. AMD Opteron 270. ⋆ scaled figures

Method	Preprocessing [hh:mm]	Overhead [byte]	Query [μs]	
CH	00:13	24	246	
Grid-TNR	≈20:00	21	63	
HH-TNR-eco	00:25	120	11	
HH-TNR-gen	01:15	247	4.30	
CH-TNR-geo-gen	00:46	193	3.30	
TNR+AF	03:49	321	1.90	
HL-0 local	00:35	1341	1.34	⋆
HL-∞ global	≈120:00	1055	0.49	⋆
CH-TNR	00:34	147	3.27	
HL-Compression	00:59	100	5.74	⋆

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pfadentpackung

Karlsruher Institut für Technologie

- analog zu Contraction Hierarchies
- interpretiere jeden Eintrag als Shortcut
- speichere für jeden Eintrag den Pfad, den er repräsentiert
- rekursiv

Vergleich zu CH

Karlsruher Institut für Technologie

Gemeinsamkeiten:

- Level-Einteilung
- Pfadentpackung ähnlich
- (TNR basiert auf CH)

Unterschiede:

- weniger level
- Suche durch Table-Lookups ersetzt
- Locality Filter nötig
- Shortcuts $\mapsto$ Tabellen-Einträgen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Übersicht: bisherige Techniken

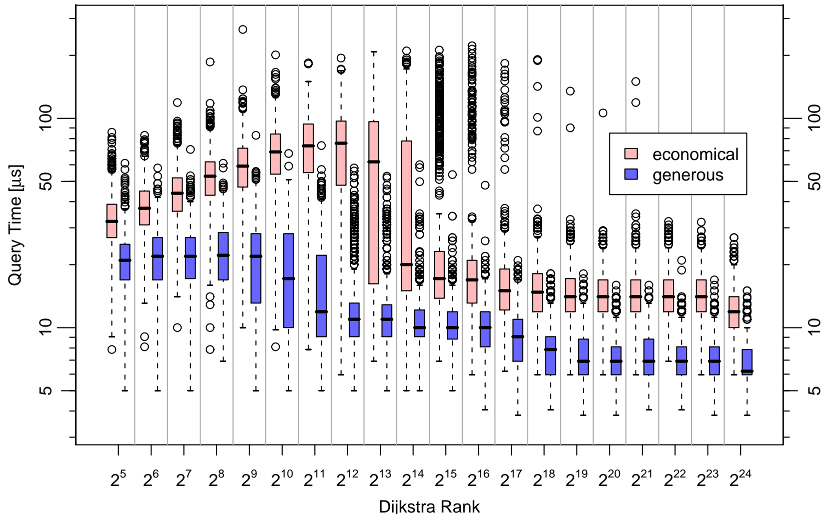
Karlsruher Institut für Technologie

	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
MLD-3	$<$ 0:01	1.7	6 074	0.91	6 143
CH	0:02	-3.0	284	0.09	62 128
HL	6:12	967.1	N/A	0.0002	ca. 22 000 000
eco TNR	0:25	120	N/A	0.011	ca. 500 000
gen TNR	1:15	247	N/A	0.0043	ca. 1 300 000
CH-TNR	00:34	147	N/A	0.0033	ca. 1 700 000

Vorsicht: ggfs. verschiedene Rechner...

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Dijkstra Rank



Zusammenfassung

Karlsruher Institut für Technologie

Transit-Node Routing

- ersetzt Suche (fast) komplett durch Table-Lookups
- 4 Zutaten:
  - Transit-Nodes
  - Distanztabelle
  - Access-Nodes
  - Locality-Filter
- Suchzeit von unter 4 μs

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ende

Karlsruher Institut für Technologie

Literatur (Transit-Node Routing):

- Peter Sanders and Dominik Schultes:
  Robust, Almost Constant Time Shortest-Path Queries in Road Networks
  In: *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, 2009
- Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders, Dominik Schultes:
  In Transit to Constant Time Shortest-Path Queries in Road Networks
  In: *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX'07)*, 2009
- Julian Arz, Dennis Luxen and Peter Sanders:
  Transit Node Routing Reconsidered
  In: *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, June 2013, to appear.

Nächste Termine

KIT
Karlsruher Institut für Technologie

Montag, 27.5.2013 (Julian)

Mittwoch, 29.5.2013 (Julian)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik