

# Algorithmen für Routenplanung

19. und letzte Sitzung, Sommersemester 2012

Thomas Pajor | 16. Juli 2012

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



## Modellierung

- Straßennetzwerk als Graph  $G$
- Kantengewichte = Reisezeit
- Beste Route  $\hat{=}$  Kürzester Weg in  $G$

## Modellierung

- Straßennetzwerk als Graph  $G$
- Kantengewichte = Reisezeit
- Beste Route  $\hat{=}$  Kürzester Weg in  $G$

## Lösung

- Klassisches Problem: Dijkstra's Algorithmus
- Laufzeit in  $O(m + n \log n)$
- **Aber:** Mehrere Sekunden auf Europa (Server-Hardware)

## Modellierung

- Straßennetzwerk als Graph  $G$
- Kantengewichte = Reisezeit
- Beste Route  $\hat{=}$  Kürzester Weg in  $G$

## Lösung

- Klassisches Problem: Dijkstra's Algorithmus
- Laufzeit in  $O(m + n \log n)$
- **Aber:** Mehrere Sekunden auf Europa (Server-Hardware)

⇒ Beschleunigungstechniken!

## Paradigma

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

## Ergebnisse

- Verschiedene sehr effiziente Verfahren
- Mehrere Millionen mal schneller als Dijkstra
- Kaum zusätzlicher Speicherbedarf

## Problem: Exakt oder Heuristisch?

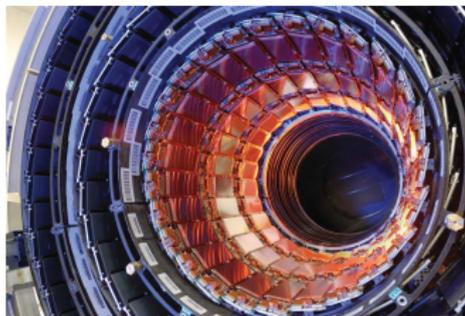
- Alle Verfahren liefern beweisbar kürzeste Wege
- **ABER:**
  - Basieren auf Intuition und experimentellen Studien
  - **Keine Garantien** bezüglich Platzverbrauch und Laufzeit
  - Meist für Straßennetzwerke “maßgeschneidert”

## Ausgenutzte Intuition (Auswahl):

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

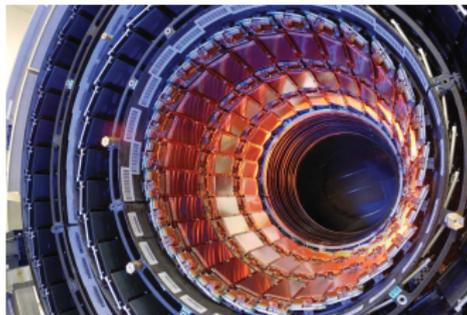
## Die Wissenschaftliche Methode

- **Beobachtung** der Wirklichkeit
- Aufstellen einer **Hypothese/Theorie**
- **Vorhersagen** treffen
- Bestätigung durch **Experiment**



## Die Wissenschaftliche Methode

- **Beobachtung** der Wirklichkeit
- Aufstellen einer **Hypothese/Theorie**
- **Vorhersagen** treffen
- Bestätigung durch **Experiment**



## Versch. Vorgehen

- Mathematics of Algorithms  $\Rightarrow$  Theoretische Aussagen
- Algorithm Engineering  $\Rightarrow$  Praktikable Algorithmen
- **Science of Algorithms**: Mathematik  $\Leftrightarrow$  Engineering  
     $\rightsquigarrow$  Benutzung der wissenschaftlichen Methode

## Lässt sich die Lücke zw. Praxis und Theorie schließen?

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

### Dieses Mal:

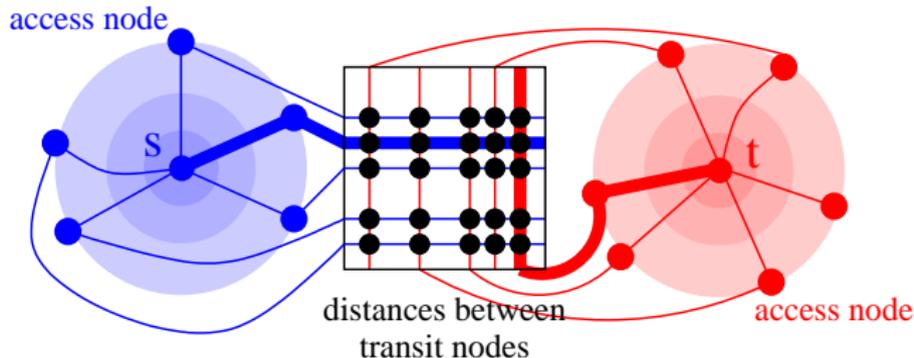
Ein erster Versuch einige Beschleunigungstechniken zu erklären.

# 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



≈ 10 000 Transit-Knoten u. ≈ 10 Access-Nodes pro Knoten (Europa)!

Wie lässt sich diese Beobachtung  
formalisieren?

## 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

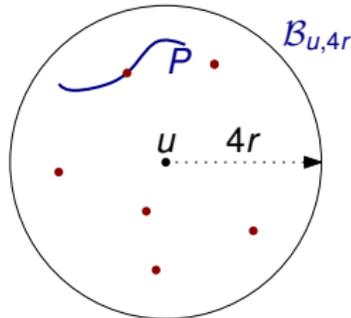
$\Delta$  bezeichne den maximalen Grad eines Knotens in  $G$ .

## 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  $S \subseteq B_{u, 4r}$  mit  $|S| \leq h$  so, dass

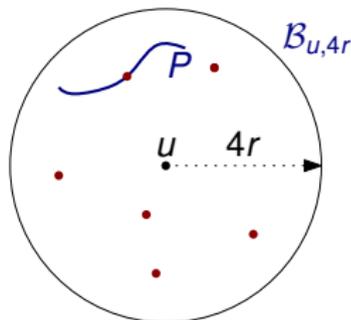
$$|P(v, w)| > r \text{ und } P(v, w) \subseteq B_{u, 4r} \Rightarrow P(v, w) \cap S \neq \emptyset$$

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

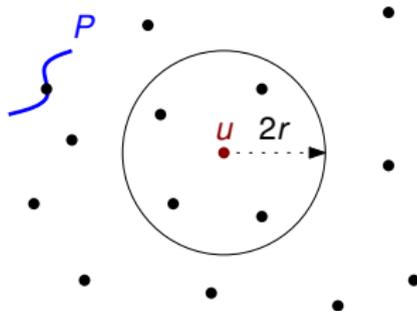
# Shortest-Path Cover (SPC)

## 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  $u \in V$ :  $|C \cap \mathcal{B}_{u,2r}| \leq k$

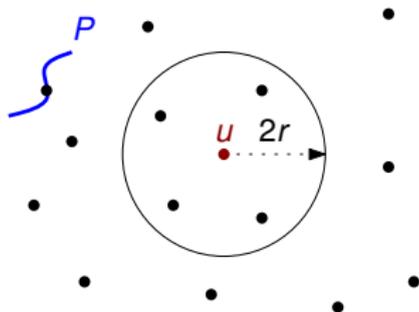
# Shortest-Path Cover (SPC)

## 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  $B$  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.

## Theorem

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

### Beweis

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

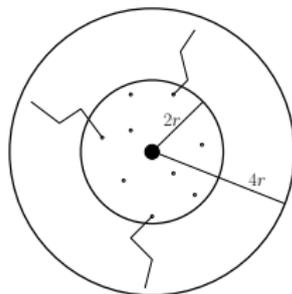
## Theorem

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

### Beweis

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

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



## Theorem

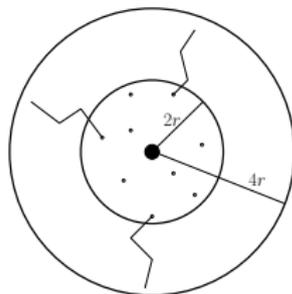
Wenn  $G$  Highway-Dimension  $h$  hat, dann  $\forall r \exists$  ein  $(r, h)$ -SPC.

## Beweis

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

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



## Theorem

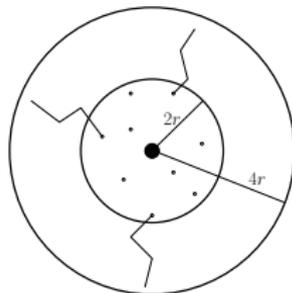
Wenn  $G$  Highway-Dimension  $h$  hat, dann  $\forall r \exists$  ein  $(r, h)$ -SPC.

## Beweis

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

**Ann.:**  $\exists u \in V$  so dass  $U := C^* \cap \mathcal{B}_{u,2r}$  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  $r < |P| \leq 2r$



## Theorem

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

## Beweis

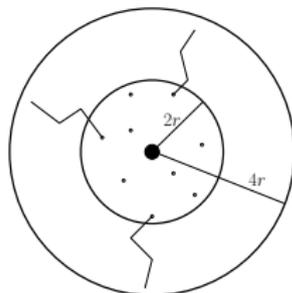
- Sei  $C^* \subseteq V$  eine *minimale* Menge die alle kürzesten Wege  $P$  mit  $r < |P| \leq 2r$  überdeckt.
- zu Zeigen:  $C^*$  ist ein  $(r, h)$ -SPC

**Ann.:**  $\exists u \in V$  so dass  $U := C^* \cap \mathcal{B}_{u,2r}$  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  $r < |P| \leq 2r$

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



## Problem:

Wie lässt sich  $C^*$  effizient konstruieren?

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

Also: Approximation.

## Theorem

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

### Idee:

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

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

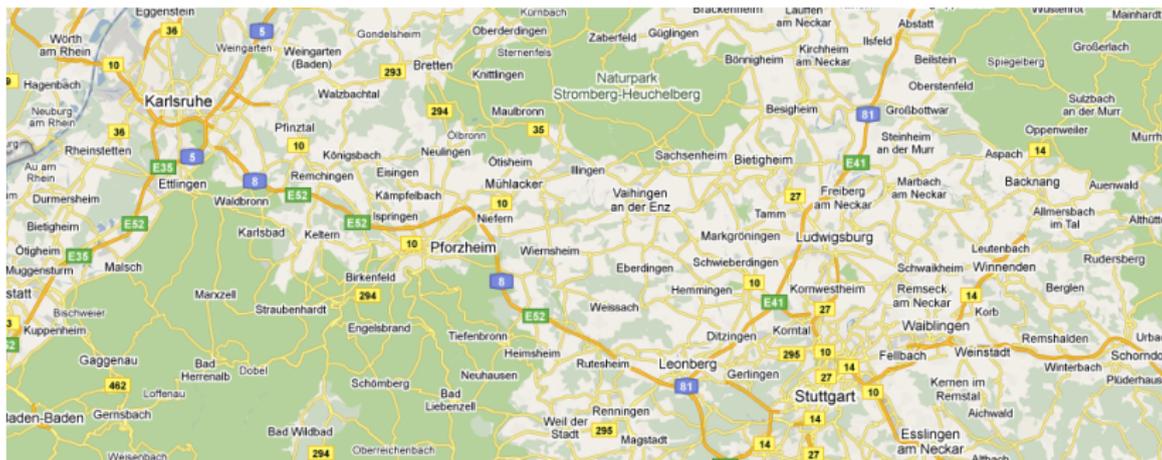
- Beginne mit  $C = \emptyset$
- Wähle iterativ den Knoten zu  $C$  der die meisten nicht-überdeckten KW überdeckt

## Beweisskizze:

- In jedem Schritt:  
Mind.  $1/h$  der unüberdeckten KW  $P$  mit  $r < |P| \leq 2r$  wird überdeckt
  - Es gibt  $O(n^2)$  KW in  $G$
- ⇒ Höchstens  $O(h \log n)$  Knoten werden ausgewählt
- Details siehe [AFGW10]

Vermutung:

Straßennetze haben kleine (konstante) Highway-Dimension.



## 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)

## Preprocessing:

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

---

KONTRAHIERE( $v$ )

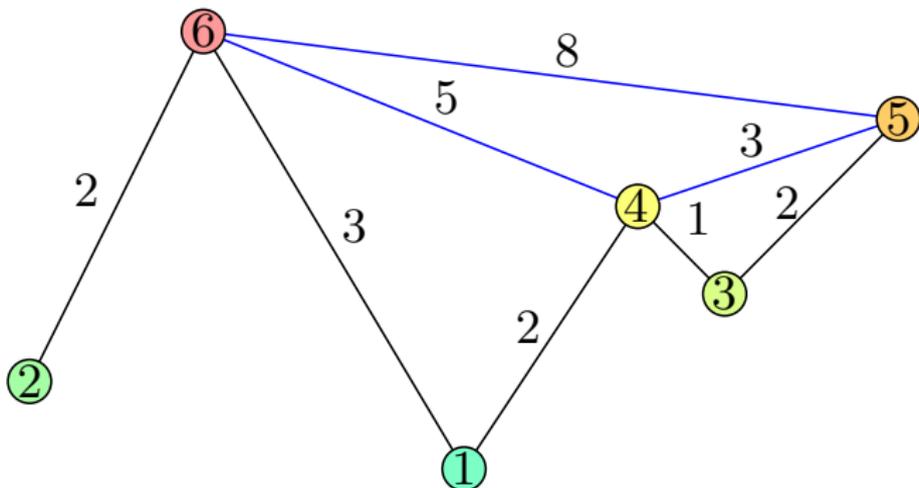
---

- 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  $\text{len}(u, v) + \text{len}(v, w)$  ein
- 

## Query:

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

**Freiheitsgrad:** In welcher Reihenfolge Knoten kontrahieren?



# Generischer Prepro.-Algorithmus

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

# Generischer Prepro.-Algorithmus

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

## Vorgehen

- Sei  $C_0 := V$
- 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 (Approx.)
- Dann ist

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

- Kontrahiere Knoten in der Reihenfolge induziert durch  $L_i$ .

# Generischer Prepro.-Algorithmus

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

## Vorgehen

- Sei  $C_0 := V$
- 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 (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  $L_0, L_1, \dots, L_{\log D}$

## Lemma

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

## 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.
- ⇒ Innere Knoten gehören zu  $L_x$  für  $x \leq i \leq j$ .
- ⇒  $w \in B_{v,2 \cdot 2^j}$  (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.

Damit folgt:

## 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



## Erinnerung:

Die meisten Query-Algorithmen sind Dijkstra-basiert.

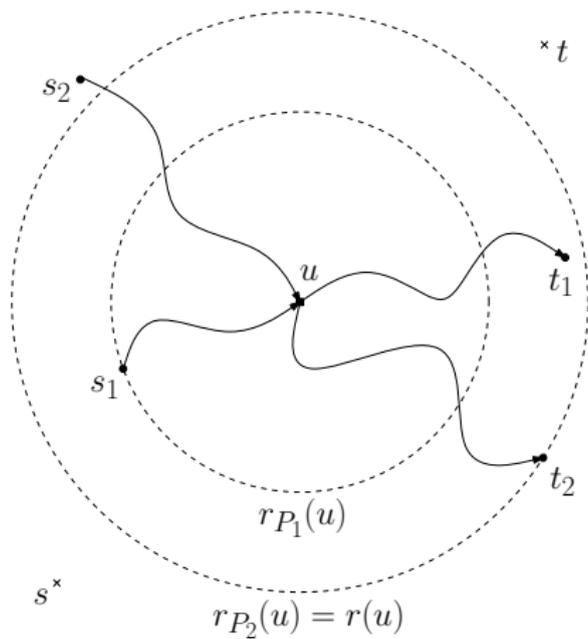
⇒ Aufwand wird dominiert von Queue-Operationen auf Knoten

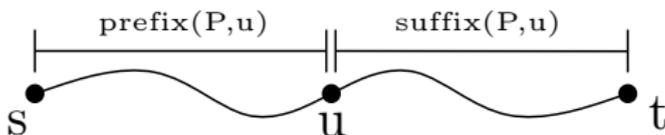
Also: Suchraumgröße  $\approx$  Queryzeit

## Vereinfachung:

Im Folgenden werden Suchraumgröße und Queryzeit gleichgesetzt.

# 1. Reach





## Definition:

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

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

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

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

## Somit:

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

## Somit:

- Reach  $r(u)$  von  $u$  gibt Suffix oder Prefix des längsten kürzesten Weges durch  $u$  an.
- 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:  $\min\{\minKey(\vec{Q}), \minKey(\overleftarrow{Q})\}$
- Bei gleich langen Wegen: Bevorzuge immer den Hopminimalen!

## Ursprüngliche Berechnung von Reach:

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

## Ursprüngliche Berechnung von Reach:

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

Modifizierte Vorberechnung durch folgendes Lemma

## Lemma

Für alle  $v \in L_j$  gilt:

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

## Lemma

Für alle  $v \in L_i$  gilt:

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

## Lemma

Für alle  $v \in L_j$  gilt:

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

### Beweis:

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

⇒  $\exists$  kürzester Weg  $P(x, v, y)$  mit  $v \in P(x, v, y)$  sowie  
 $|P(x, v)| > 2 \cdot 2^i$  und  $|P(v, y)| > 2 \cdot 2^i$ .

⇒ Sowohl  $P(x, v)$  als auch  $P(v, y)$  enthalten Knoten aus  $L_j$  mit  $j > i$ .

⇒ Es gibt einen Shortcut zwischen  $P(x, v)$  und  $P(v, y)$ .

⇒  $P(x, v, y)$  ist kein kürzester Weg (# Hops). ⚡

# 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*

# 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:

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

# 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

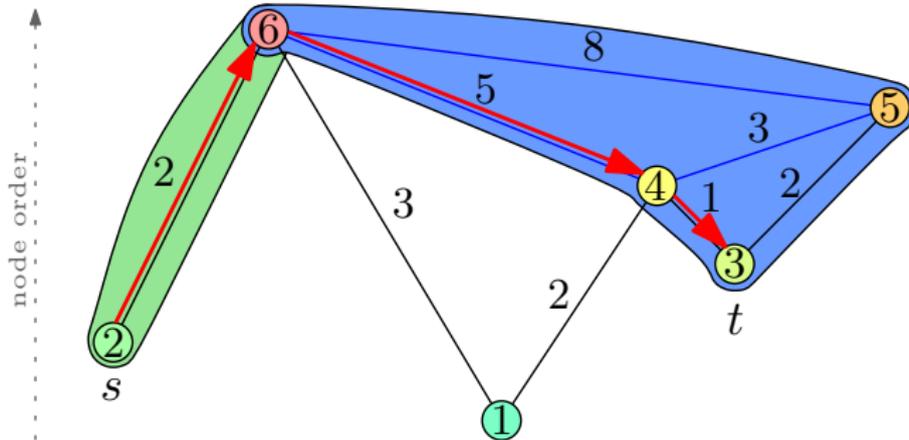
**Beweis:** Zusammensetzung der Schranke:

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

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  $|L_i \cap \mathcal{B}_{s, 2 \cdot 2^i}| \leq k$
- ⇒ Höchstens  $k$  Knoten werden pro Level abgearbeitet

## 2. Contraction Hierarchies



Betrachte vereinfachte Query:

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

Variante funktioniert in der Praxis bereits sehr gut

## **Problem:**

Reach-Schranken gelten nicht für CH-Query.

## **Lösungsansätze:**

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

## 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  $v \in P(u, w)$ .
- Benutze als Rank von Knoten  $v$  lediglich sein Level  $L_i$  (also  $i$ ).

## 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

## 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  
↪ 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

## Generisches Preprocessing:

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

## Labeling-Eigenschaft:

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

## Generisches Preprocessing:

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

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

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

**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  $v \in V$ :

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

**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  $v \in V$ :

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

## Label-Größe

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

## 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  $2^i < \text{dist}(s, t) \leq 2 \cdot 2^i$ .
  - Dann gilt:  $P(s, t) \subseteq \mathcal{B}_{2 \cdot 2^i}(s)$  und  $P(s, t) \subseteq \mathcal{B}_{2 \cdot 2^i}(t)$ .
- ⇒ Nach Def. SPC  $\exists u \in P(s, t) \cap C_i$  und somit  $u \in L(s) \cap L(t)$ .



## 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.

## 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.

## 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  $n$

## 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 voraus
- ↪ Hub-Labels schnellster Algorithmus auf Straßennetzen

# Ende



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

## Themen (Auswahl)

- Public Transportation
- Multi-Modale Routenplanung
- Elektromobilität
- Mitfahrgelegenheitsvermittlungssystementwicklung
- Implementierung auf mobile Geräte (Smartphones)
- Theoretische Fragestellungen
- ...

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

## Praktikum zur Routenplanung

- im Wintersemester 2012/2013 (Master)
- Betreuung: Moritz Baum, Julian Dibbelt, Thomas Pajor
- 6 ECTS-Punkte
- Praktische Implementierung einiger Aspekte aus der Vorlesung
  - Beschleunigungstechniken
  - Alternativrouten
  - Public Transport
  - Hub-Labels (in Datenbanken)
  - ...
- Möglichkeit aktuelle Forschung im Bereich Routenplanung praktisch auszuprobieren

Details werden auf unserer Seite bekanntgegeben unter  
<http://illwww.iti.kit.edu>

## Algorithmen zur Visualisierung von Graphen

- Dozenten: Tamara Mchedlidze, Martin Nöllenburg, Ignaz Rutter
- 5 ECTS-Punkte
- Termin: Dienstags 9:45 Uhr und mittwochs 14 Uhr
- Themen (Auswahl):
  - Kräftebasierte Layouts
  - Knickminimierung, Winkelauflösung
  - Lagenlayouts, Baumlayouts, Gitterzeichnungen
  - Zeichnen von Metromaps
  - ...

Mehr Informationen werden auf unserer Seite bekanntgegeben unter  
<http://illwww.itl.kit.edu>

## Geometrische Algorithmen für Anwendungen in der Geovisualisierung

- Andreas Gemsa, Martin Nöllenburg, Ignaz Rutter
- Termin: Dienstags 15:45 Uhr
- 4 ECTS-Punkte
- Themen (Auswahl):
  - Schematische Karten
  - Kartogramme
  - Beschriftung von Karten
  - Dynamische Darstellung von Karten
  - Generalisierung
  - ...

Mehr Informationen werden auf unserer Seite bekanntgegeben unter  
<http://illwww.itl.kit.edu>

## Synergien aus Graph-Theorie und Data-Mining zur Analyse von Netzwerkdaten

- Proseminar und Seminar (Algorithmentechnik B)
- Zusammen mit dem IPD
- Methoden zur Analyse komplexer Netzwerke  
z.B. soziale Netzwerke, Zitationsdatenbanken, Sensornetzwerke
- Schwerpunkte:
  - Maße zur Bewertung von Knoten und Subgraphen
  - Graph Clustering zur Gruppierung von ähnlichen Graph-Strukturen
  - Graph Outlier Mining zur Erkennung von Graph-Anomalien

Erster Termin: 16. Oktober 2012, 14 Uhr (SR 348, Geb. 50.34)



Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck.  
Highway Dimension, Shortest Paths, and Provably Efficient Algorithms.  
In Moses Charikar, editor, *Proceedings of the 21st Annual ACM–SIAM  
Symposium on Discrete Algorithms (SODA'10)*, pages 782–793. SIAM, 2010.