Karlsruher Institut für Technologie

Algorithmen für Routenplanung

2. Vorlesung, Sommersemester 2016

Tobias Zündorf | 25. April 2016

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

Organisatorisches

Vorlesung

- Montags 14:00–15:30 Uhr, SR 301 (hier)
- Mittwochs 11:30–13:00 Uhr, SR 301 (hier)

Prüfung

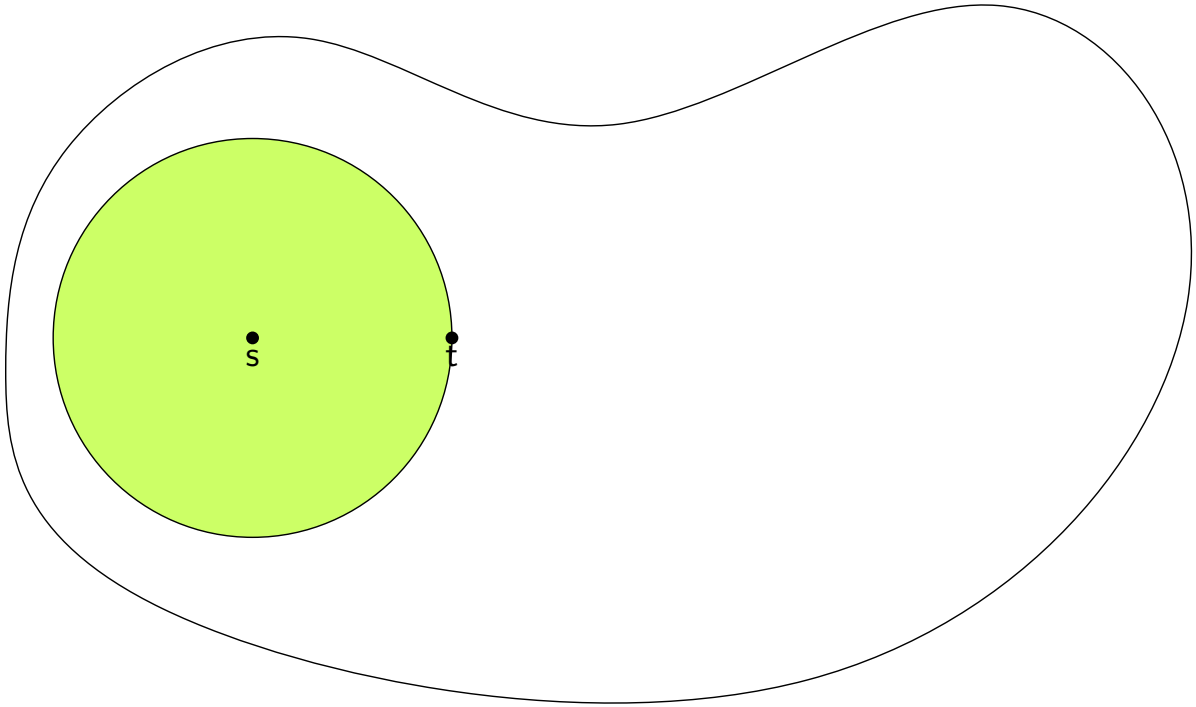
- Prüfbar im Master (und Hauptstudium Diplom)
- Im Master: 5 ECTS Leistungspunkte
- VF: 1 (Theoretische Grundlagen), 2 (Algorithmentechnik)

Vorlesungswebseite:
`http://i11www.iti.kit.edu/teaching/sommer2016/routenplanung/index`

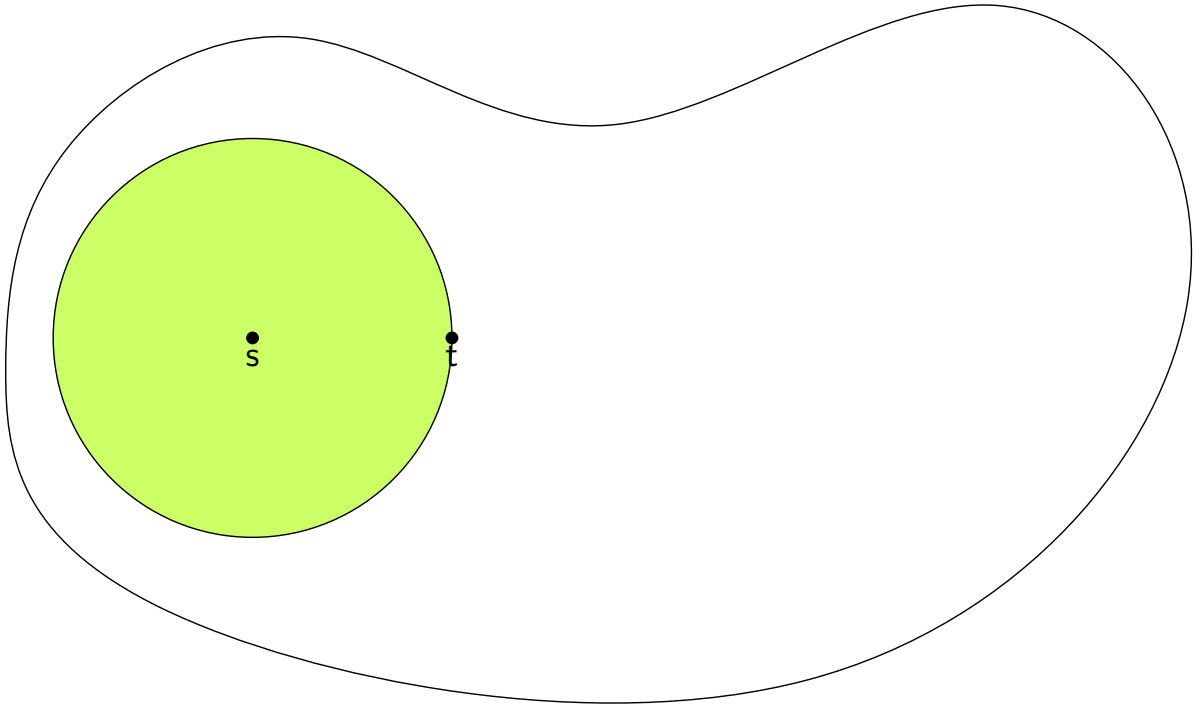
Nachtrag – Bidirektionale Suche

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Nachtrag – Bidirektionale Suche



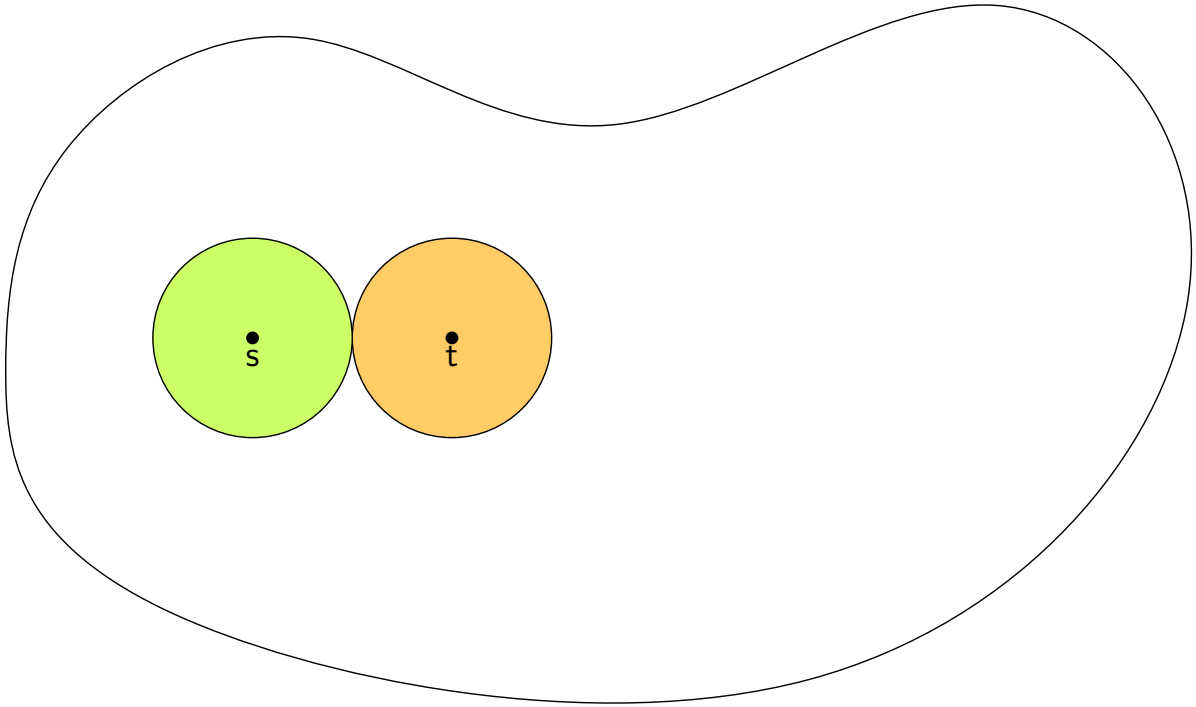
Nachtrag – Bidirektionale Suche



Beobachtung: Ein kürzester s-t-Weg lässt sich finden durch

- Normaler Dijkstra (Vorwärtssuche) von s
- Dijkstra auf Graph mit rückwärts Kanten von t

Nachtrag – Bidirektionale Suche



Idee: Kombiniere beide Suchen

- „Gleichzeitig“ Vor- und Rückwärtssuche
- Abbruch wenn beide Suchen „weit genug fortgeschritten“
- Weg dann zusammensetzen

Abbruchstrategie (1)

Abbruchstrategie (1)

Abbruch, sobald ein Knoten v existiert, der von beiden Suchen abgearbeitet wurde.

Abbruchstrategie (1)

Abbruchstrategie (1)

Abbruch, sobald ein Knoten v existiert, der von beiden Suchen abgearbeitet wurde.

Abbruchstrategie (1) berechnet $d(s, t)$ korrekt. Beweisskizze:

- O.B.d.A: Sei $d(s, t) < \infty$ (andernfalls klar)
- Klar: $\overrightarrow{d}[v] + \overleftarrow{d}[v] \geq \text{dist}(s, t)$
- Seien $\overrightarrow{S}, \overleftarrow{S}$ die abgearbeiteten Knoten von Vor- und Rückwärtssuche nach Terminierung
- Sei $P = (v_1, \ldots v_k)$ ein kürzester s-t-Weg. Wir zeigen:

$$\{v_1, \ldots v_k\} \subseteq \overrightarrow{S} \cup \overleftarrow{S} \quad \text{oder} \quad \text{len}(P) = \text{dist}(s, v) + \text{dist}(v, t)$$

Abbruchstrategie (1)

Abbruchstrategie (1)

Abbruch, sobald ein Knoten v existiert, der von beiden Suchen abgearbeitet wurde.

- Sei $P = (v_1, \dots v_k)$ ein kürzester s-t-Weg. Wir zeigen:

$$\{v_1, \dots v_k\} \subseteq \overrightarrow{S} \cup \overleftarrow{S} \quad \text{oder} \quad \text{len}(P) = \text{dist}(s, v) + \text{dist}(v, t)$$

- Angenommen es gibt $v_i \notin \overrightarrow{S} \cup \overleftarrow{S}$. Dann gilt:

$$\begin{aligned} \text{dist}(s, v_i) &\geq \text{dist}(s, v) \\ \text{dist}(v_i, t) &\geq \text{dist}(v, t) \end{aligned}$$

- Woraus folgt:

$$\text{len}(P) = \text{dist}(s, v_i) + \text{dist}(v_i, t) \geq \text{dist}(s, v) + \text{dist}(v, t)$$

Abbruchstrategie (2)

Abbruchstrategie (2)

Abbruch, sobald $\mu \leq \text{minKey}(\overrightarrow{Q}) + \text{minKey}(\overleftarrow{Q})$

Abbruchstrategie (2)

Abbruchstrategie (2)

Abbruch, sobald $\mu \leq \text{minKey}(\overrightarrow{Q}) + \text{minKey}(\overleftarrow{Q})$

Abbruchstrategie (2) berechnet $d(s, t)$ korrekt. Beweisskizze:

- O.B.d.A: Sei $d(s, t) < \infty$ (andernfalls klar)
- Klar: $\overrightarrow{d}[v] + \overleftarrow{d}[v] \geq \text{dist}(s, t)$

Abbruchstrategie (2)

Abbruchstrategie (2)

Abbruch, sobald $\mu \leq \text{minKey}(\overrightarrow{Q}) + \text{minKey}(\overleftarrow{Q})$

Annahme: $\mu > d(s,t)$ nach Terminierung

- Dann gibt es einen s-t Pfad P der kürzer als μ ist
- Auf P gibt es eine Kante (u,v) mit:

$$d(s,u) \leq \mathit{minKey}(\overrightarrow{Q}) \quad \text{und} \quad d(v,t) \leq \mathit{minKey}(\overleftarrow{Q}).$$

- Also müssen u und v schon abgearbeitet worden sein (o.B.d.A. u vor v)
- Beim relaxieren von (u,v) wäre:
  - P entdeckt worden
  - μ aktualisiert worden

Damit ist $\mu \leq d(s,t)$. Widerspruch!

Wechselstrategien

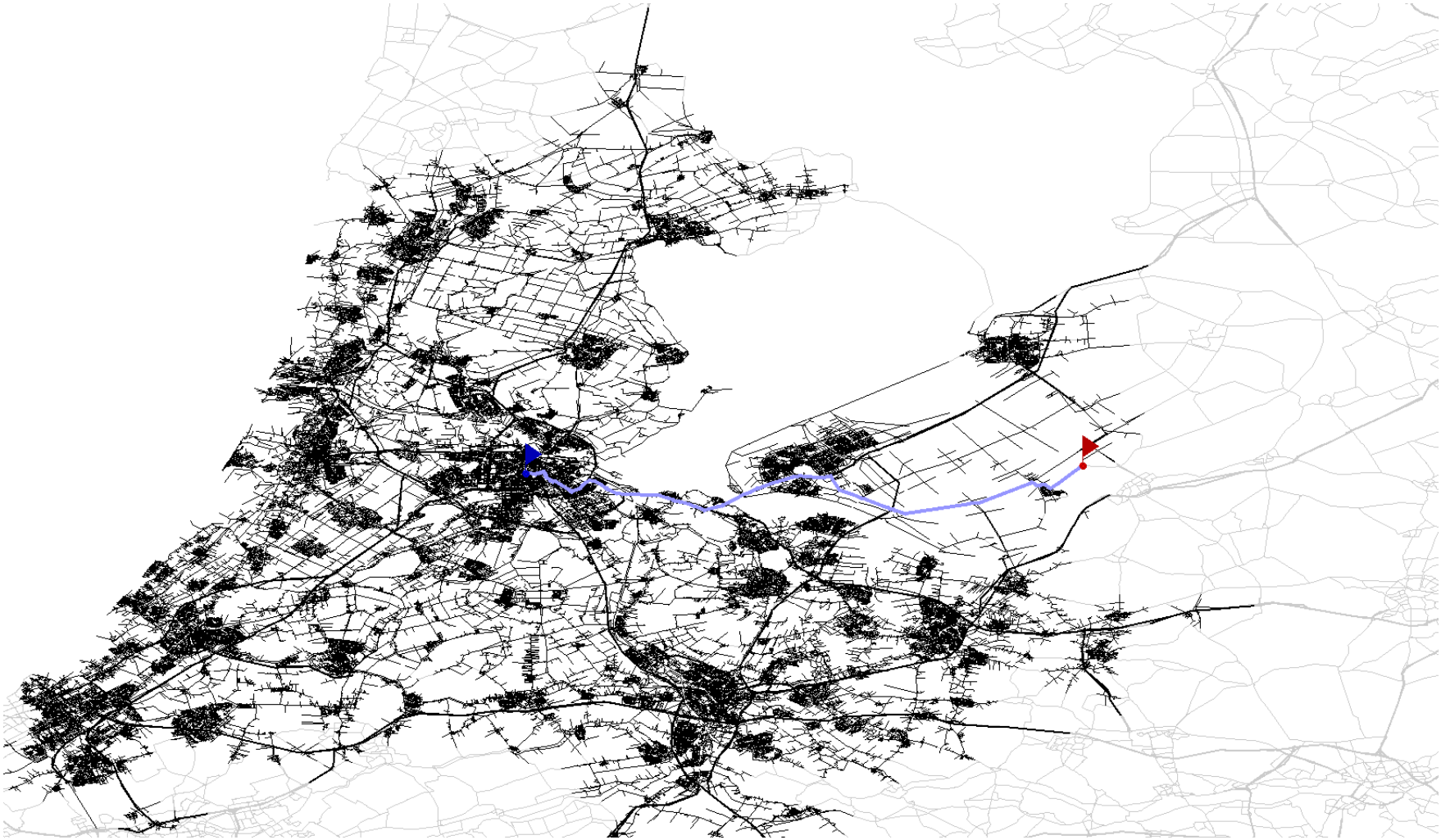
Frage: Was sind mögliche Wechselstrategien?

Wechselstrategien

Mögliche Wechselstrategien

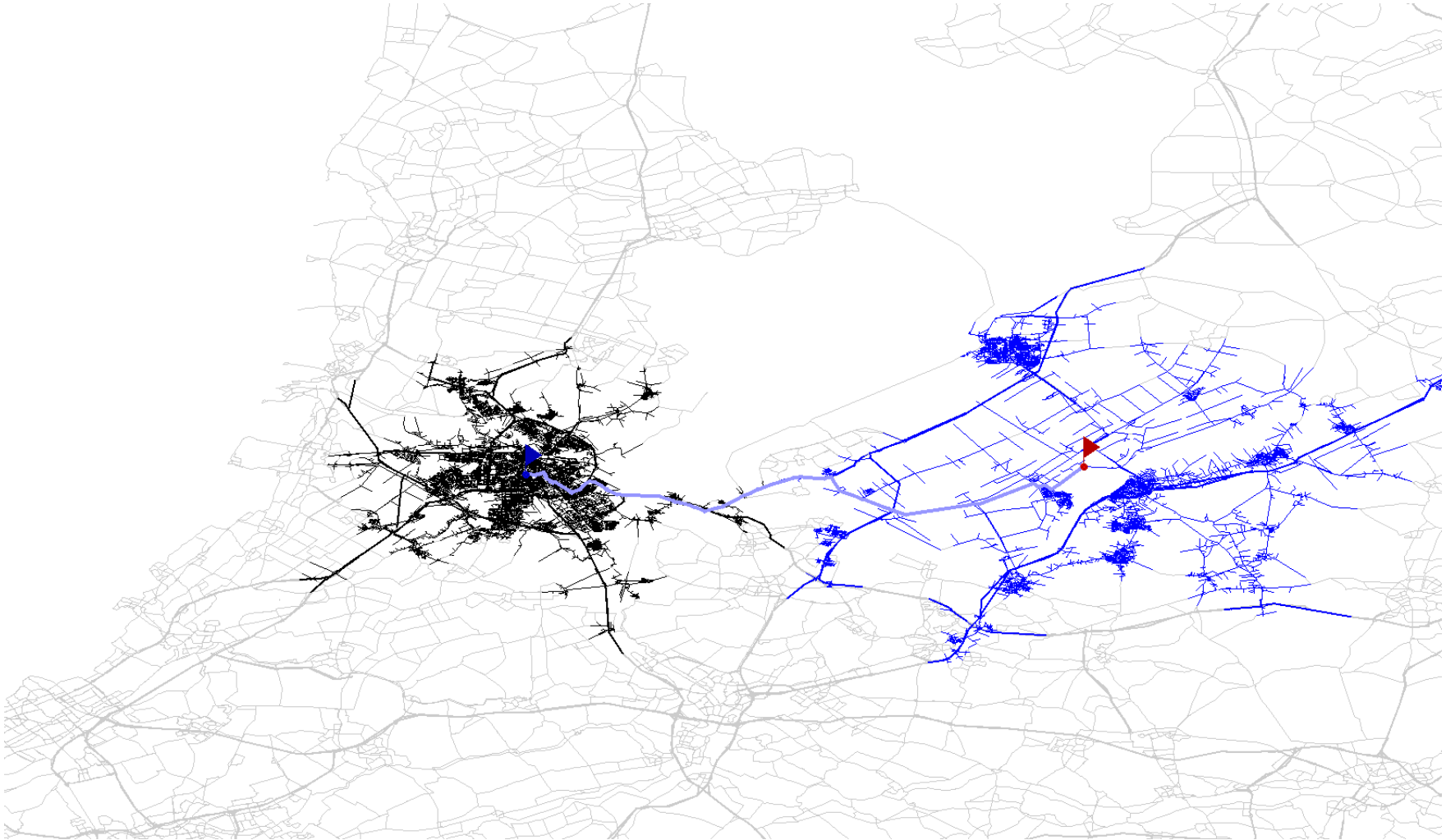
- Prinzipiell jede Wechselstrategie möglich
- Wechsle nach jedem Schritt zur entgegengesetzten Suche
- Wechsel zu Suche mit der weniger Elementen in der Queue
- Wechsel zu Suche mit dem kleineren minimalen Queueelement
- Oder: Parallele Ausführung auf zwei Kernen

Beispiel



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bemerkungen

Beschleunigung

- Annahme: Suchraum ist Kreisscheibe mit Radius r

$\Rightarrow$ Speedup bzgl. Suchraum (ca.):

$$\frac{\text{Dijkstra}}{\text{Bidir. Suche}} \approx \frac{\pi r^2}{2 \cdot \pi \left(\frac{r}{2}\right)^2} = 2$$

- Führe Suchen *parallel* aus

$\Rightarrow$ Gesamtspeedup ca. 4

Bemerkungen

Beschleunigung

- Annahme: Suchraum ist Kreisscheibe mit Radius r

$\Rightarrow$ Speedup bzgl. Suchraum (ca.):

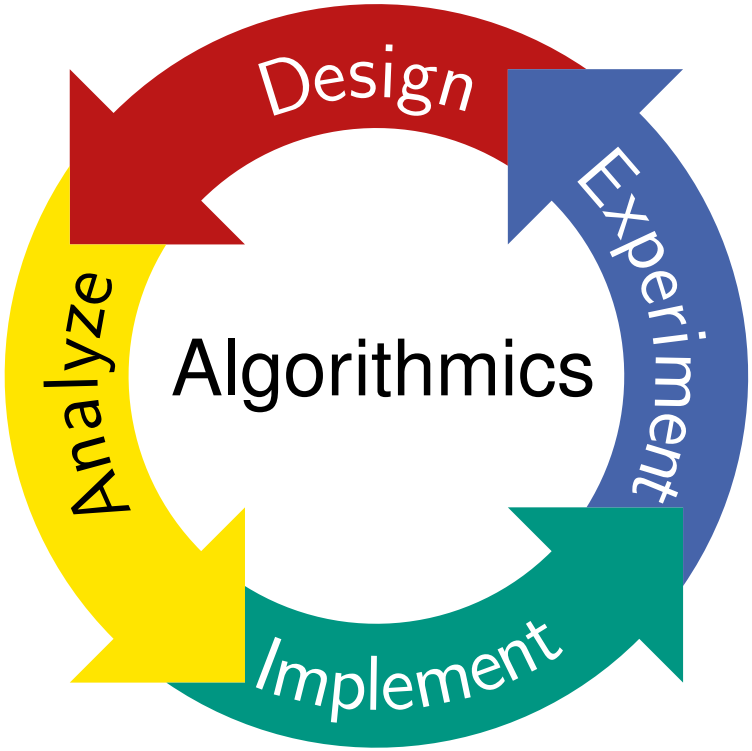
$$\frac{\text{Dijkstra}}{\text{Bidir. Suche}} \approx \frac{\pi r^2}{2 \cdot \pi \left(\frac{r}{2}\right)^2} = 2$$

- Führe Suchen *parallel* aus

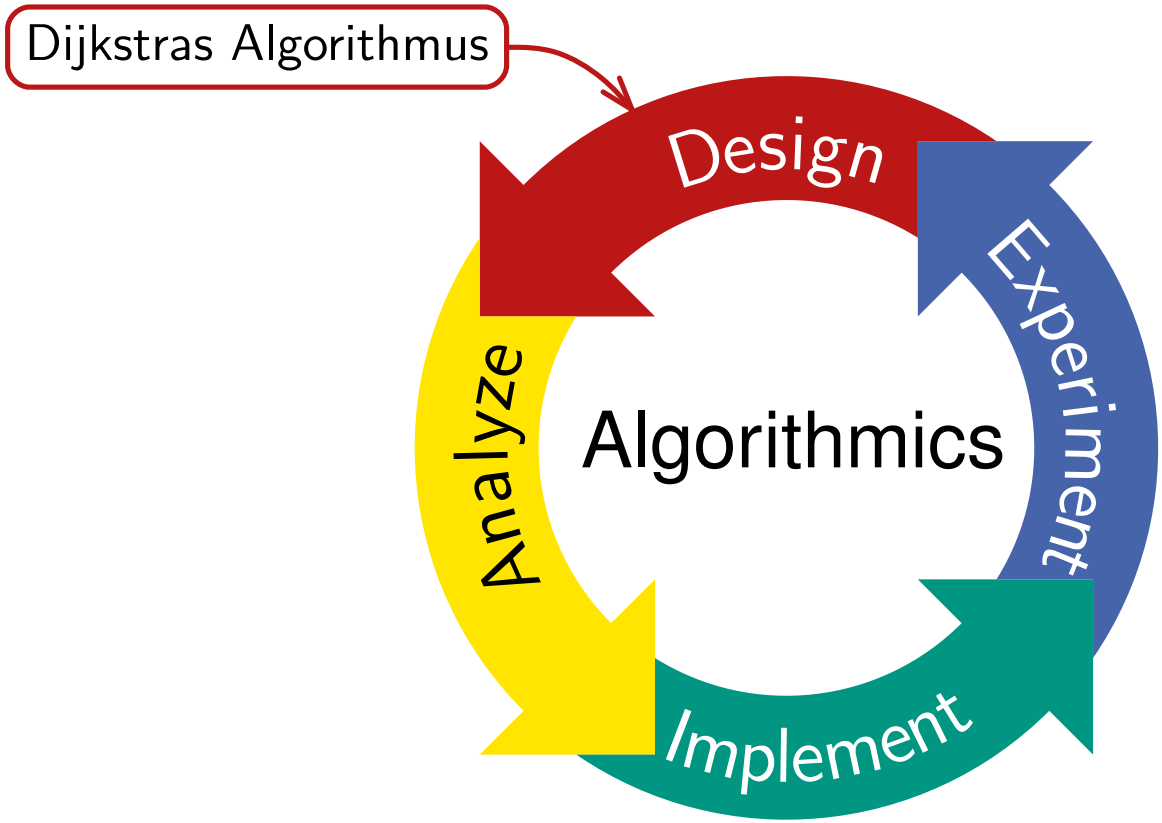
$\Rightarrow$ Gesamtspeedup ca. 4

Wichtiger Bestandteil vieler effizienter Techniken!

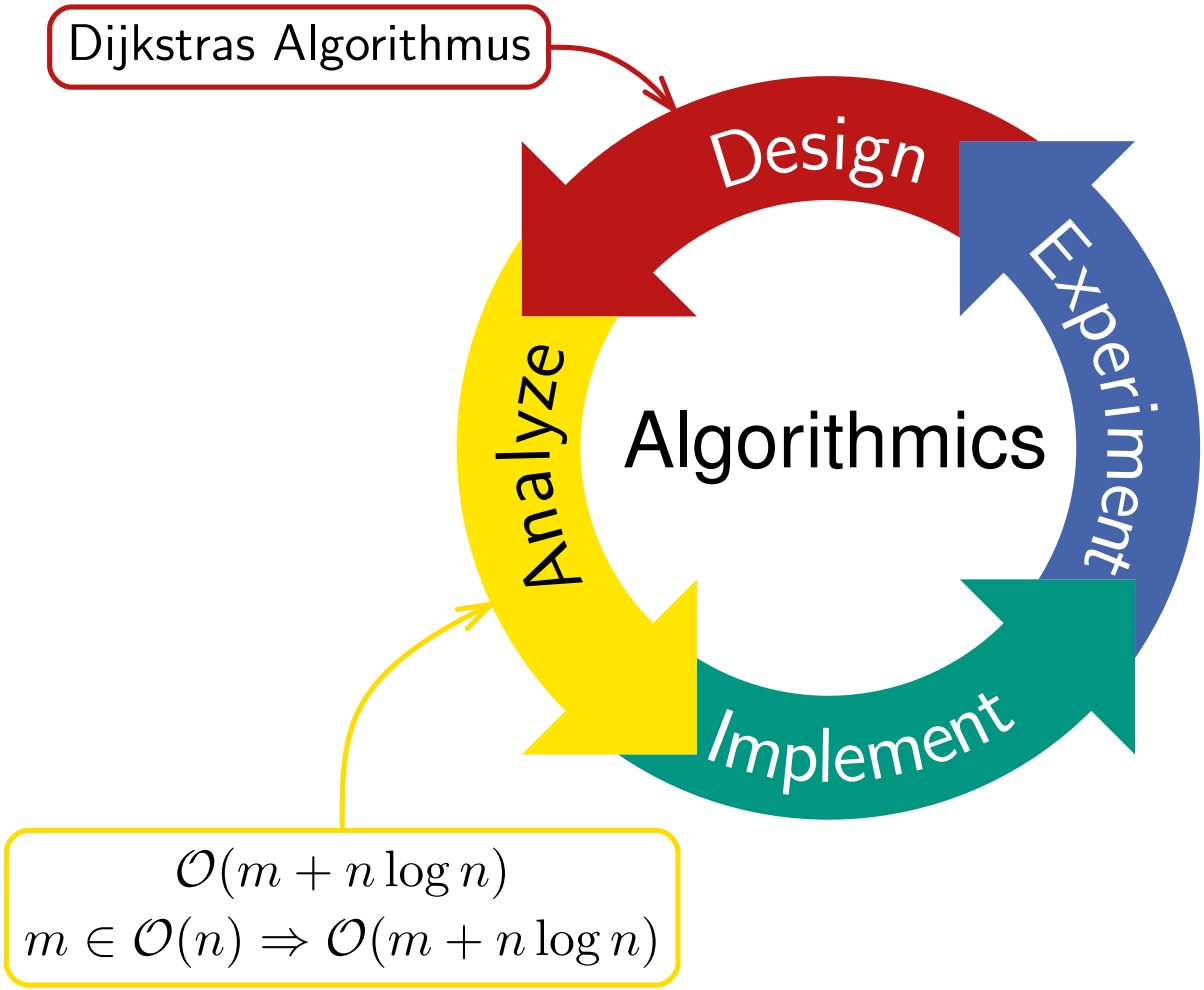
Algorithm Engineering



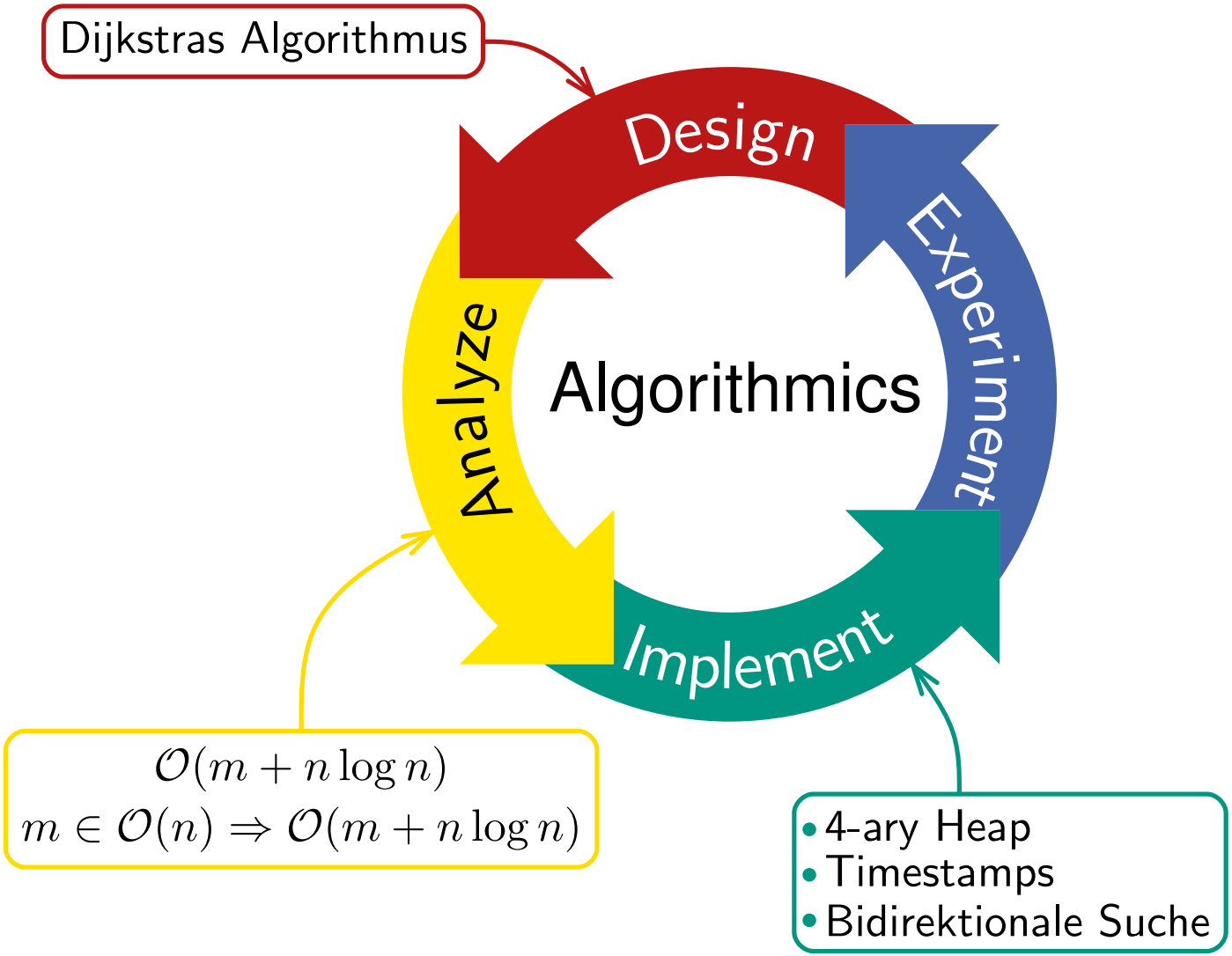
Algorithm Engineering



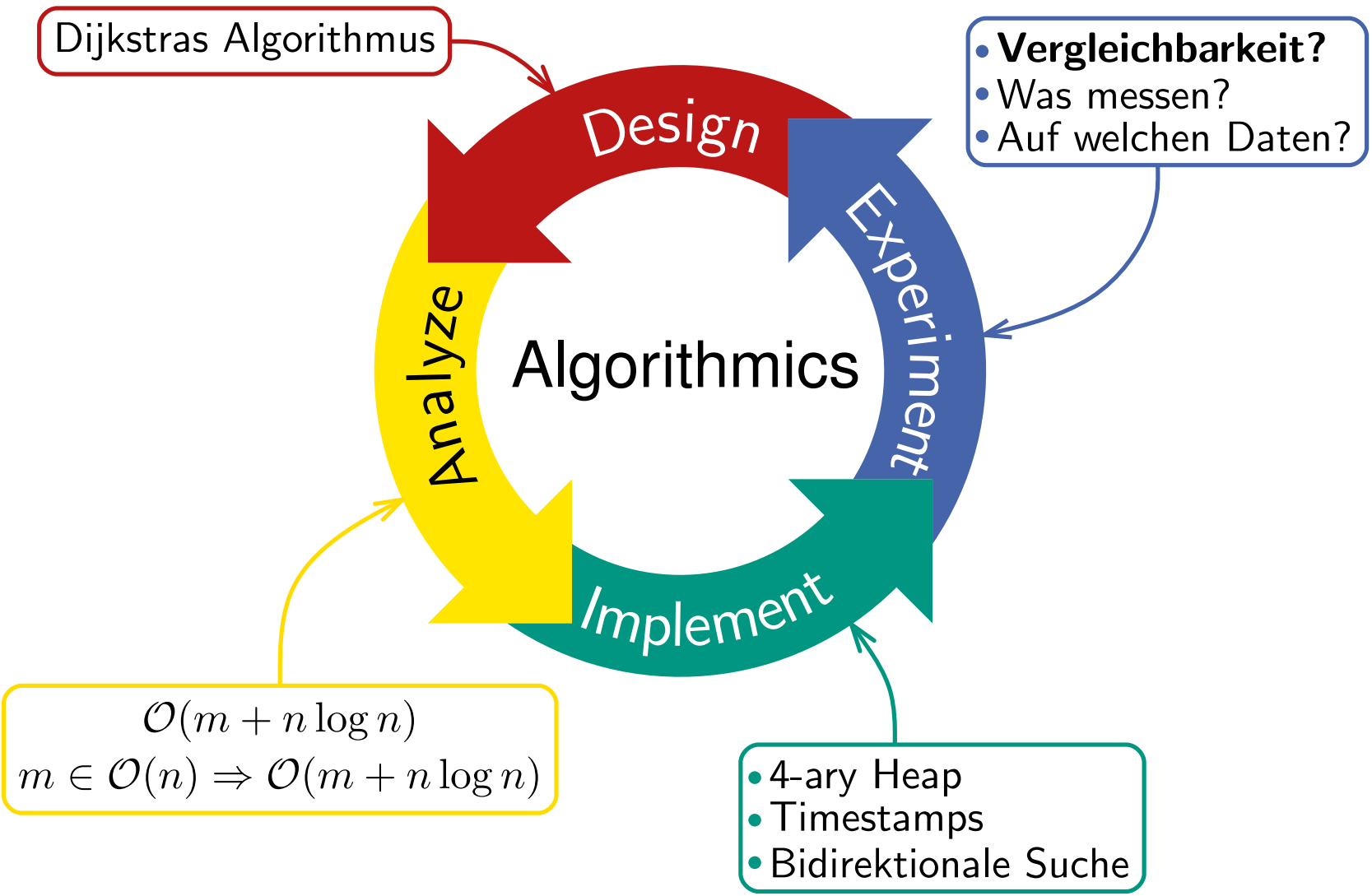
Algorithm Engineering



Algorithm Engineering



Algorithm Engineering



Experiment – Dijkstra-Rank

Problem:

- Zufallsanfragen geben wenig Informationen
- Wie ist die Varianz?
- Werden nahe oder ferne Anfragen beschleunigt?

Experiment – Dijkstra-Rank

Problem:

- Zufallsanfragen geben wenig Informationen
- Wie ist die Varianz?
- Werden nahe oder ferne Anfragen beschleunigt?

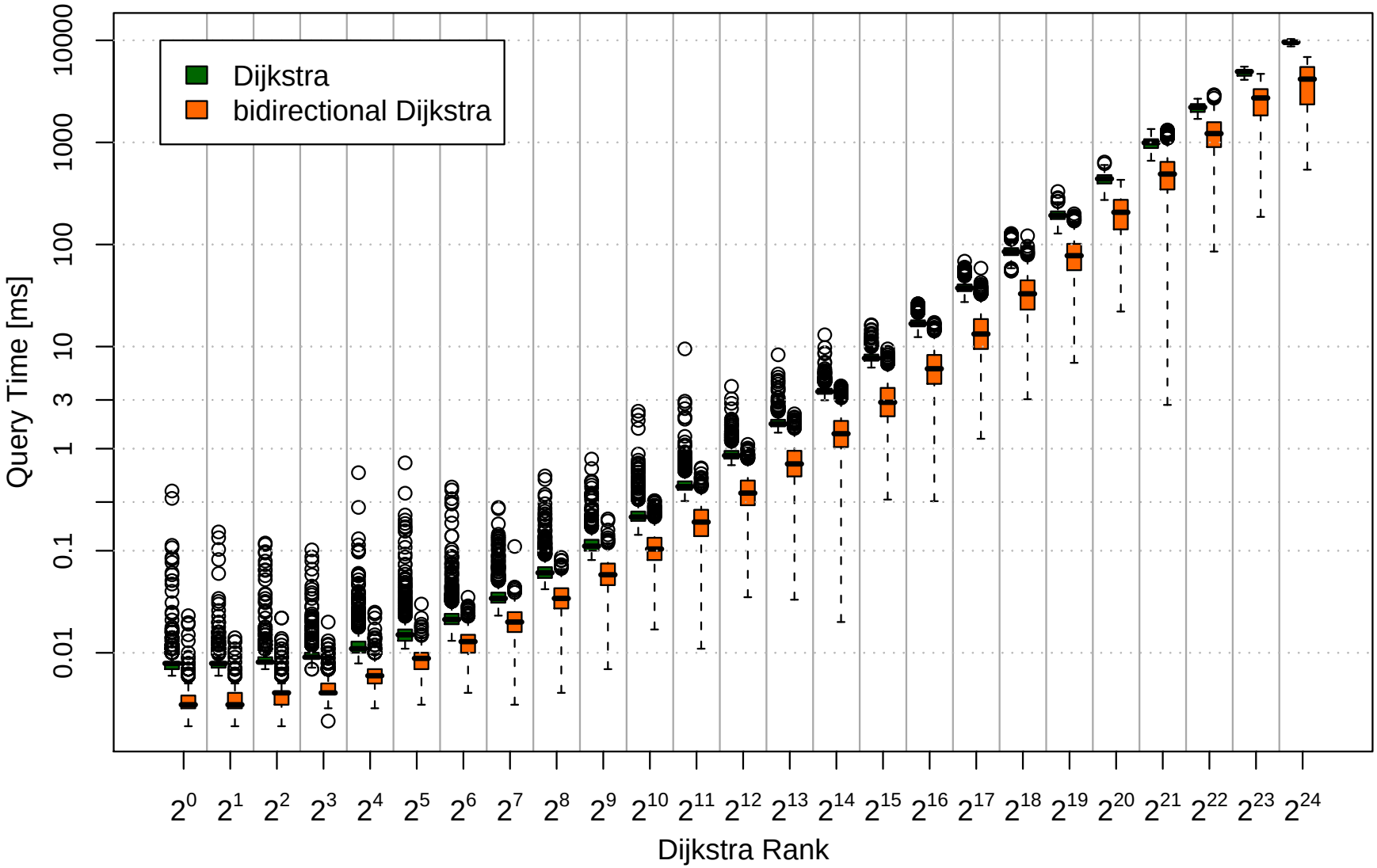
Idee:

- DIJKSTRA definiert für Startknoten s Ordnung für auf den Knoten
- DIJKSTRA-Rang $r_s(u)$ eines Knoten u für gegebenes s
- Wähle 1000 zufällige Startknoten, analysiere jeweils den Suchraum

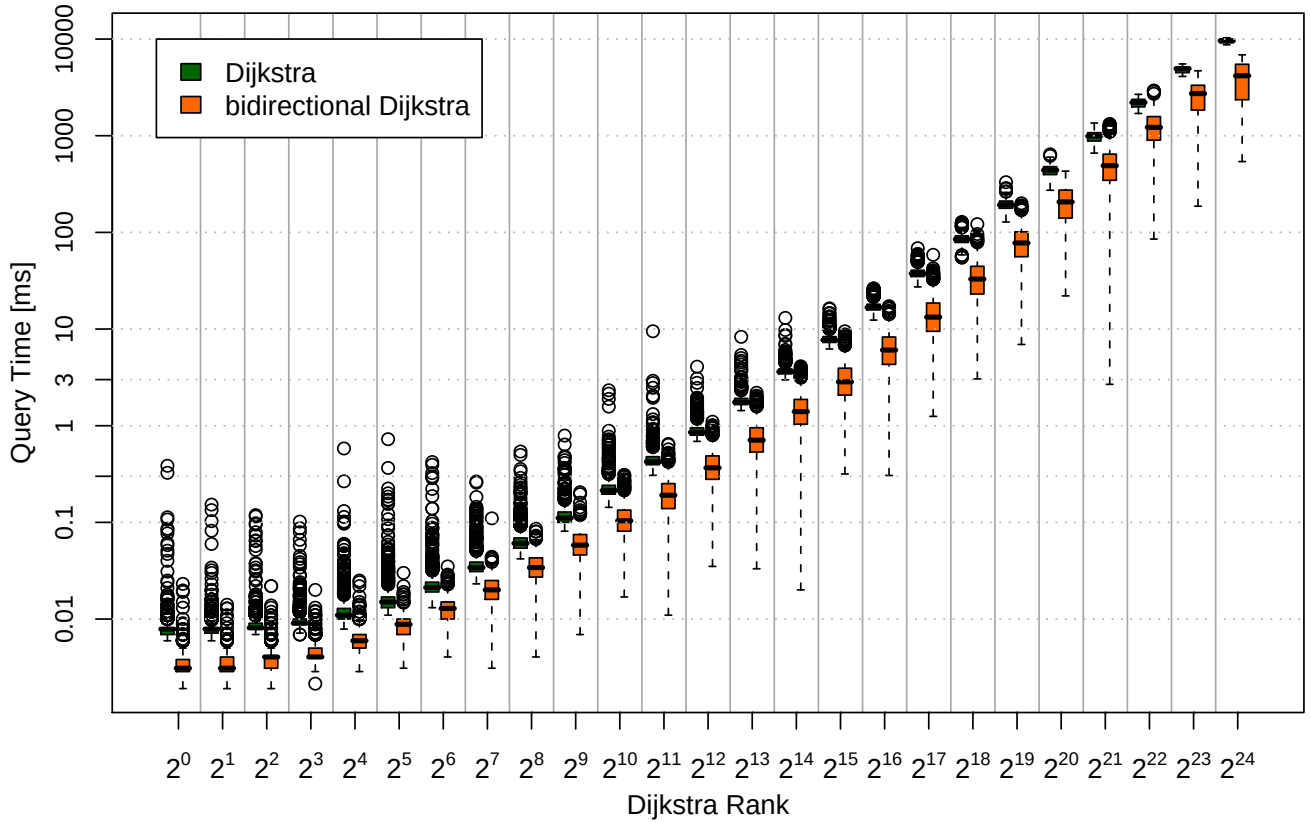
$$\text{Zieknoten } t := \text{ Knoten mit Rang } 2^1, \ldots, 2^{\log n}$$

- Zeichne Plot (x-Achse = Rang, y-Achse = Laufzeit)

Dijkstra-Rank – Bidirektionale Suche



Dijkstra-Rank – Bidirektionale Suche



- Ausreißer bei nahen Anfragen (vor allem unidirektional)
- Beschleunigung unabhängig vom Rang (immer ca. Faktor 2)
- Varianz etwas höher als bei unidirektionaler Suche
- Manche Anfragen sehr schnell

Experiment – Datensatz

Welche Graphen für Experimente nutze:

- Reales Straßennetz
- Synthetische Daten
- Zufallsgraphen

Experiment – Datensatz

Welche Graphen für Experimente nutze:

- Reales Straßennetz
- Synthetische Daten
- Zufallsgraphen

Zielsetzung:

- Anwendungsnähe
- Effizienz auf echten Daten

Experiment – Datensatz

Welche Graphen für Experimente nutze:

- Reales Straßennetz
  - Kartendaten: z.B.: OpenStreetMap
  - Benchmark Instanzen: z.B.: Dimacs Implementation Challenge
- Synthetische Daten
- Zufallsgraphen

Zielsetzung:

- Anwendungsnähe
- Effizienz auf echten Daten

Experiment – Datensatz

Welche Graphen für Experimente nutze:

- Reales Straßennetz
  - Kartendaten: z.B.: OpenStreetMap
  - Benchmark Instanzen: z.B.: Dimacs Implementation Challenge
- Synthetische Daten
- Zufallsgraphen

Zielsetzung:

- Anwendungsnähe
- Effizienz auf echten Daten

Welchen Einfluss haben die Daten auf die Experimente?

OpenStreetMap (OSM)

Straßengraphen extrahiert von OSM:

- \+ Echtes Straßennetz
- \+ Frei verfügbar
- \+ Große Datenmengen ($>$ 173 Mio Knoten für Europa)

OpenStreetMap (OSM)

Straßengraphen extrahiert von OSM:

+ Echtes Straßennetz
+ Frei verfügbar
+ Große Datenmengen ($>$ 173 Mio Knoten für Europa)

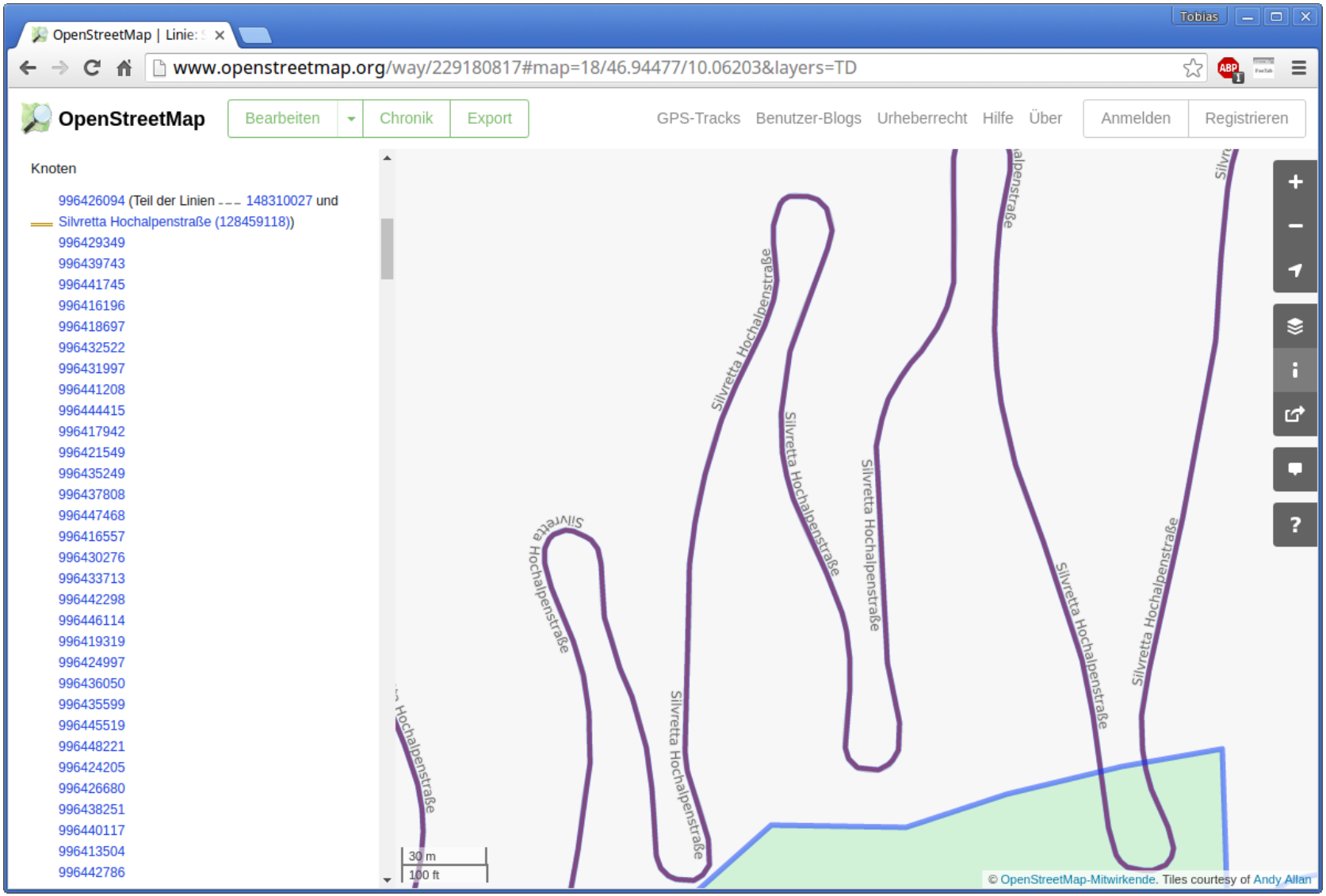
− Zur Darstellung in Karten gedacht
− Viele Freiheiten beim erstellen des Graphen

$\Rightarrow$ Experimente schwer vergleichbar

OpenStreetMap (OSM)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

OpenStreetMap (OSM)



OpenStreetMap (OSM)

OSM Graphen:

- Knoten zum Modellieren des Straßenverlaufs
- Knoten irrelevant für das Routing
- Hat großen Einfluss auf Laufzeit & gemessene Beschleunigung

OpenStreetMap (OSM)

OSM Graphen:

- Knoten zum Modellieren des Straßenverlaufs
- Knoten irrelevant für das Routing
- Hat großen Einfluss auf Laufzeit & gemessene Beschleunigung

Wie Graphen vereinheitlichen?

OpenStreetMap (OSM)

OSM Graphen:

- Knoten zum Modellieren des Straßenverlaufs
- Knoten irrelevant für das Routing
- Hat großen Einfluss auf Laufzeit & gemessene Beschleunigung

Wie Graphen vereinheitlichen?

- Grad zwei Knoten löschen?

OpenStreetMap (OSM)

OSM Graphen:

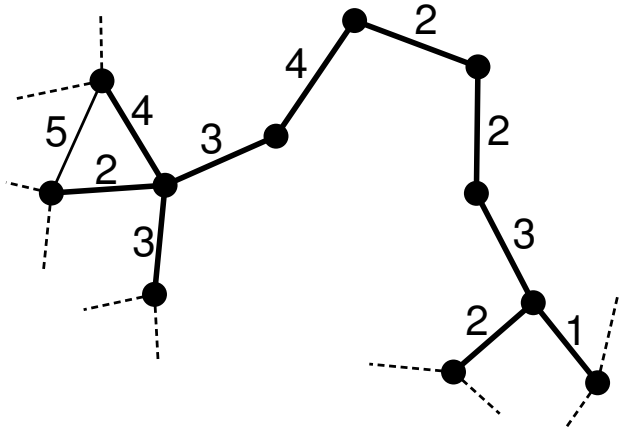
- Knoten zum Modellieren des Straßenverlaufs
- Knoten irrelevant für das Routing
- Hat großen Einfluss auf Laufzeit & gemessene Beschleunigung

Wie Graphen vereinheitlichen?

- Grad zwei Knoten löschen?
- Knoten kontrahieren und Overlay Graphen!

Knoten Kontraktionen

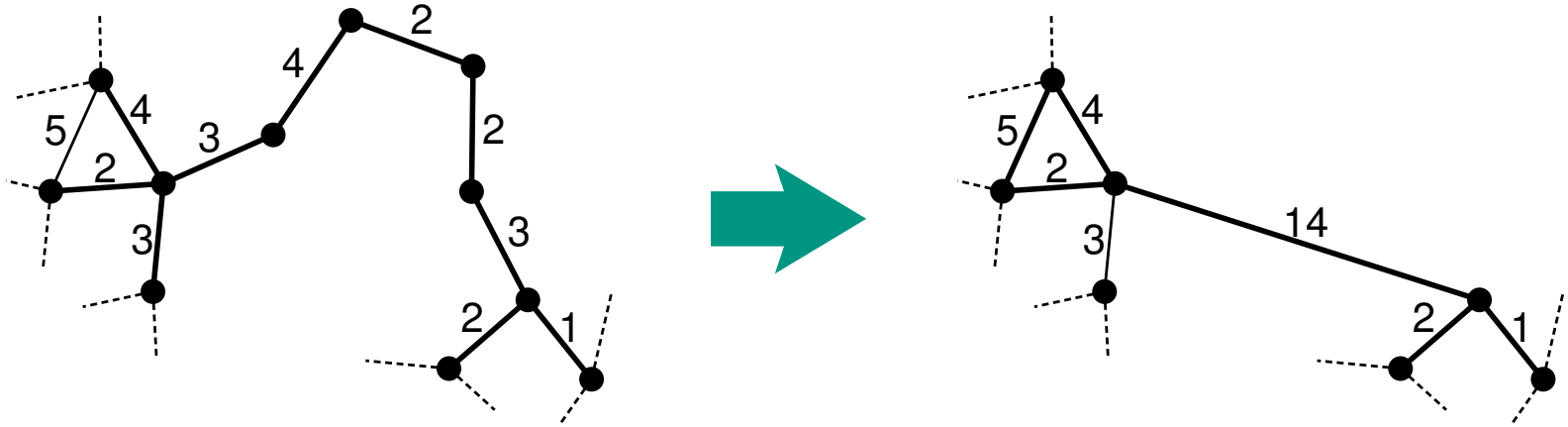
Beispiel: Kontraktion eines Pfades



- Finde Pfade im Graphen

Knoten Kontraktionen

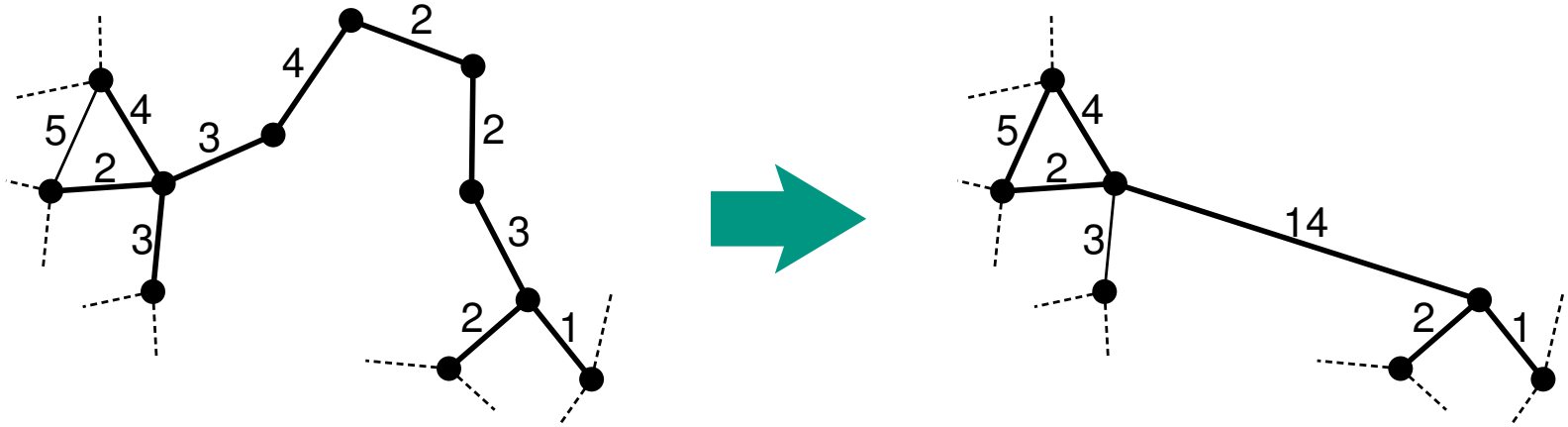
Beispiel: Kontraktion eines Pfades



- Finde Pfade im Graphen
- Ersetze jeden Pfad durch einen Shortcut
- Länge des Shortcut = Länge des Pfades

Knoten Kontraktionen

Beispiel: Kontraktion eines Pfades

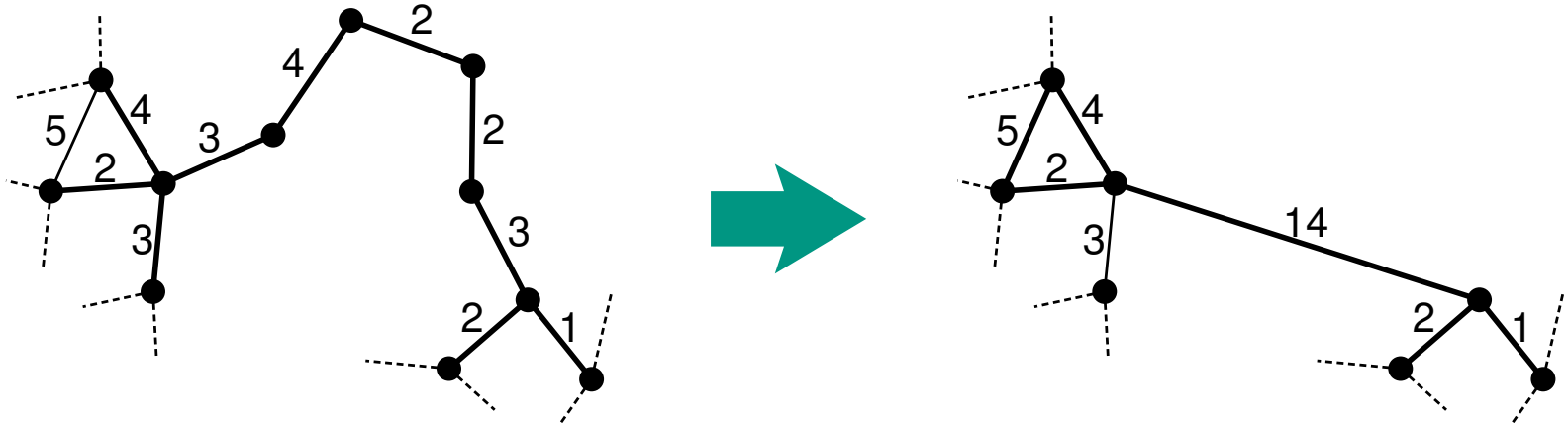


- Finde Pfade im Graphen
- Ersetze jeden Pfad durch einen Shortcut
- Länge des Shortcut = Länge des Pfades

Probleme?

Knoten Kontraktionen

Beispiel: Kontraktion eines Pfades



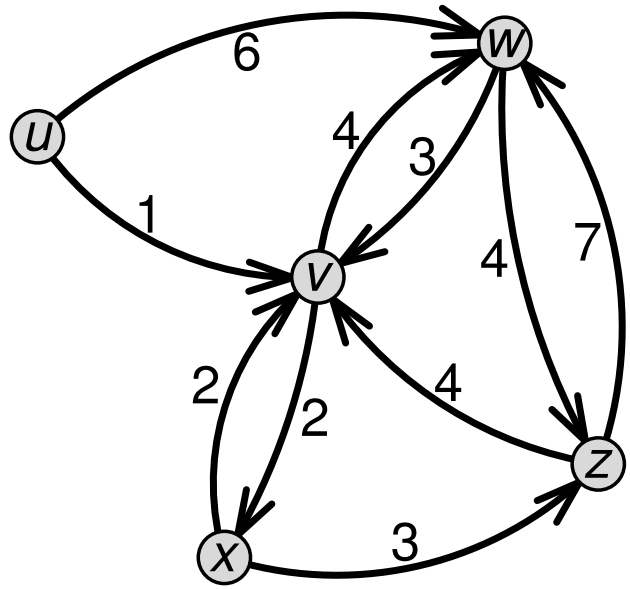
- Finde Pfade im Graphen
- Ersetze jeden Pfad durch einen Shortcut
- Länge des Shortcut = Länge des Pfades

Probleme:

- Gelöschte Knoten können nicht mehr Start/Ziel sein
- Entfernt nur Grad 2 Knoten – Gibt es weitere unnötige Strukturen?

Knoten Kontraktionen

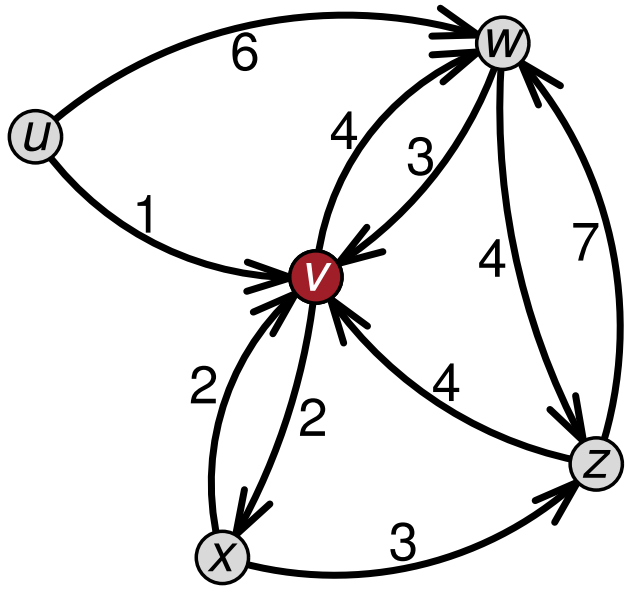
Allgemeine Knoten Kontraktion:



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

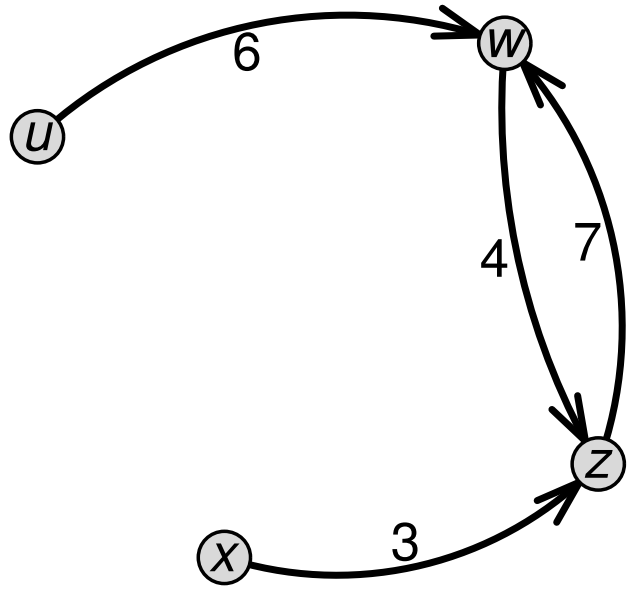
- Knoten v soll kontrahiert werden



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

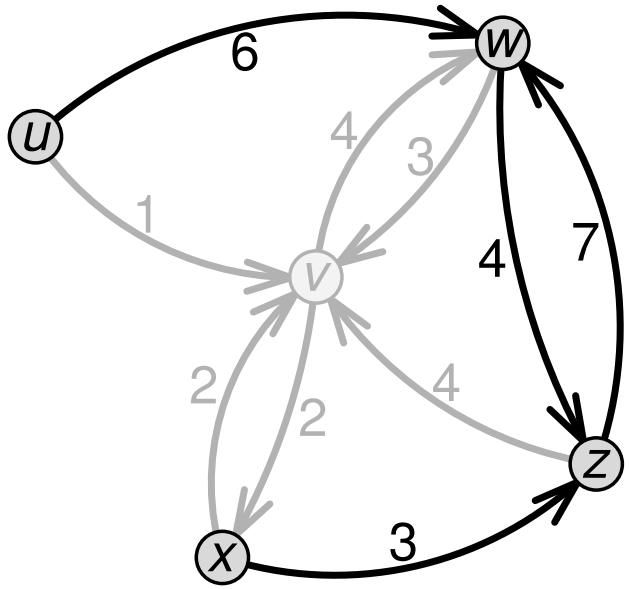
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

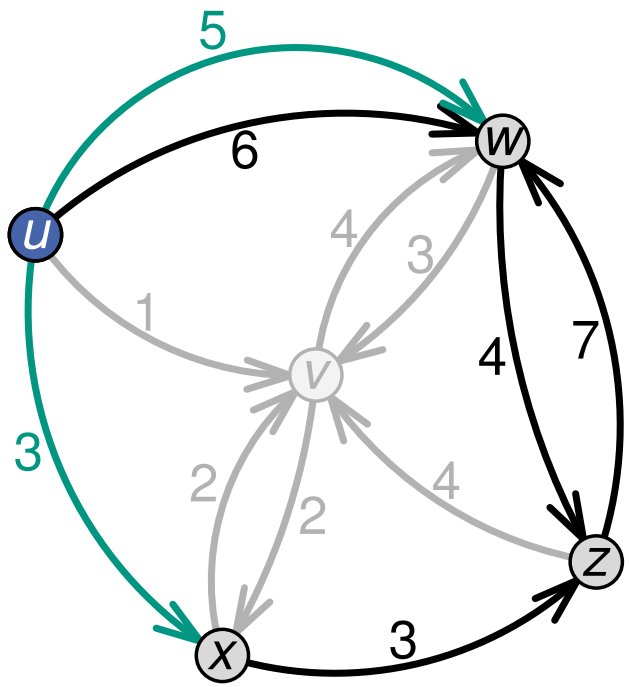
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

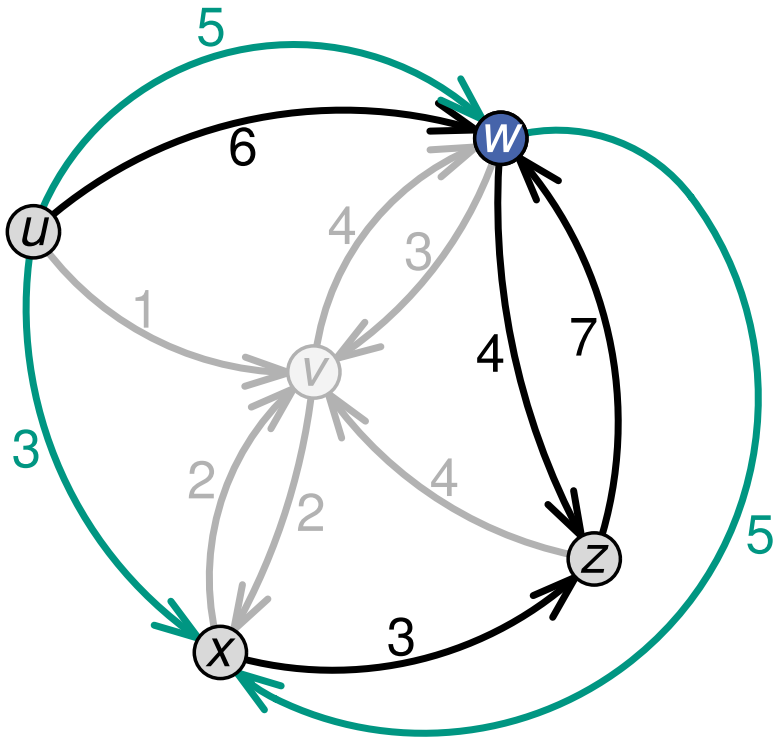
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

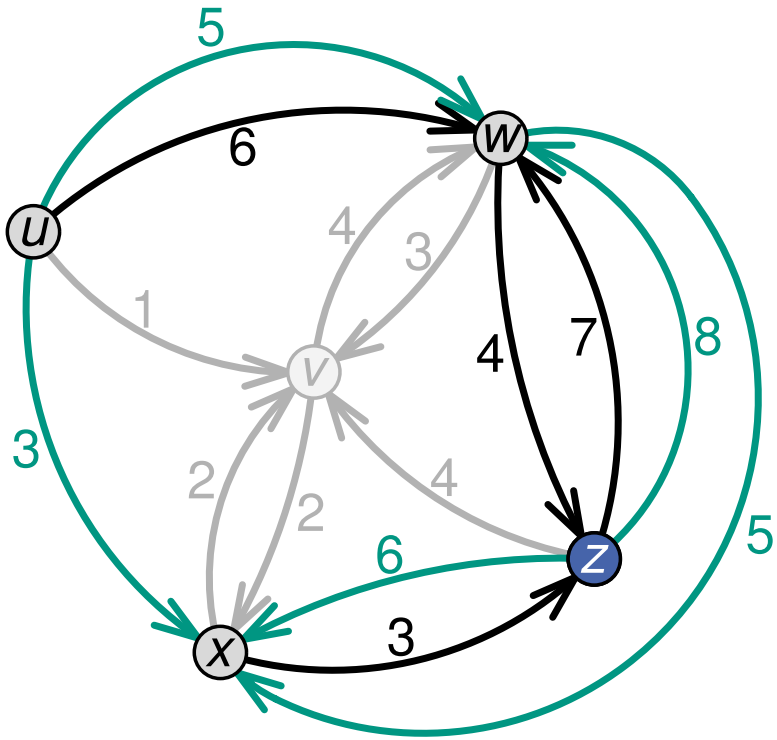
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

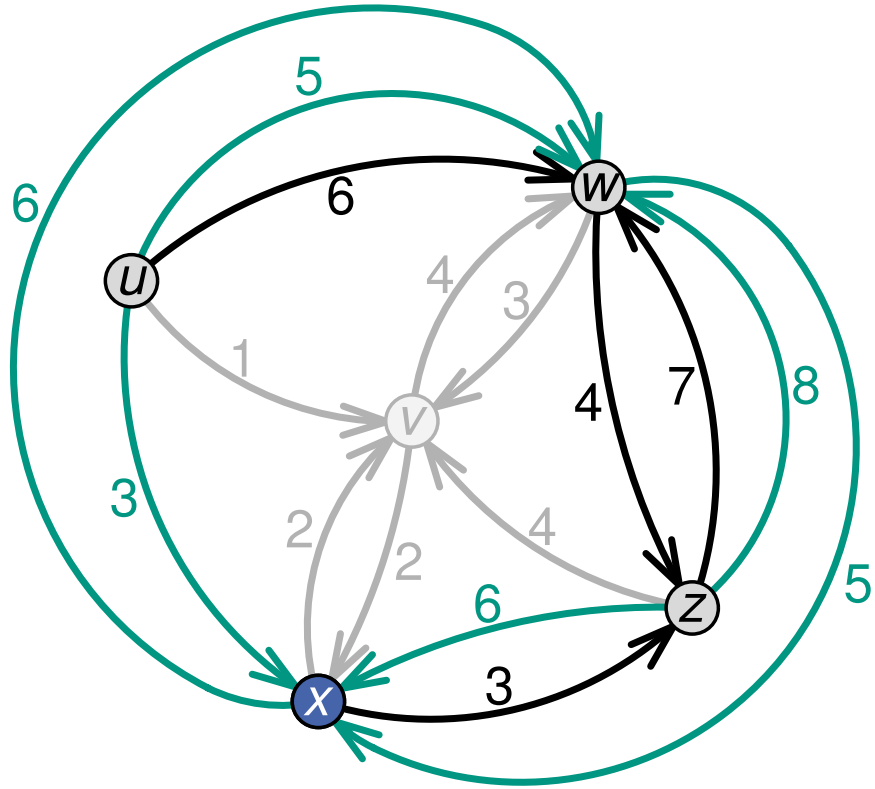
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

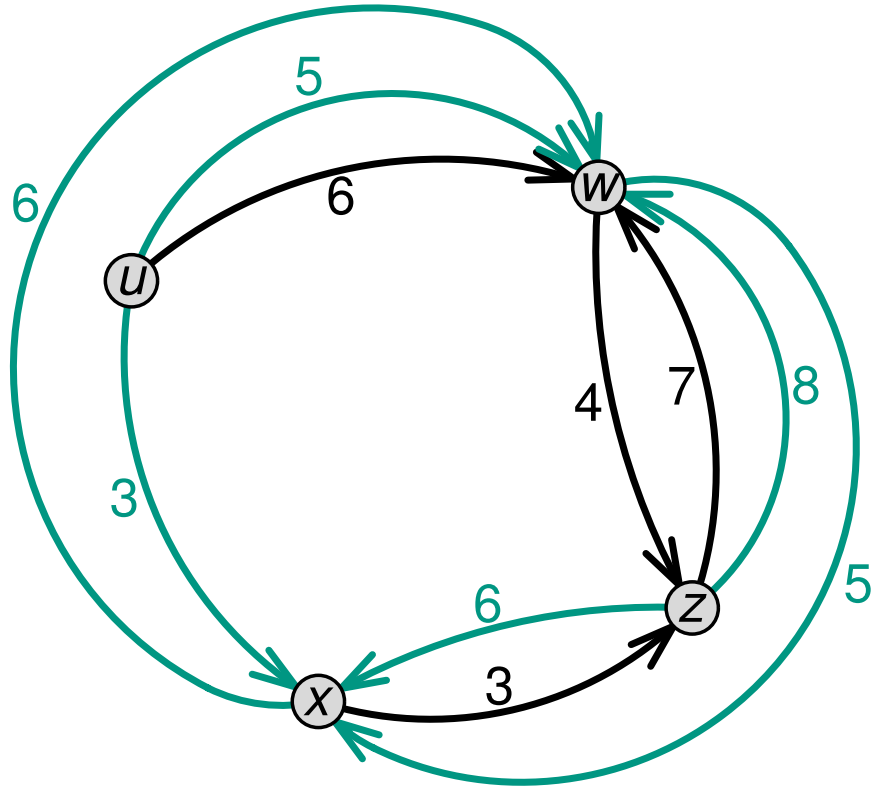
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

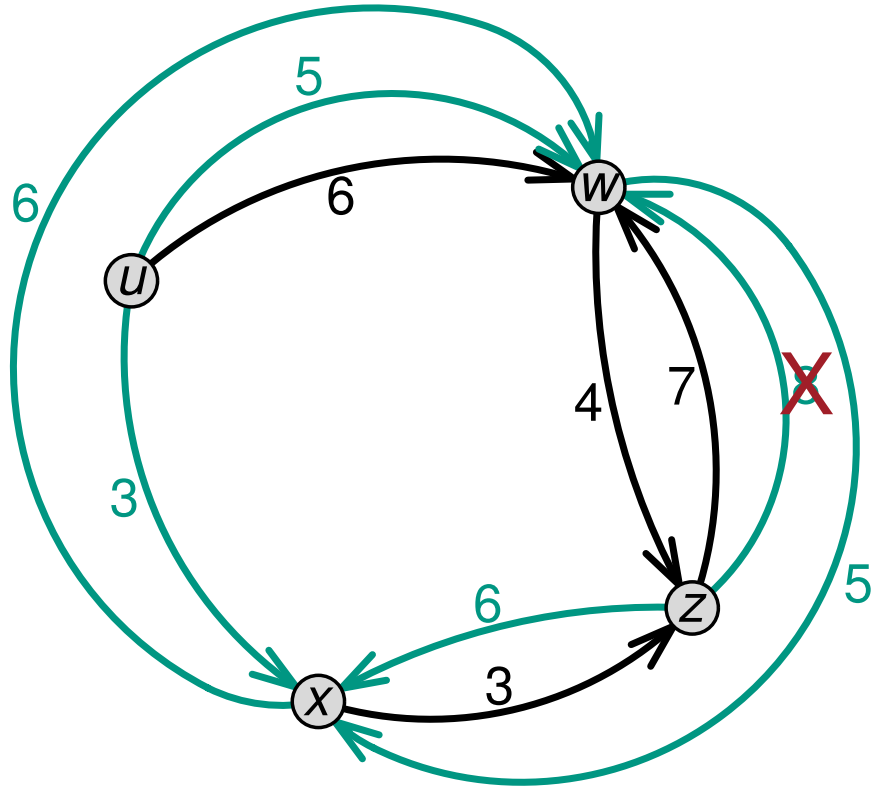
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)
- Lösche Knoten v



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

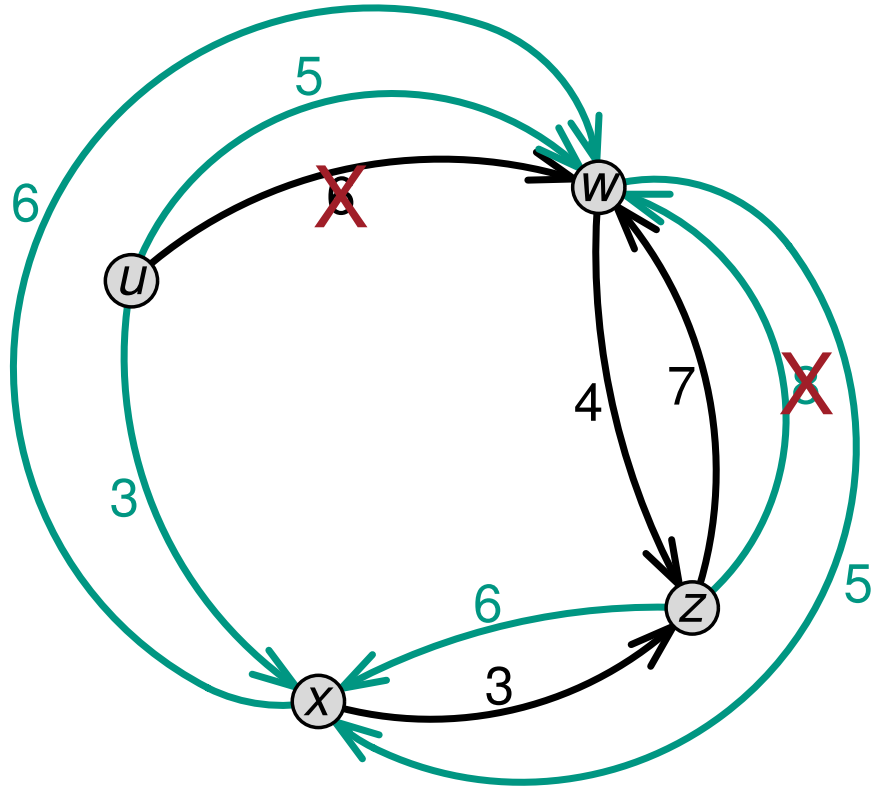
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)
- Lösche Knoten v
- Reduziere Multikanten



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

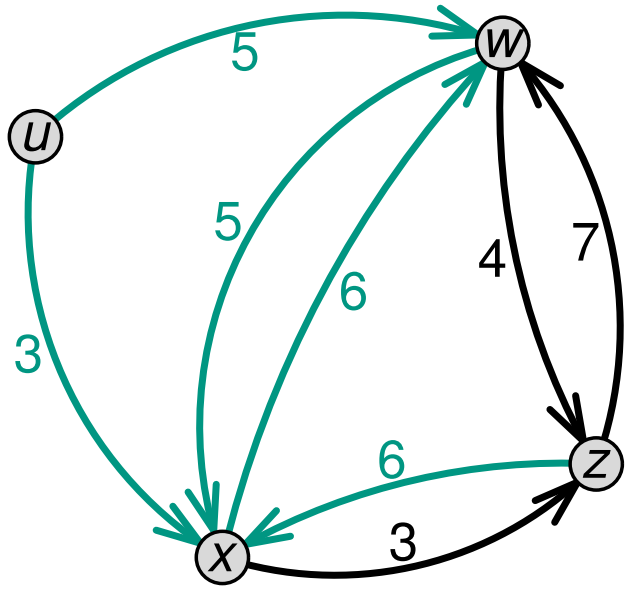
- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)
- Lösche Knoten v
- Reduziere Multikanten



Knoten Kontraktionen

Allgemeine Knoten Kontraktion:

- Knoten v soll kontrahiert werden
- Erhalte Distanzen in $G \setminus \{v\}$
- $\forall$ Nachbarpaare $\{a, b\}$ von v:
  - Sind (a, v) und (v, b) Kanten?
  - Dann erstelle Shortcut (a, b)
- Lösche Knoten v
- Reduziere Multikanten



Grundstein vieler weiterer Beschleunigungstechniken

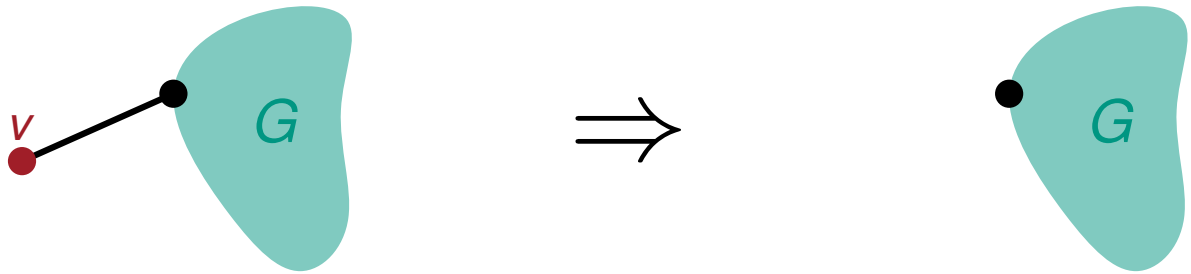
Knoten Kontraktionen – TopoCore

Welche Kontraktionen reduzieren die Komplexität des Graphen?

Knoten Kontraktionen – TopoCore

Welche Kontraktionen reduzieren die Komplexität des Graphen?

- Grad 1 Knoten:

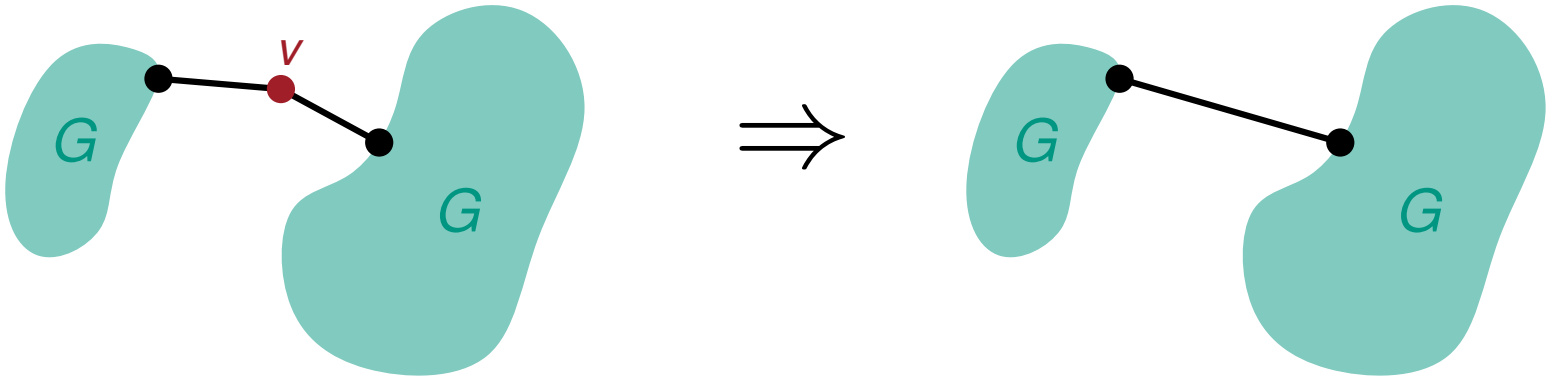


- Sackgasse liegen nie auf kürzesten Wegen
- Kontraktion reduziert Anzahl Kanten
- Entfernt komplette Bäume!

Knoten Kontraktionen – TopoCore

Welche Kontraktionen reduzieren die Komplexität des Graphen?

- Grad 2 Knoten:

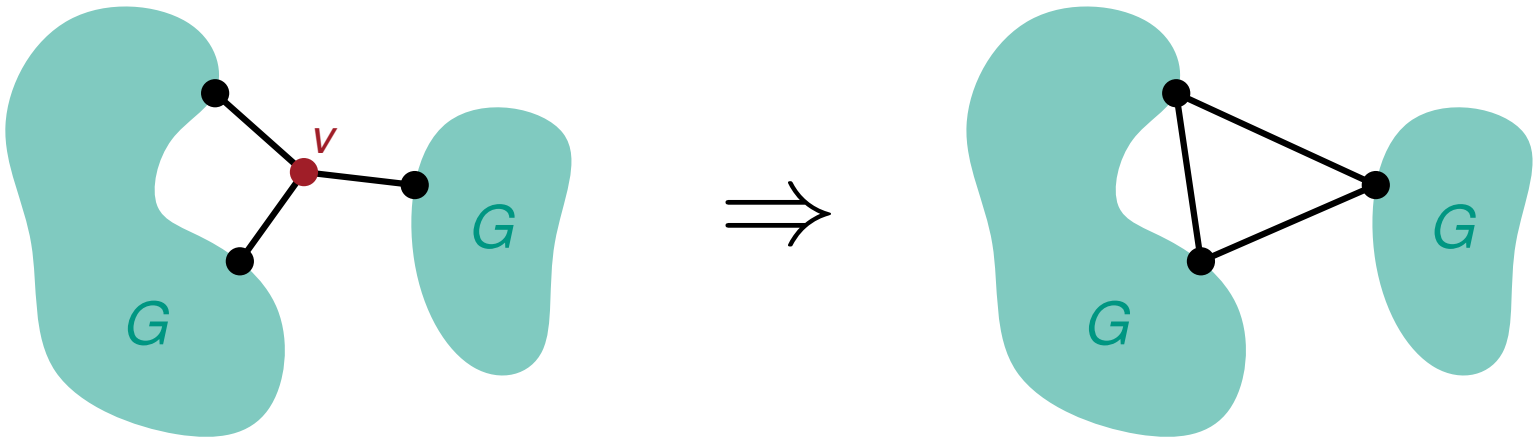


- Nur eine Möglichkeit für Routing auf Pfaden
- Kontraktion reduziert Anzahl Kanten

Knoten Kontraktionen – TopoCore

Welche Kontraktionen reduzieren die Komplexität des Graphen?

- Grad 3 Knoten:

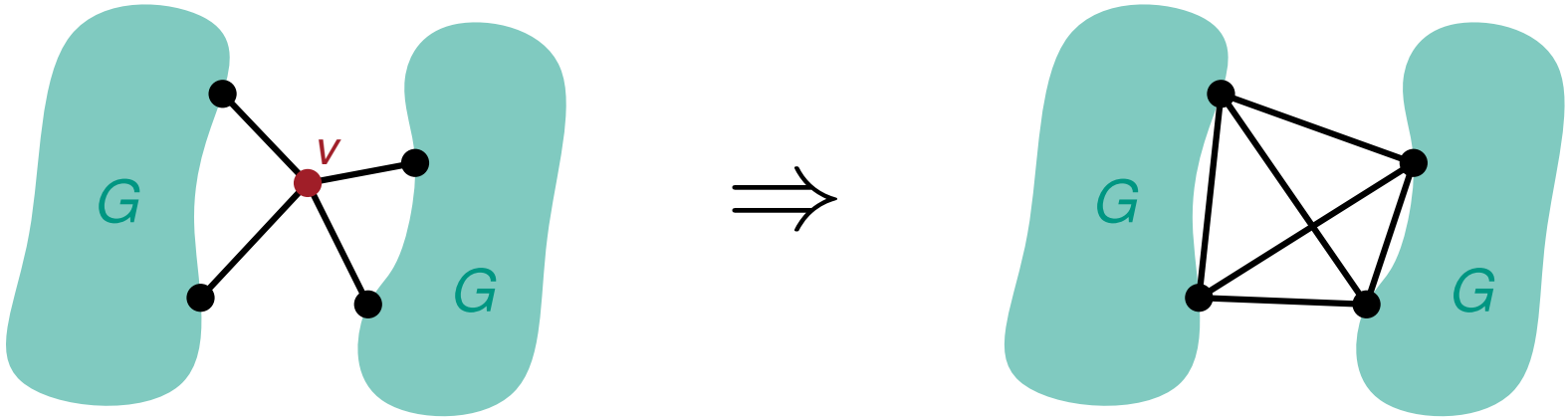


- Wenig Möglichkeiten während Routing (Alle vorberechnen)
- Kontraktion lässt Anzahl Kanten unverändert
  (Anzahl Knoten wird reduziert)

Knoten Kontraktionen – TopoCore

Welche Kontraktionen reduzieren die Komplexität des Graphen?

- Grad ≥ 4 Knoten:



- Struktur nicht mehr ausnutzbar
- Kontraktion erhöht komplexität des Graphen
- Nachbarknoten von v bilden Clique

Overlays

Bisher:

- Kontraktionen reduzieren komplexität des Graphen
- Knoten werden endgültig gelöscht
- Mögliche Start-/Zielpunkte gehen verloren

Overlays

Bisher:

- Kontraktionen reduzieren komplexität des Graphen
- Knoten werden endgültig gelöscht
- Mögliche Start-/Zielpunkte gehen verloren

Gesucht:

- Technik die alle Knoten im Graphen beibehält
- Dabei weiterhin von Shortcuts profitieren

Overlays

Bisher:

- Kontraktionen reduzieren komplexität des Graphen
- Knoten werden endgültig gelöscht
- Mögliche Start-/Zielpunkte gehen verloren

Gesucht:

- Technik die alle Knoten im Graphen beibehält
- Dabei weiterhin von Shortcuts profitieren

Lösung: Overlays

Overlays

Definition:

- Ein Overlay O ist eine Teilmenge der Knoten $O \subset V$
- Für Knoten $s, t \in O$ gilt:

  Der kürzeste s-t-Pfad enthält nur Knoten aus O

Overlays

Definition:

- Ein Overlay O ist eine Teilmenge der Knoten $O \subset V$
- Für Knoten $s, t \in O$ gilt:

  Der kürzeste s-t-Pfad enthält nur Knoten aus O

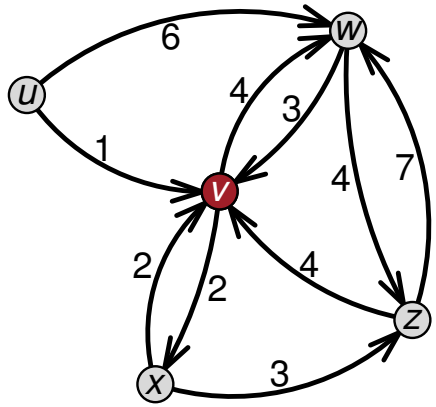
Kürzeste Wege Suche mit Overlays:

- Bidirektionale Variante von Dijkstra
- Suche erreicht Overlay $\Rightarrow$ Beschränke Suche auf Overlay

  Prune Kanten (u, v) mit $u \in O$ und $v \notin O$

Overlays

Beispiel:

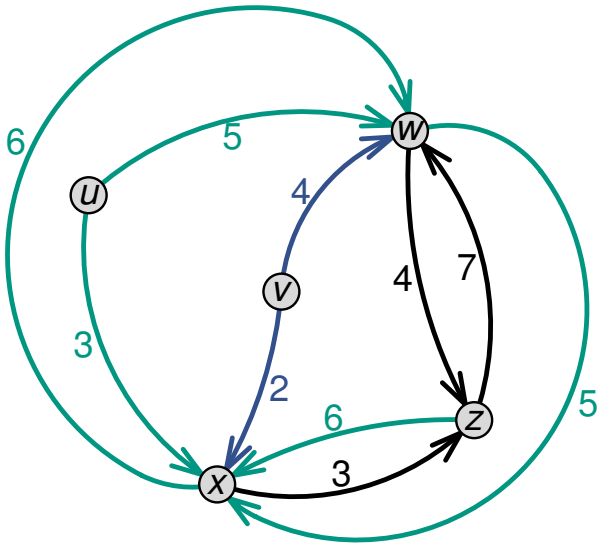


- Nutze speziellen Vorwärtsgraph $\overrightarrow{G}$ und Rückwärtsgraph $\overleftarrow{G}$
- $\overrightarrow{G}$ enthält Kanten ausgehen von kontrahierten Knoten
- $\overleftarrow{G}$ enthält Kanten zu kontrahierten Knoten

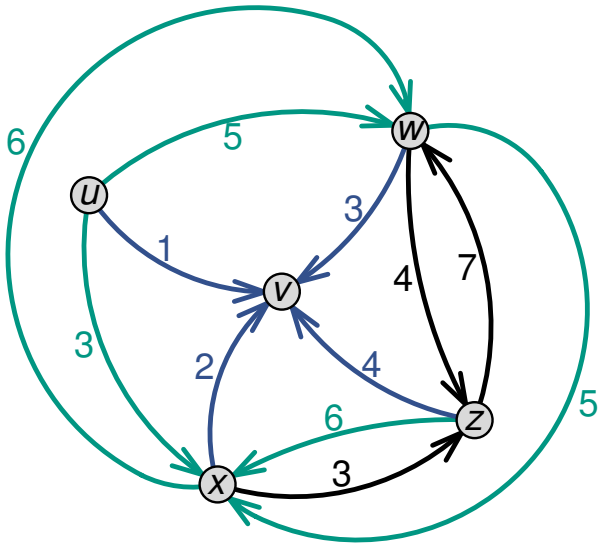
Overlays

Beispiel:

$\overrightarrow{G}$:



$\overleftarrow{G}$:

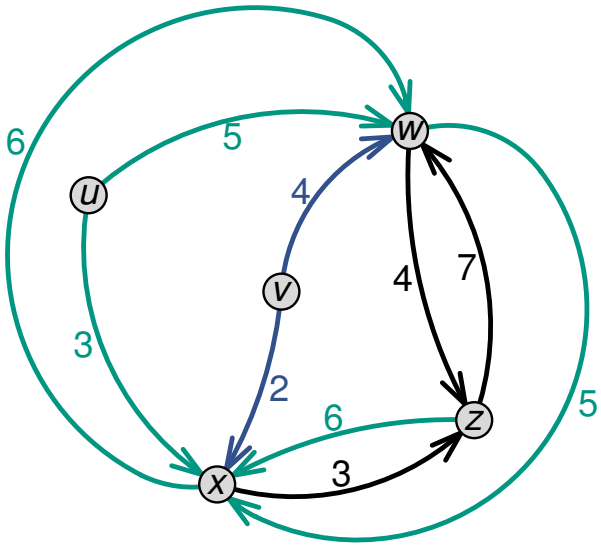


- Nutze speziellen Vorwärtsgraph $\overrightarrow{G}$ und Rückwärtsgraph $\overleftarrow{G}$
- $\overrightarrow{G}$ enthält Kanten ausgehen von kontrahierten Knoten
- $\overleftarrow{G}$ enthält Kanten zu kontrahierten Knoten

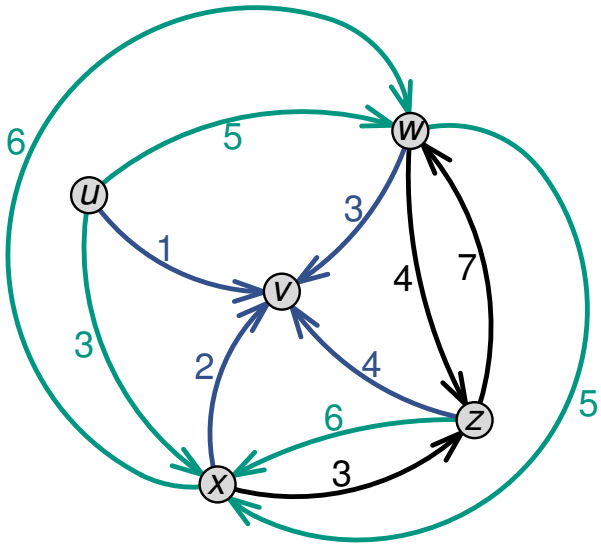
Overlays

Beispiel:

$\overrightarrow{G}$:



$\overleftarrow{G}$:



- Nutze speziellen Vorwärtsgraph $\overrightarrow{G}$ und Rückwärtsgraph $\overleftarrow{G}$
- $\overrightarrow{G}$ enthält Kanten ausgehen von kontrahierten Knoten
- $\overleftarrow{G}$ enthält Kanten zu kontrahierten Knoten

Danach: Bidirektionaler Dijkstra unverändert benutzbar

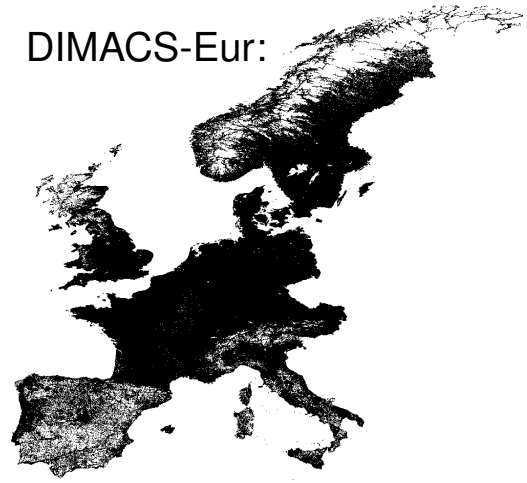
Graphen Vergleich

- Größe der Graphen:

Graph	$\lvert V \rvert [\cdot 10^3]$	$\lvert E \rvert [\cdot 10^3]$
OSM-Ger	20 690	41 792
OSM-Eur	173 789	347 997
DIMACS-Eur	18 010	42 189

OSM-Eur:

DIMACS-Eur:



OSM-Ger:

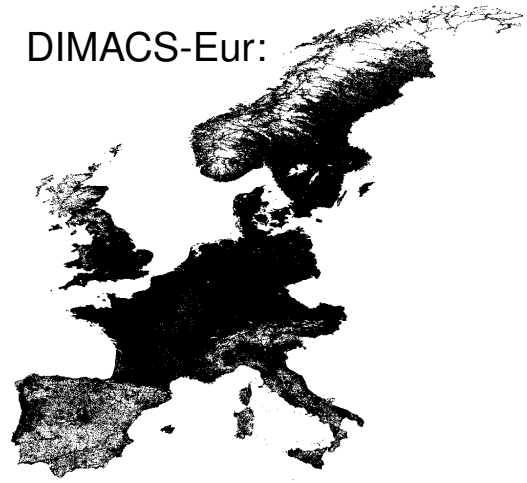
Graphen Vergleich

- Größe der Graphen:

Graph	$\lvert V \rvert [\cdot 10^3]$	$\lvert E \rvert [\cdot 10^3]$
OSM-Ger	20 690	41 792
OSM-Eur	173 789	347 997
DIMACS-Eur	18 010	42 189

OSM-Eur:

DIMACS-Eur:



- Verteilung der Knotengrade:

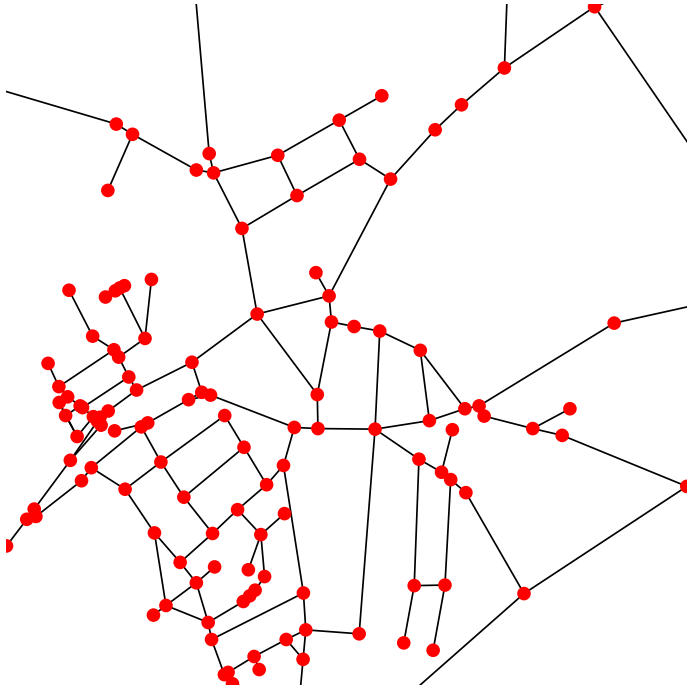
Graph	#Knoten pro Grad: 1	2	3	≥ 4
OSM-Ger	14%	71%	14%	1%
OSM-Eur	12%	77%	10%	1%
DIMACS-Eur	26%	19%	49%	6%

OSM-Ger:

Graphen Vergleich – TopoCore

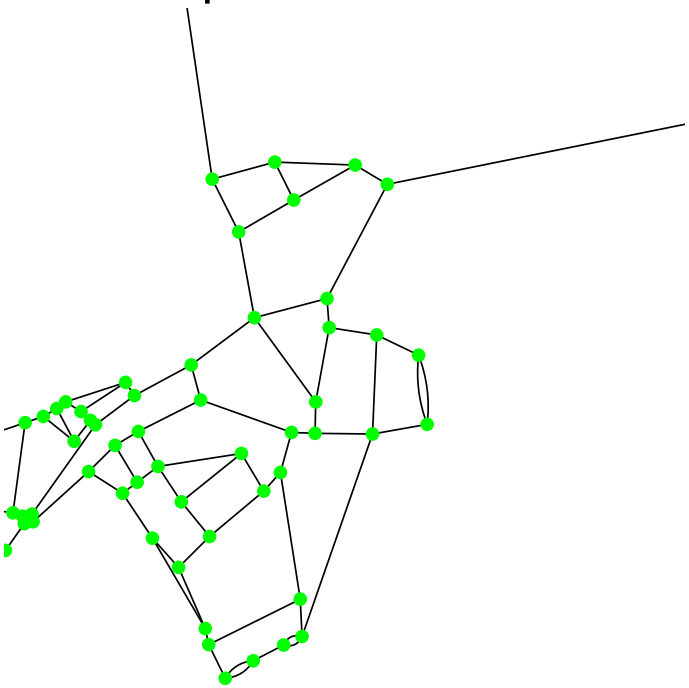
OSM Input:

Dimacs Input:

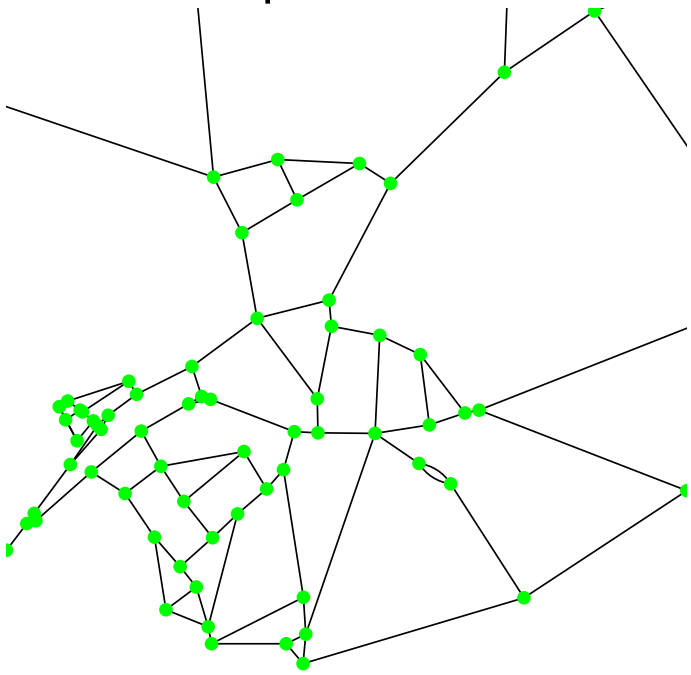


Graphen Vergleich – TopoCore

OSM TopoCore:



Dimacs TopoCore:



Graphen Vergleich – TopoCore

Algorithmus	#Attr.	Graph	$\lvert V \rvert$ [$\cdot 10^6$]	$\lvert E \rvert$ [$\cdot 10^6$]	Prepro. [h:m:s]	Query [ms]
CH [GSSV'12]	1	DIMACS-Eur	18.0	42.2	2:45	0.15
Pareto-SHARC [DW'09]	2	DIMACS-Eur	18.0	42.2	7:12:00	35.4
FlexCH [GKS'10]	2	DIMACS-Eur	18.0	42.2	5:12:00	0.98
MultiCH [FS'13]	2	OSM-BaWü	2.5	5.0	2:01	0.42
MultiCH [FS'13]	3	OSM-BaWü	2.5	5.0	1:08	3.16
k-Path Cover [FNS'14]	8	OSM-BaWü	2.2	4.6	12	35
k-Path Cover [FNS'14]	8	**OSM-Ger**	17.7	36.1	**2:29**	**249**
PRP [FS'15]	10	**OSM-Ger**	21.7	44.1	**n/r**	**128**
TopoCore [DSW'15]	8	OSM-BaWü	3.1	6.2	3	9
TopoCore [DSW'15]	8	**OSM-Ger**	20.7	41.8	**35**	**86**
TopoCore [DSW'15]	8	OSM-Eur	173.8	348.0	10:57	621
TopoCore [DSW'15]	8	DIMACS-Eur	18.0	42.2	36	279
TopoCore [DSW'15]	8	DIMACS-US	23.9	57.7	43	386

Nächster Termin

Karlsruher Institut für Technologie

Mittwoch, 27. April 2016

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bibliographie I

Edsger W. Dijkstra.

A Note on Two Problems in Connexion with Graphs.
Numerische Mathematik, 1(1):269–271, 1959.

Julian Dibbelt, Ben Strasser, and Dorothea Wagner.

Fast Exact Shortest Path and Distance Queries on Road Networks with Parametrized Costs.
In *Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM Press, 2015.

Hermann Kaindl and Gerhard Kainz.

Bidirectional Heuristic Search Reconsidered.
Journal of Artificial Intelligence Research, 7:283–317, December 1997.