Karlsruher Institut für Technologie

Algorithmen für Routenplanung

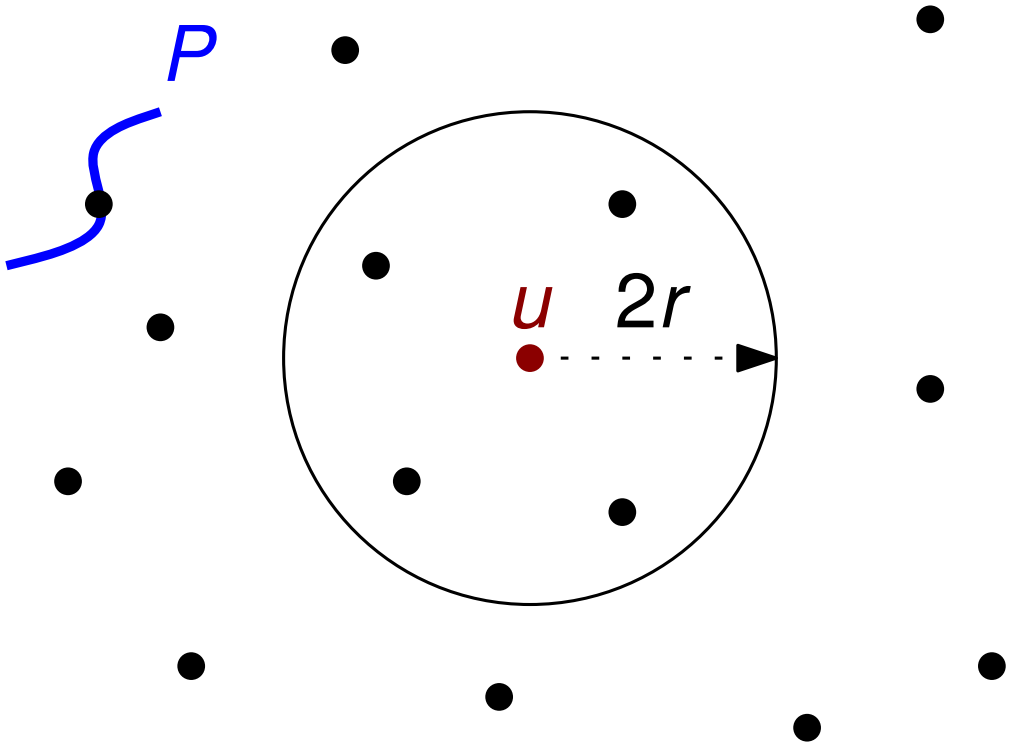
10. Sitzung, Sommersemester 2010

Thomas Pajor | 14. Juni 2010

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

Letztes Mal

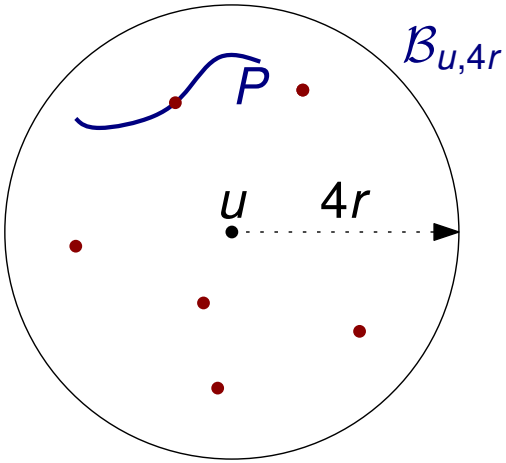
Highway-Dimension und Shortest-Path Cover



Highway Dimension

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

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



Definition

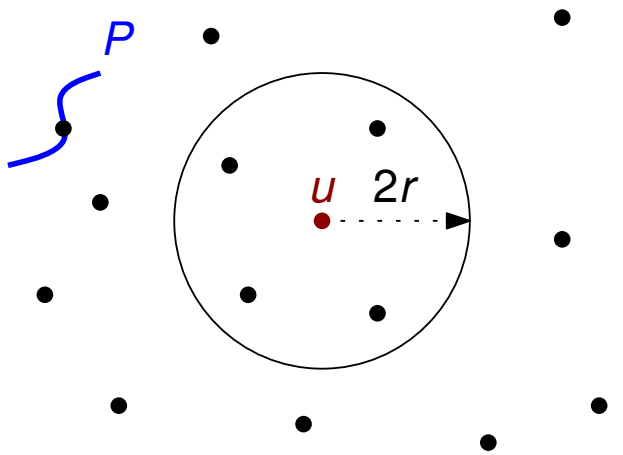
Die *Highway-Dimension* von G ist die kleinste Zahl $h \in \mathbb{N}$, so dass

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

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

Shortest-Path Cover (SPC)

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



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

Definition

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

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

Highway-Dimension und SPCs

Theorem

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

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

Vermutung: Straßennetzwerke haben kleine (konstante) Highway-Dimension

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

Theorem

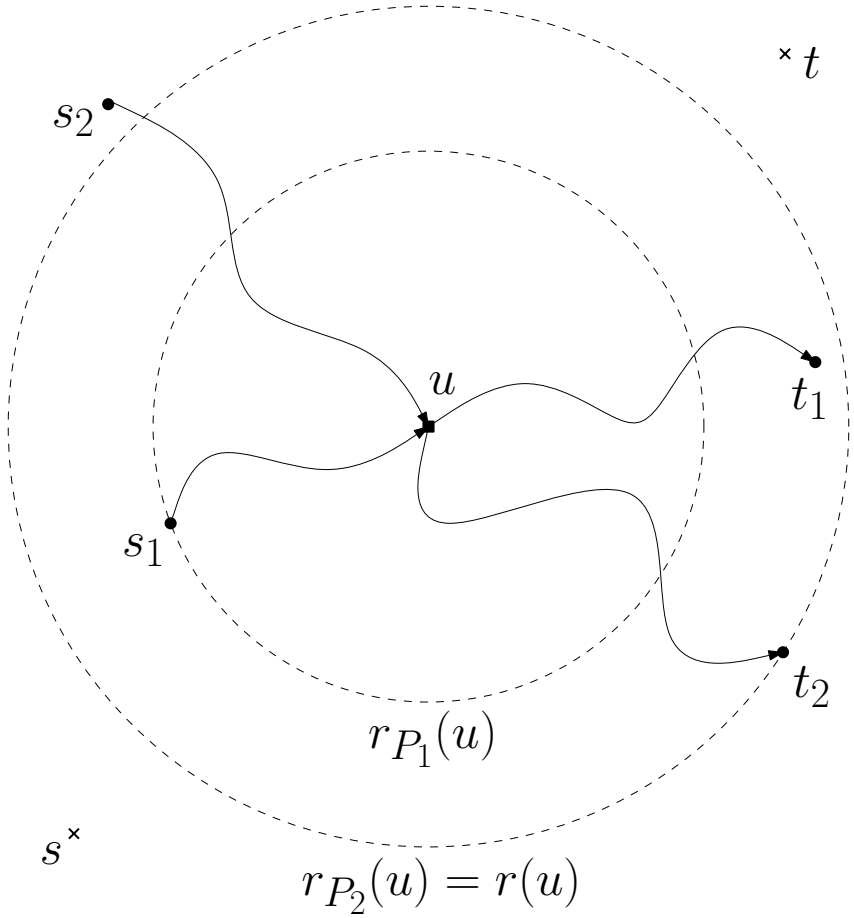
Für jeden Graphen G mit Highway Dimension h gibt es eine Knotenordnung, durch die das CH-Preprocessing eine Menge von Shortcuts E^+ so erzeugt, dass gilt

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

1. Reach



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:

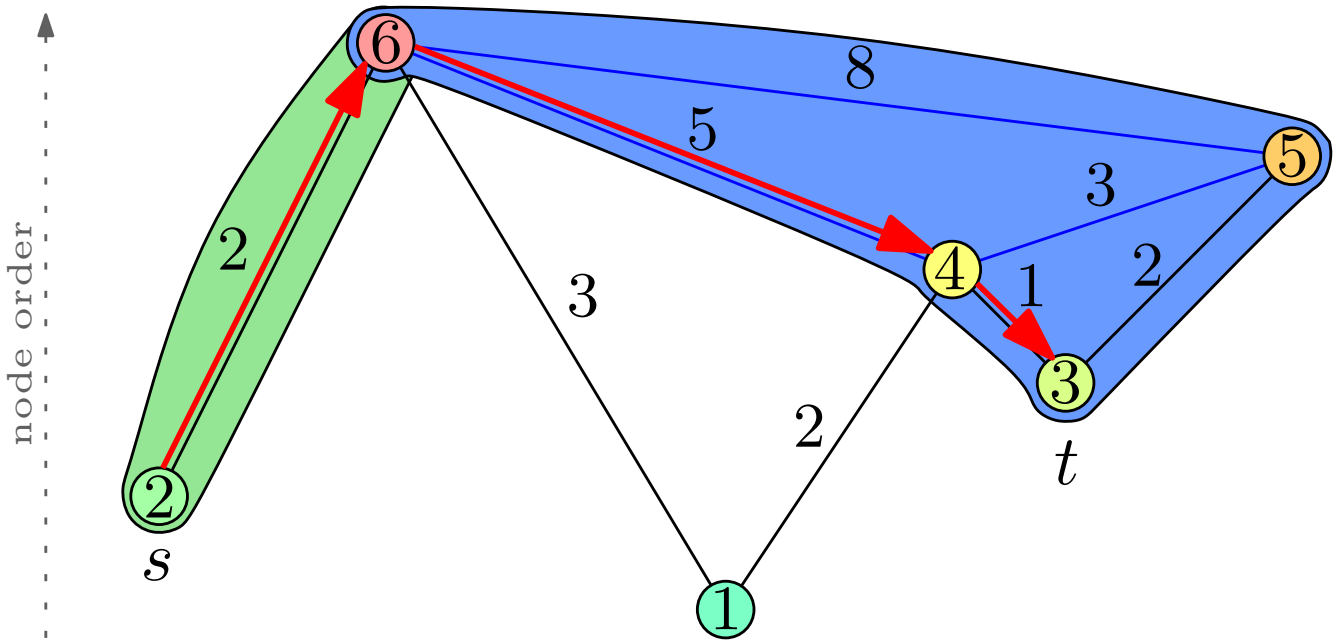
$$\underbrace{(\Delta + k \log D)}_{\text{max. Knotengrad in } G^+} \cdot \underbrace{(k \log D)}_{\text{max. Suchraum}}$$

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.

- Betrachte Knoten $v \in L_i$ und KW $P(s, v)$ mit $|P(s, v)| > 2 \cdot 2^i$.

$\Rightarrow$ $\exists u \in L_j$ mit $j > i$ auf $P(s, v)$.

$\Rightarrow$ $\text{rank}(u) > \text{rank}(v)$

$\Rightarrow$ Teilweg $P(u, v)$ wird nicht betrachtet.

ABER (bei CH):
Nicht jeder Weg im "KW"-Baum ist KW!

$\Rightarrow$ Alternativer Weg $P'(s, v)$ könnte gefunden werden.

$\Rightarrow$ Mehr als k Knoten pro Level können abgearbeitet werden.

CH: Query

Problem:

Reach-Schranken gelten nicht für CH-Query.

- Betrachte Knoten $v \in L_i$ und KW $P(s, v)$ mit $|P(s, v)| > 2 \cdot 2^i$.

$\Rightarrow$ $\exists u \in L_j$ mit $j > i$ auf $P(s, v)$.

$\Rightarrow$ $\text{rank}(u) > \text{rank}(v)$

$\Rightarrow$ Teilweg $P(u, v)$ wird nicht betrachtet.

ABER (bei CH):

Nicht jeder Weg im "KW"-Baum ist KW!

$\Rightarrow$ Alternativer Weg $P'(s, v)$ könnte gefunden werden.

$\Rightarrow$ Mehr als k Knoten pro Level können abgearbeitet werden.

CH: Query

Problem:

Reach-Schranken gelten nicht für CH-Query.

- Betrachte Knoten $v \in L_i$ und KW $P(s, v)$ mit $|P(s, v)| > 2 \cdot 2^i$.

$\Rightarrow$ $\exists u \in L_j$ mit $j > i$ auf $P(s, v)$.

$\Rightarrow$ $\text{rank}(u) > \text{rank}(v)$

$\Rightarrow$ Teilweg $P(u, v)$ wird nicht betrachtet.

ABER (bei CH):
Nicht jeder Weg im “KW”-Baum ist KW!

$\Rightarrow$ Alternativer Weg $P'(s, v)$ könnte gefunden werden.

$\Rightarrow$ Mehr als k Knoten pro Level können abgearbeitet werden.

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

CH: Query

Lösungsansätze:

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

Hier letzteres:

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

CH: Analyse

Beobachtungen:

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

Theorem

Die Query-Zeit von CH ist in

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

CH: Analyse

Beobachtungen:

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

Theorem

Die Query-Zeit von CH ist in

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

TNR und SHARC

Mit ähnlichen Analysen erhält man...

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

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

TNR und SHARC

Mit ähnlichen Analysen erhält man...

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)

Zeitabhängige Routenplanung

Zeitabhängiges Szenario

Szenario:

- Historische Daten für Verkehrssituation verfügbar
- Verkehrssituation vorhersagbar
- Berechne schnellsten Weg bezüglich der erwarteten Verkehrssituation (zu einem gegebenen Startzeitpunkt)

Beobachtung:

- Kein konzeptioneller Unterschied zu Public Transport
- Somit (eventuell) Techniken übertragbar

Herausforderung

Hauptproblem:

- Kürzester Weg hängt von Abfahrtszeitpunkt ab
- Eingabegröße steigt massiv an

Vorgehen:

- Modellierung
- Anpassung Dijkstra
- Anpassung der Basismodule für Beschleunigungstechniken

Heute:

- Modellierung und Dijkstra

Herausforderung

Hauptproblem:

- Kürzester Weg hängt von Abfahrtszeitpunkt ab
- Eingabegröße steigt massiv an

Vorgehen:

- Modellierung
- Anpassung Dijkstra
- Anpassung der Basismodule für Beschleunigungstechniken

Heute:

- Modellierung und Dijkstra

Herausforderung

Hauptproblem:

- Kürzester Weg hängt von Abfahrtszeitpunkt ab
- Eingabegröße steigt massiv an

Vorgehen:

- Modellierung
- Anpassung Dijkstra
- Anpassung der Basismodule für Beschleunigungstechniken

Heute:

- Modellierung und Dijkstra

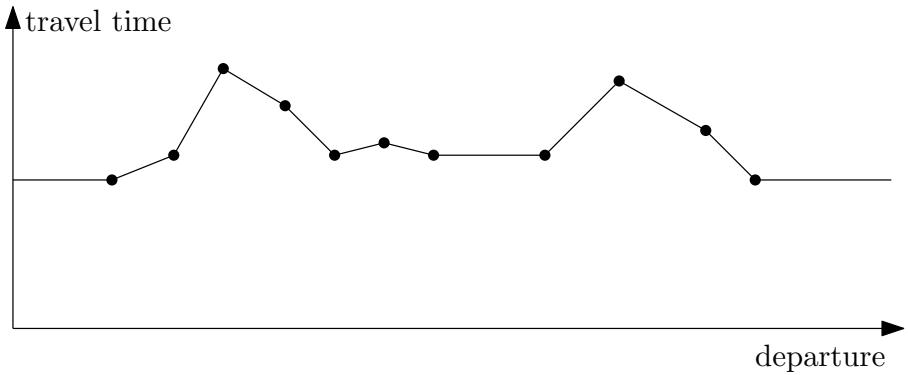
Zeitabhängige Straßennetzwerke

Eingabe:

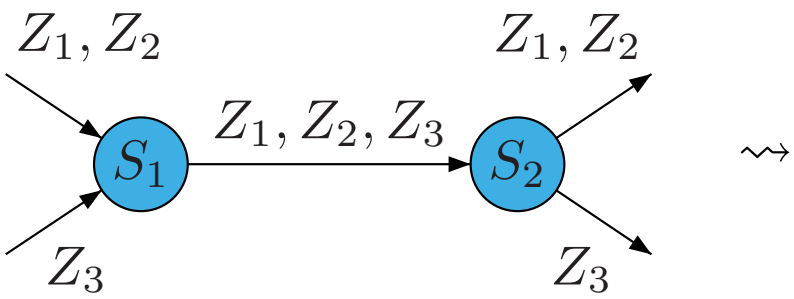
- Durchschnittliche Reisezeit zu bestimmten Zeitpunkten
- Jeden Wochentag verschieden
- Sonderfälle: Urlaubszeit

Somit an jeder Kante:

- Periodische stückweise lineare Funktion
- Definiert durch Stützpunkte
- Interpoliere linear zwischen Stützpunkten



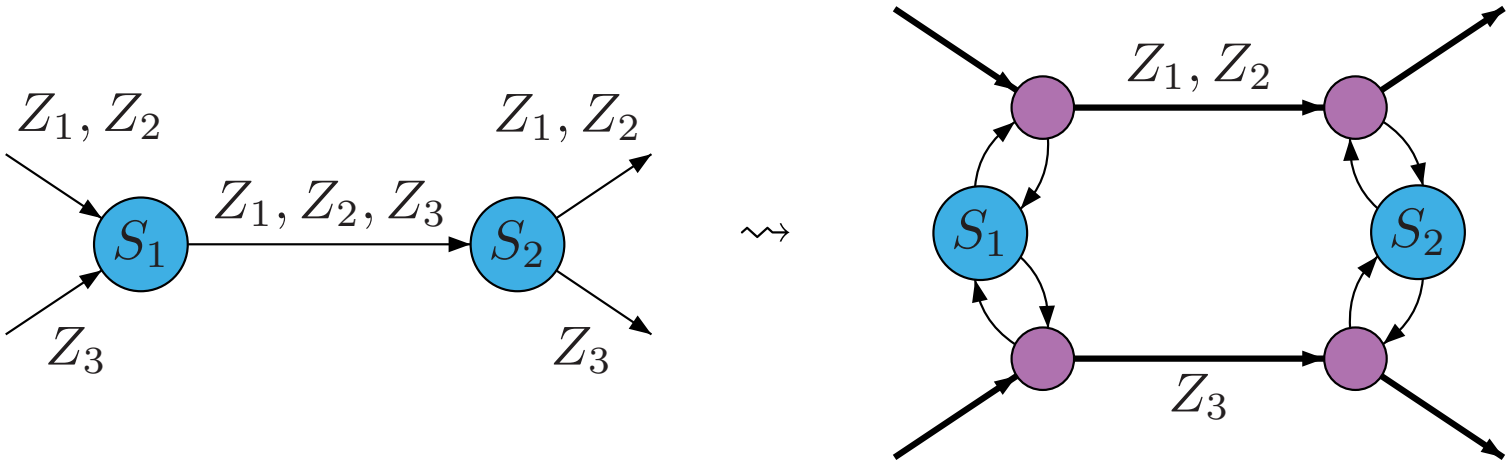
Eisenbahn-Netzwerke



$\rightsquigarrow$

- Teile Züge Z_i in Routen ein
- Für alle Routen, die eine Station bedienen: Ein extra Knoten
  - Modelliert Umstiege zwischen Zügen
  - Mindesttransferzeit für jede Station: an Stations-Kanten

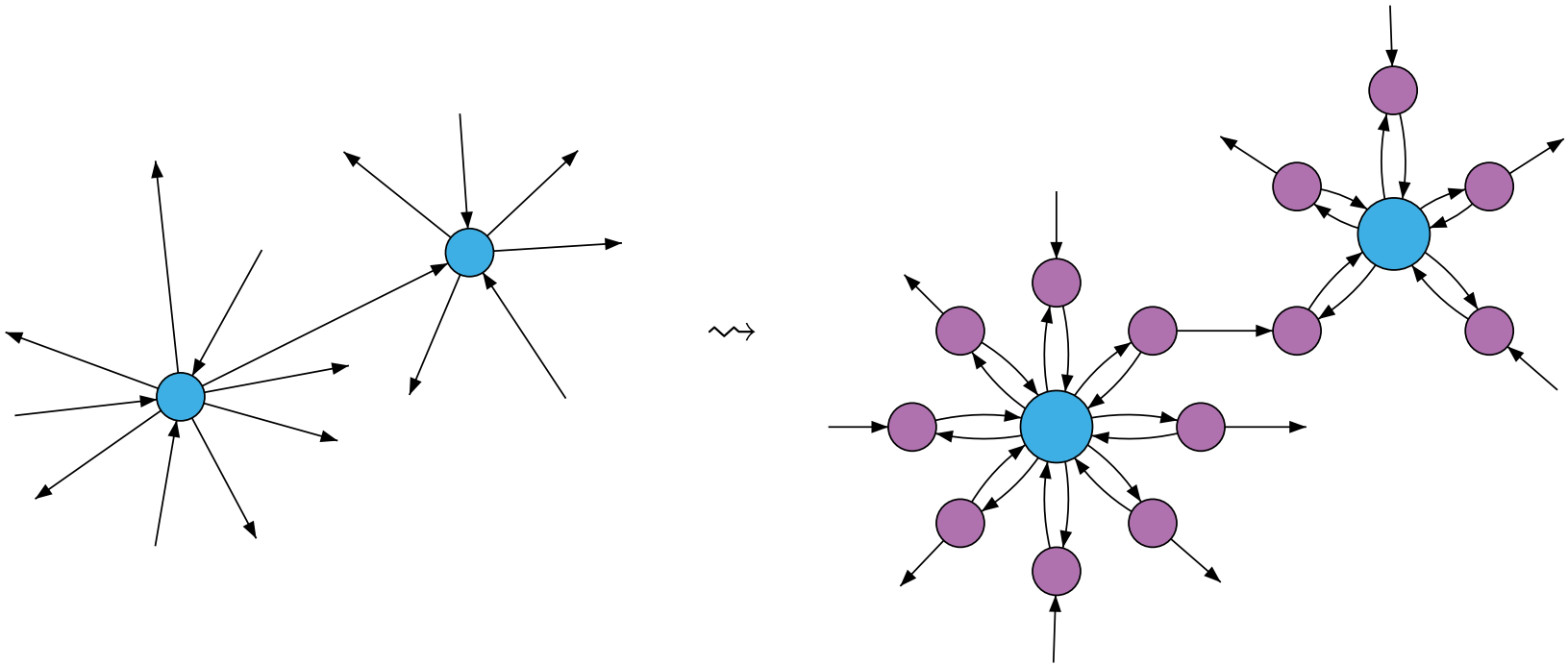
Eisenbahn-Netzwerke



- Teile Züge Z_i in Routen ein
- Für alle Routen, die eine Station bedienen: Ein extra Knoten
  - Modelliert Umstiege zwischen Zügen
  - Mindesttransferzeit für jede Station: an Stations-Kanten

Flugnetzwerke

Nutzung des Eisenbahnansatzes nicht sinnvoll

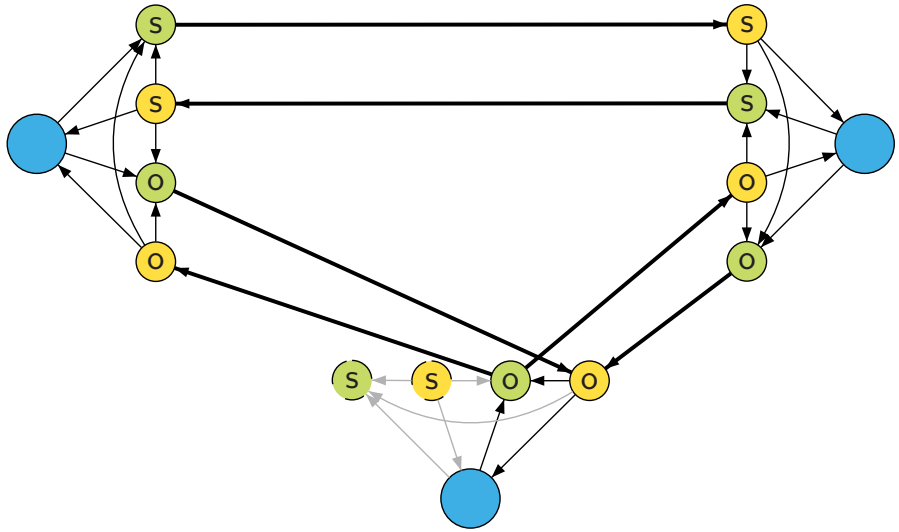


Graphen werden zu groß.

Flugnetzwerke

Modellierung:

- Check-in,
- Check-out,
- Transfer,
- zwischen Allianzen.

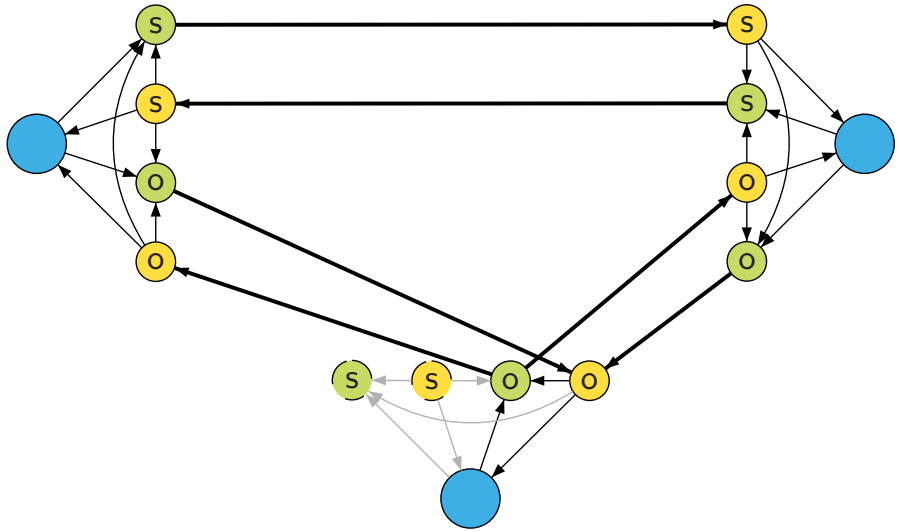


⇒ Zwei Knoten pro Allianz (Abflug und Ankunftsknoten)

Flugnetzwerke

Modellierung:

- Check-in,
- Check-out,
- Transfer,
- zwischen Allianzen.



$\Rightarrow$ Zwei Knoten pro Allianz (Abflug und Ankunftsknoten)

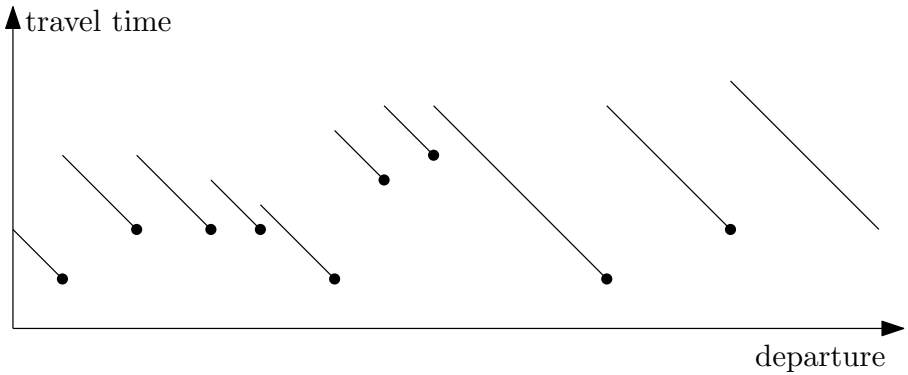
Public Transport Funktionen

Eingabe:

- Reisezeit zu bestimmten Zeitpunkten (Fahrzeiten der Züge)
- Jeden Tag verschieden

Somit:

- Periodische stückweise lineare Funktionen
- Definiert durch Stützpunkte
- Wartezeit zur nächsten Verbindung + Reisezeit



FIFO-Eigenschaft

Definition

Sei $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ eine Funktion. f erfüllt die *FIFO-Eigenschaft*, wenn für jedes $\varepsilon > 0$ und alle $\tau \in \mathbb{R}_0^+$ gilt, dass

$$f(\tau) \leq \varepsilon + f(\tau + \varepsilon).$$

Diskussion

- Interpretation: “Warten lohnt sich nie”
- Kürzeste Wege auf Graphen mit non-FIFO Funktionen zu finden ist NP-schwer.
  (wenn warten an Knoten nicht erlaubt ist)

  $\Rightarrow$ Sicherstellen, dass Funktionen FIFO-Eigenschaft erfüllen.

Diskussion

Eigenschaften:

- Topologie ändert sich nicht
- Kanten gemischt zeitabhängig und konstant
- variable (!) Anzahl Interpolationspunkte pro Kante

Beobachtungen:

- FIFO gilt auf allen Kanten
- später wichtig

Diskussion

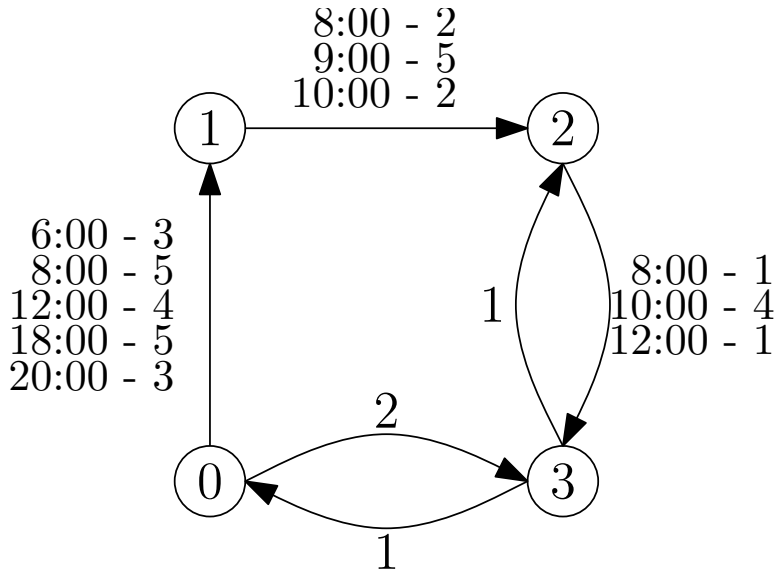
Eigenschaften:

- Topologie ändert sich nicht
- Kanten gemischt zeitabhängig und konstant
- variable (!) Anzahl Interpolationspunkte pro Kante

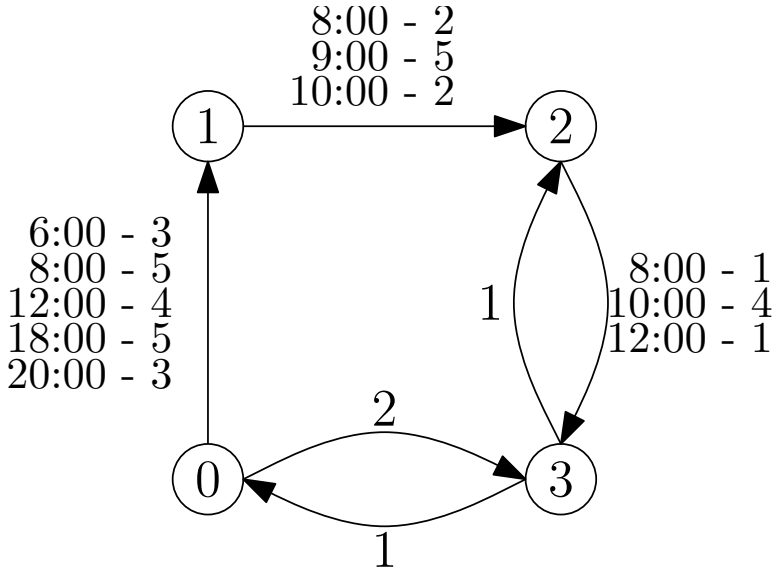
Beobachtungen:

- FIFO gilt auf allen Kanten
- später wichtig

Datenstruktur



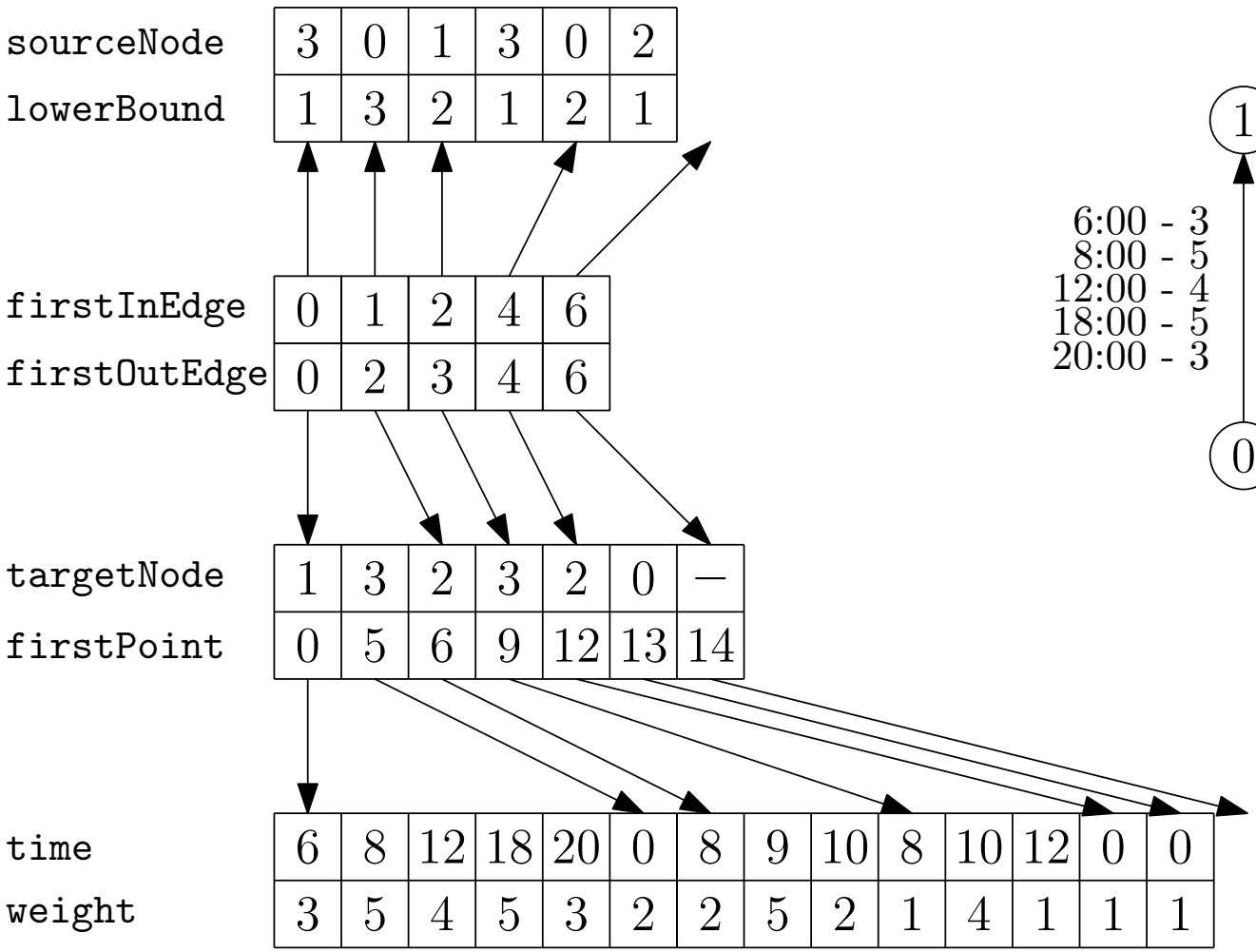
Datenstruktur

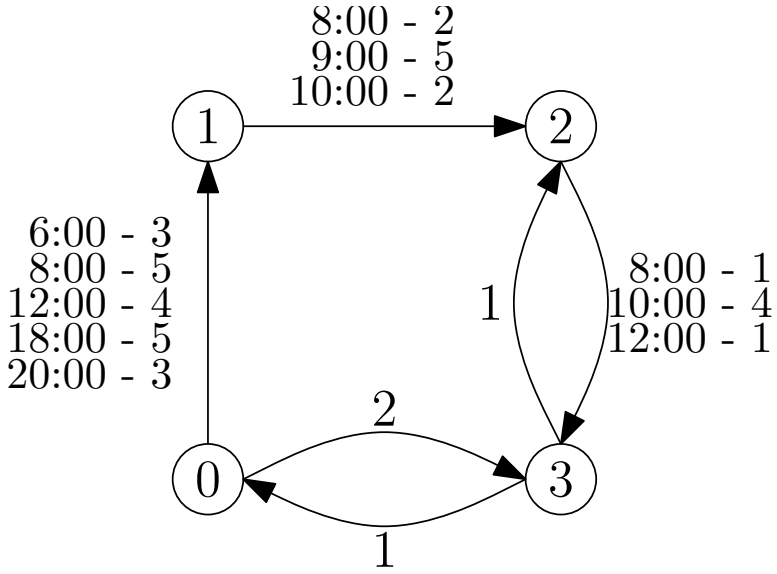


firstEdge	0	2	3	4	6		
targetNode	1	3	2	3	2	0	–
firstPoint	0	5	6	9	12	13	14

time	6	8	12	18	20	0	8	9	10	8	10	12	0	0
weight	3	5	4	5	3	2	2	5	2	1	4	1	1	1

Datenstruktur





Anfrageszenarien

Zeit-Anfrage:

- finde kürzesten Weg für Abfahrtszeit τ
- analog zu Dijkstra?

Profil-Anfrage:

- finde kürzesten Weg für alle Abfahrtszeitpunkte
- analog zu Dijkstra?

Anfrageszenarien

Zeit-Anfrage:

- finde kürzesten Weg für Abfahrtszeit τ
- analog zu Dijkstra?

Profil-Anfrage:

- finde kürzesten Weg für alle Abfahrtszeitpunkte
- analog zu Dijkstra?

Zeit-Anfragen

Karlsruher Institut für Technologie

`Time-Dijkstra`$(G = (V, E), s, \tau)$

```
1  d_τ[s] = 0
2  Q.clear(), Q.add(s, 0)
3  while !Q.empty() do
4      u ← Q.deleteMin()
5      for all edges e = (u, v) ∈ E do
6          if d_τ[u] + len(e, τ + d_τ[u]) < d_τ[v] then
7              d_τ[v] ← d_τ[u] + len(e, τ + d_τ[u])
8              p_τ[v] ← u
9              if v ∈ Q then Q.decreaseKey(v, d_τ[v])
10             else Q.insert(v, d_τ[v])
```

Diskussion Zeit-Anfragen

Beobachtung:

- Nur ein Unterschied zu Dijkstra
- Auswertung der Kanten

non-FIFO Netzwerke:

- Im Kreis fahren kann sich lohnen
- NP-schwer (wenn warten an Knoten nicht erlaubt ist)
- Transportnetzwerke sind FIFO modellierbar (notfalls Multikanten)

Diskussion Zeit-Anfragen

Beobachtung:

- Nur ein Unterschied zu Dijkstra
- Auswertung der Kanten

non-FIFO Netzwerke:

- Im Kreis fahren kann sich lohnen
- NP-schwer (wenn warten an Knoten nicht erlaubt ist)
- Transportnetzwerke sind FIFO modellierbar (notfalls Multikanten)

Ende

Literatur (Highway-Dimension):

- 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`

Ende

Literatur (Zeitabhängige Routenplanung):

- Daniel Delling:
  Enginering and Augmenting Route Planning Algorithms
  Ph.D. Thesis, Universität Karlsruhe (TH), 2009.

Username: `routePlanning` Passwort: `ss10`