Karlsruher Institut für Technologie

Algorithmen für Routenplanung

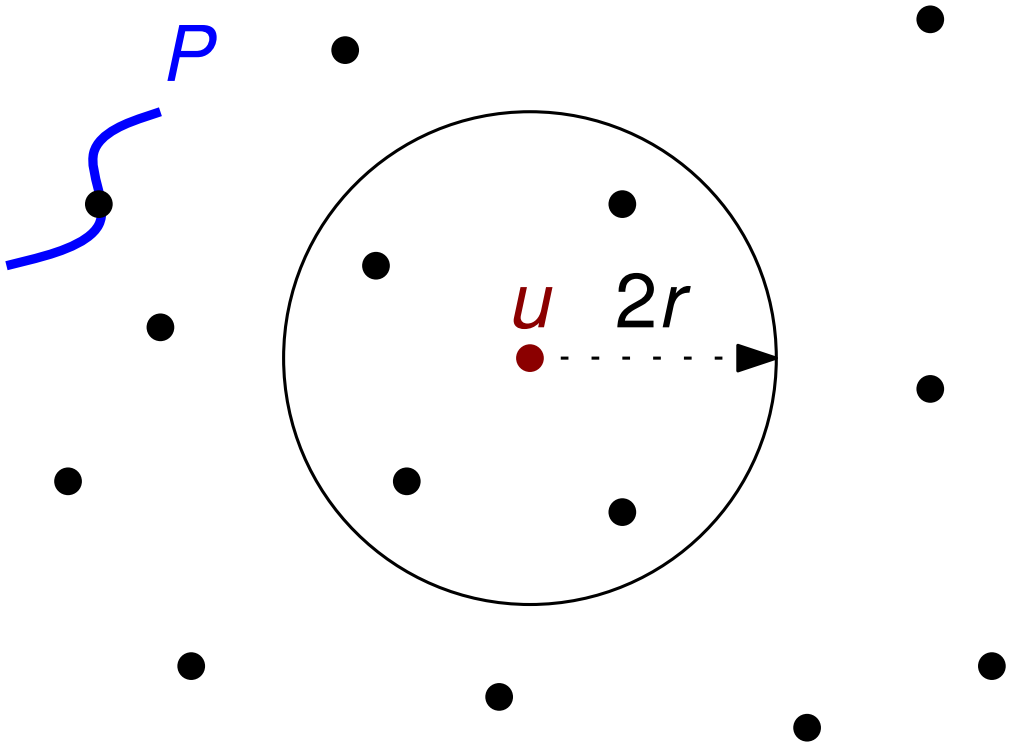
17. Sitzung, Sommersemester 2011

Thomas Pajor | 7. Juli 2011

Institut für Theoretische Informatik · Algorithmik I · Prof. Dr. Dorothea Wagner

Letztes Mal

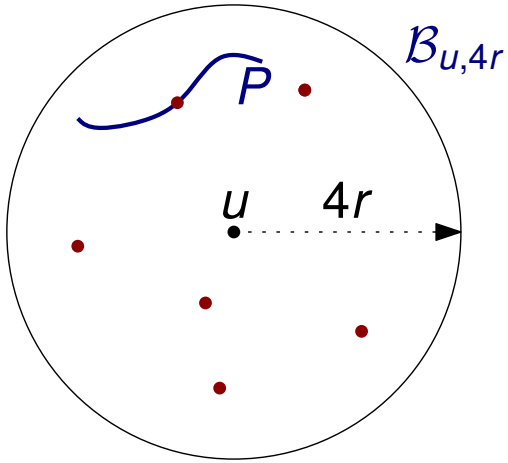
Highway-Dimension und Shortest-Path Cover



Highway Dimension

Idee:
Lokal überdeckt eine kleine Menge Knoten alle hinreichend langen kürzesten Wege.

Gegeben:
Ungerichteter Graph $G = (V, E, \text{len})$.



Definition

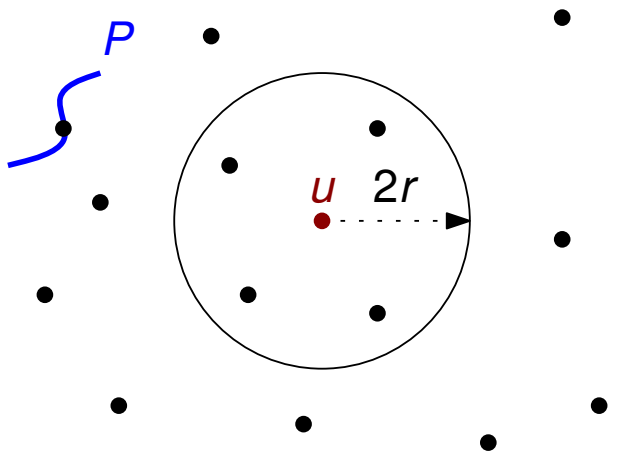
Die *Highway-Dimension* von G ist die kleinste Zahl $h \in \mathbb{N}$, so dass

- Für alle $r \in \mathbb{R}^+$ und
- für alle Knoten $u \in V$
- existiert eine Menge $S \subseteq \mathcal{B}_{u,4r}$ mit $|S| \leq h$ so, dass

$$|P(v,w)| > r \text{ und } P(v,w) \subseteq \mathcal{B}_{u,4r} \quad \Rightarrow \quad P(v,w) \cap S \neq \emptyset$$

Shortest-Path Cover (SPC)

Idee:
Alle kürzesten Wege einer gewissen Länge können mit einer kleinen Menge von Knoten überdeckt werden.



Gegeben:
Ungerichteter Graph $G = (V, E, \text{len})$.

Definition

Eine Menge C heißt (r, k)-SPC von G genau dann wenn

- Für alle kürzesten Wege P mit $r < |P| \leq 2r$ gilt dass
- $P \cap C \neq \emptyset$ und
- für alle $u \in V$: $|C \cap \mathcal{B}_{u,2r}| \leq k$

Highway-Dimension und SPCs

Karlsruher Institut für Technologie

Theorem

Wenn G Highway-Dimension h hat, dann $\forall r \; \exists$ ein (r, h)-SPC.

- Kleine Highway-Dimension führt zu kleinen SPCs
- Für alle r!

Highway Dimension und SPCs

Problem:
Wie lässt sich C^* effizient konstruieren?

Ein minimales (r, h)-SPC zu finden ist $\mathcal{NP}$-schwer.

Also: Approximation.

Highway Dimension und SPCs

Theorem

Wenn G Highway-Dimension h hat, dann lässt sich für alle r in polynomieller Zeit ein $(r, O(h \log n))$-SPC konstruieren.

Idee:

Greedy Set-Cover Algorithmus liefert $O(\log n)$-Approximation von C^*.

Highway Dimension und SPCs

Vorgehen zur Konstruktion eines $(r, O(h \log n))$-SPC:

- Beginne mit $C = \emptyset$
- Wähle iterativ den Knoten zu C der die meisten nicht-überdeckten KW überdeckt

Beweis

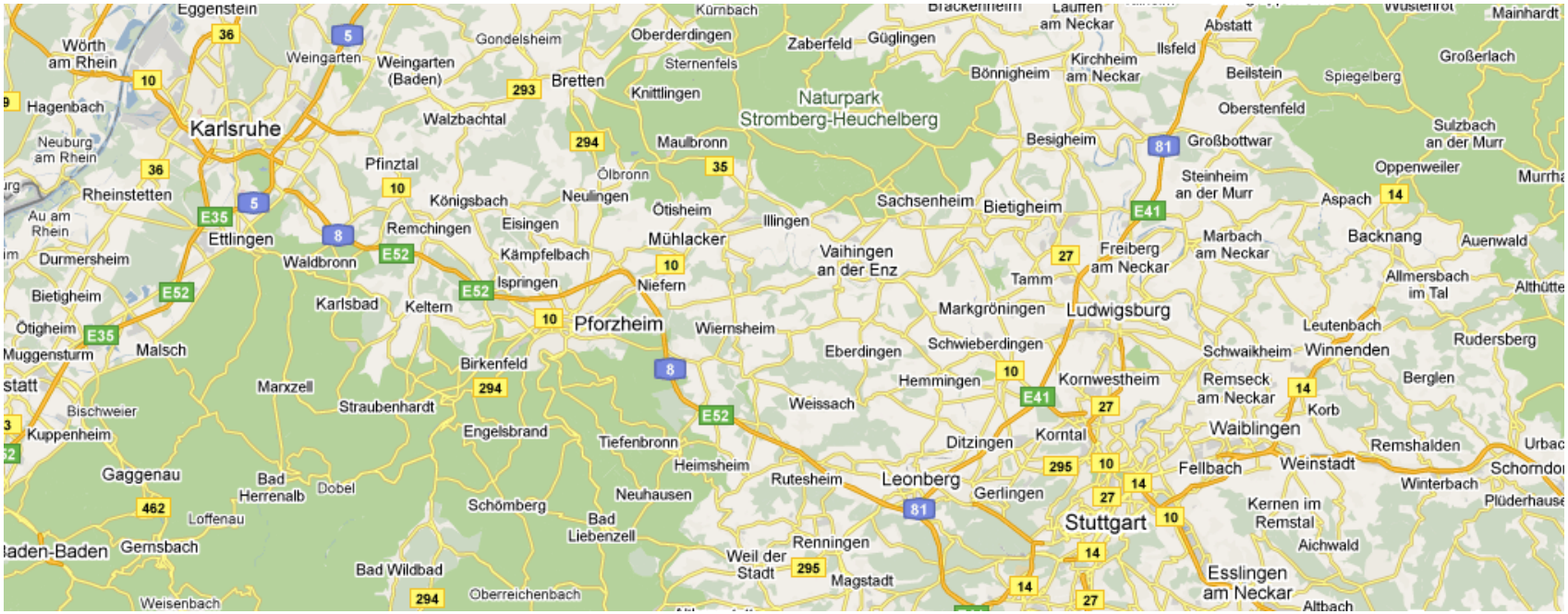
- In jedem Schritt:
  Mind. $1/h$ der unüberdeckten KW P mit $r < |P| \leq 2r$ wird überdeckt
- Es gibt $O(n^2)$ KW in G

$\Rightarrow$ Höchstens $O(h \log n)$ Knoten werden ausgewählt

HD von Straßennetzwerken

Vermutung:

Straßennetzwerke haben kleine (konstante) Highway-Dimension.



Generischer Prepro.-Algorithmus

Ziel:
Konstruktion eines Preprocessing-Verfahrens, so dass

- Platzverbrauch “klein” (möglichst linear in n)
- über die Query später Gütegarantien ausgesagt werden können
  - RE
  - CH
  - TNR
  - SHARC
  - . . . ?
- in Abhängigkeit von h statt n
- Zum Vergleich: Dijkstra $\in O(m + n \log n)$

Idee:
Benutze Preprocessing von Contraction Hierarchies (modifiziert)

Wdh.: Contraction Hierarchies

Preprocessing:

- ordne Knoten nach “Wichtigkeit”
- kontrahier Knoten in dieser Ordnung
- Knoten v wird kontrahiert durch

KONTRAHIERE(v)

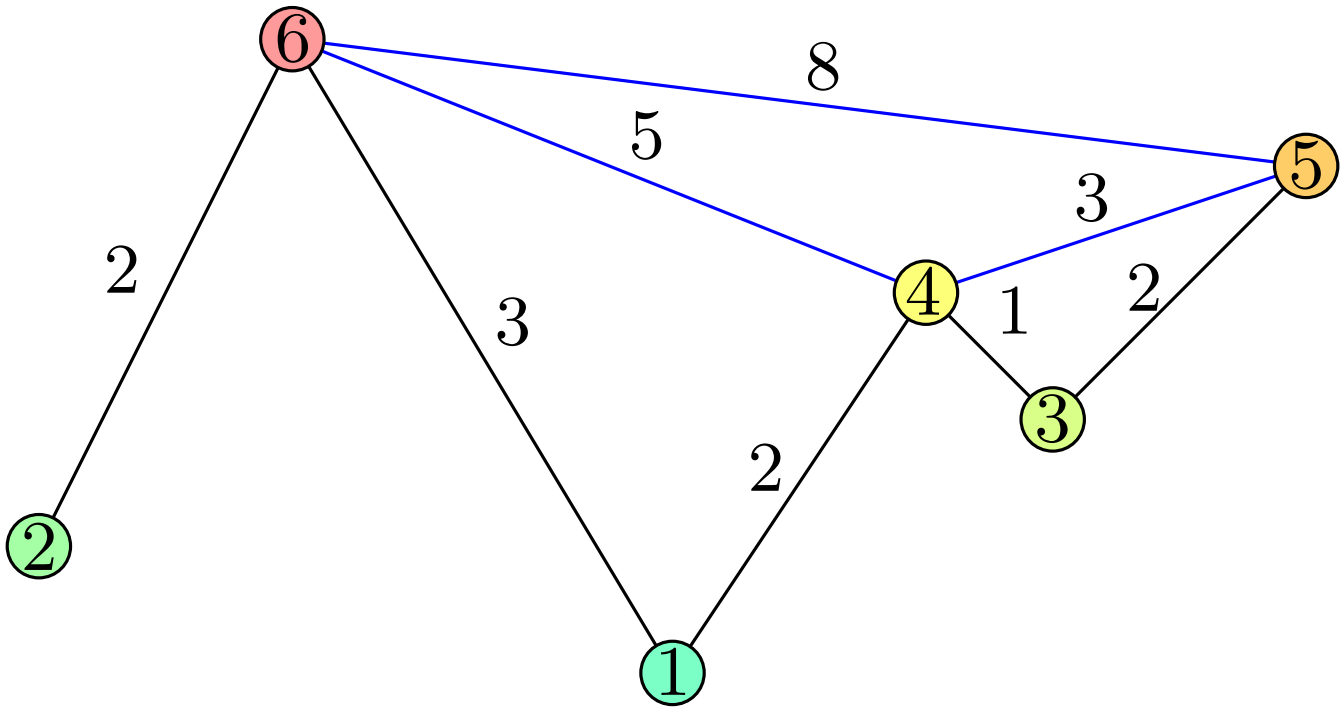
```
1 für alle Paare (u, v) und (v, w) von Kanten tue
2     wenn (u, v, w) eindeutiger kürzester Weg dann
3         Füge Shortcut (u, w) mit Gewicht len(u, v) + len(v, w) ein
```

Query:

- Bidirektional
- Vorwärtssuche: Relaxiert nur Kanten zu wichtigeren Knoten
- Rückwärtssuche: Relaxiert nur Kanten von wichtigeren Knoten

Wdh.: Contraction Hierarchies

Freiheitsgrad: In welcher Reihenfolge Knoten kontrahieren?



Generischer Prepro.-Algorithmus

Idee: Partitioniere V zu einer Hierarchie $L_0, L_1, \ldots, L_{\log D}$

Vorgehen

- Sei $C_0 := V$
- Sei C_i ein $(2^i, k)$-SPC für $i = 1 \ldots \log D$ wobei
  - $k = h$ für unbeschränktes Preprocessing
  - $k = h \log n$ für polynomialzeit Preprocessing (Approx.)
- Dann ist

$$L_i := C_i \setminus \bigcup_{j=i+1}^{\log D} C_j$$

- Kontrahiere Knoten in der Reihenfolge induziert durch L_i.

Ausgabe:

- Menge E^+ von eingefügten Shortcuts
- Hierarchie $L_0, L_1, \ldots, L_{\log D}$

Generischer Prepro.-Algorithmus

Idee: Partitioniere V zu einer Hierarchie $L_0, L_1, \ldots, L_{\log D}$

Vorgehen

- Sei $C_0 := V$
- Sei C_i ein $(2^i, k)$-SPC für $i = 1 \ldots \log D$ wobei
  - $k = h$ für unbeschränktes Preprocessing
  - $k = h \log n$ für polynomialzeit Preprocessing (Approx.)
- Dann ist

$$L_i := C_i \setminus \bigcup_{j=i+1}^{\log D} C_j$$

- Kontrahiere Knoten in der Reihenfolge induziert durch L_i.

Ausgabe:

- Menge E^+ von eingefügten Shortcuts
- Hierarchie $L_0, L_1, \ldots, L_{\log D}$

Generischer Prepro.-Algorithmus

Idee: Partitioniere V zu einer Hierarchie $L_0, L_1, \ldots, L_{\log D}$

Vorgehen

- Sei $C_0 := V$
- Sei C_i ein $(2^i, k)$-SPC für $i = 1 \ldots \log D$ wobei
  - $k = h$ für unbeschränktes Preprocessing
  - $k = h \log n$ für polynomialzeit Preprocessing (Approx.)
- Dann ist

$$L_i := C_i \setminus \bigcup_{j=i+1}^{\log D} C_j$$

- Kontrahiere Knoten in der Reihenfolge induziert durch L_i.

Ausgabe:

- Menge E^+ von eingefügten Shortcuts
- Hierarchie $L_0, L_1, \ldots, L_{\log D}$

Preprocessing: Platz

Lemma

Für $v \in L_i$ und festes $j > i$ ist die Anzahl Kanten $(v, w) \in E^+$ mit $w \in L_j$ höchstens k.

Beweis:

- Kante (v, w) repräsentiert Pfad P dessen innere Knoten *vor* v und w kontrahiert wurden.

$\Rightarrow$ Innere Knoten gehören zu L_x für $x \leq i \leq j$.

$\Rightarrow$ $w \in \mathcal{B}_{v,2 \cdot 2^j}$ (da sonst P durch u mit $u \in L_{y>j}$ abgedeckt sein müsste).

- Def. $(2^j, k)$-SPC besagt, dass $B_{v,2 \cdot 2^j}$ höchstens k Knoten aus L_j enthält.

Preprocessing: Platz

Lemma

Für $v \in L_i$ und festes $j > i$ ist die Anzahl Kanten $(v, w) \in E^+$ mit $w \in L_j$ höchstens k.

Beweis:

- Kante (v, w) repräsentiert Pfad P dessen innere Knoten *vor* v und w kontrahiert wurden.

$\Rightarrow$ Innere Knoten gehören zu L_x für $x \leq i \leq j$.

$\Rightarrow$ $w \in \mathcal{B}_{v,2\cdot 2^j}$ (da sonst P durch u mit $u \in L_{y>j}$ abgedeckt sein müsste).

- Def. $(2^j, k)$-SPC besagt, dass $B_{v,2\cdot 2^j}$ höchstens k Knoten aus L_j enthält.

Preprocessing: Platz

Theorem

Für jeden Graphen G mit Highway Dimension h gibt es eine Knotenordnung, durch die das CH-Preprocessing eine Menge von Shortcuts E^+ so erzeugt, dass gilt

- *Der Grad jedes Knotens in $G^+ = (V, E \cup E^+)$ ist höchstens $\Delta + h \log D$*
- $|E^+| \in O(nh \log D)$

Für Polynomialzeit-Preprocessing:

- Maximalgrad in G^+ ist in $O(\Delta + h \log n \log D)$
- $|E^+| \in O(nh \log n \log D)$

Queries

Suchraumgröße und Queryzeit

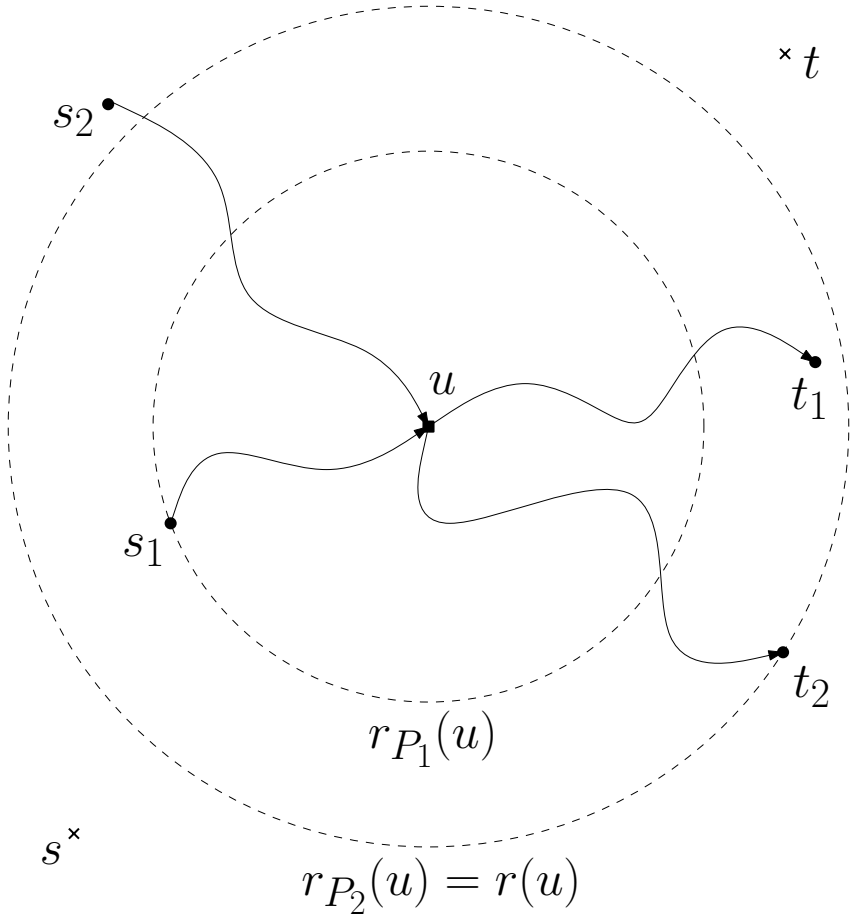
Erinnerung:
Die meisten Query-Algorithmen sind Dijkstra-basiert.

$\Rightarrow$ Aufwand wird dominiert von Queue-Operationen auf Knoten

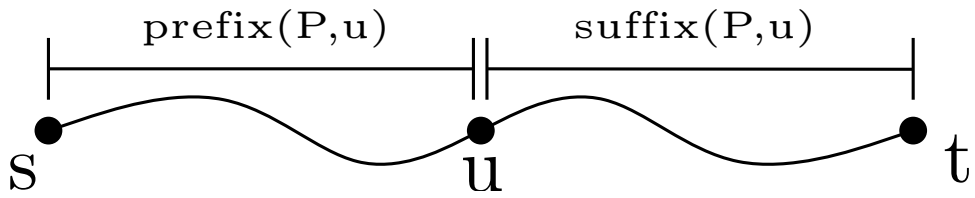
Also: Suchraumgröße $\approx$ Queryzeit

Vereinfachung:
Im Folgenden werden Suchraumgröße und Queryzeit gleichgesetzt.

1. Reach



Reach



Definition:

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

$$r_P(u) := \min\{\text{len}(P_{su}), \text{len}(P_{ut})\}$$

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

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

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, kann u geprunt werden

Query-Variante:

- Bidirectional Distance-Bounding Reach-Dijkstra
- Alternierungsstrategie: $\min\{\text{minKey}(\overrightarrow{Q}), \text{minKey}(\overleftarrow{Q})\}$
- Bei gleich langen Wegen: Bevorzuge immer den Hopminimalen!

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, kann u geprunt werden

Query-Variante:

- Bidirectional Distance-Bounding Reach-Dijkstra
- Alternierungsstrategie: $\min\{\text{minKey}(\overrightarrow{Q}), \text{minKey}(\overleftarrow{Q})\}$
- Bei gleich langen Wegen: Bevorzuge immer den Hopminimalen!

Berechnung von Reach (RE)

Ursprüngliche Berechnung von Reach:

- wähle ϵ frei
- iterativ, solange $E \neq \emptyset$
  - berechne obere Schranken mit part. KW-Bäumen der Höhe $\approx 2\epsilon$
  - entferne alle Kanten mit Reach $< \epsilon$ aus Graphen
  - setze $\epsilon = k \cdot \epsilon$
- wandle Kanten-Reach in Knoten-Reach um

Modifizierte Vorberechnung durch folgendes Lemma

Lemma

Für alle $v \in L_i$ gilt:

$$r(v) \leq 2 \cdot 2^i \text{ in } G^+ = (V, E \cup E^+)$$

Berechnung von Reach (RE)

Ursprüngliche Berechnung von Reach:

- wähle ϵ frei
- iterativ, solange $E \neq \emptyset$
  - berechne obere Schranken mit part. KW-Bäumen der Höhe $\approx 2\epsilon$
  - entferne alle Kanten mit Reach $< \epsilon$ aus Graphen
  - setze $\epsilon = k \cdot \epsilon$
- wandle Kanten-Reach in Knoten-Reach um

Modifizierte Vorberechnung durch folgendes Lemma

Lemma

Für alle $v \in L_i$ gilt:

$$r(v) \leq 2 \cdot 2^i \text{ in } G^+ = (V, E \cup E^+)$$

Berechnung von Reach (RE)

Lemma

Für alle $v \in L_i$ gilt:

$$r(v) \leq 2 \cdot 2^i \text{ in } G^+ = (V, E \cup E^+)$$

Beweis:

Ann.: Es gibt $v \in L_i$ mit $r(v) > 2 \cdot 2^i$.

- $\Rightarrow$ $\exists$ kürzester Weg $P(x, v, y)$ mit $v \in P(x, v, y)$ sowie $|P(x, v)| > 2 \cdot 2^i$ und $|P(v, y)| > 2 \cdot 2^i$.
- $\Rightarrow$ Sowohl $P(x, v)$ alsauch $P(v, y)$ enthalten Knoten aus L_j mit $j > i$.
- $\Rightarrow$ Es gibt einen Shortcut zwischen $P(x, v)$ und $P(v, y)$.
- $\Rightarrow$ $P(x, v, y)$ ist kein kürzester Weg (# Hops). $\lightning$

Berechnung von Reach (RE)

Lemma

Für alle $v \in L_i$ gilt:

$$r(v) \leq 2 \cdot 2^i \text{ in } G^+ = (V, E \cup E^+)$$

Beweis:

Ann.: Es gibt $v \in L_i$ mit $r(v) > 2 \cdot 2^i$.

- $\Rightarrow$ $\exists$ kürzester Weg $P(x, v, y)$ mit $v \in P(x, v, y)$ sowie $|P(x, v)| > 2 \cdot 2^i$ und $|P(v, y)| > 2 \cdot 2^i$.
- $\Rightarrow$ Sowohl $P(x, v)$ alsauch $P(v, y)$ enthalten Knoten aus L_j mit $j > i$.
- $\Rightarrow$ Es gibt einen Shortcut zwischen $P(x, v)$ und $P(v, y)$.
- $\Rightarrow$ $P(x, v, y)$ ist kein kürzester Weg (# Hops). $\lightning$

Theorem

Die Query-Zeit von RE ist in

- $O((\Delta + h \log D)(h \log D))$ *für unbeschränktes Preprocessing*
- $O((\Delta + h \log n \log D)(h \log n \log D))$ *für pol. Preprocessing*

Beweis: Zusammensetzung der Schranke:

$$\underbrace{(\Delta + k \log D)}_{\text{max. Knotengrad in } G^+} \cdot \underbrace{(k \log D)}_{\text{max. Suchraum}}$$

Betrachte die Vorwärtssuche von s (Rückwärtssuche analog)

- Betrachte die Kugel $\mathcal{B}_{s,2 \cdot 2^i}$ um s.
- Die Suche arbeitet wegen $r(v) \leq 2 \cdot 2^i$ keine Knoten $v \in L_i$ mit $v \notin \mathcal{B}_{s,2 \cdot 2^i}$ ab.
- L_i ist ein $(2^i, k)$-SPC, also $|L_i \cap \mathcal{B}_{s,2 \cdot 2^i}| \leq k$

$\Rightarrow$ Höchstens k Knoten werden pro Level abgearbeitet

Theorem

Die Query-Zeit von RE ist in

- $O((\Delta + h \log D)(h \log D))$ *für unbeschränktes Preprocessing*
- $O((\Delta + h \log n \log D)(h \log n \log D))$ *für pol. Preprocessing*

Beweis: Zusammensetzung der Schranke:

$$\underbrace{(\Delta + k \log D)}_{\text{max. Knotengrad in } G^+} \cdot \underbrace{(k \log D)}_{\text{max. Suchraum}}$$

Betrachte die Vorwärtssuche von s (Rückwärtssuche analog)

- Betrachte die Kugel $\mathcal{B}_{s,2 \cdot 2^i}$ um s.
- Die Suche arbeitet wegen $r(v) \leq 2 \cdot 2^i$ keine Knoten $v \in L_i$ mit $v \notin \mathcal{B}_{s,2 \cdot 2^i}$ ab.
- L_i ist ein $(2^i, k)$-SPC, also $|L_i \cap \mathcal{B}_{s,2 \cdot 2^i}| \leq k$

$\Rightarrow$ Höchstens k Knoten werden pro Level abgearbeitet

Theorem

Die Query-Zeit von RE ist in

- $O((\Delta + h \log D)(h \log D))$ *für unbeschränktes Preprocessing*
- $O((\Delta + h \log n \log D)(h \log n \log D))$ *für pol. Preprocessing*

Beweis: Zusammensetzung der Schranke:

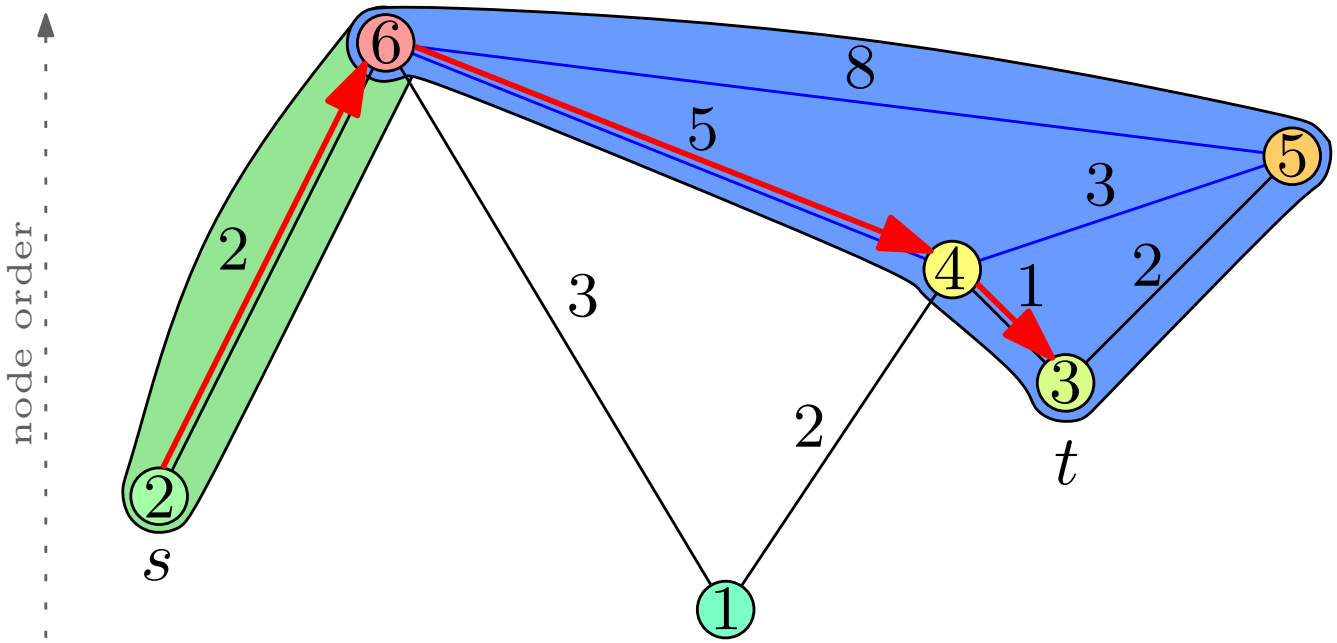
$$\underbrace{(\Delta + k \log D)}_{\text{max. Knotengrad in } G^+} \cdot \underbrace{(k \log D)}_{\text{max. Suchraum}}$$

Betrachte die Vorwärtssuche von s (Rückwärtssuche analog)

- Betrachte die Kugel $\mathcal{B}_{s,2 \cdot 2^i}$ um s.
- Die Suche arbeitet wegen $r(v) \leq 2 \cdot 2^i$ keine Knoten $v \in L_i$ mit $v \notin \mathcal{B}_{s,2 \cdot 2^i}$ ab.
- L_i ist ein $(2^i, k)$-SPC, also $|L_i \cap \mathcal{B}_{s,2 \cdot 2^i}| \leq k$

$\Rightarrow$ Höchstens k Knoten werden pro Level abgearbeitet

2. Contraction Hierarchies



CH: Query

Betrachte vereinfachte Query:

- Führe volle bidirektionale Suche durch
- Benutze *kein* Stoppkriterium
  - $\Rightarrow$ Der gesamte erreichbare Graph wird abgearbeitet!
- Relaxiere nur Kanten zu wichtigeren Knoten (wie gehabt)

Variante funktioniert in der Praxis bereits sehr gut

CH: Query

Problem:
Reach-Schranken gelten nicht für CH-Query.

Lösungsansätze:

1. Benutze zusätzlich Reach-Pruning
2. Füge zusätzliche Shortcuts beim Preprocessing ein

Hier letzteres:

- Bei Knotenreduktion von $v \in L_i$:
  Erzeuge Shortcut (u, w) für jedes Paar $u, w \in \mathcal{B}_{v,2 \cdot 2^i}$ für das $v \in P(u, w)$.
- Gib für jeden Knoten v lediglich sein Level L_i (also i) aus.

CH: Query

Problem:
Reach-Schranken gelten nicht für CH-Query.

Lösungsansätze:

1. Benutze zusätzlich Reach-Pruning
2. Füge zusätzliche Shortcuts beim Preprocessing ein

Hier letzteres:

- Bei Knotenreduktion von $v \in L_i$:
  Erzeuge Shortcut (u, w) für jedes Paar $u, w \in \mathcal{B}_{v,2 \cdot 2^i}$ für das $v \in P(u, w)$.
- Gib für jeden Knoten v lediglich sein Level L_i (also i) aus.

CH: Analyse

Beobachtungen:

- Platz-Schranken gelten weiterhin
- Für jeden kürzesten s-t-Weg:
  Es gibt weiterhin Weg in G^+, so dass zwei aufeinanderfolgende Knoten stets verschiedene Levels haben
- Ausnahme: oberstes Level kann genau zwei Nachbarn haben
  - $\rightsquigarrow$ Sonderbehandlung

Theorem

Die Query-Zeit von CH ist in

- $O((\Delta + h \log D)(h \log D))$ *für unbeschränktes Preprocessing*
- $O((\Delta + h \log n \log D)(h \log n \log D))$ *für pol. Preprocessing*

CH: Analyse

Beobachtungen:

- Platz-Schranken gelten weiterhin
- Für jeden kürzesten s-t-Weg:
  Es gibt weiterhin Weg in G^+, so dass zwei aufeinanderfolgende Knoten stets verschiedene Levels haben
- Ausnahme: oberstes Level kann genau zwei Nachbarn haben
  - $\rightsquigarrow$ Sonderbehandlung

Theorem

Die Query-Zeit von CH ist in

- $O((\Delta + h \log D)(h \log D))$ *für unbeschränktes Preprocessing*
- $O((\Delta + h \log n \log D)(h \log n \log D))$ *für pol. Preprocessing*

Labeling-Algorithmus

Generisches Preprocessing:

- $\forall v \in V$: Berechne Label $L(v) \subseteq V$ (sortiert nach Knoten-ID).
- $\forall w \in L(v)$: Berechne $\mathrm{dist}(v, w)$.

Labeling-Eigenschaft:
$\forall s, t \in V$: Es existiert KW $P(s, t)$ mit $L(s) \cap L(t) \cap P(s, t) \neq \emptyset$.

Query

Berechne $\min_{u \in L(s) \cap L(t)} (\mathrm{dist}(s, u) + \mathrm{dist}(u, t))$.

- Sehr einfacher Algorithmus: Linearer Mengen-Schnitt-Test.
- Schnell, wenn Label klein sind.

Labeling-Algorithmus

Generisches Preprocessing:

- $\forall v \in V$: Berechne Label $L(v) \subseteq V$ (sortiert nach Knoten-ID).
- $\forall w \in L(v)$: Berechne $\mathrm{dist}(v, w)$.

Labeling-Eigenschaft:
$\forall s, t \in V$: Es existiert KW $P(s, t)$ mit $L(s) \cap L(t) \cap P(s, t) \neq \emptyset$.

Query

Berechne $\min_{u \in L(s) \cap L(t)} (\mathrm{dist}(s, u) + \mathrm{dist}(u, t))$.

- Sehr einfacher Algorithmus: Linearer Mengen-Schnitt-Test.
- Schnell, wenn Label klein sind.

SPC-Labels

Idee: “TNR-Variante“ mit nur einem Access-Node pro Weg

Vorgehen: *Shortest-Path-Cover-Label*

- Sei C_i ein $(2^i, k)$-SPC für $i = 1 \dots \log D$ wobei
  - $k = h$ für unbeschränktes Preprocessing
  - $k = h \log n$ für polynomialzeit Preprocessing
- Setze für alle $v \in V$:

$$L(v) := \bigcup_i \left(C_i \cap \mathcal{B}_{2 \cdot 2^i}(v) \right)$$

Label-Größe

Für alle $v \in V$ ist $|L(v)| \in O(k \log D)$.

SPC-Labels

Idee: “TNR-Variante“ mit nur einem Access-Node pro Weg

Vorgehen: *Shortest-Path-Cover-Label*

- Sei C_i ein $(2^i, k)$-SPC für $i = 1 \dots \log D$ wobei
  - $k = h$ für unbeschränktes Preprocessing
  - $k = h \log n$ für polynomialzeit Preprocessing
- Setze für alle $v \in V$:

$$L(v) := \bigcup_i \left(C_i \cap \mathcal{B}_{2 \cdot 2^i}(v) \right)$$

Label-Größe

Für alle $v \in V$ ist $|L(v)| \in O(k \log D)$.

SPC-Labels

Theorem

Shortest Path Cover Label erfüllen die Labeling-Eigenschaft, das heißt für alle $s, t \in V$ gilt $P(s,t) \cap L(s) \cap L(t) \neq \emptyset$.

Beweis:

- Wähle i so dass $2^i < \mathrm{dist}(s,t) \leq 2 \cdot 2^i$.
- Dann gilt: $P(s,t) \subseteq \mathcal{B}_{2 \cdot 2^i}(s)$ und $P(s,t) \subseteq \mathcal{B}_{2 \cdot 2^i}(t)$.

$\Rightarrow$ Nach Def. SPC $\exists u \in P(s,t) \cap C_i$ und somit $u \in L(s) \cap L(t)$.

□

SPC-Labels

Corollary

Die Query-Zeit von SPC-Labels ist in

- $O(h \log D)$ *für unbeschränktes Preprocessing, und*
- $O(h \log n \log D)$ *für pol. Preprocessing.*

Zum Vergleich: Reach und CH

- $O((\Delta + h \log D)(h \log D))$ für unbeschränktes Preprocessing, und
- $O((\Delta + h \log n \log D)(h \log n \log D))$ für pol. Preprocessing.

SPC-Labels also echt besser als CH oder Reach?

SPC-Labels

Corollary

Die Query-Zeit von SPC-Labels ist in

- $O(h \log D)$ *für unbeschränktes Preprocessing, und*
- $O(h \log n \log D)$ *für pol. Preprocessing.*

Zum Vergleich: Reach und CH

- $O((\Delta + h \log D)(h \log D))$ für unbeschränktes Preprocessing, und
- $O((\Delta + h \log n \log D)(h \log n \log D))$ für pol. Preprocessing.

SPC-Labels also echt besser als CH oder Reach?

SPC-Labels

Corollary

Die Query-Zeit von SPC-Labels ist in

- $O(h \log D)$ *für unbeschränktes Preprocessing, und*
- $O(h \log n \log D)$ *für pol. Preprocessing.*

Zum Vergleich: Reach und CH

- $O((\Delta + h \log D)(h \log D))$ für unbeschränktes Preprocessing, und
- $O((\Delta + h \log n \log D)(h \log n \log D))$ für pol. Preprocessing.

SPC-Labels also echt besser als CH oder Reach?

CH-Labels

Problem:
SPC-Labels zwar theoretisch gut, aber inpraktikabel zu berechnen.

Beobachtung

Für jeden Knoten $v \in V$ ist der Suchraum einer uneingeschränkten CH-Aufwärtssuche ein gültiges Label.

Beweis:

- Betrachte Knoten $s, t \in V$ und kürzesten Weg $P(s, t)$.
- Knoten $u \in P(s, t)$ mit höchstem Rang ist in $L(s)$ und $L(t)$.

$\Rightarrow$ (Korrektheit von CH) u hat korrekte Distanzen $\mathrm{dist}(s, u)$, $\mathrm{dist}(u, t)$.

□

Also: Pratikabler Labeling Algorithmus auf Basis von CH-Labels?

CH-Labels

Problem:
SPC-Labels zwar theoretisch gut, aber inpraktikabel zu berechnen.

Beobachtung

Für jeden Knoten $v \in V$ ist der Suchraum einer uneingeschränkten CH-Aufwärtssuche ein gültiges Label.

Beweis:

- Betrachte Knoten $s, t \in V$ und kürzesten Weg $P(s, t)$.
- Knoten $u \in P(s, t)$ mit höchstem Rang ist in $L(s)$ und $L(t)$.

$\Rightarrow$ (Korrektheit von CH) u hat korrekte Distanzen $\mathrm{dist}(s, u)$, $\mathrm{dist}(u, t)$.

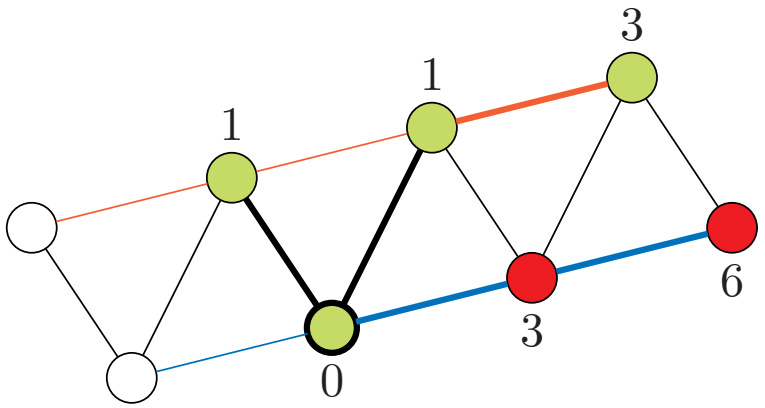
□

Also: Pratikabler Labeling Algorithmus auf Basis von CH-Labels?

CH-Labels

Problem: Durchschnittlicher CH-Suchraum: $\approx$ 500 Knoten

Beob.: Viele Knoten im CH-Suchraum haben inkorrekte Distanz.



Legende der Kantenfarben:

- Länge 1
- Länge 2
- Länge 3

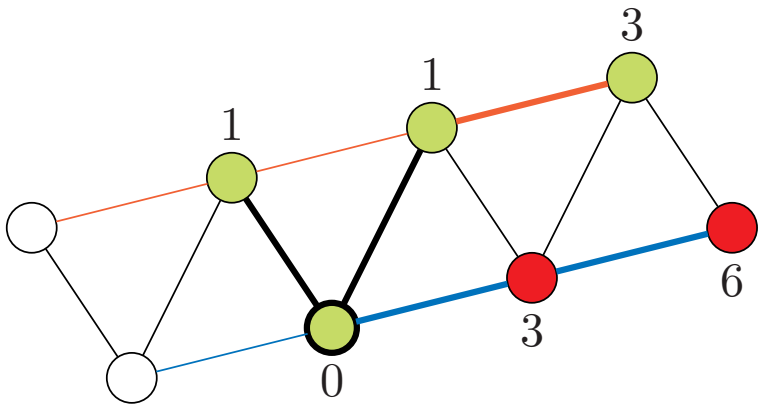
Pruning der Labels via Bootstrapping

- Berechne Label top-down bezüglich CH-Ordering.
- Für $v \in V$ und $w \in L(v)$ ist Label $L(w)$ bereits berechnet.
- Benutze v-w-Labeling-Query um w ggf. zu prunen.
- Reduziert Labelgröße auf $\approx$ 110.

CH-Labels

Problem: Durchschnittlicher CH-Suchraum: $\approx$ 500 Knoten

Beob.: Viele Knoten im CH-Suchraum haben inkorrekte Distanz.



Legende der Kantenfarben:

- Länge 1
- Länge 2
- Länge 3

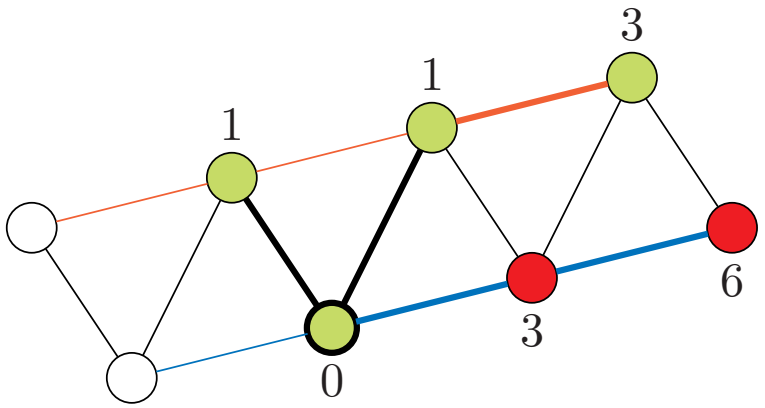
Pruning der Labels via Bootstrapping

- Berechne Label top-down bezüglich CH-Ordering.
- Für $v \in V$ und $w \in L(v)$ ist Label $L(w)$ bereits berechnet.
- Benutze v-w-Labeling-Query um w ggf. zu prunen.
- Reduziert Labelgröße auf $\approx$ 110.

CH-Labels

Problem: Durchschnittlicher CH-Suchraum: $\approx$ 500 Knoten

Beob.: Viele Knoten im CH-Suchraum haben inkorrekte Distanz.



Legende der Kantenfarben:

- Länge 1
- Länge 2
- Länge 3

Pruning der Labels via Bootstrapping

- Berechne Label top-down bezüglich CH-Ordering.
- Für $v \in V$ und $w \in L(v)$ ist Label $L(w)$ bereits berechnet.
- Benutze v-w-Labeling-Query um w ggf. zu prunen.
- Reduziert Labelgröße auf $\approx$ 110.

CH-Labels

Weitere Reduktion der Labelgrößen

- Benutze Shortest-Path-Cover für oberste Knoten in CH
  $\rightsquigarrow$ Labelgröße ca. 80
- Kompression der Label (verbessert Lokalität)

	Vorberechnung		Anfrage
	Zeit [h:m]	Platz GB	Zeit [ns]
CH	0:13	0.4	93 995
TNR	0:58	3.7	1 775
TNR+AF	2:00	5.7	992
CHL prefix	3:16	5.7	527
CHL global	2:45	21.3	276
Table Lookup	$>$ 11:03	1 208 358 7	56

CHL ist schnellster Algorithmus in Straßennetzwerken.

Zusammenfassung

Theoretische Garantien für Beschleunigungstechniken

- Highway-Dimension formalisiert Intuition hinter Transit-Node Routing
- Kleine Highway-Dimension führt zu kleinen Shortest-Path-Covern
- Vermutung: Straßennetzwerke haben kleine Highway-Dimension
- Generisches Preprocessing basierend auf CH
- Laufzeitgarantien für RE/CH in Abhängigkeit von h statt n

Labeling Algorithmus

- Theorie sagt effizienten Algorithmus vorraus
- CH-Labels schnellster Algorithmus auf Straßennetzen

Zusammenfassung

Theoretische Garantien für Beschleunigungstechniken

- Highway-Dimension formalisiert Intuition hinter Transit-Node Routing
- Kleine Highway-Dimension führt zu kleinen Shortest-Path-Covern
- Vermutung: Straßennetzwerke haben kleine Highway-Dimension
- Generisches Preprocessing basierend auf CH
- Laufzeitgarantien für RE/CH in Abhängigkeit von h statt n

Labeling Algorithmus

- Theorie sagt effizienten Algorithmus vorraus
- CH-Labels schnellster Algorithmus auf Straßennetzen

Praktikum Routenplanung

Karlsruher Institut für Technologie

Praktikum zur Routenplanung

- im Wintersemester 2011/2012 (vmtl. Bachelor+Master)
- Betreeung durch Julian und Thomas
- Praktische Implementierung einiger Aspekte aus der Vorlesung
  - Beschleunigungstechniken
  - Alternativrouten
  - Public Transport
  - PHAST
  - ...
- Möglichkeit aktuelle Forschung im Bereich Routenplanung praktisch auszuprobieren

Details werden auf unserer Seite bekanntgegeben unter

`http://i11www.ira.uka.de`

Offene Probleme

Wir sind immer interessiert an
Studien-/Diplom-/Bachelor-/Master-Arbeiten!

Themen (Auswahl)

- Public Transportation
- Multi-Modale Routenplanung
- Dynamische Szenarien
- Implementierung auf mobile Geräte (Smartphones)
- ...

Kommt einfach vorbei und meldet euch bei Julian Dibbelt oder Thomas Pajor!

Ende

Literatur:

- Ittai Abraham, Amos Fiat, Andrew Goldberg und Renato Werneck:
  Highway Dimension, Shortest Paths, and Provably Efficient Algorithms
  In: *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA10)*, 2010.
- Ittai Abraham, Daniel Delling, Andrew Goldberg und Renato Werneck:
  A Hub-Based Labeling Algorithm for Shortest Paths on Road Networks
  In: *Proc. International Symposium on Experimental Algorithms (SEA'11)*, 2011