Karlsruher Institut für Technologie

Algorithmen für Routenplanung

6. Sitzung, Sommersemester 2013

Julian Dibbelt | 8. Mai 2013

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

Hierarchische Techniken

Karlsruher Institut für Technologie

Bisher: bidirektionale Suche, zielgerichtete Suche (ALT, Arc-Flags)

Heute: Nutze “Hierarchie” im Graphen

in etwa: Seiten-, Haupt-, Land-, Bundesstraßen, Autobahnen

Ideensammlung:

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Hierarchische Techniken

Karlsruher Institut für Technologie

Bisher: bidirektionale Suche, zielgerichtete Suche (ALT, Arc-Flags)

Heute: Nutze “Hierarchie” im Graphen

in etwa: Seiten-, Haupt-, Land-, Bundesstraßen, Autobahnen

Ideensammlung:

- identifiziere wichtige Knoten mit Zentralitätsmaß
- überspringe unwichtige Teile des Graphen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Motivation 1: Netzwerkanalyse

Karlsruher Institut für Technologie

- Zentralitätsmaße bewerten Wichtigkeit von Knoten oder Kanten in einem Netzwerk
- Geläufige Beispiele
  - Google Page Rank
  - Erdös-Zahl

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zentralitätsmaße - Beispiele

Karlsruher Institut für Technologie

Zentralitätsmaße - Beispiele

Karlsruher Institut für Technologie

- degree centrality

$$C_D(v) = \frac{\deg(v)}{n-1} \in [0, 1]$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zentralitätsmaße - Beispiele

- degree centrality

$$C_D(v) = \frac{\deg(v)}{n-1} \in [0, 1]$$

Geeignetes Maß?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zentralitätsmaße - Beispiele

Karlsruher Institut für Technologie

- degree centrality

$$C_D(v) = \frac{\deg(v)}{n-1} \in [0,1]$$

- betweenness centrality

$$C_B(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}} \in [0,1]$$

wobei

- σ_{st} die Anzahl der kürzesten s-t-Wege ist (meistens 1)
- $\sigma_{st}(v)$ die Anzahl solcher Wege, die durch v gehen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zentralitätsmaße - Beispiele

KIT — Karlsruher Institut für Technologie

- degree centrality

$$C_D(v) = \frac{\deg(v)}{n-1} \in [0,1]$$

- betweenness centrality

$$C_B(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}} \in [0,1]$$

wobei

- σ_{st} die Anzahl der kürzesten s-t-Wege ist (meistens 1)
- $\sigma_{st}(v)$ die Anzahl solcher Wege, die durch v gehen

Geeignetes Maß?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Motivation 1: Netzwerkanalyse

Karlsruher Institut für Technologie

Reach:

- Zentralitätsmaß, das groß ist, falls eine Kante in der Mitte eines langen kürzesten Weges liegt.

Motivation 2

Karlsruher Institut für Technologie

- Zeichne um jede Kante (u, v) Kreis mit Radius $r(u, v)$, so dass gilt:
  - Für jedes Paar s, t für das (u, v) auf einem kürzesten s-t-Weg liegt gilt, dass entweder s oder t in dem Kreis liegt.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Motivation 2

Karlsruher Institut für Technologie

- Zeichne um jede Kante (u, v) Kreis mit Radius $r(u, v)$, so dass gilt:
  - Für jedes Paar s, t für das (u, v) auf einem kürzesten s-t-Weg liegt gilt, dass entweder s oder t in dem Kreis liegt.

Möglichkeiten für diesen Kreis

- „geometrischer Kreis" in euklidischer Ebene
- „graphentheoretischer Kreis" im Eingabegraph

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Motivation 2

Karlsruher Institut für Technologie

- Zeichne um jede Kante (u, v) Kreis mit Radius $r(u, v)$, so dass gilt:
  - Für jedes Paar s, t für das (u, v) auf einem kürzesten s-t-Weg liegt gilt, dass entweder s oder t in dem Kreis liegt.

Möglichkeiten für diesen Kreis

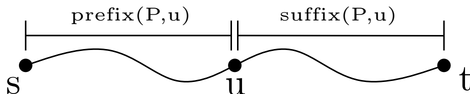
- „geometrischer Kreis“ in euklidischer Ebene
- „graphentheoretischer Kreis“ im Eingabegraph

Strategie für Beschleunigungstechnik:

- Beachte Kante (u, v) nicht, wenn s und t nicht im Kreis um (u, v) liegen.
- wie überprüfen?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Das Zentralitätsmaß Reach

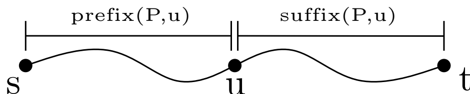


Definition:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Das Zentralitätsmaß Reach

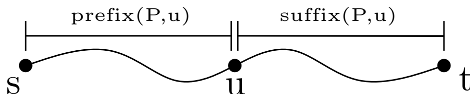


Definition:

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

$$r_P(u) := \min\{\text{len}(\textit{prefix}(P, u)), \text{len}(\textit{suffix}(P, u))\}$$

Das Zentralitätsmaß Reach



Definition:

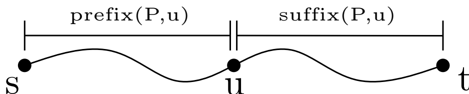
- Sei $P = \langle s, \dots, u, \dots, t \rangle$ Pfad durch u
- dann Reach von u bezüglich P:

$$r_P(u) := \min\{\mathrm{len}(\mathit{prefix}(P,u)), \mathrm{len}(\mathit{suffix}(P,u))\}$$

- Reach von u:

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Das Zentralitätsmaß Reach



Definition:

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

$$r_P(u) := \min\{\mathrm{len}(\mathit{prefix}(P,u)), \mathrm{len}(\mathit{suffix}(P,u))\}$$

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

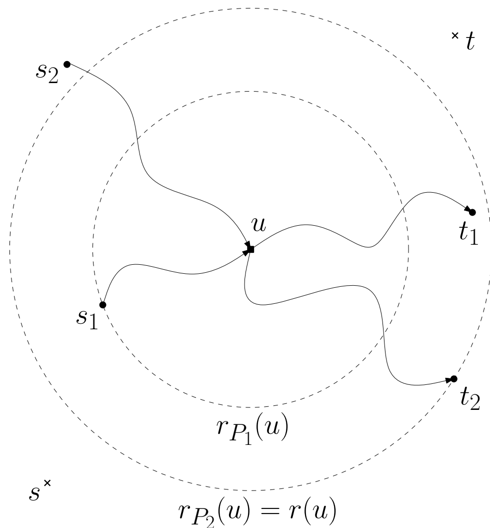
$$r(u) := \max\{r_P(u) \mid P \text{ kürzester Weg mit } u \in P\}$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

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, muss u nicht beachtet werden



Uni Reach Dijkstra - Pseudocode

Karlsruher Institut für Technologie

ReachDijkstra($G = (V, E), s, t$)

1. $d[s] = 0$
2. $Q.clear(), Q.add(s, 0)$
3. **while** $!Q.empty()$ **do**
4. $\quad u \leftarrow Q.deleteMin()$
5. $\quad$ **if** $r(u) < d[u]$ **and** $r(u) < d(u, t)$ **then continue**
6. $\quad$ **forall** $edges\ e = (u, v) \in E$ **do**
7. $\quad\quad$ **if** $d[u] + \text{len}(e) < d[v]$ **then**
8. $\quad\quad\quad d[v] \leftarrow d[u] + \text{len}(e)$
9. $\quad\quad\quad$ **if** $v \in Q$ **then** $Q.decreaseKey(v, d[v])$
10. $\quad\quad\quad$ **else** $Q.insert(v, d[v])$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Problem

Problem?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Problem

Problem:

- Abfrage $r(u) < d(u, t)$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Problem

Karlsruher Institut für Technologie

Problem:

- Abfrage $r(u) < d(u, t)$

- Im geometrischen Fall ist die Überprüfung einfach
- Auch möglich: benutze Landmarken
- Weiteres?

Bidir. Self-Bounding Reach-Dijkstra

Karlsruher Institut für Technologie

Idee (für Vorwärtssuche):

- ignoriere knoten u wenn $d[u] > r(u)$ gilt
- überlasse den Check $d(u, t) > r(u)$ der Rückwärtssuche
- Rückwärtssuche analog (umgekehrt)
- ändere das Stoppkriterium

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bidir. Self-Bounding Reach-Dijkstra

Karlsruher Institut für Technologie

Idee (für Vorwärtssuche):

- ignoriere knoten u wenn $d[u] > r(u)$ gilt
- überlasse den Check $d(u, t) > r(u)$ der Rückwärtssuche
- Rückwärtssuche analog (umgekehrt)
- ändere das Stoppkriterium

neues Stoppkriterium:

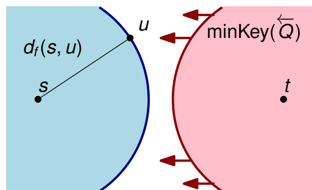
- stoppe Suche in eine Richtung wenn Queue leer oder es gilt: $\text{minKey}(Q) > \mu/2$
- stoppe Anfrage, wenn beide Suchrichtungen gestoppt haben
- Korrektheit gute Fingerübung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bidir. Dist.-Bounding Reach-Dijkstra

Idee (für Vorwärtssuche):

- wenn u von Rückwärtssuche noch nicht erreicht, ist minKey($\overleftarrow{Q}$) eine untere Schranke für $d(u,t)$
- wenn u von Rückwärtssuche abgearbeitet, $d(u,t)$ bekannt

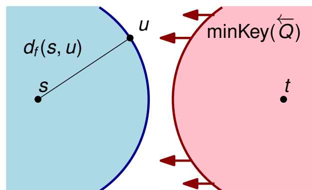


Bidir. Dist.-Bounding Reach-Dijkstra

KIT – Karlsruher Institut für Technologie

Idee (für Vorwärtssuche):

- wenn u von Rückwärtssuche noch nicht erreicht, ist $\text{minKey}(\overleftarrow{Q})$ eine untere Schranke für $d(u, t)$
- wenn u von Rückwärtssuche abgearbeitet, $d(u, t)$ bekannt



somit:

- ignoriere u, wenn $r(u) < d_f[u]$ und $r(u) < \min\{d_b[u], \text{minKey}(\overleftarrow{Q})\}$
- Stoppkriterium bleibt erhalten (also $\text{minKey}(\overrightarrow{Q}) + \text{minKey}(\overleftarrow{Q}) \geq \mu$)
- wenn als Alternierungsstrategie $\min\{\text{minKey}(\overrightarrow{Q}), \text{minKey}(\overleftarrow{Q})\}$ gewählt, gilt für Vorwärtssuche: $d_f[u] \leq \text{minKey}(\overleftarrow{Q})$

Optimierungen

Karlsruher Institut für Technologie

mögliche Verbesserungen:

- Early (Kanten-)Pruning:
  (u, v) muss nicht relaxiert werden, wenn gilt:
  - $d_f[u] + \text{len}(u, v) > r(v)$
  - und $r(v) < \min\{d_b[v], \text{minKey}(\overleftarrow{Q})\}$

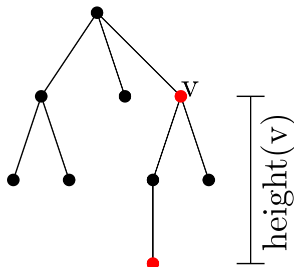
- Kanten sortieren:
  - sortiere ausgehende Kanten (u, v_i) absteigende nach $r(v_i)$
  - wenn Kante relaxiert wird mit $r(v_i) < \text{minKey}(\overleftarrow{Q})$ und $r(v_i) < d_f[u]$ müssen die restlichen Kanten ausgehend von u nicht relaxiert werden

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Vorberechnung: Erster Ansatz

Wie kann man Reach-Werte vorberechnen?

- initialisieren $r(u) = 0$ für alle Knoten
- für jeden Knoten u
  - konstruiere kürzeste Wege-Baum
  - höhe von Knoten v: Abstand von v zum am weitesten entfernten Nachfolger
  - für jeden Knoten v:
    $r(v) = \max\{r(v), \min\{d(u,v), \text{height}(v)\}\}$

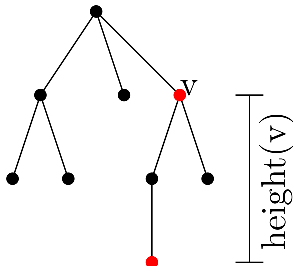


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Vorberechnung: Erster Ansatz

Wie kann man Reach-Werte vorberechnen?

- initialisieren $r(u) = 0$ für alle Knoten
- für jeden Knoten u
  - konstruiere kürzeste Wege-Baum
  - höhe von Knoten v: Abstand von v zum am weitesten entfernten Nachfolger
  - für jeden Knoten v: $r(v) = \max\{r(v), \min\{d(u,v), \text{height}(v)\}\}$



altes Problem:

- Vorberechnung basiert auf all-pair-shortest paths

Approximation

Karlsruher Institut für Technologie

Beobachtung:

- es genügt, für jeden Knoten eine obere Schranke des Reach-Wertes zu haben

Problem:

- untere Schranken einfach zu finden:
  - breche Konstruktion der Bäume einfach bei bestimmter Größe ab
- aber: untere Schranken sind unbrauchbar
- Berechnung von oberen Schranken deutlich schwieriger
- möglich, aber sehr aufwendig
- nicht (mehr) Bestandteil der Vorlesung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Literatur

Karlsruher Institut für Technologie

Literatur (Reach, RE, REAL):

- R. Gutman:
  Reach-based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks
  In: *ALENEX, 2004*
- Andrew V. Goldberg and Haim Kaplan and Renato F. Werneck:
  Reach for A*: Shortest Path Algorithms with Preprocessing
  In: *Shortest Paths: Ninth DIMACS Implementation Challenge, 2009.*

Hierarchische Techniken

Ideensammlung:

- identifiziere wichtige Knoten mit Zentralitätsmaß
- überspringe unwichtige Teile des Graphen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Overlay Graph

Karlsruher Institut für Technologie

Gegeben

- Eingabegraph $G = (V, E, \text{len})$
- Knotenmenge $V \supseteq V_L$

Berechne

- Berechne $G_L = (V_L, E_L, \text{len}_L)$, so dass Distanzen in G_L wie in G

Wahl der Knotenmenge

Karlsruher Institut für Technologie

Ideensammlung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wahl der Knotenmenge

Karlsruher Institut für Technologie

Ideensammlung

- Randknoten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wahl der Knotenmenge

Karlsruher Institut für Technologie

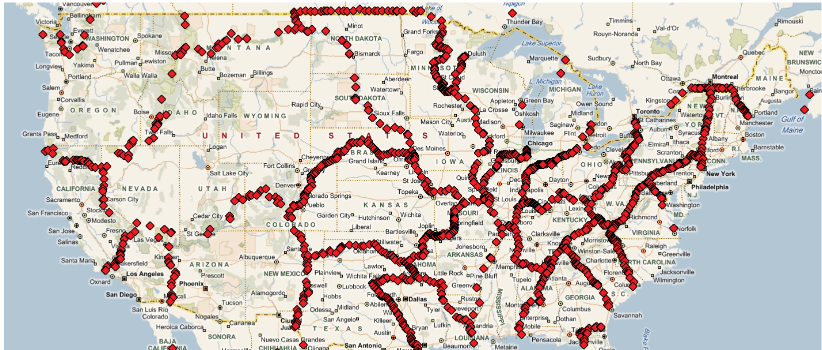
Ideensammlung

- Randknoten
- (approximiertes) Zentralitätsmaß

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Partitionierung

Letztes Mal: Strassengraphen haben natürliche Schnitte



- Jeder Pfad durch eine Zelle betritt/verlässt die Zelle durch einen Randknoten

$\Rightarrow$ Minimiiere # Schnittkanten mit Zellgrösse $\leq U$ (Eingabeparameter)

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

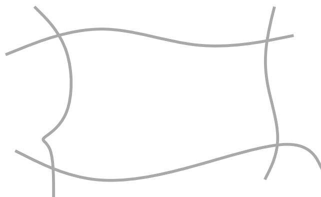
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

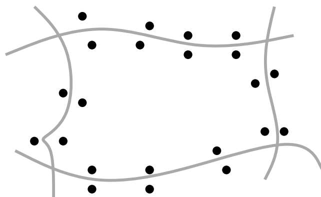


Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



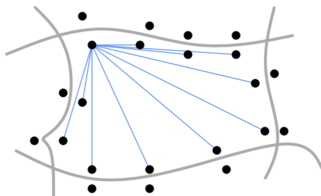
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

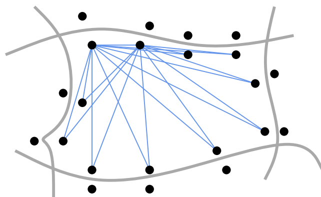


Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



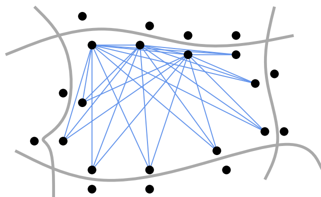
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



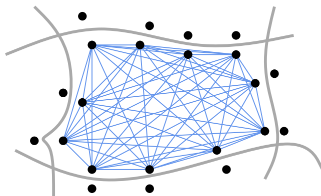
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

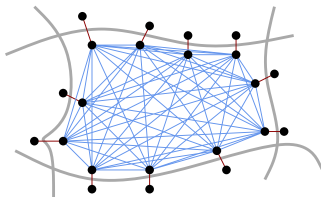
- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

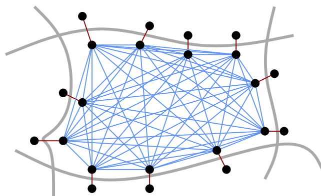


Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

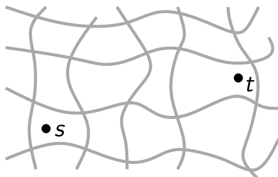
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Suchgraph:

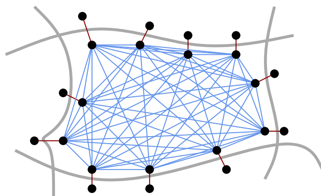
- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

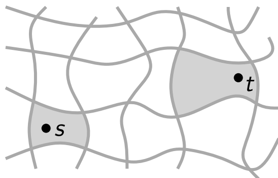
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

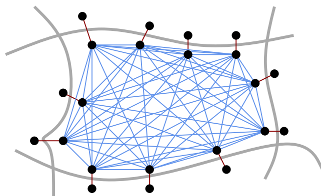
- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

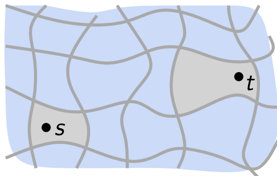
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





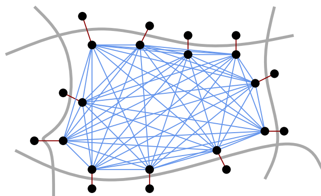
Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

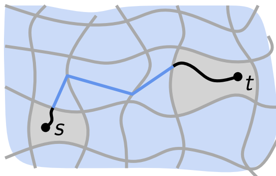
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

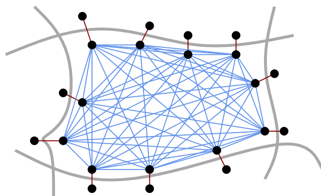
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

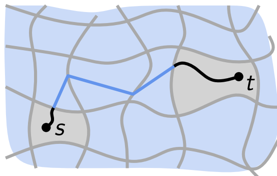
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

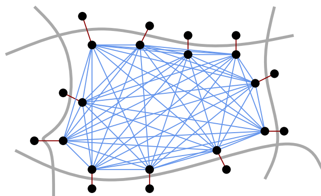
Ausnutzung der Partition

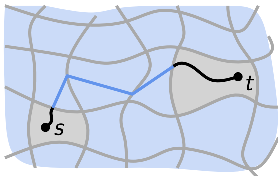
Karlsruher Institut für Technologie

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

Example

2^{15} Knoten pro Zelle, 626 Zellen $\Rightarrow$ 34 k Knoten im Overlaygraphen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ergebnisse

Karlsruher Institut für Technologie

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ergebnisse

Karlsruher Institut für Technologie

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Beispiel:

	Metric Customization		Queries	
metric	time [s]	space [MB]	scans	time [ms]
Travel time	20	10	45134	10
Distance	20	10	47127	11

(partition: $\leq 2^{14}$ nodes/cell)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ergebnisse

Karlsruher Institut für Technologie

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Beispiel:

	Metric Customization		Queries	
metric	time [s]	space [MB]	scans	time [ms]
Travel time	20	10	45134	10
Distance	20	10	47127	11

(partition: $\leq 2^{14}$ nodes/cell)

West Europa (18 M nodes, 42 M edges)
Intel Core-i7 920 (four cores at 2.67 GHz)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ergebnisse

Karlsruher Institut für Technologie

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Beispiel:

	Metric Customization		Queries	
metric	time [s]	space [MB]	scans	time [ms]
Travel time	20	10	45134	10
Distance	20	10	47127	11

(partition: $\leq 2^{14}$ nodes/cell)

West Europa (18 M nodes, 42 M edges)
Intel Core-i7 920 (four cores at 2.67 GHz)

Diskussion: Unterschied zu Multilevel-ArcFlags?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Diskussion

Karlsruher Institut für Technologie

Verbesserungen?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Diskussion

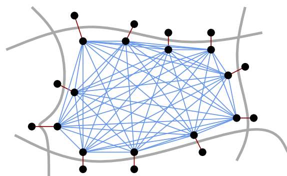
Karlsruher Institut für Technologie

Verbesserungen?

- Ausdünnung der Graphen
- Kombination mit zielgerichteter Suche
- Multi-Level Partitionierung

Naheliegendes

Ausdünnung des Overlaygraphen:

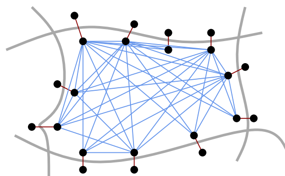


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten

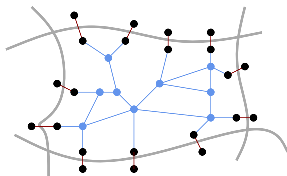


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

Ausdünnung des Overlaygraphen:

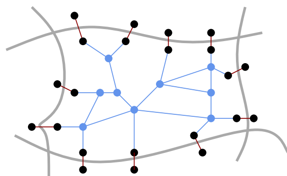
- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)



Naheliegendes

Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)
- etwas schnellere Anfragen

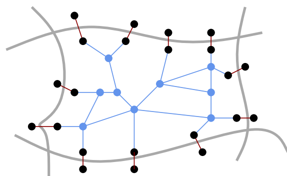


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)
- etwas schnellere Anfragen



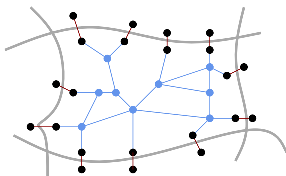
Kombination mit zielgerichteter Suche:

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

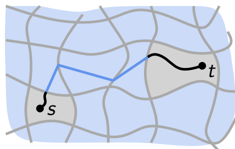
Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)
- etwas schnellere Anfragen



Kombination mit zielgerichteter Suche:

- nur auf (kleinem) Overlaygraphen
- ALT/Arc-Flags
- beschleunigt Anfragen, macht Vorberechnung und Queries komplizierter

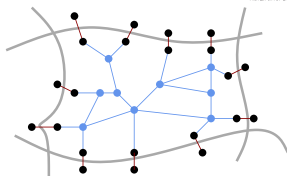


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Naheliegendes

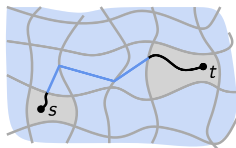
Ausdünnung des Overlaygraphen:

- entferne unnötige Kanten
- füge Knoten hinzu (aus Originalgraphen)
- etwas schnellere Anfragen



Kombination mit zielgerichteter Suche:

- nur auf (kleinem) Overlaygraphen
- ALT/Arc-Flags
- beschleunigt Anfragen, macht Vorberechnung und Queries komplizierter



Es geht besser (und einfacher!)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multilevel-Overlay Graphs

Karlsruher Institut für Technologie

Gegeben

- Eingabegraph $G = (V, E, \text{len})$
- Folge $V := V_0 \supseteq V_1 \supseteq \ldots \supseteq V_L$ von Teilmengen von V.

Berechne

- Folge $G_0 = (V_0, E_0, \text{len}_0), \ldots, G_L = (V_L, E_L, \text{len}_L)$ von Graphen, so dass Distanzen in G_i wie in G_0

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

Karlsruher Institut für Technologie

Multi-Level Partition:

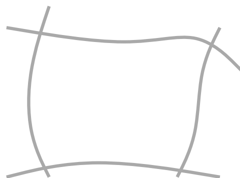
- benutze mehrere Partitionen (mit PUNCH)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

Multi-Level Partition:

- benutze mehrere Partitionen (mit PUNCH)



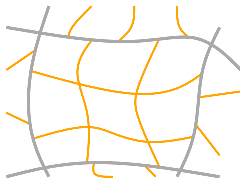
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

Karlsruher Institut für Technologie

Multi-Level Partition:

- benutze mehrere Partitionen (mit PUNCH)



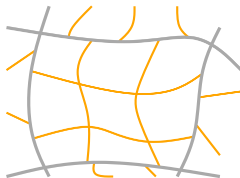
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

KIT
Karlsruher Institut für Technologie

Multi-Level Partition:

- benutze mehrere Partitionen (mit PUNCH)
- berechne Cliquen bottom-up
- benutze G_{i-1} um G_i zu bestimmen
- trade-off zwischen Platz und Suchzeiten

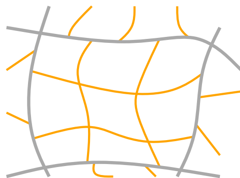


Multi-Level Ansatz

Karlsruher Institut für Technologie

Multi-Level Partition:

- benutze mehrere Partitionen (mit PUNCH)
- berechne Cliquen bottom-up
- benutze G_{i-1} um G_i zu bestimmen
- trade-off zwischen Platz und Suchzeiten

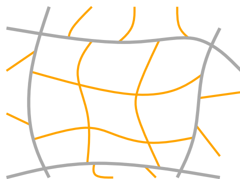


Suchgraph:

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Ansatz

Karlsruher Institut für Technologie



Multi-Level Partition:

- benutze mehrere Partitionen (mit PUNCH)
- berechne Cliquen bottom-up
- benutze G_{i-1} um G_i zu bestimmen
- trade-off zwischen Platz und Suchzeiten

Suchgraph:

- Overlay auf obersten Level (G_L) ...
- und alle Ziel- und Startzellen (auf jedem Level)
- (bidirektionaler) Dijkstra

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Phantom Level

Beobachtung: Viele Level $\Rightarrow$ schnellere Vorberechnung, mehr Platz

Phantom Level

Beobachtung: Viele Level ⇒ schnellere Vorberechnung, mehr Platz

- Benutze mehr Level für Vorberechnung (32 Knoten pro Zelle)
- Speichere die unteren Level aber nicht dauerhaft
- Schnellere Vorberechnung
- Werden aber nicht für Queries benutzt

Cell Size	Cust Time
$[2^{14}]$	20.0 s
$[2^{8} : 2^{14}]$	4.9 s

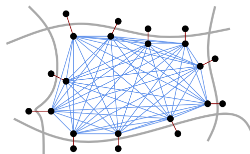
(metric: travel time)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Implementation

Repräsentation des Overlaygraphen:

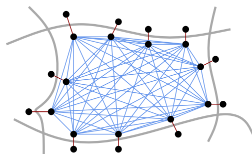
- wirklich Graph-DS nötig?



Implementation

Repräsentation des Overlaygraphen:

- wirklich Graph-DS nötig?
- speicher Cliquen als Matrizen
- Beschleunigt Vorberechnung und Queries um einen Faktor 2
  Schneller als ausgedünnter Overlay!

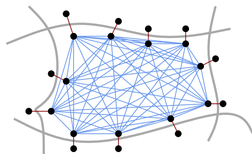


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Implementation

Repräsentation des Overlaygraphen:

- wirklich Graph-DS nötig?
- speicher Cliquen als Matrizen
- Beschleunigt Vorberechnung und Queries um einen Faktor 2
  Schneller als ausgedünnter Overlay!
- Entkoppelung von Metrik und Topology
- reduziert Platzverbrauch: 32 Bit pro Cliquenkante (pro Metrik) + 1 x 32 Bit fuer Topology



3-Phasen Algorithmus

Karlsruher Institut für Technologie

1. Metrik-unabhängige Vorberechnung

- Partitionierung des Graphen
- Bauen der Topology des Overlay Graphen
- Linearer Platz für Partition, kann ein wenig dauern (nur einmal)
- PUNCH: 15-30 Minuten (mit aggressiven Parametern)

3-Phasen Algorithmus

Karlsruher Institut für Technologie

1. Metrik-unabhängige Vorberechnung

- Partitionierung des Graphen
- Bauen der Topology des Overlay Graphen
- Linearer Platz für Partition, kann ein wenig dauern (nur einmal)
- PUNCH: 15-30 Minuten (mit aggressiven Parametern)

2. Metrik-abhängige Vorberechnung

- Berechnung der Gewicht der Matrix-Einträge
- Mit Hilfe von lokalen (hoch-parallelisierbaren) Dijkstra-Suchen
- Overhead pro Metrik gering (ein Array)

3-Phasen Algorithmus

Karlsruher Institut für Technologie

1. Metrik-unabhängige Vorberechnung

- Partitionierung des Graphen
- Bauen der Topology des Overlay Graphen
- Linearer Platz für Partition, kann ein wenig dauern (nur einmal)
- PUNCH: 15-30 Minuten (mit aggressiven Parametern)

2. Metrik-abhängige Vorberechnung

- Berechnung der Gewicht der Matrix-Einträge
- Mit Hilfe von lokalen (hoch-parallelisierbaren) Dijkstra-Suchen
- Overhead pro Metrik gering (ein Array)

3. Queries

- Benutze Graph und beide Vorberechnungen
- (paralleler) bidirektionaler Dijkstra

Performance

	Algorithm	Customization time [s]	Customization space [MB]	Queries scans	Queries time [ms]
travel time	MLD-1 $[2^{14}]$	4.9	9.8	45134	5.67
	MLD-2 $[2^{12} : 2^{18}]$	5.0	18.4	12722	1.79
	MLD-3 $[2^{10} : 2^{15} : 2^{20}]$	5.2	32.3	6074	0.91
	MLD-4 $[2^{8} : 2^{12} : 2^{16} : 2^{20}]$	5.2	59.0	3897	0.71
distances	MLD-1 $[2^{14}]$	4.7	9.8	47127	6.19
	MLD-2 $[2^{12} : 2^{18}]$	4.9	18.4	13114	1.85
	MLD-3 $[2^{10} : 2^{15} : 2^{20}]$	5.1	32.3	6315	1.01
	MLD-4 $[2^{8} : 2^{12} : 2^{16} : 2^{20}]$	4.7	59.0	4102	0.77

Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Performance

Karlsruher Institut für Technologie

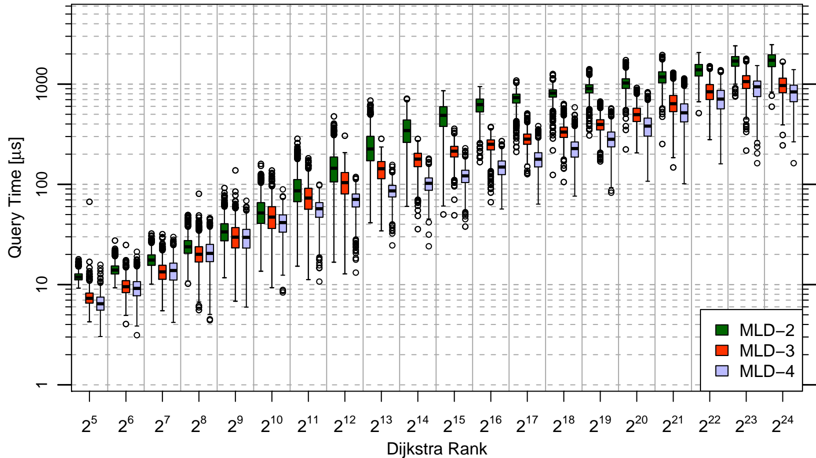
	Algorithm	Customization time [s]	Customization space [MB]	Queries scans	Queries time [ms]
travel time	MLD-1 $[2^{14}]$	4.9	9.8	45134	5.67
	MLD-2 $[2^{12} : 2^{18}]$	5.0	18.4	12722	1.79
	MLD-3 $[2^{10} : 2^{15} : 2^{20}]$	5.2	32.3	6074	0.91
	MLD-4 $[2^{8} : 2^{12} : 2^{16} : 2^{20}]$	5.2	59.0	3897	0.71
distances	MLD-1 $[2^{14}]$	4.7	9.8	47127	6.19
	MLD-2 $[2^{12} : 2^{18}]$	4.9	18.4	13114	1.85
	MLD-3 $[2^{10} : 2^{15} : 2^{20}]$	5.1	32.3	6315	1.01
	MLD-4 $[2^{8} : 2^{12} : 2^{16} : 2^{20}]$	4.7	59.0	4102	0.77

MLD: fast customization / compact / real-time queries / robust

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

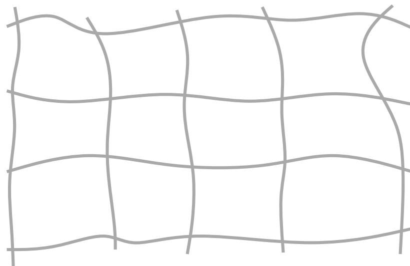
Local Queries

KIT
Karlsruher Institut für Technologie



Entpacken der Shortcuts

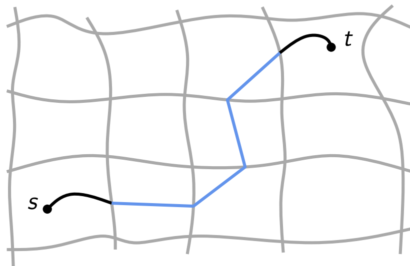
Von Shortcuts zu vollen Pfaden:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entpacken der Shortcuts

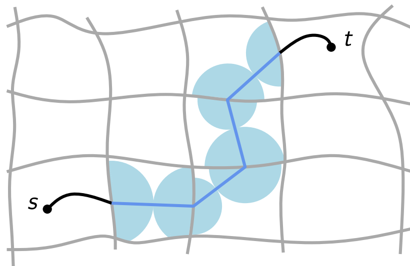
Von Shortcuts zu vollen Pfaden:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

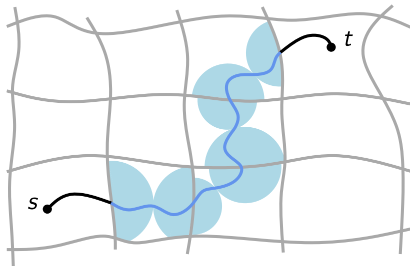
Entpacken der Shortcuts

Von Shortcuts zu vollen Pfaden:



Entpacken der Shortcuts

Von Shortcuts zu vollen Pfaden:

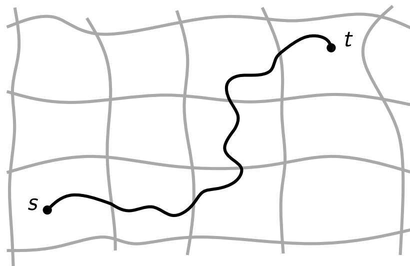


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entpacken der Shortcuts

Von Shortcuts zu vollen Pfaden:

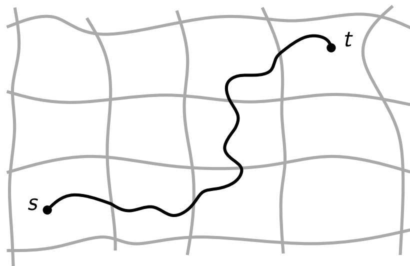
- bidirektionale Suche
- beschränkt auf die Zelle
- benutze untere Level rekursiv
- parallelisierbar
- benutze LRU-cache



Entpacken der Shortcuts

Von Shortcuts zu vollen Pfaden:

- bidirektionale Suche
- beschränkt auf die Zelle
- benutze untere Level rekursiv
- parallelisierbar
- benutze LRU-cache



Kaum zusätzlicher Speicher, 20-30% Zeit-Overhead.

Customizable Route Planning

Karlsruher Institut für Technologie

Eigenschaften:

- viele Metriken
- schnelle lokale and globale updates
  Zelle in Millisekunden, gesamter Graph in Sekunden
- real-time queries (5000$\times$ Dijkstra)
- Bing Maps Routing Engine

Partition ist der Schlüssel.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Literatur

Karlsruher Institut für Technologie

Literatur (Multilevel Overlay Graph):

- Frank Schulz, Dorothea Wagner, Christos Zaroliagis
  Using Multi-Level Graphs for Timetable Information in Railway Systems
  In : *Proceedings of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02)*, 2002
- Peter Sanders, Dominik Schultes
  Dynamic Highway Node Routing
  In: *Proceedings of the 6th Workshop on Experimental Algorithms (WEA '07)*, 2007
- Daniel Delling, Andrew V. Goldberg, Thomas Pajor, Renato Werneck
  Customizable Route Planning
  In: *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA '11)*, 2011

Hierarchische Techniken

Ideensammlung:

- identifiziere wichtige Knoten mit Zentralitätsmaß
- überspringe unwichtige Teile des Graphen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Overlay Graph

Karlsruher Institut für Technologie

Gegeben

- Eingabegraph $G = (V, E, \text{len})$
- Knotenmenge $V \supseteq V_L$

Berechne

- Berechne $G_L = (V_L, E_L, \text{len}_L)$, so dass Distanzen in G_L wie in G

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multilevel-Overlay Graphs

Karlsruher Institut für Technologie

Gegeben

- Eingabegraph $G = (V, E, \text{len})$
- Folge $V := V_0 \supseteq V_1 \supseteq \ldots \supseteq V_L$ von Teilmengen von V.

Berechne

- Folge $G_0 = (V_0, E_0, \text{len}_0), \ldots, G_L = (V_L, E_L, \text{len}_L)$ von Graphen, so dass Distanzen in G_i wie in G_0

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wahl der Knotenmenge

- Randknoten
- können auch beliebige Knotenmengen sein

Contraction Hierarchies

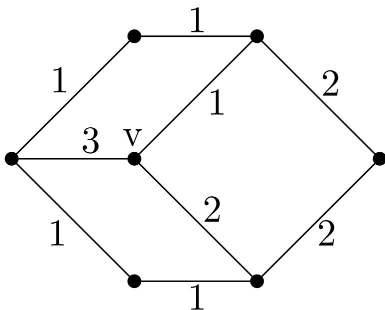
Zutaten:

- Ordne Knoten nach Wichtigkeit
  $r : V \to [0, \ldots, n-1]$
- Shortcut Operation

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Karlsruher Institut für Technologie



Zutaten:

- Ordne Knoten nach Wichtigkeit
  $r : V \to [0, \ldots, n-1]$
- Shortcut Operation

Shortcut Operation (von Knoten v):

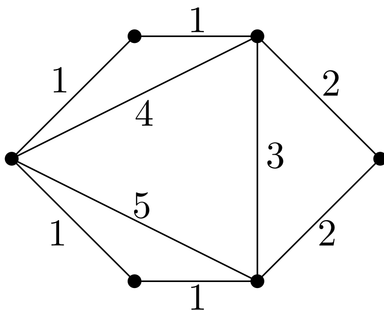
Contraction Hierarchies

Zutaten:

- Ordne Knoten nach Wichtigkeit $r : V \rightarrow [0, \ldots, n-1]$
- Shortcut Operation

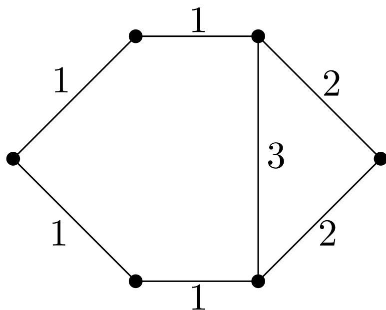
Shortcut Operation (von Knoten v):

- entferne v



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies



Zutaten:

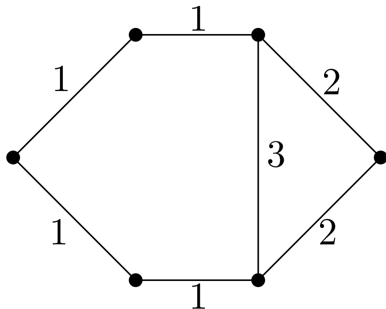
- Ordne Knoten nach Wichtigkeit
  $r : V \to [0, \ldots, n-1]$
- Shortcut Operation

Shortcut Operation (von Knoten *v*):

- entferne *v*
- für jedes Paar *u*, *w* füge Kanten (*u*, *w*) zum Graphen wenn *v* auf dem einzigen kürzesten Weg von *u* nach *w* liegt

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies



Zutaten:

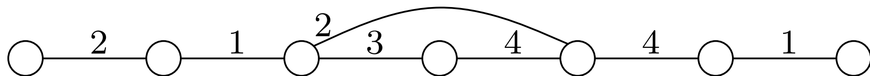
- Ordne Knoten nach Wichtigkeit $r : V \to [0, \ldots, n-1]$
- Shortcut Operation

Shortcut Operation (von Knoten v):

- entferne v
- für jedes Paar u, w füge Kanten (u, w) zum Graphen wenn v auf dem einzigen kürzesten Weg von u nach w liegt
- kann durch lokale Dijkstra Suchen von den Nachbarn berechnet werden

Contraction Hierarchies

Vorberechnung:

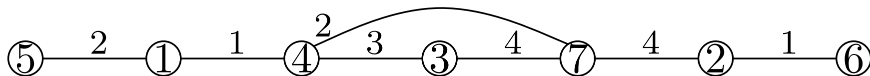


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Vorberechnung:

- ordne Knoten

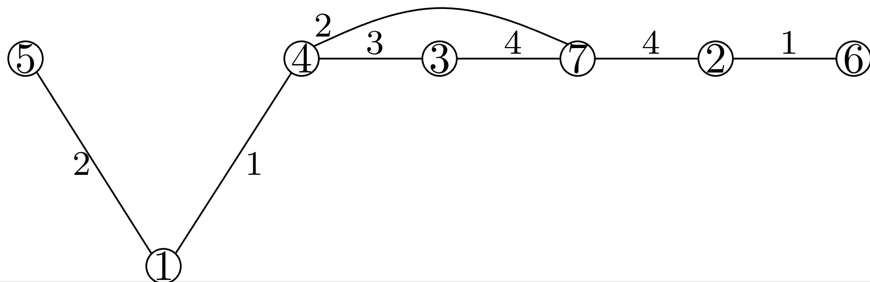


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Vorberechnung:

- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)

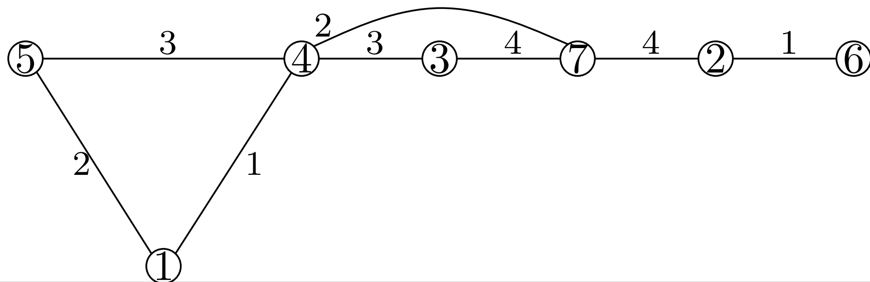


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Vorberechnung:

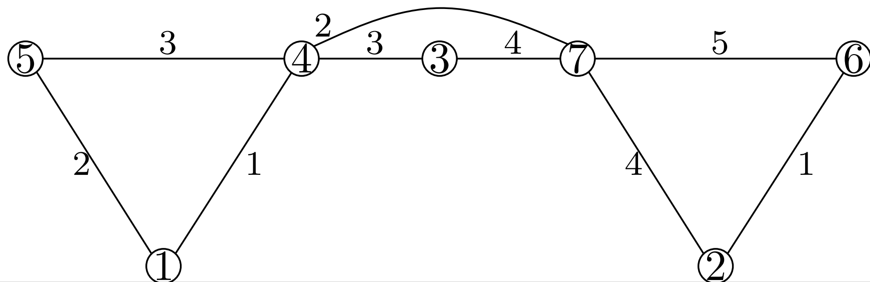
- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)



Contraction Hierarchies

Vorberechnung:

- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)

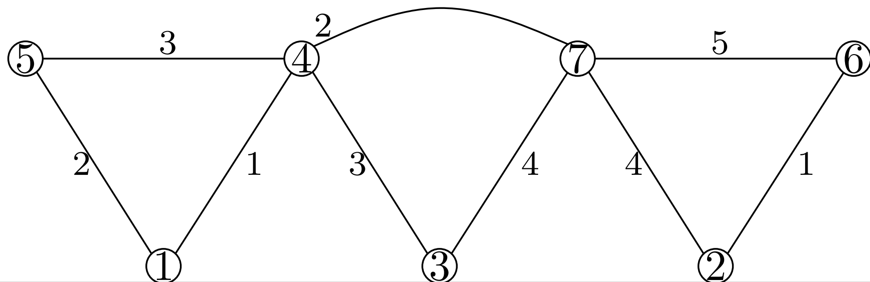


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Vorberechnung:

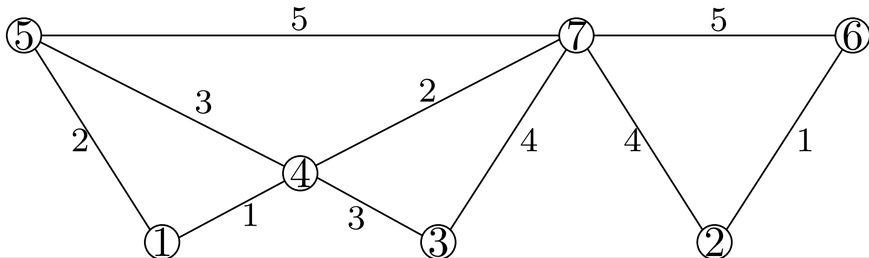
- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)



Contraction Hierarchies

Vorberechnung:

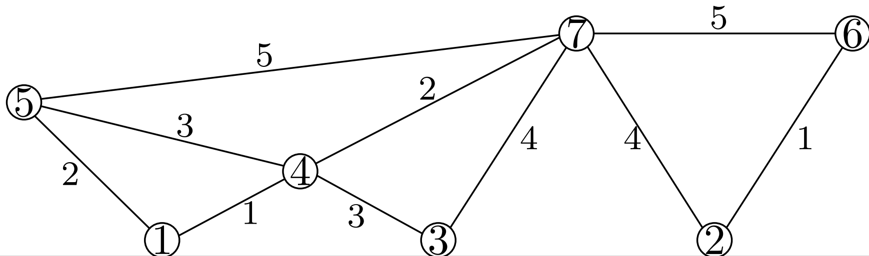
- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)



Contraction Hierarchies

Vorberechnung:

- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)

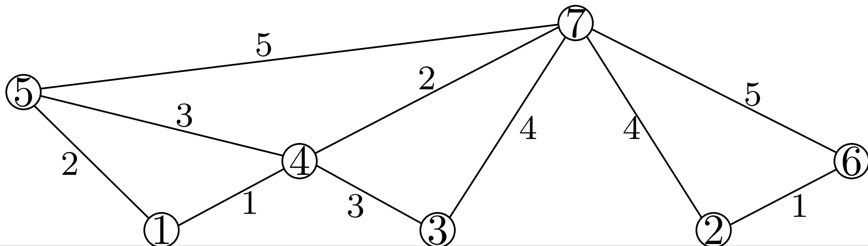


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Vorberechnung:

- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)

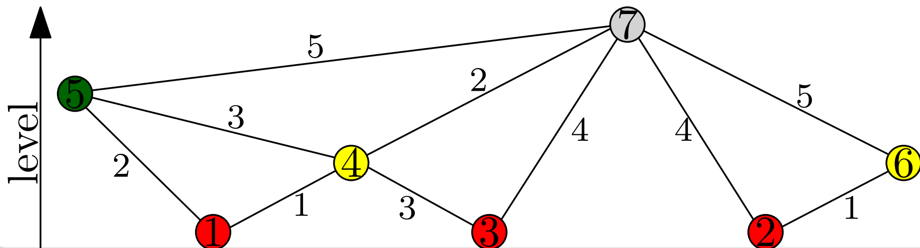


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Vorberechnung:

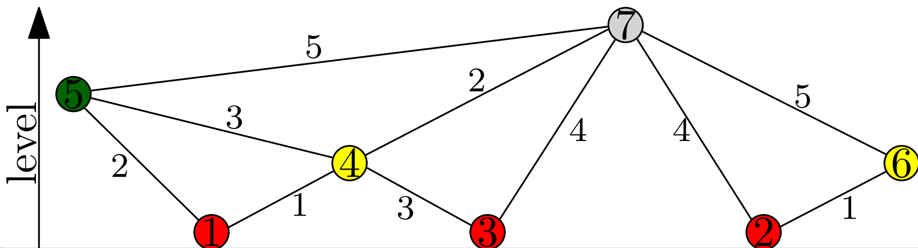
- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)
- bestimme Level des Knoten



Contraction Hierarchies

Vorberechnung:

- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)
- bestimme Level des Knoten
- erzeugt einen Graph $G^+ = (V, A \cup A^+)$

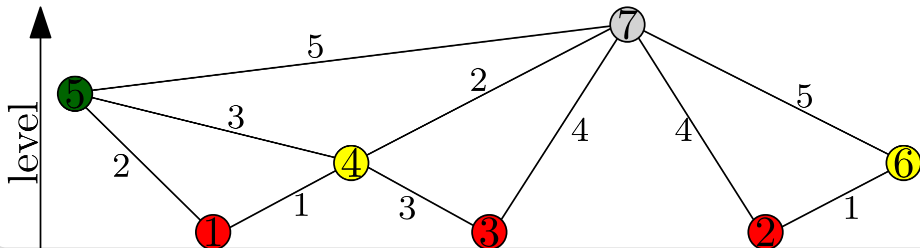


Contraction Hierarchies

KIT
Karlsruher Institut für Technologie

Anfrage

- modifizierter bidirektionaler Dijkstra
- folge nur Kanten zu wichtigeren Knoten
  Aufwärtsgraph $G_\uparrow := (V, E_\uparrow)$ with $E_\uparrow := \{(u, v) \in E : u < v\}$
  Abwärtsgraph $G_\downarrow := (V, E_\downarrow)$ with $E_\downarrow := \{(u, v) \in E : u > v\}$
- Vorwärtssuche in $G_\uparrow$ und Rückwärtssuche in $G_\downarrow$
- konservatives Stoppkriterium
- jede Suche stoppt wenn Queue leer oder $minKey(Q) > \mu$

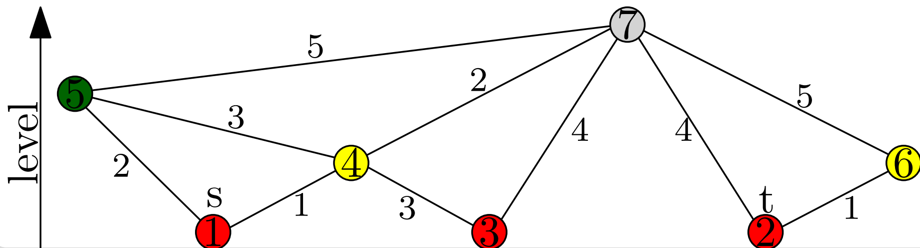


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Anfrage

- modifizierter bidirektionaler Dijkstra
- folge nur Kanten zu wichtigeren Knoten
  Aufwärtsgraph $G_\uparrow := (V, E_\uparrow)$ with $E_\uparrow := \{(u, v) \in E : u < v\}$
  Abwärtsgraph $G_\downarrow := (V, E_\downarrow)$ with $E_\downarrow := \{(u, v) \in E : u > v\}$
- Vorwärtssuche in $G_\uparrow$ und Rückwärtssuche in $G_\downarrow$
- konservatives Stoppkriterium
- jede Suche stoppt wenn Queue leer oder $minKey(Q) > \mu$



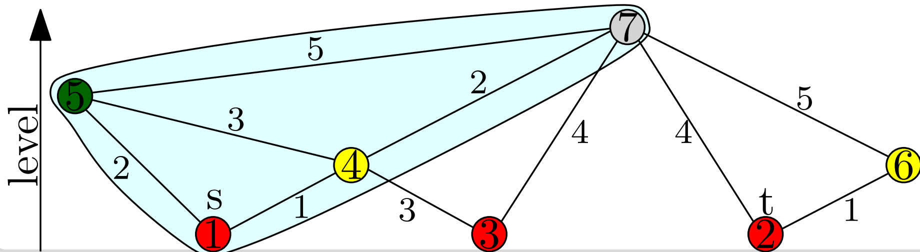
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Contraction Hierarchies

Karlsruher Institut für Technologie

Anfrage

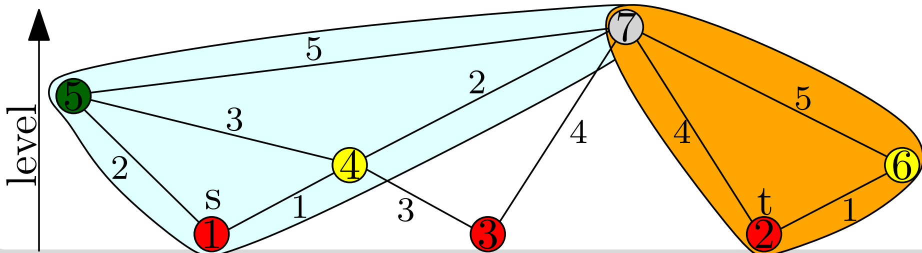
- modifizierter bidirektionaler Dijkstra
- folge nur Kanten zu wichtigeren Knoten
  Aufwärtsgraph $G_\uparrow := (V, E_\uparrow)$ with $E_\uparrow := \{(u,v) \in E : u < v\}$
  Abwärtsgraph $G_\downarrow := (V, E_\downarrow)$ with $E_\downarrow := \{(u,v) \in E : u > v\}$
- Vorwärtssuche in $G_\uparrow$ und Rückwärtssuche in $G_\downarrow$
- konservatives Stoppkriterium
- jede Suche stoppt wenn Queue leer oder $minKey(Q) > \mu$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Anfrage

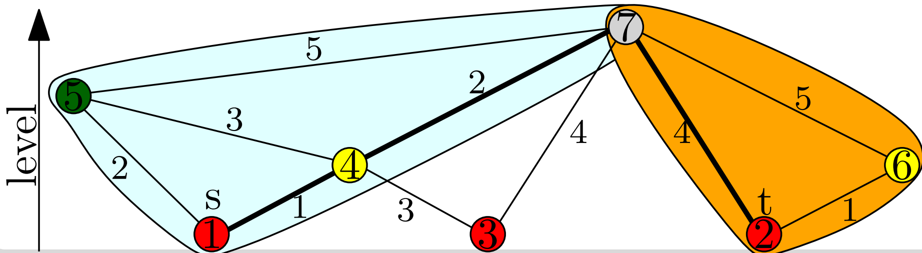
- modifizierter bidirektionaler Dijkstra
- folge nur Kanten zu wichtigeren Knoten
  Aufwärtsgraph $G_\uparrow := (V, E_\uparrow)$ with $E_\uparrow := \{(u, v) \in E : u < v\}$
  Abwärtsgraph $G_\downarrow := (V, E_\downarrow)$ with $E_\downarrow := \{(u, v) \in E : u > v\}$
- Vorwärtssuche in $G_\uparrow$ und Rückwärtssuche in $G_\downarrow$
- konservatives Stoppkriterium
- jede Suche stoppt wenn Queue leer oder $minKey(Q) > \mu$



Contraction Hierarchies

Anfrage

- modifizierter bidirektionaler Dijkstra
- folge nur Kanten zu wichtigeren Knoten
  Aufwärtsgraph $G_\uparrow := (V, E_\uparrow)$ with $E_\uparrow := \{(u,v) \in E : u < v\}$
  Abwärtsgraph $G_\downarrow := (V, E_\downarrow)$ with $E_\downarrow := \{(u,v) \in E : u > v\}$
- Vorwärtssuche in $G_\uparrow$ und Rückwärtssuche in $G_\downarrow$
- konservatives Stoppkriterium
- jede Suche stoppt wenn Queue leer oder $minKey(Q) > \mu$



Korrektheit

Karlsruher Institut für Technologie

Beobachtung:

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Korrektheit

Beobachtung:

- der kürzeste s–t Weg P existiert auch in G^+
  (wir fügen nur Kanten hinzu)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Korrektheit

Beobachtung:

- der kürzeste s–t Weg P existiert auch in G^+ (wir fügen nur Kanten hinzu)
- aber wir suchen nur aufwärts

Korrektheit

Beobachtung:

- der kürzeste s–t Weg P existiert auch in G^+ (wir fügen nur Kanten hinzu)
- aber wir suchen nur aufwärts
- es gibt einen wichtigsten Knoten v auf P

Korrektheit

Karlsruher Institut für Technologie

Beobachtung:

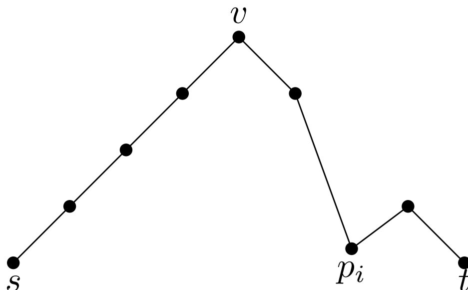
- der kürzeste s–t Weg P existiert auch in G^+
  (wir fügen nur Kanten hinzu)
- aber wir suchen nur aufwärts
- es gibt einen wichtigsten Knoten v auf P
- wir müssen zeigen, dass es einen s–aufwärts–v–abwärts–t Pfad gibt

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Korrektheit

Beobachtung:

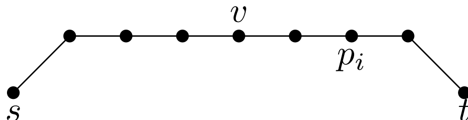
- der kürzeste s–t Weg P existiert auch in G^+ (wir fügen nur Kanten hinzu)
- aber wir suchen nur aufwärts
- es gibt einen wichtigsten Knoten v auf P
- wir müssen zeigen, dass es einen s–aufwärts–v–abwärts–t Pfad gibt



Korrektheit

Beobachtung:

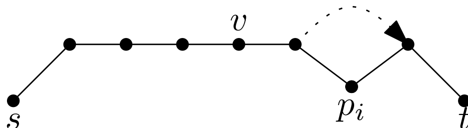
- der kürzeste s–t Weg P existiert auch in G^+ (wir fügen nur Kanten hinzu)
- aber wir suchen nur aufwärts
- es gibt einen wichtigsten Knoten v auf P
- wir müssen zeigen, dass es einen s–aufwärts–v–abwärts–t Pfad gibt



Korrektheit

Beobachtung:

- der kürzeste *s*–*t* Weg *P* existiert auch in G^+
  (wir fügen nur Kanten hinzu)
- aber wir suchen nur aufwärts
- es gibt einen wichtigsten Knoten *v* auf *P*
- wir müssen zeigen, dass es einen *s*–aufwärts–*v*–abwärts–*t* Pfad gibt

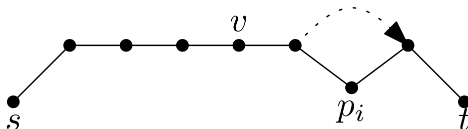


Korrektheit

Karlsruher Institut für Technologie

Beobachtung:

- der kürzeste s–t Weg P existiert auch in G^+
  (wir fügen nur Kanten hinzu)
- aber wir suchen nur aufwärts
- es gibt einen wichtigsten Knoten v auf P
- wir müssen zeigen, dass es einen s–aufwärts–v–abwärts–t Pfad gibt
- verändertes Stoppkriterium:
  es kann $v = s$ oder $v = t$ sein

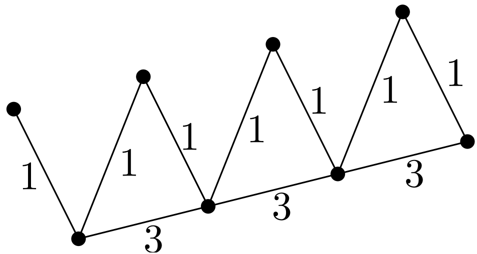


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Stall-On-Demand

Beobachtung:

- Aufwärtssuchen können Knoten mit falscher Distanz scannen

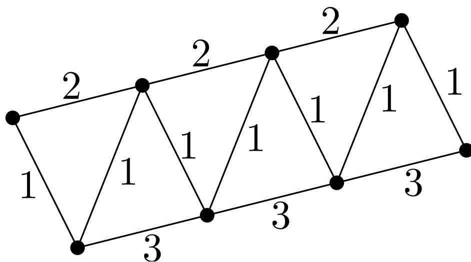


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Stall-On-Demand

Beobachtung:

- Aufwärtssuchen können Knoten mit falscher Distanz scannen

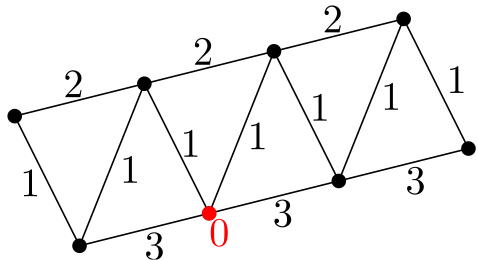


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Stall-On-Demand

Beobachtung:

- Aufwärtssuchen können Knoten mit falscher Distanz scannen

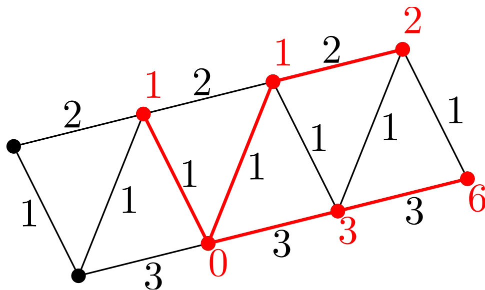


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Stall-On-Demand

Beobachtung:

- Aufwärtssuchen können Knoten mit falscher Distanz scannen

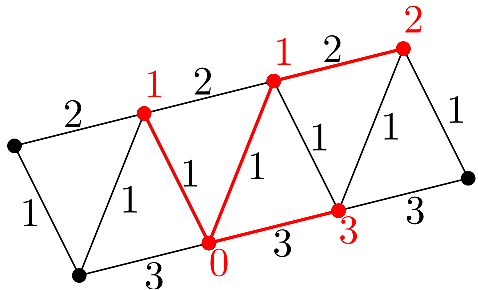


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Stall-On-Demand

Beobachtung:

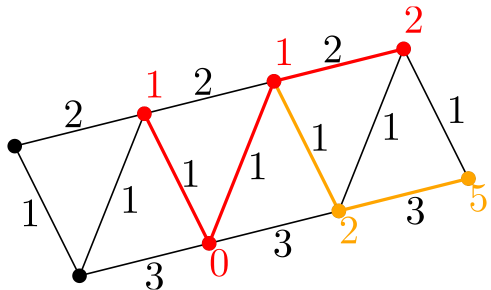
- Aufwärtssuchen können Knoten mit falscher Distanz scannen
- Heuristik versucht das durch scannen von Abwärtskanten zu verhindern
- diese Knoten werden deaktiviert, können aber wieder aktiviert werden



Stall-On-Demand

Beobachtung:

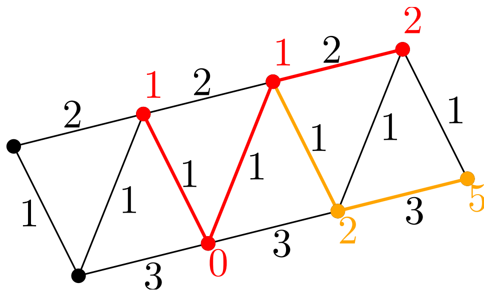
- Aufwärtssuchen können Knoten mit falscher Distanz scannen
- Heuristik versucht das durch scannen von Abwärtskanten zu verhindern
- diese Knoten werden deaktiviert, können aber wieder aktiviert werden



Stall-On-Demand

Beobachtung:

- Aufwärtssuchen können Knoten mit falscher Distanz scannen
- Heuristik versucht das durch scannen von Abwärtskanten zu verhindern
- diese Knoten werden deaktiviert, können aber wieder aktiviert werden
- reduziert Suchraum um einen Faktor 4



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Nächste Termine

Karlsruher Institut für Technologie

Montag, 13.5.2012

Mittwoch, 22.5.2012

Institut für Theoretische Informatik
Lehrstuhl Algorithmik