Karlsruher Institut für Technologie

Algorithmen für Routenplanung

9. Sitzung, Sommersemester 2010

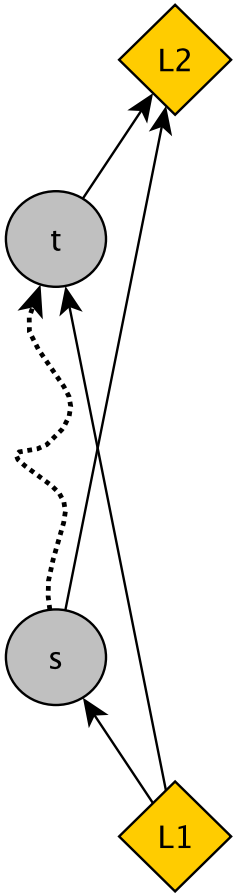
Thomas Pajor | 7. Juni 2010

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

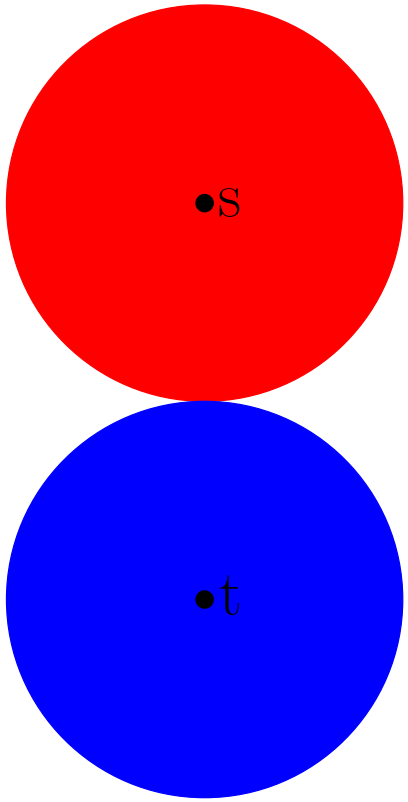
Letztes Mal: Kombinationen

Karlsruher Institut für Technologie

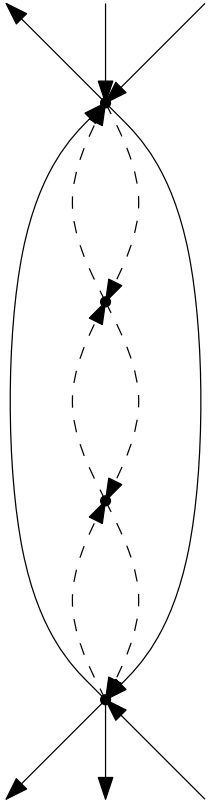
Landmarken



Bidirektionale Suche



Kontraktion



Arc-Flags

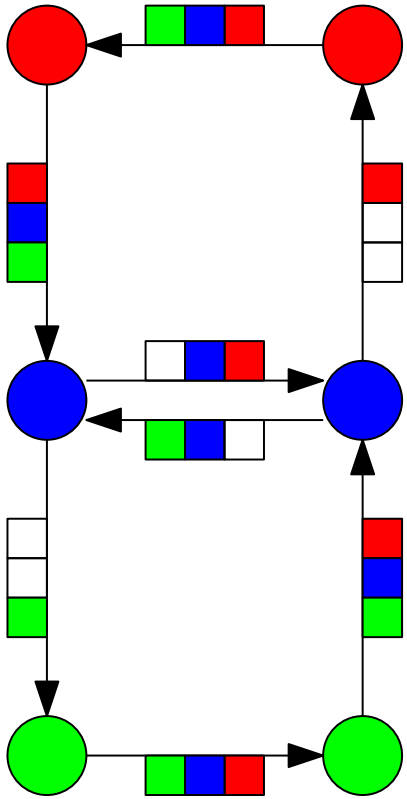
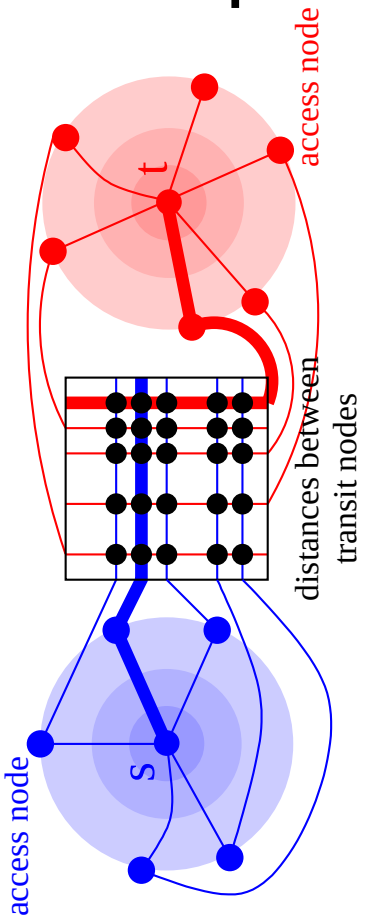


Table-Lookups



Ergebnisse auf Straßennetzwerken

Karlsruher Institut für Technologie

Paradigma

- Offline-Phase:
  Vorberechnung zusätzlicher Informationen
- Online-Phase:
  Beschleunigung der Anfragen mit Hilfe der Informationen
  - Meist Dijkstra-basierte Algorithmen

Ergebnisse

- Verschiedene sehr effiziente Verfahren
- Bis zu 3 Millionen mal schneller als Dijkstra
- Kaum zusätzlicher Speicherbedarf

Ergebnisse auf Straßennetzwerken

	Vorberechnung		Anfrage		
	Zeit [h:m]	Platz [byte/n]	Such-raum	Zeit [ms]	Beschl.
Dijkstra	0:00	0	9 114 385	5 591.6	1
Bi-Dijkstra	0:00	0	4 764 110	2 713.2	2
ALT-16	1:25	128	74 669	53.6	104
Arc-Flags-128	17:08	10	2 764	0.8	6 988
RE	1:22	13	4 643	3.5	1 597
REAL-(64,16)	2:20	35	679	1.1	5 037
CH	0:32	-3.0	359	0.15	37 273
TNR	1:15	247	N/A	0.0043	≈ 1 300 000
CALT-(med.,32)	0:10	10.6	1 704	1.6	3 368
gen. SHARC	1:21	14.5	654	0.29	19 281
gen. CHASE	1:39	12	45	0.0173	323 213
TNR+AF	3:49	321	N/A	0.0019	≈ 3 000 000

Input: Europa, ≈ 18 Mio Knoten und ≈ 42 Mio Kanten

Ergebnisse auf Straßennetzwerken

Karlsruher Institut für Technologie

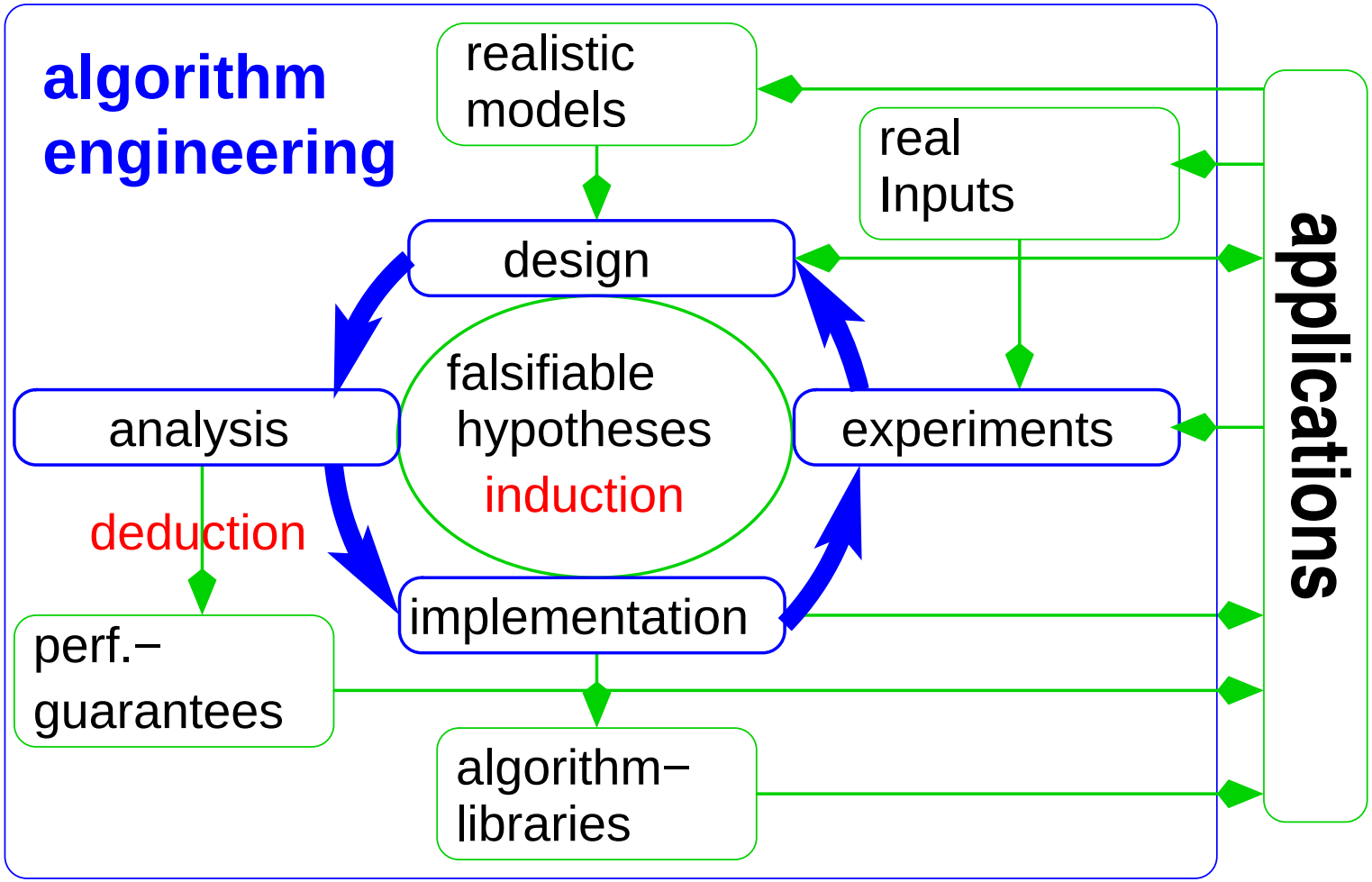
Problem: Exakt oder Heuristisch?

- Alle Verfahren liefern beweisbar kürzeste Wege
- **ABER:**
  - Basieren auf Intuition und Experimentellen Studien
  - Keine Garantien bezüglich Platzverbrauch und Laufzeit
  - Meist für Straßennetzwerke “maßgeschneidert”

Ausgenutzte Intuition (Auswahl):

- Kürzeste Wege sind oft eindeutig
- Lange in die falsche Richtung fahren lohnt meist nicht
- Inheränte Hierarchie in Straßennetzwerken
- Es gibt wenige Knoten die für “lange” Wege wichtig sind
- Lokale Suchen sind “billig”

Algorithm Engineering



Lücke: Praxis vs. Theorie

Lässt sich die Lücke zw. Praxis und Theorie schließen?

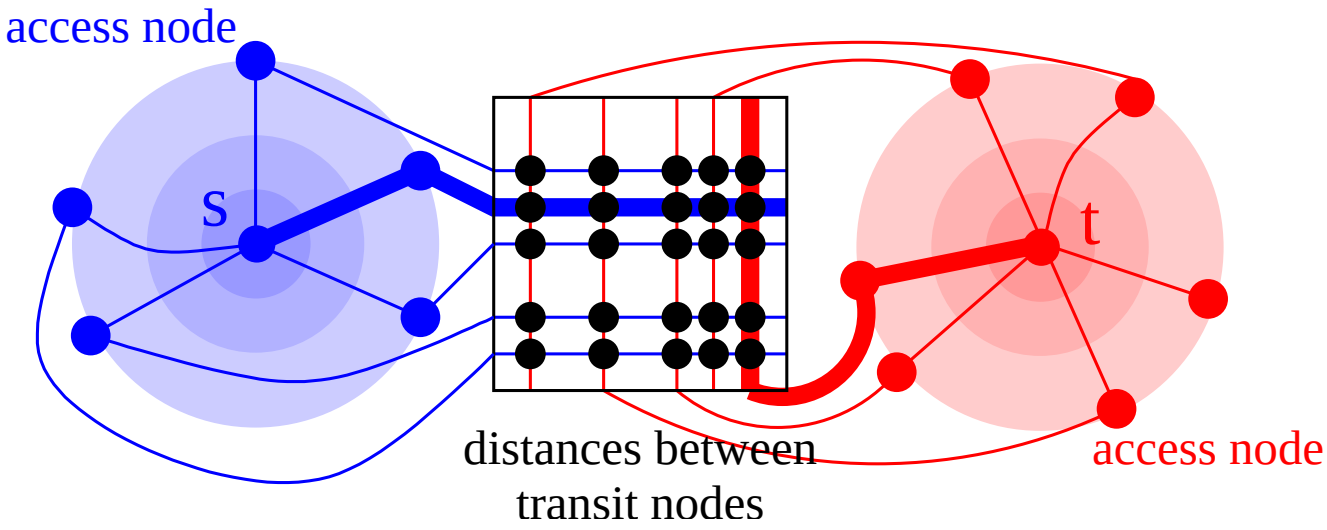
- **Warum** funktionieren die Beschleunigungstechniken so gut?
- **Was** zeichnet die Straßennetzwerke dazu aus?

Dieses Mal:
Ein erster Versuch einige Beschleunigungstechniken zu erklären.

Beob.: Transit-Node Routing

Idee:
Es gibt eine **kleine** Menge von **(Transit)-Knoten**, so dass

- jeder “lange” kürzeste Weg mindestens einen Transit-Knoten enthält
- für jeden Knoten *seine* wichtigen Transit-Knoten (Access-Nodes) nur sehr wenige sind



$\approx$ 10 000 Transit-Knoten u. $\approx$ 10 Access-Nodes pro Knoten (Europa)!

Formalisierung

Wie lässt sich diese Beobachtung formalisieren?

Werkzeuge

Voraussetzungen

- Graph $G = (V, E, \text{len})$
- ungerichtet (lässt sich auf gerichtet verallgemeinern)
- kürzeste Wege sind eindeutig
- Jede Kante $e = \{u, v\}$ ist kürzester Weg zw. u und v (sonst lösche e)

Kugel

Zu $r \in \mathbb{R}^+$ und $u \in V$ ist $\mathcal{B}_{u,r} := \{v \in V \ : \ |P(u, v)| \leq r\}$ die Kugel mit Radius r um u.

Durchmesser

Der Durchmesser D eines Graphen ist $D := \max_{u,v \in V} |P(u, v)|$

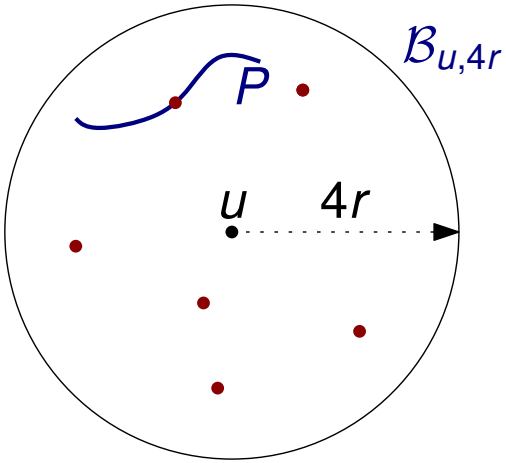
Maximaler Grad

Δ bezeichne den maximalen Grad eines Knotens in G.

Highway Dimension

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

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



Definition

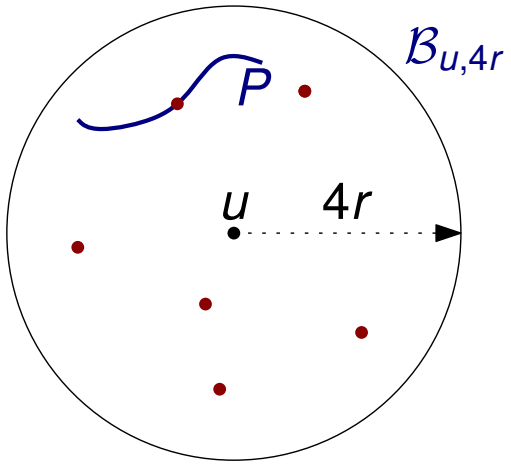
Die *Highway-Dimension* von G ist die kleinste Zahl $h \in \mathbb{N}$, so dass

- 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

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

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 jede Kugel $\mathcal{B}$ in G (mit Radius $4r$)
- eine Knoten-Menge S mit $|S| \leq h$ existiert, so dass
- jeder kürzeste Weg in $\mathcal{B}$ mit Mindestlänge r einen Knoten aus S enthält.

Highway vs. Doubling-Dimension

Definition

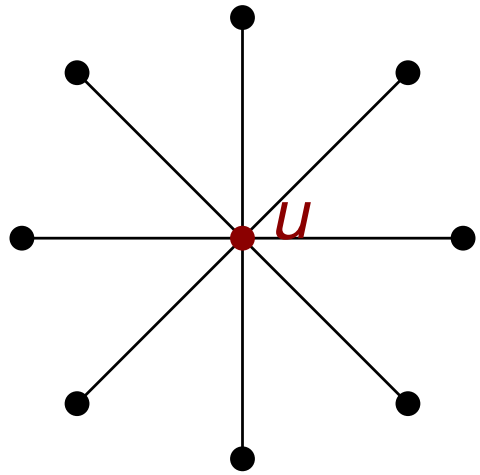
Ein Graph hat Doubling-Dimension α wenn jede Kugel $\mathcal{B}$ mit Radius r durch 2^{α} Kugeln mit Radius $r/2$ überdeckt werden kann.

Beispiel: Pfad (kleine HD und kleine DD)

Highway-Dimension 7 ($O(1)$) und Doubling-Dimension 1.

Highway vs. Doubling-Dimension

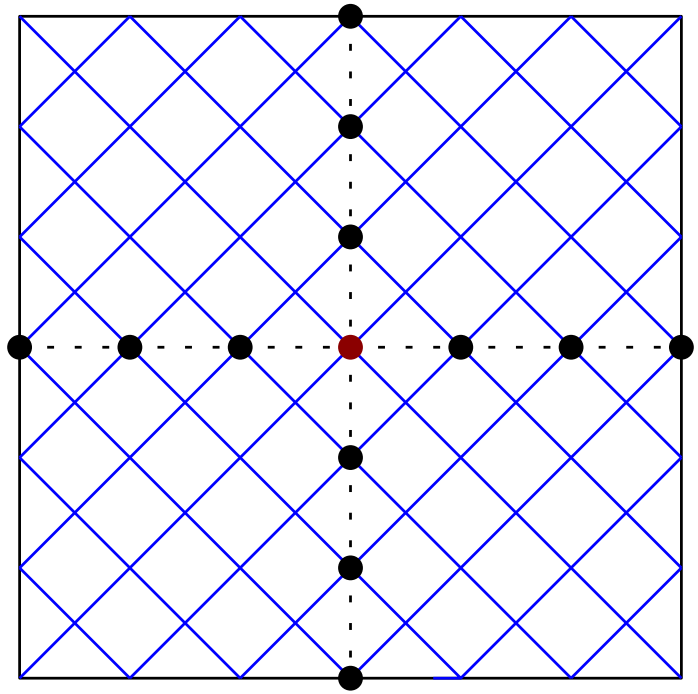
Beispiel: Stern (kleine HD und große DD)



Highway-Dimension 1 und Doubling-Dimension $\Theta(\log n)$.

Highway vs. Doubling-Dimension

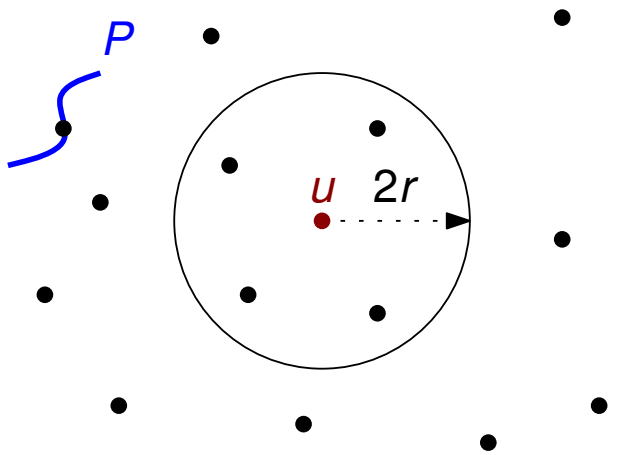
Beispiel: Gitter (große HD und kleine DD)



Highway-Dimension $\Theta(\sqrt{n})$ und Doubling-Dimension 2.

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

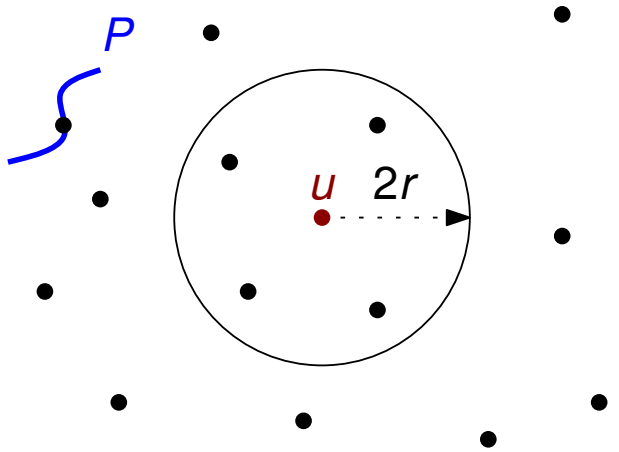
Eine Menge C heißt (r, k)-SPC von G genau dann wenn

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

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

- jede Kugel $\mathcal{B}$ mit Radius $2r$ höchstens k Knoten aus C enthält und
- jeder kürzeste Weg P der Länge $r \leq |P| \leq 2r$ einen Knoten aus C enthält.

Highway Dimension und SPCs

Theorem

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

Beweis

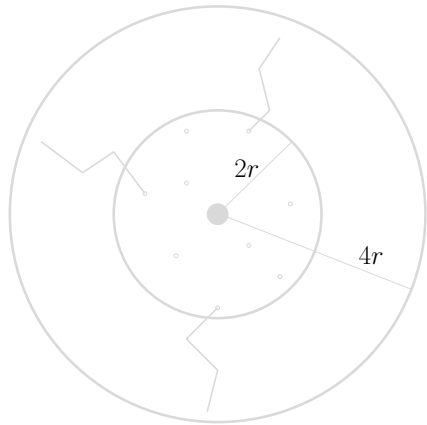
- Sei S^* eine *minimale* Menge die alle kürzesten Wege P mit $r < |P| \leq 2r$ überdeckt.
- zu Zeigen: S^* ist ein (r, h)-SPC

Ann.: $\exists u \in V$ so dass $U := S^* \cap \mathcal{B}_{u,2r}$ und $|U| > h$.

$\Rightarrow$ Nach Def. von h gibt es $H \subseteq V$ mit $|H| \leq h$ die alle kürzesten Wege überdeckt die in $\mathcal{B}_{u,4r}$ enthalten sind und länger als r sind.

$\Rightarrow$ H überdeckt alle KW P mit $r < |P| \leq 2r$

$\Rightarrow$ $|(S^* \setminus U) \cup H| < |S^*|$ und ist (r, h)-SPC. $\lightning$



Highway Dimension und SPCs

Theorem

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

Beweis

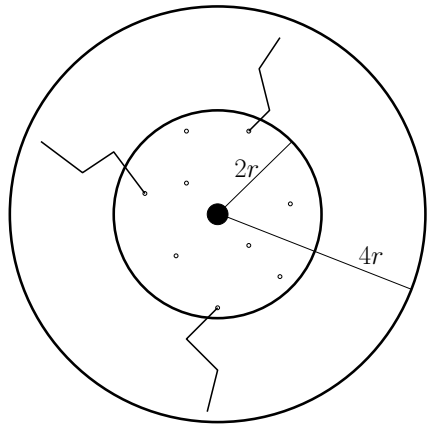
- Sei S^* eine *minimale* Menge die alle kürzesten Wege P mit $r < |P| \leq 2r$ überdeckt.
- zu Zeigen: S^* ist ein (r, h)-SPC

Ann.: $\exists u \in V$ so dass $U := S^* \cap \mathcal{B}_{u,2r}$ und $|U| > h$.

$\Rightarrow$ Nach Def. von h gibt es $H \subseteq V$ mit $|H| \leq h$ die alle kürzesten Wege überdeckt die in $\mathcal{B}_{u,4r}$ enthalten sind und länger als r sind.

$\Rightarrow$ H überdeckt alle KW P mit $r < |P| \leq 2r$

$\Rightarrow$ $|(S^* \setminus U) \cup H| < |S^*|$ und ist (r, h)-SPC. $\lightning$



Highway Dimension und SPCs

Theorem

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

Beweis

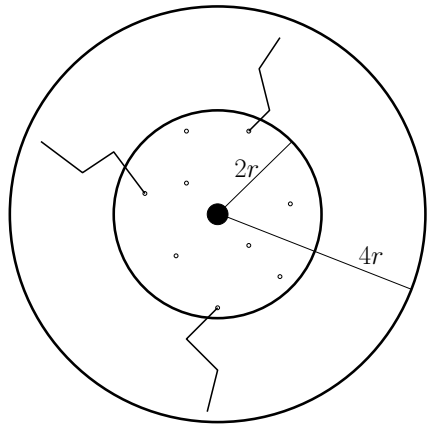
- Sei S^* eine *minimale* Menge die alle kürzesten Wege P mit $r < |P| \leq 2r$ überdeckt.
- zu Zeigen: S^* ist ein (r, h)-SPC

Ann.: $\exists u \in V$ so dass $U := S^* \cap \mathcal{B}_{u,2r}$ und $|U| > h$.

$\Rightarrow$ Nach Def. von h gibt es $H \subseteq V$ mit $|H| \leq h$ die alle kürzesten Wege überdeckt die in $\mathcal{B}_{u,4r}$ enthalten sind und länger als r sind.

$\Rightarrow$ H überdeckt alle KW P mit $r < |P| \leq 2r$

$\Rightarrow$ $|(S^* \setminus U) \cup H| < |S^*|$ und ist (r, h)-SPC. $\lightning$



Highway Dimension und SPCs

Theorem

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

Beweis

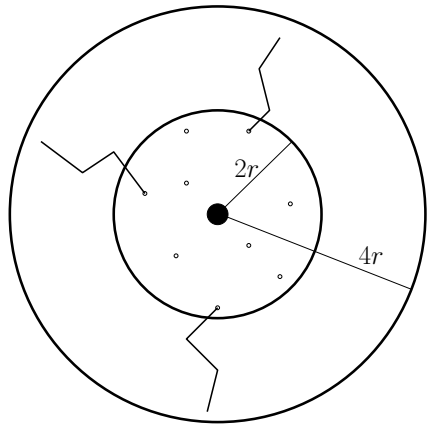
- Sei S^* eine *minimale* Menge die alle kürzesten Wege P mit $r < |P| \leq 2r$ überdeckt.
- zu Zeigen: S^* ist ein (r, h)-SPC

Ann.: $\exists u \in V$ so dass $U := S^* \cap \mathcal{B}_{u,2r}$ und $|U| > h$.

$\Rightarrow$ Nach Def. von h gibt es $H \subseteq V$ mit $|H| \leq h$ die alle kürzesten Wege überdeckt die in $\mathcal{B}_{u,4r}$ enthalten sind und länger als r sind.

$\Rightarrow$ H überdeckt alle KW P mit $r < |P| \leq 2r$

$\Rightarrow$ $|(S^* \setminus U) \cup H| < |S^*|$ und ist (r, h)-SPC. $\lightning$



Highway Dimension und SPCs

Theorem

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

Beweis

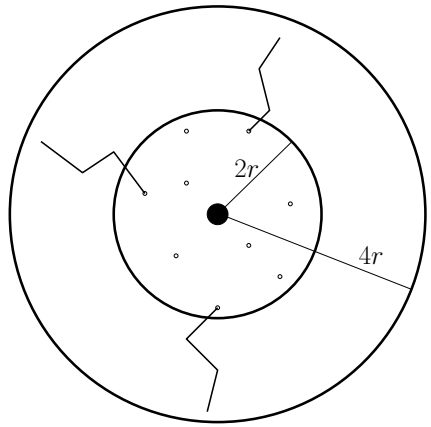
- Sei S^* eine *minimale* Menge die alle kürzesten Wege P mit $r < |P| \leq 2r$ überdeckt.
- zu Zeigen: S^* ist ein (r, h)-SPC

Ann.: $\exists u \in V$ so dass $U := S^* \cap \mathcal{B}_{u,2r}$ und $|U| > h$.

$\Rightarrow$ Nach Def. von h gibt es $H \subseteq V$ mit $|H| \leq h$ die alle kürzesten Wege überdeckt die in $\mathcal{B}_{u,4r}$ enthalten sind und länger als r sind.

$\Rightarrow$ H überdeckt alle KW P mit $r < |P| \leq 2r$

$\Rightarrow$ $|(S^* \setminus U) \cup H| < |S^*|$ und ist (r, h)-SPC. $\lightning$



Highway Dimension und SPCs

Problem:
Wie lässt sich S^* 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 S^*.

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

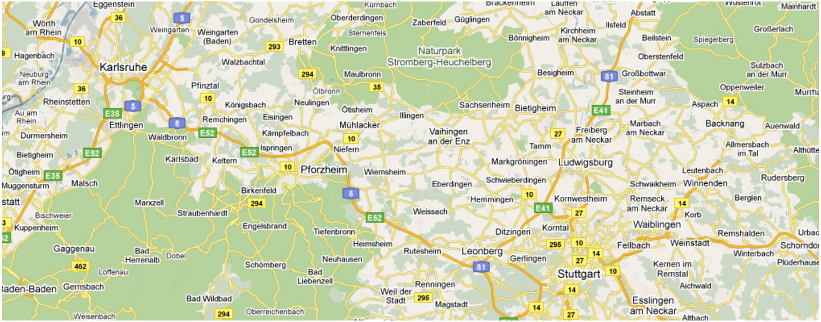
$\Rightarrow$ Höchstens $O(h \log n)$ Knoten werden ausgewählt

HD von Straßennetzwerken

Karlsruher Institut für Technologie

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)

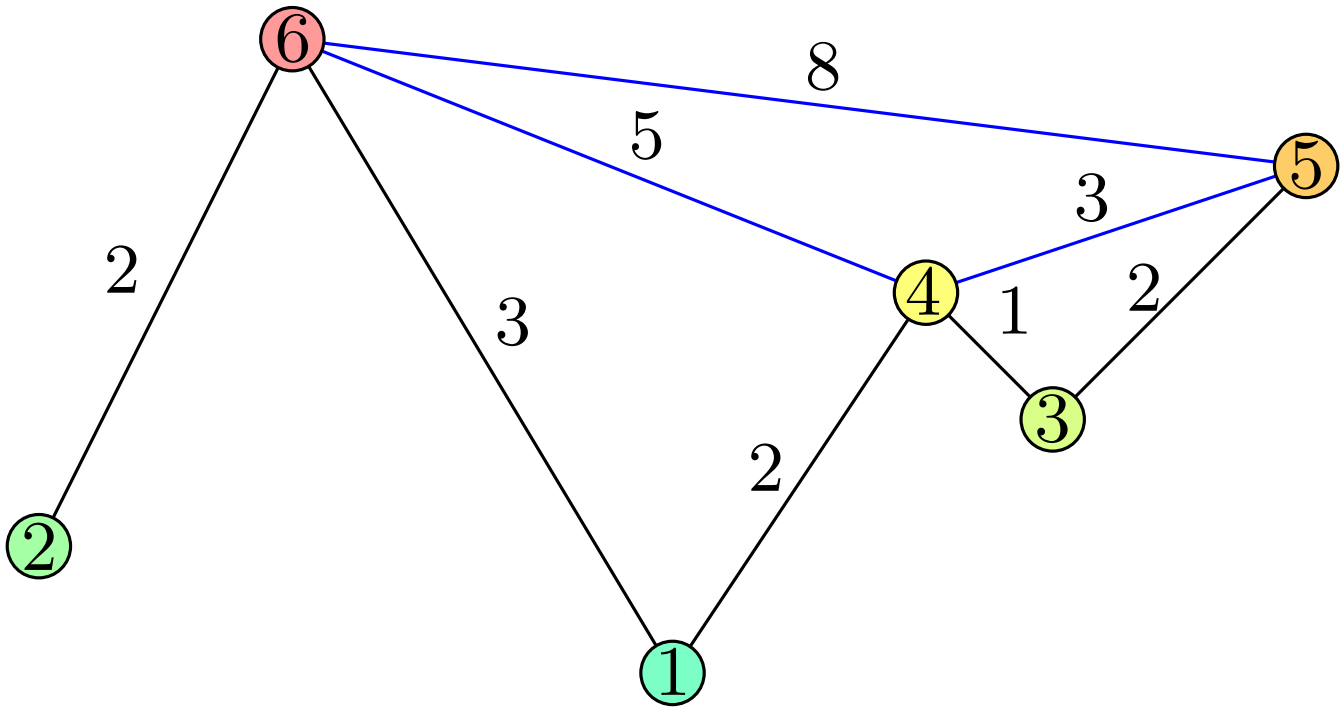
```
1 für alle Paare (u, v) und (v, w) von Kanten tue
2     wenn (u, v, w) eindeutiger kürzester Weg dann
3         Füge Shortcut (u, w) mit Gewicht len(u, v) + len(v, w) ein
```

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 $S_0 := V$
- Sei S_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

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

- 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 $S_0 := V$
- Sei S_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

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

- 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 $S_0 := V$
- Sei S_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

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

- 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. $(2 \cdot 2^j, k)$-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. $(2 \cdot 2^j, k)$-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

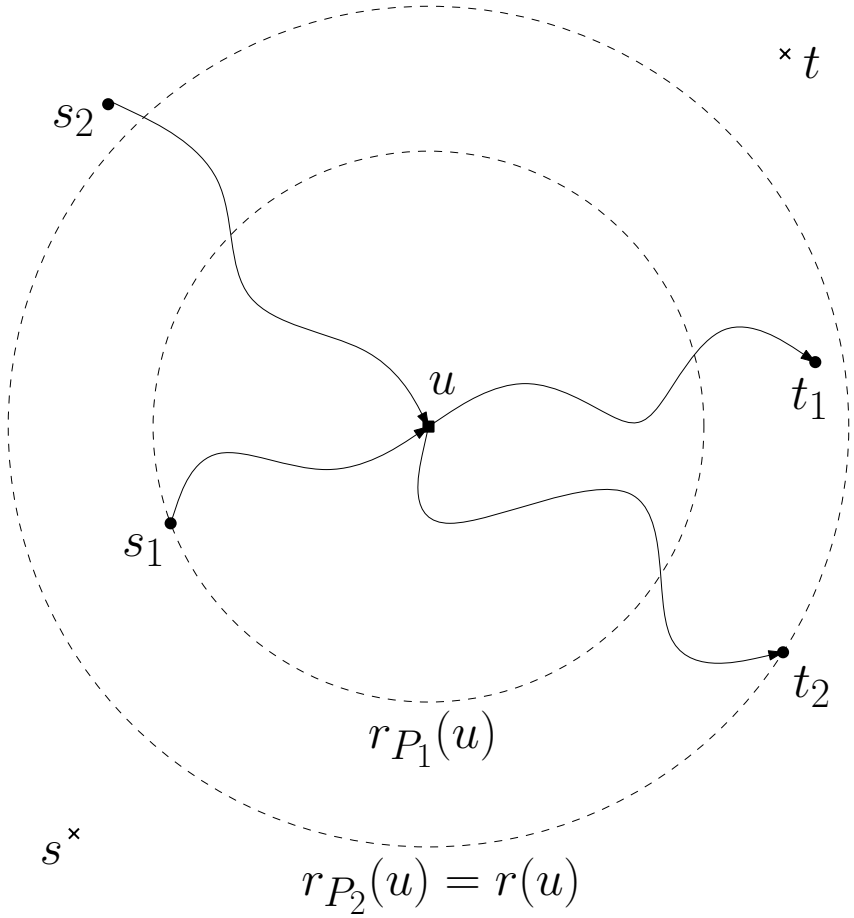
Erinnerung:
Die meisten Query-Algorithmen sind Dijkstra-basiert.

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

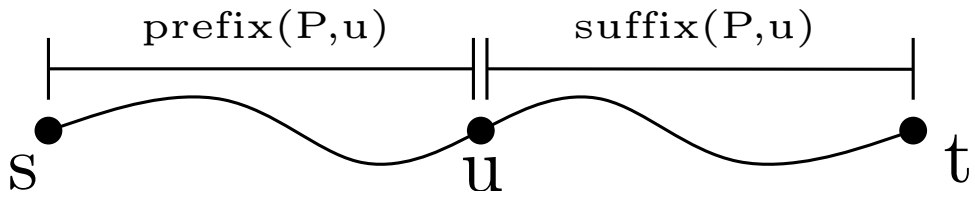
Also: Suchraumgröße $\approx$ Queryzeit

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. $\lightning$

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. $\lightning$

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:

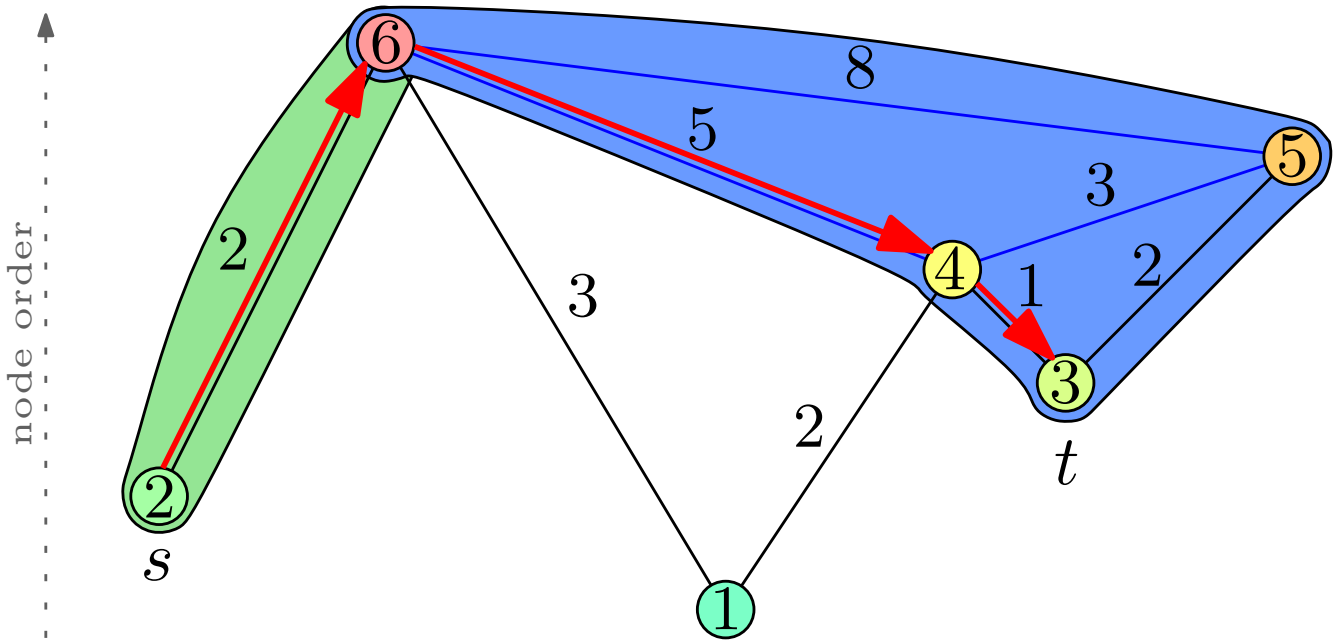
$$\underbrace{(\Delta + k \log D)}_{\text{max. Knotengrad in } G^+} \cdot \underbrace{(k \log D)}_{\text{max. Suchraum}}$$

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 $u, w \in L_i \cap \mathcal{B}_{v,2^{i+1}}$ 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 $u, w \in L_i \cap \mathcal{B}_{v,2^{i+1}}$ 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*

TNR und SHARC

Theorem

Die Query-Zeit von TNR ist in

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

Theorem

SHARC (mit Modifikationen) hat polynomialzeit Preprocessing mit Platzverbrauch in

$$O(nh \log n \log D)$$

so dass für die Query-Zeit gilt

$$O((h \log n \log D)^2).$$

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
- Führt zu Laufzeitgarantien für RE, CH, TNR und SHARC in Abhängigkeit von h statt n

Offene Probleme

- Wie Highway-Dimension effizient berechnen?
- Analyse für zielgerichtete Techniken (Arc-Flags, ALT)

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
- Führt zu Laufzeitgarantien für RE, CH, TNR und SHARC in Abhängigkeit von h statt n

Offene Probleme

- Wie Highway-Dimension effizient berechnen?
- Analyse für zielgerichtete Techniken (Arc-Flags, ALT)

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.

Username: `routePlanning` Passwort: `ss10`