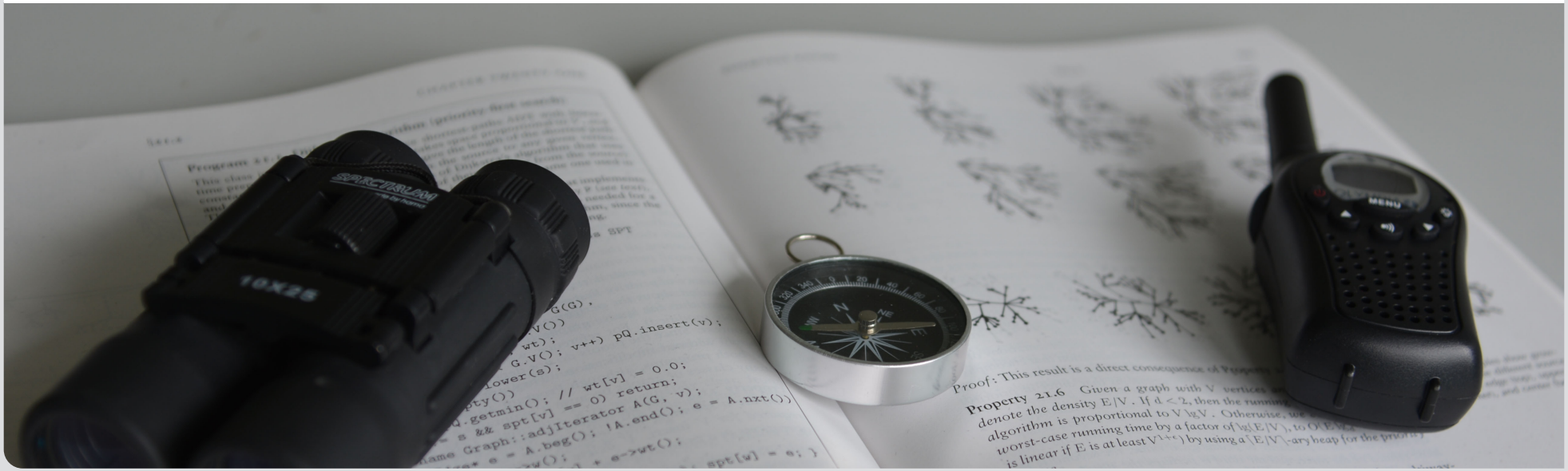
Karlsruher Institut für Technologie

An Algorithmic View on Sensor Networks –
Surveillance, Localization, and Communication

Dissertationsvortrag von Dipl.-Phys., Dipl.-Inform. Dennis Schieferdecker



Einführung

Einführung

- verteilte Sensoren

Einführung

- verteilte Sensoren
- lokale Kommunikation

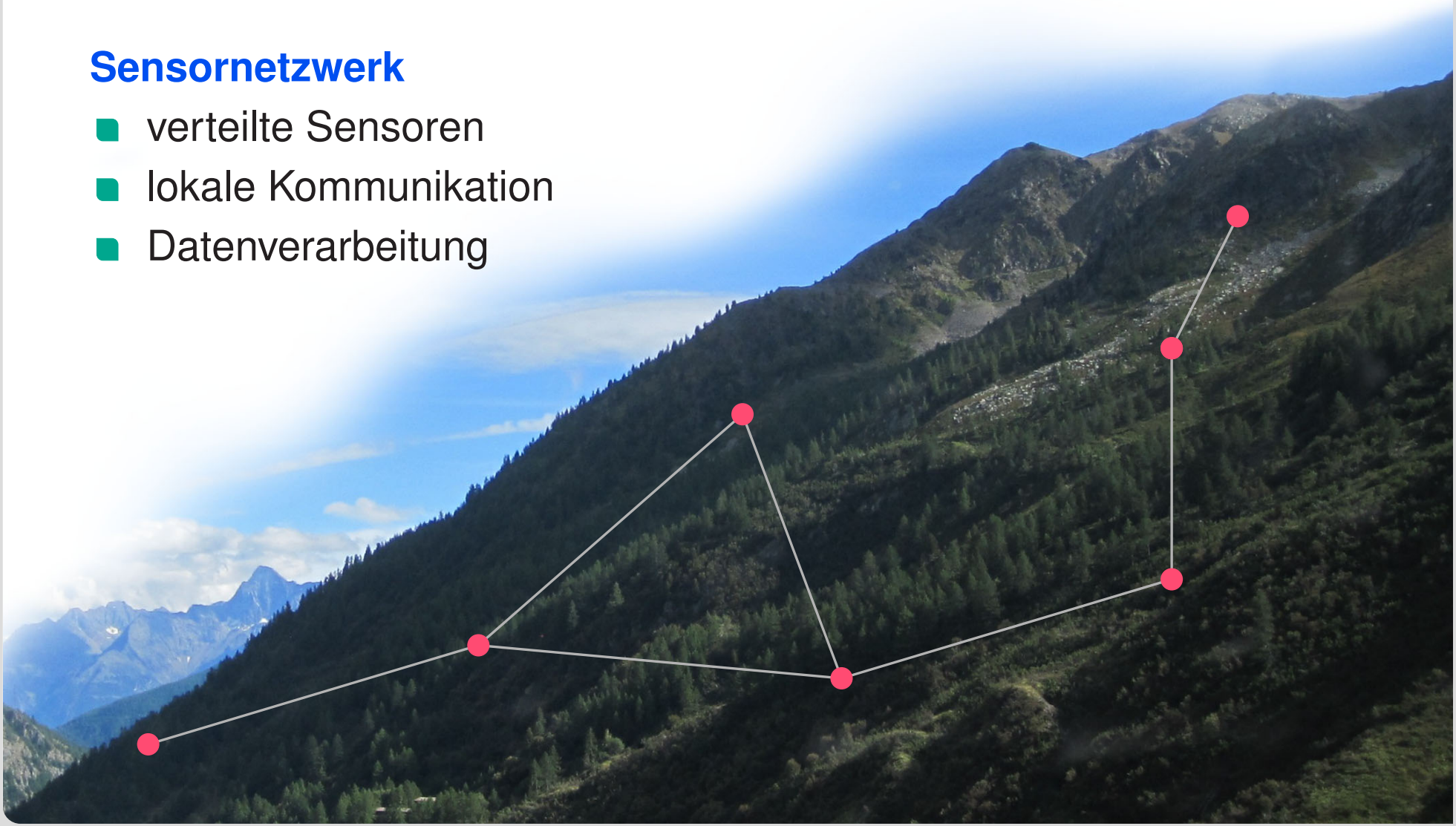
Einführung

- verteilte Sensoren
- lokale Kommunikation
- Datenverarbeitung

Einführung

Sensornetzwerk

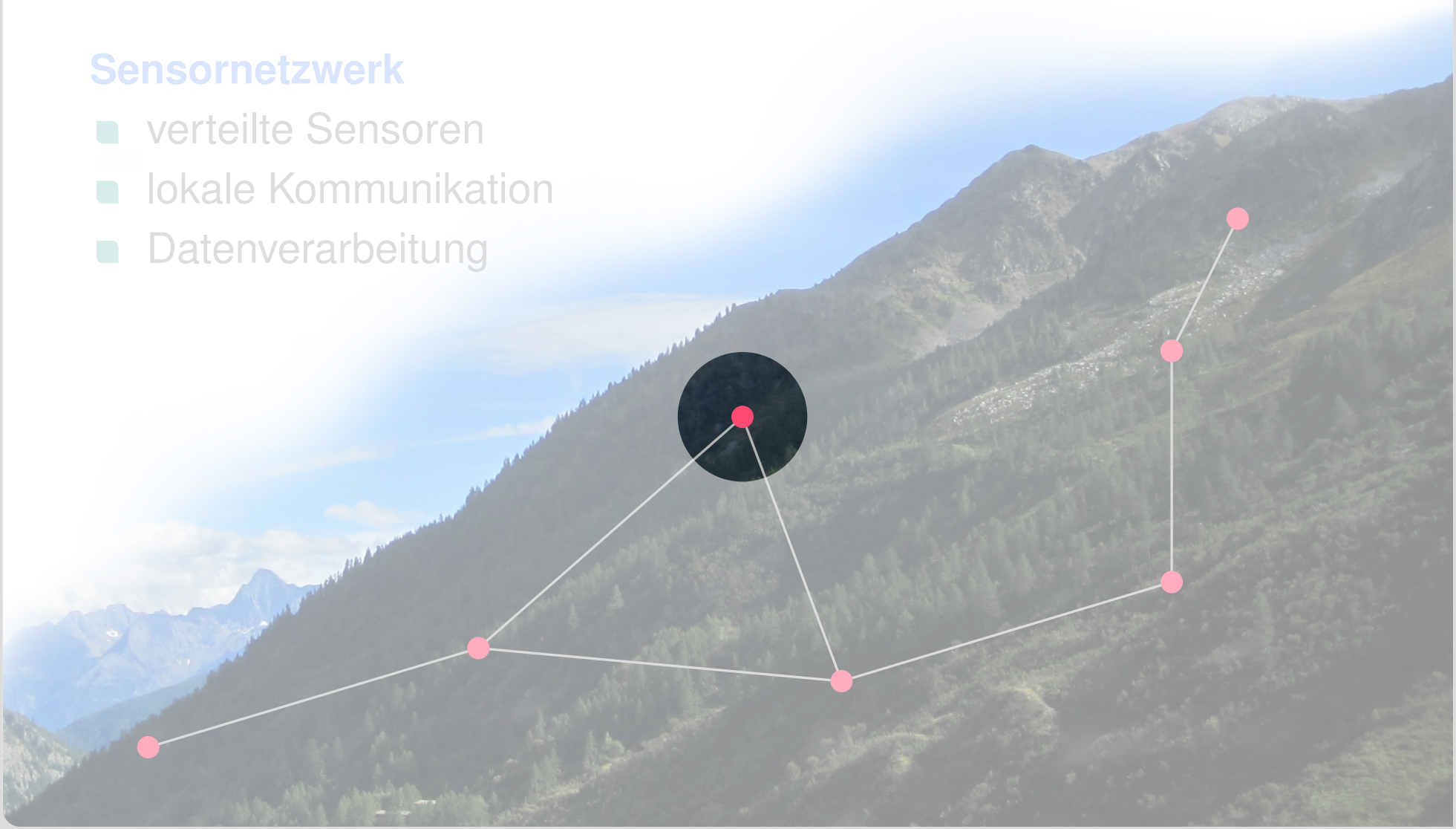
- verteilte Sensoren
- lokale Kommunikation
- Datenverarbeitung



Einführung

Sensornetzwerk

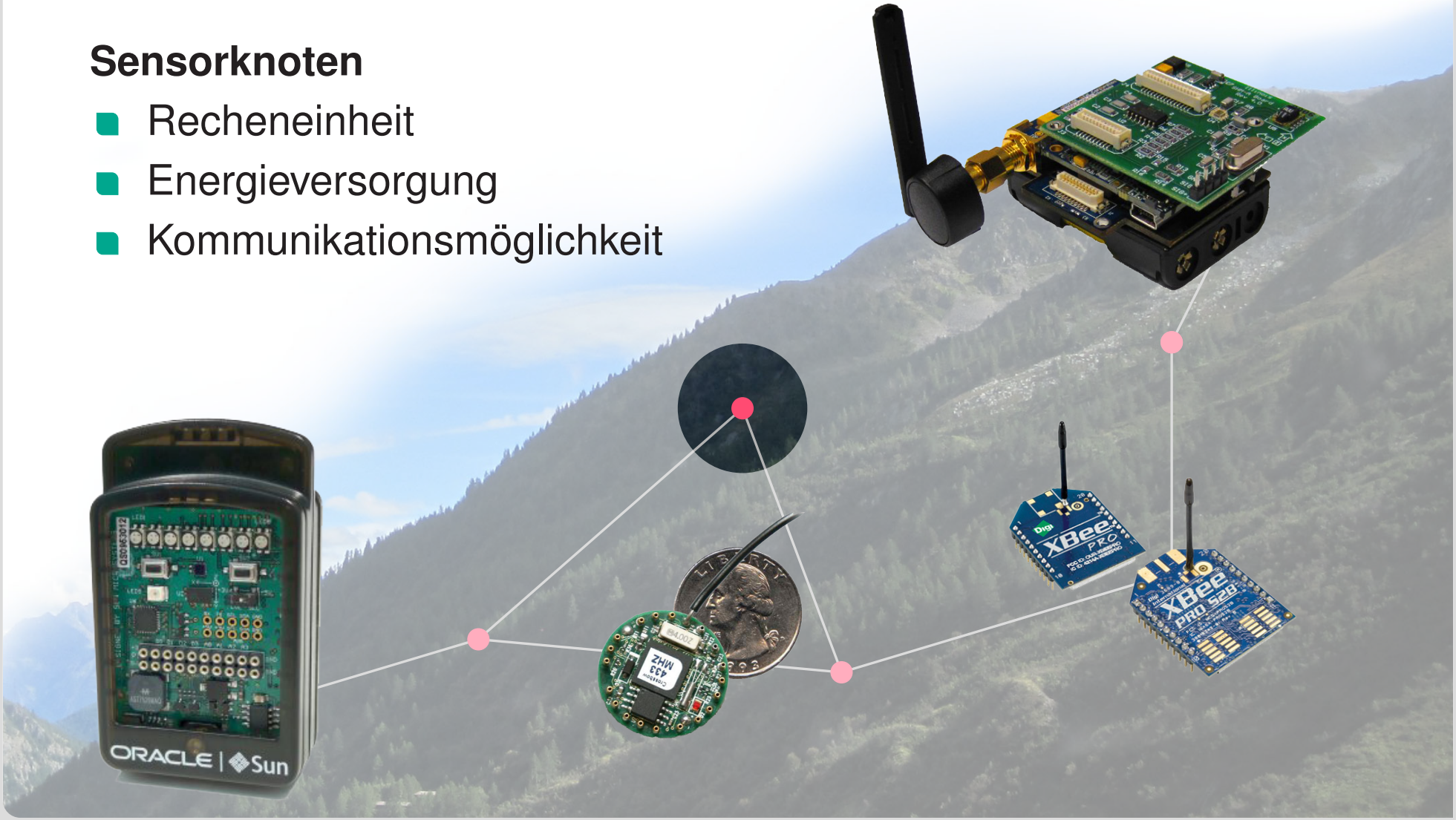
- verteilte Sensoren
- lokale Kommunikation
- Datenverarbeitung



Einführung

Sensorknoten

- Recheneinheit
- Energieversorgung
- Kommunikationsmöglichkeit



Einführung

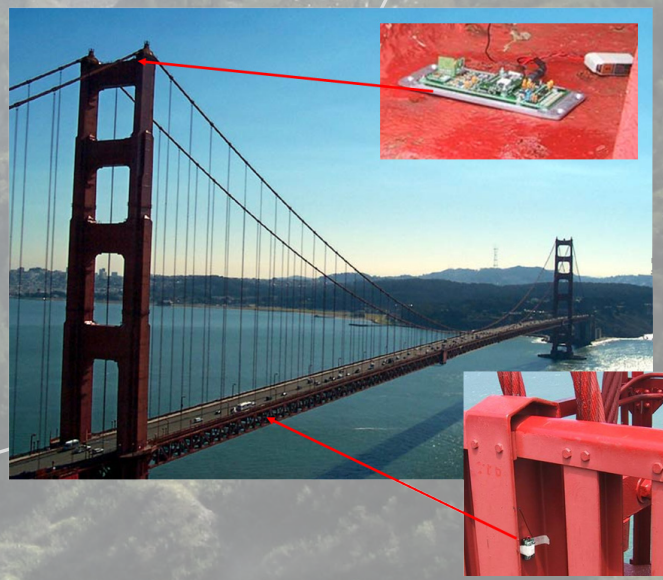
Sensorknoten

- Recheneinheit
- Energieversorgung
- Kommunikationsmöglichkeit

Einführung

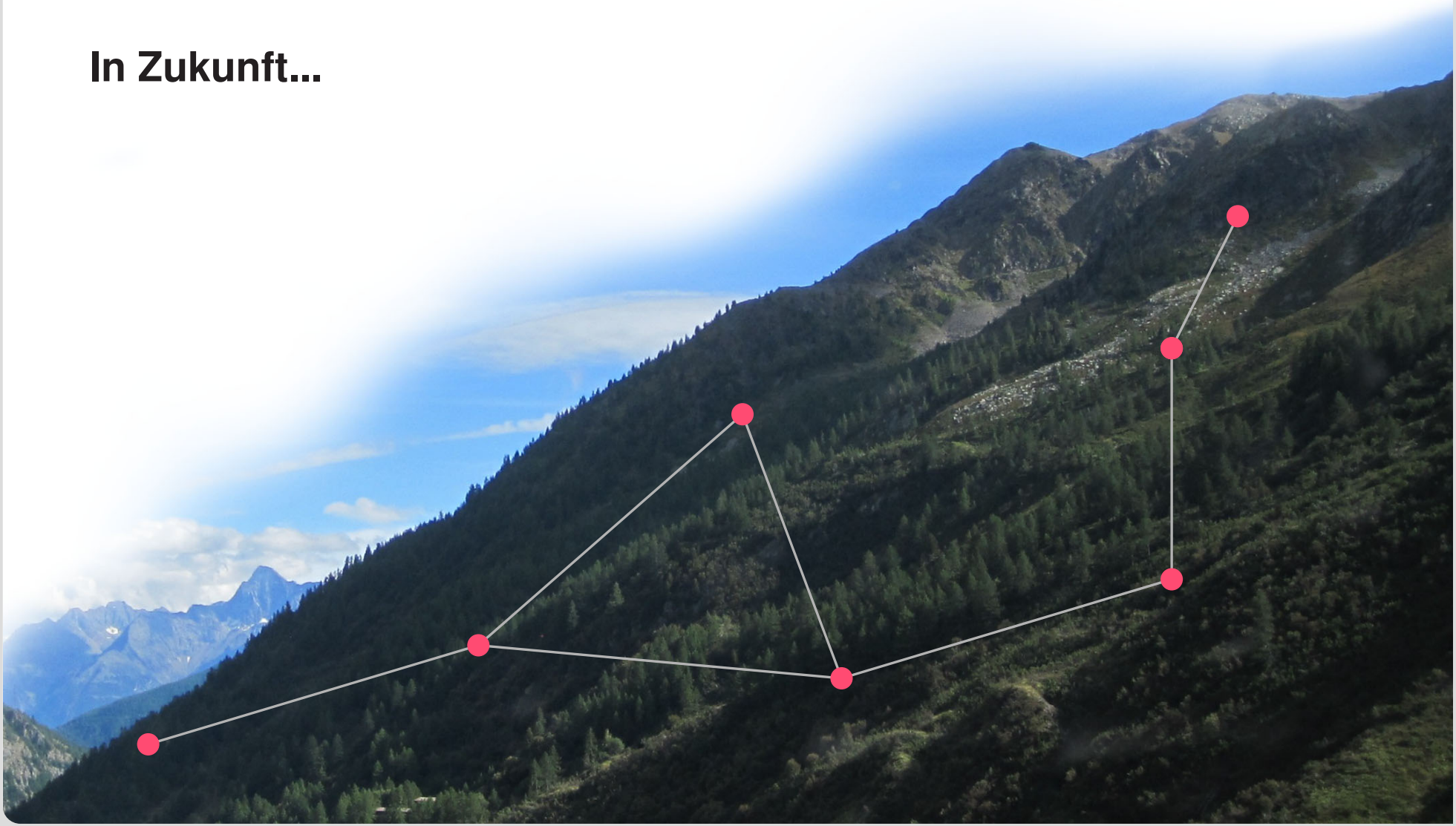
Sensorknoten

- Recheneinheit
- Energieversorgung
- Kommunikationsmöglichkeit



Einführung

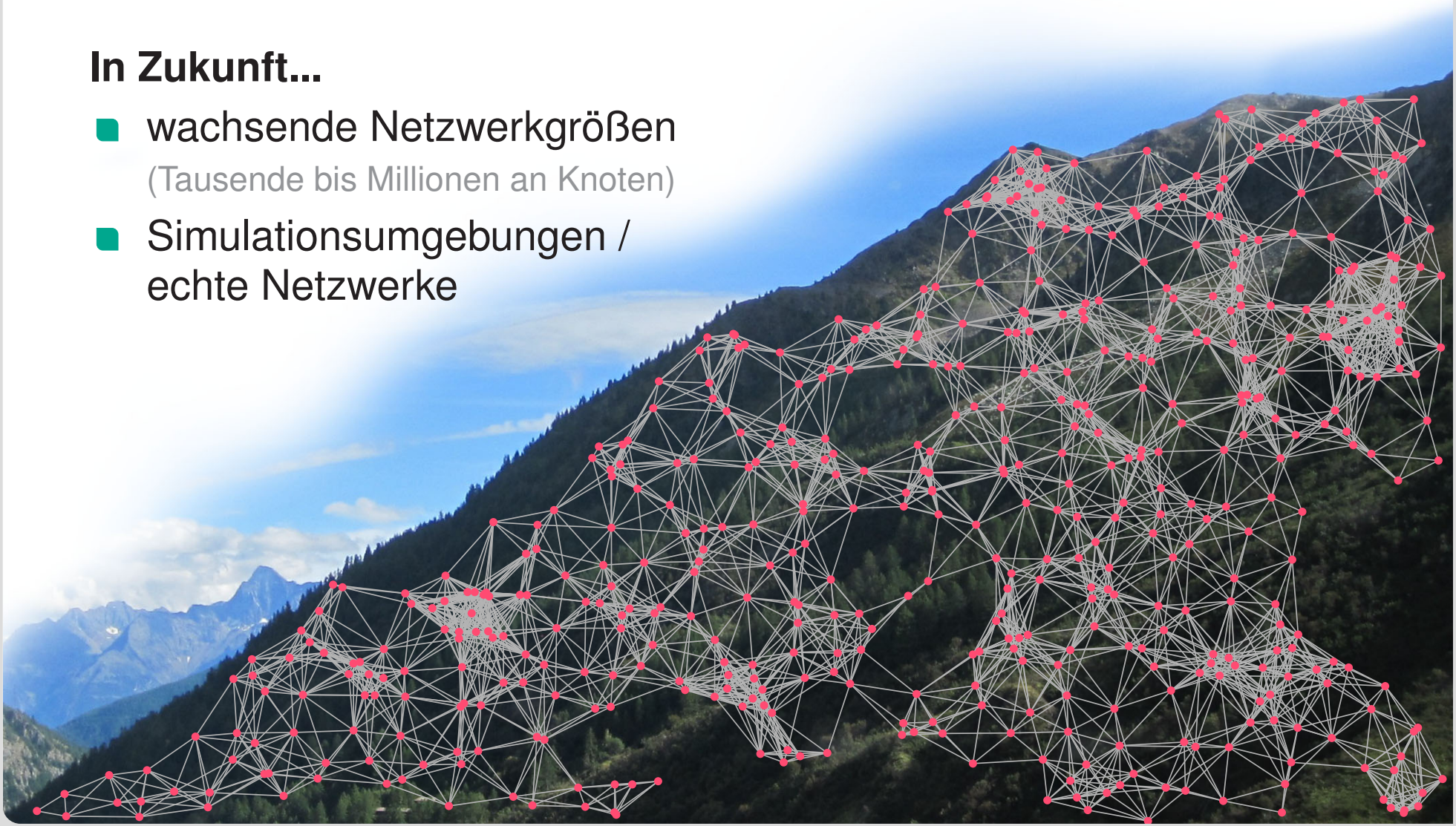
In Zukunft...



Einführung

In Zukunft...

- wachsende Netzwerkgrößen
  (Tausende bis Millionen an Knoten)
- Simulationsumgebungen /
  echte Netzwerke



Einführung

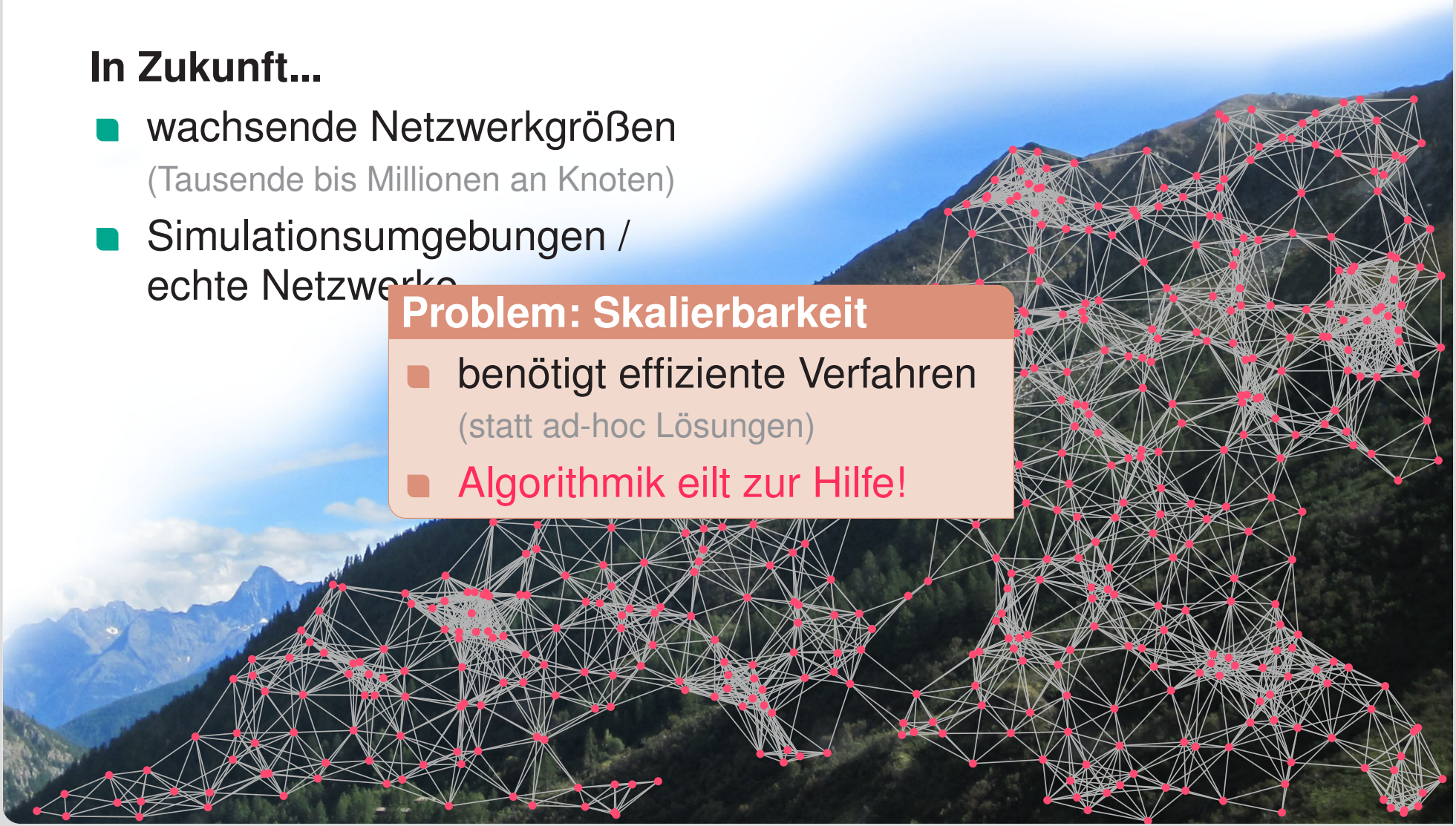
Karlsruher Institut für Technologie

In Zukunft...

- wachsende Netzwerkgrößen
  (Tausende bis Millionen an Knoten)
- Simulationsumgebungen /
  echte Netzwerke

Problem: Skalierbarkeit

- benötigt effiziente Verfahren
  (statt ad-hoc Lösungen)
- Algorithmik eilt zur Hilfe!



An Algorithmic View on Sensor Networks –
Surveillance, Localization, and Communication

An Algorithmic View on Sensor Networks – **Surveillance**, Localization, and Communication

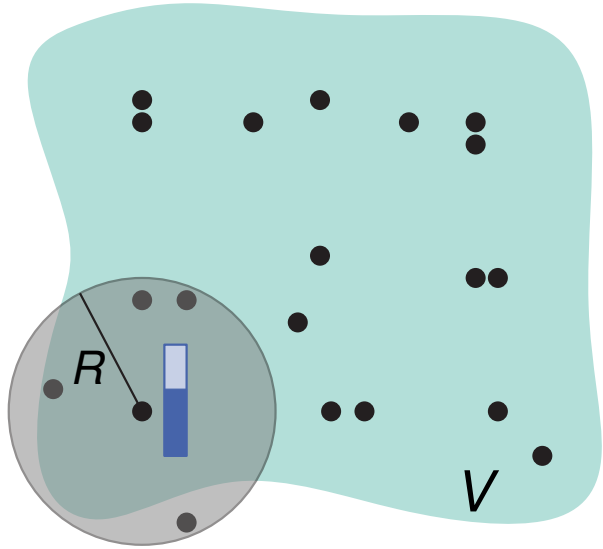
Sensoreinsatzplanung

[ALGOSENSORS'10]

Problemstellung

- beschränktes Gebiet
- stationäre Sensorknoten
  - kreisförmiger Beobachtungsbereich
  - beschränkte Batteriekapazität

→ Sensoreinsatzplan maximaler Dauer bei vollständiger Abdeckung des Gebiets

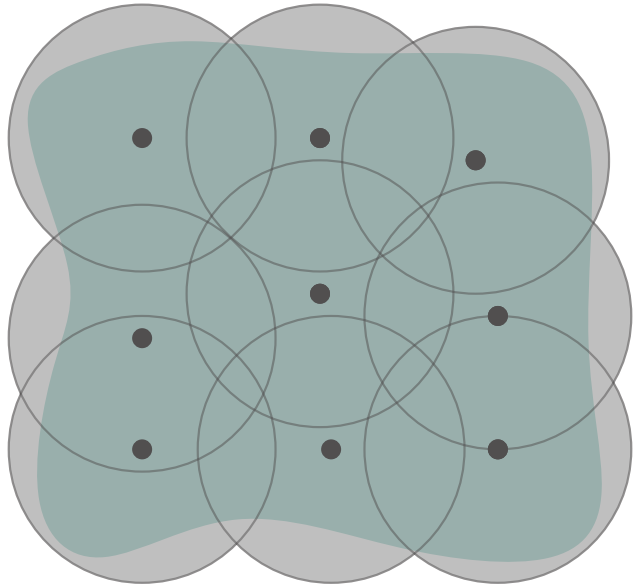


Anwendungen

- lückenlose Beobachtung (z.B. Studium von Wildtieren, Schutz vor Waldbränden)
- kontinuierliche Messungen mit garantierter minimaler Auflösung
  (z.B. Kontrolle von Schadstoffbelastungen)

Sensoreinsatzplanung

[ALGOSENSORS'10]



Problemstellung

- beschränktes Gebiet
- stationäre Sensorknoten
  - kreisförmiger Beobachtungsbereich
  - beschränkte Batteriekapazität

→ Sensoreinsatzplan maximaler Dauer bei vollständiger Abdeckung des Gebiets

Anwendungen

- lückenlose Beobachtung (z.B. Studium von Wildtieren, Schutz vor Waldbränden)
- kontinuierliche Messungen mit garantierter minimaler Auflösung
  (z.B. Kontrolle von Schadstoffbelastungen)

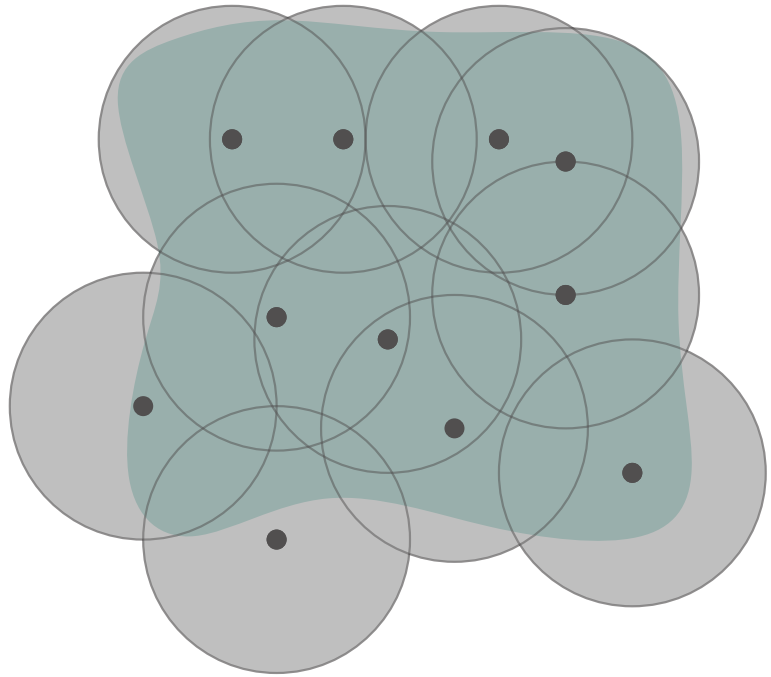
Sensoreinsatzplanung

[ALGOSENSORS'10]

Problemstellung

- beschränktes Gebiet
- stationäre Sensorknoten
  - kreisförmiger Beobachtungsbereich
  - beschränkte Batteriekapazität

→ Sensoreinsatzplan maximaler Dauer bei vollständiger Abdeckung des Gebiets

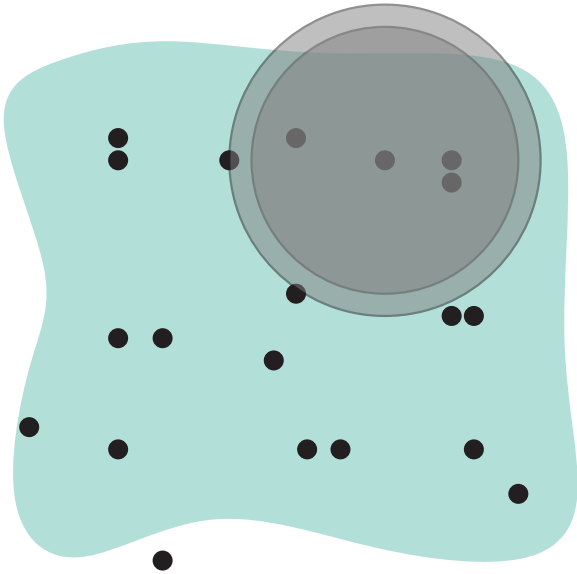


Anwendungen

- lückenlose Beobachtung (z.B. Studium von Wildtieren, Schutz vor Waldbränden)
- kontinuierliche Messungen mit garantierter minimaler Auflösung
  (z.B. Kontrolle von Schadstoffbelastungen)

Sensoreinsatzplanung

[ALGOSENSORS'10]



Beiträge dieser Arbeit

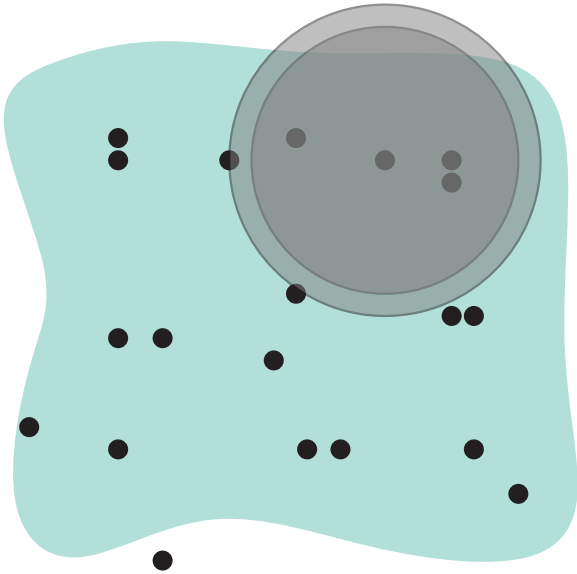
- Beweis der $\mathcal{NP}$-Vollständigkeit
  (über wMDS auf UDGs, Dualität von LPs)
- effizient polynomielles Approximationsschema (EPTAS)
  (lineare Laufzeit, zwei Relaxationen, zentralisiert)

Frühere Arbeiten

- Verfahren auf Basis von ILPs (exakt / Approximationsschemata)
- superlineare Algorithmen (schlechtere / ohne Approximationsgüte)
- Beweis der $\mathcal{NP}$-Vollständigkeit [CTLW'05]

Sensoreinsatzplanung

[ALGOSENSORS'10]



Beiträge dieser Arbeit

- Beweis der $\mathcal{NP}$-Vollständigkeit
  (über wMDS auf UDGs, Dualität von LPs)
- effizient polynomielles Approximationsschema (EPTAS)
  (lineare Laufzeit, zwei Relaxationen, zentralisiert)

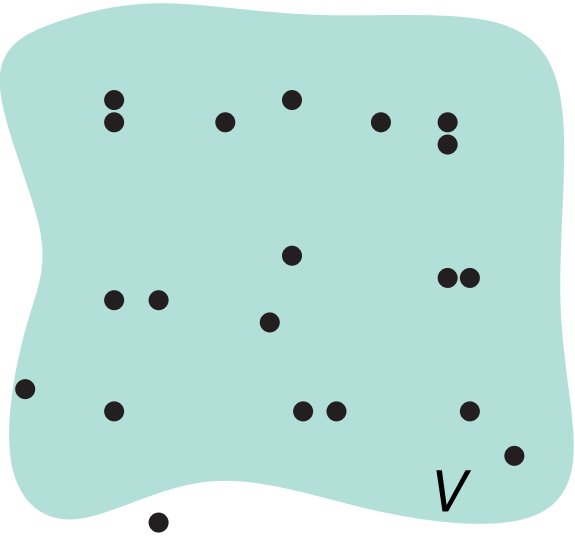
Frühere Arbeiten

- Verfahren auf Basis von ILPs (exakt / Approximationsschemata)
- superlineare Algorithmen (schlechtere / ohne Approximationsgüte)
- Beweis der $\mathcal{NP}$-Vollständigkeit [CTLW'05] $\rightarrow$ unvollständig
  (keine Beachtung von Geometrie, Batteriekapazitäten)

Ablauf

Approximationsalgorithmus

Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)



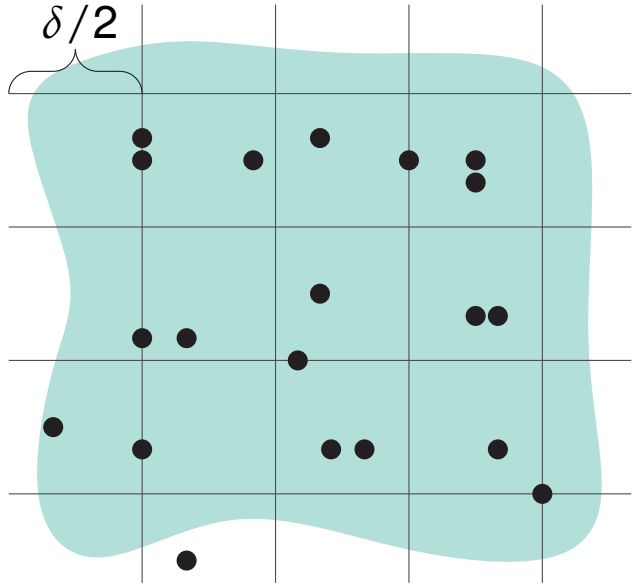
- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Ablauf

Approximationsalgorithmus

Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)



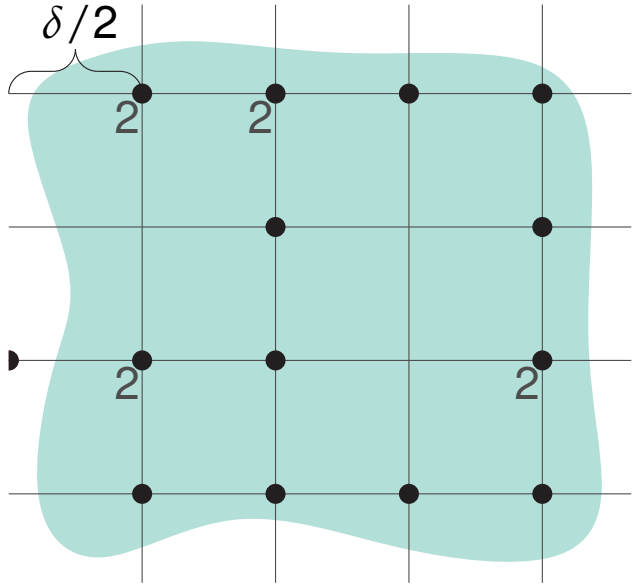
- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Ablauf

Approximationsalgorithmus

Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)



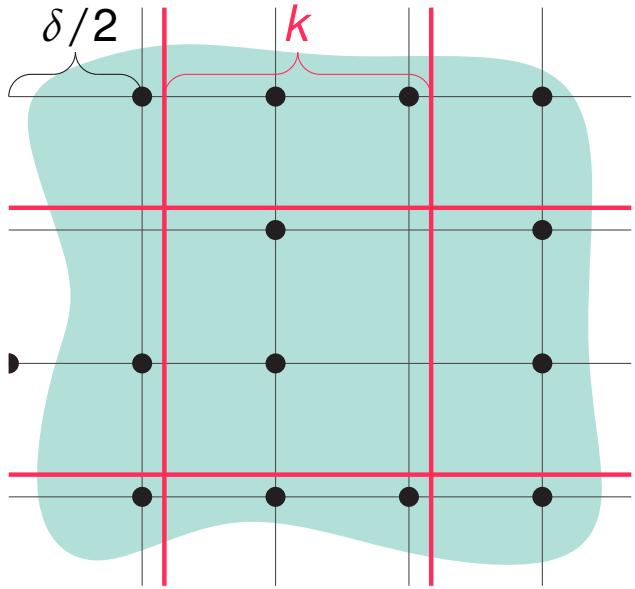
- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Ablauf

Approximationsalgorithmus

Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)



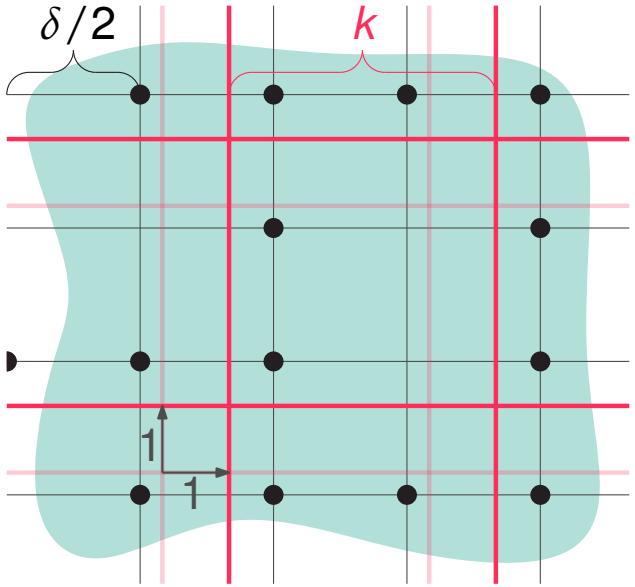
- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Ablauf

Approximationsalgorithmus

Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)



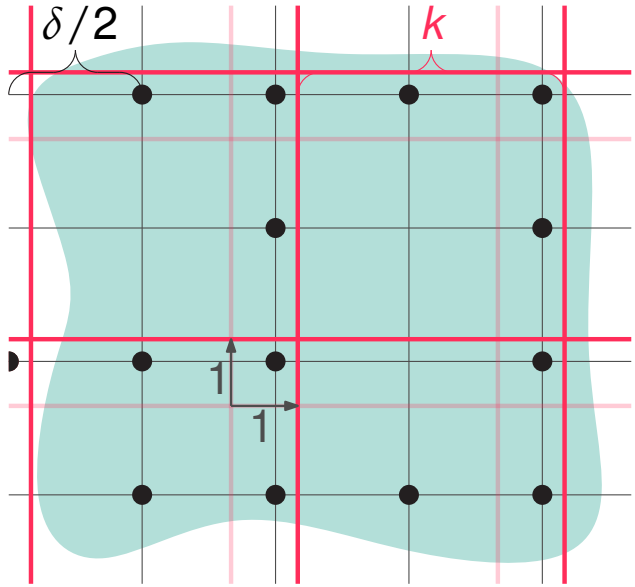
- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Ablauf

Approximationsalgorithmus

Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)



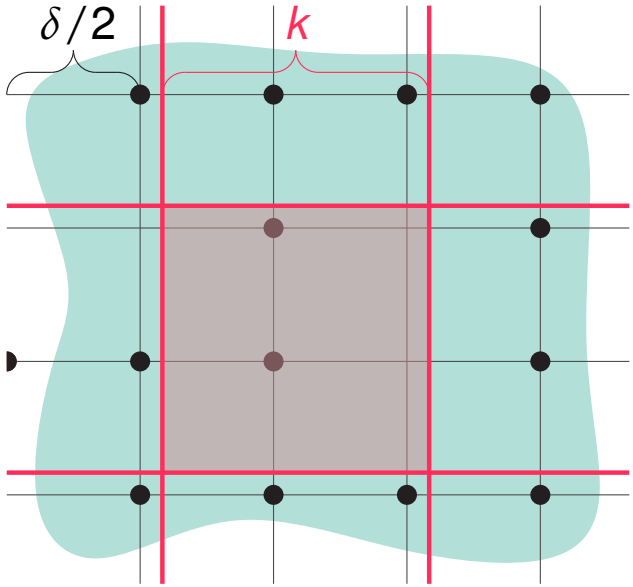
- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Ablauf

Approximationsalgorithmus

Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)



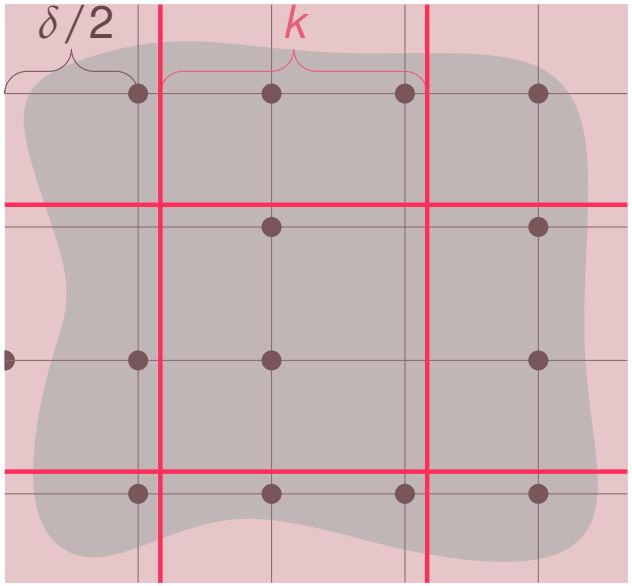
- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Ablauf

Approximationsalgorithmus

Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)

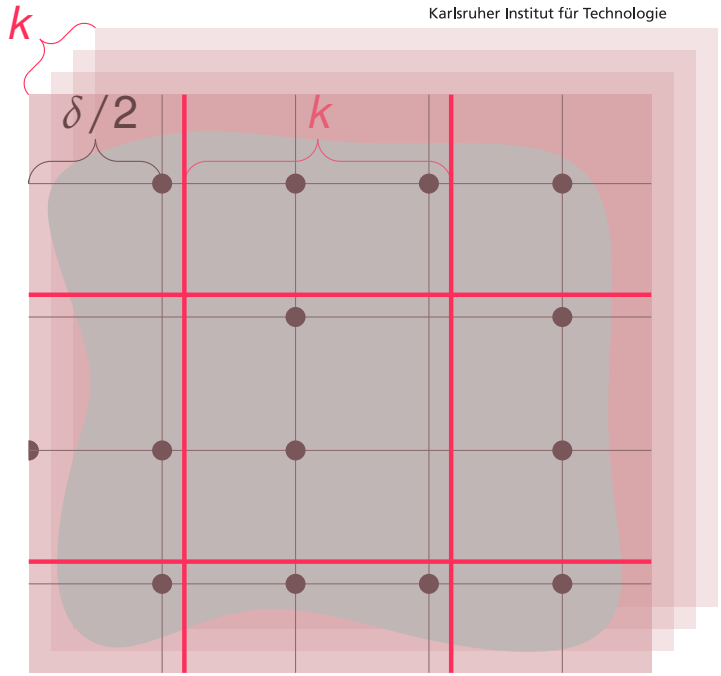


- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Ablauf

Approximationsalgorithmus



Eingabe: Instanz $(V, 1)$, Algorithmus $\mathcal{A}$
(Beobachtungsradien auf 1 normiert)

- bewege Knoten auf Gitterplätze
- partitioniere Gebiet $k = \lceil \frac{10}{\epsilon} \rceil$ mal
- löse Problem unabhängig für jede Zelle
  (mit Algorithmus $\mathcal{A}$)
- füge Teillösungen für jede Partitionierung zusammen
- skaliere Aktivierungsdauern mit $\frac{1-\epsilon}{k}$

Ausgabe: approximierte Lösung für $(V, (1+\delta) \cdot 1)$
(zulässiger Einsatzplan für jeden Knoten mit fast optimaler Lebensdauer)

Behauptungen

Approximationsalgorithmus

- lineare Laufzeit

$$\mathcal{O}\left(|V| + \tfrac{1}{\epsilon}|V| \cdot t_{\mathcal{A}}(\mathcal{O}(\tfrac{1}{\delta^2\epsilon^2}))\right) = \mathcal{O}(|V|)$$

- Approximationsgüte

$$T\langle V, 1+\delta\rangle \geq (1-\epsilon) \cdot f \cdot T_{\text{opt}}\langle V, 1\rangle$$

- Korrektheit

Energieverbrauch kleiner als Batteriekapazität

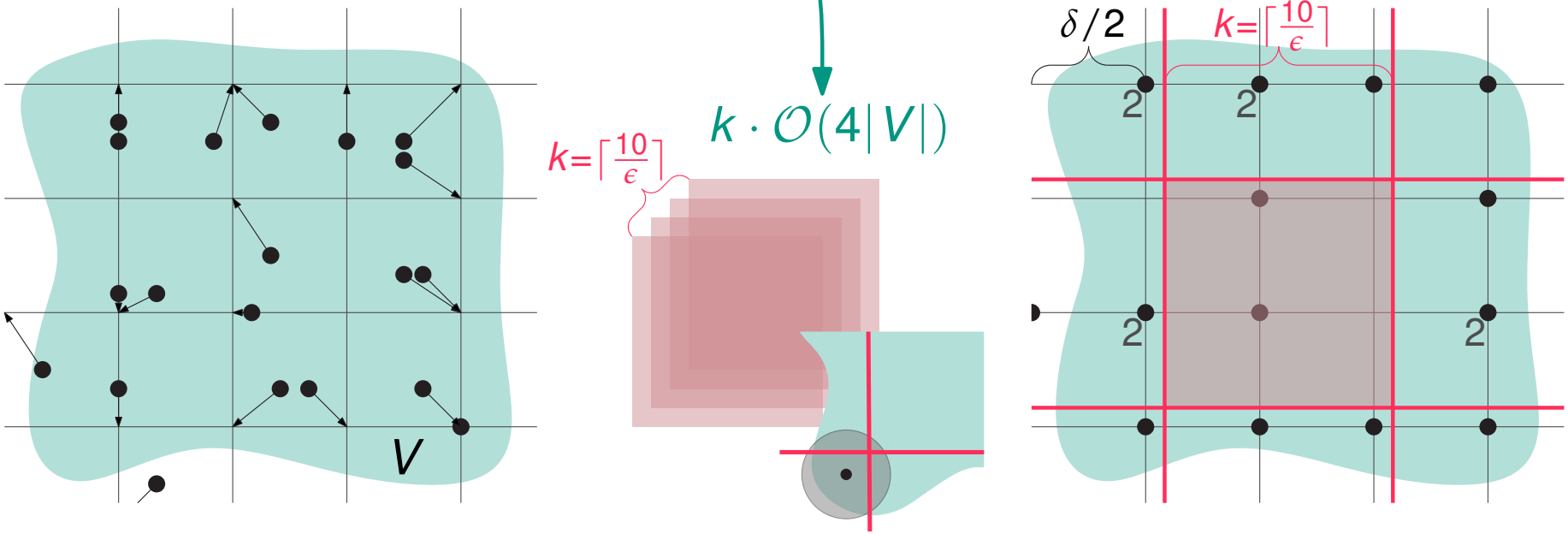
Behauptungen

Approximationsalgorithmus

- lineare Laufzeit

Algorithmus $\mathcal{A}$

$$\mathcal{O}\left(|V| + \tfrac{1}{\epsilon}|V| \cdot t_{\mathcal{A}}(\mathcal{O}(\tfrac{1}{\delta^2\epsilon^2}))\right) = \mathcal{O}(|V|)$$



Behauptungen

Approximationsalgorithmus

- lineare Laufzeit

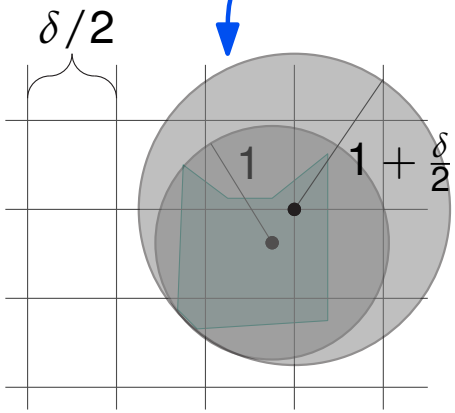
$$\mathcal{O}\left(|V| + \tfrac{1}{\epsilon}|V| \cdot t_{\mathcal{A}}(\mathcal{O}(\tfrac{1}{\delta^2\epsilon^2}))\right) = \mathcal{O}(|V|)$$

- Approximationsgüte

optimale Lösung

$$T\langle V, 1+\delta\rangle \geq (1-\epsilon) \cdot f \cdot T_{\text{opt}}\langle V, 1\rangle$$

Algorithmus $\mathcal{A}$



$k \cdot \frac{1-\epsilon}{k}$ k

Behauptungen

Approximationsalgorithmus

- lineare Laufzeit

$$\mathcal{O}\left(|V| + \tfrac{1}{\epsilon}|V| \cdot t_{\mathcal{A}}(\mathcal{O}(\tfrac{1}{\delta^2\epsilon^2}))\right) = \mathcal{O}(|V|)$$

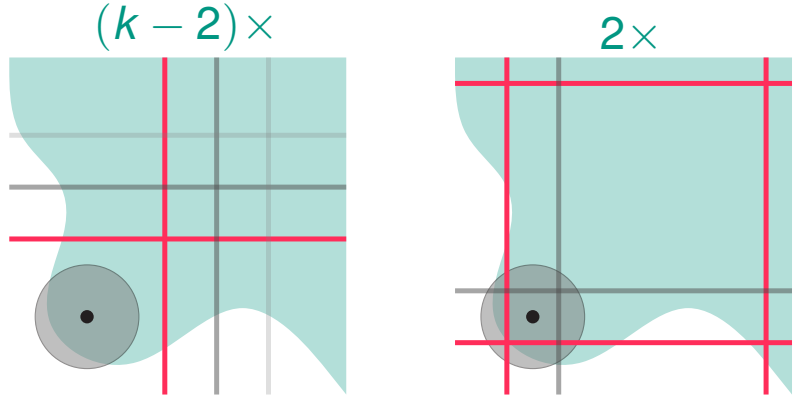
- Approximationsgüte

$$T\langle V, 1+\delta\rangle \geq (1-\epsilon) \cdot f \cdot T_{\text{opt}}\langle V, 1\rangle$$

- Korrektheit

Energieverbrauch

$$((k-2)\cdot 1 + 2\cdot 4) \overset{!}{\leq} 1$$



Behauptungen

Approximationsalgorithmus

- lineare Laufzeit

$$\mathcal{O}\left(|V| + \tfrac{1}{\epsilon}|V| \cdot t_{\mathcal{A}}(\mathcal{O}(\tfrac{1}{\delta^2\epsilon^2}))\right) = \mathcal{O}(|V|)$$

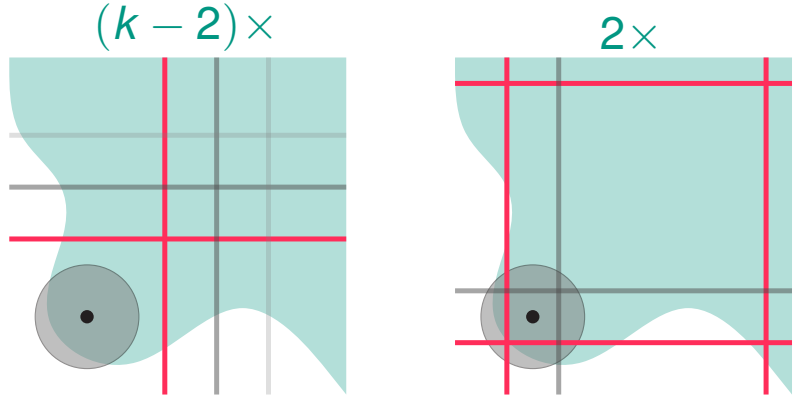
- Approximationsgüte

$$T\langle V, 1+\delta\rangle \geq (1-\epsilon) \cdot f \cdot T_{\text{opt}}\langle V, 1\rangle$$

- Korrektheit

Energieverbrauch

$$\tfrac{1-\epsilon}{k} \cdot \left((k-2)\cdot 1 + 2\cdot 4\right) \leq 1 \qquad k = \lceil \tfrac{10}{\epsilon} \rceil$$



Ergebnisse

Approximationsalgorithmus

- lineare Laufzeit

$$\mathcal{O}\left(|V| + \tfrac{1}{\epsilon}|V| \cdot t_{\mathcal{A}}(\mathcal{O}(\tfrac{1}{\delta^2\epsilon^2}))\right) = \mathcal{O}(|V|)$$

- Approximationsgüte

$$T\langle V, 1+\delta\rangle \geq (1-\epsilon) \cdot f \cdot T_{\text{opt}}\langle V, 1\rangle$$

- Korrektheit

Energieverbrauch kleiner als Batteriekapazität

$\rightarrow$ effizient polynomielles Approximationsschema (EPTAS)

Weitere Ergebnisse

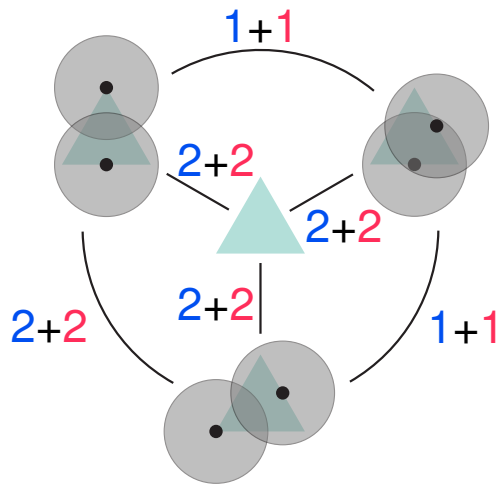
exakter Algorithmus

- Formulierung als lineares Programm [BCSZ'05]
  - verzögerte Spaltenerzeugung [DW'60]
  - Garg-Könemann Algorithmus [GK'07]
- effizient für kleinere Instanzen ($\approx$ 1 000 Knoten)

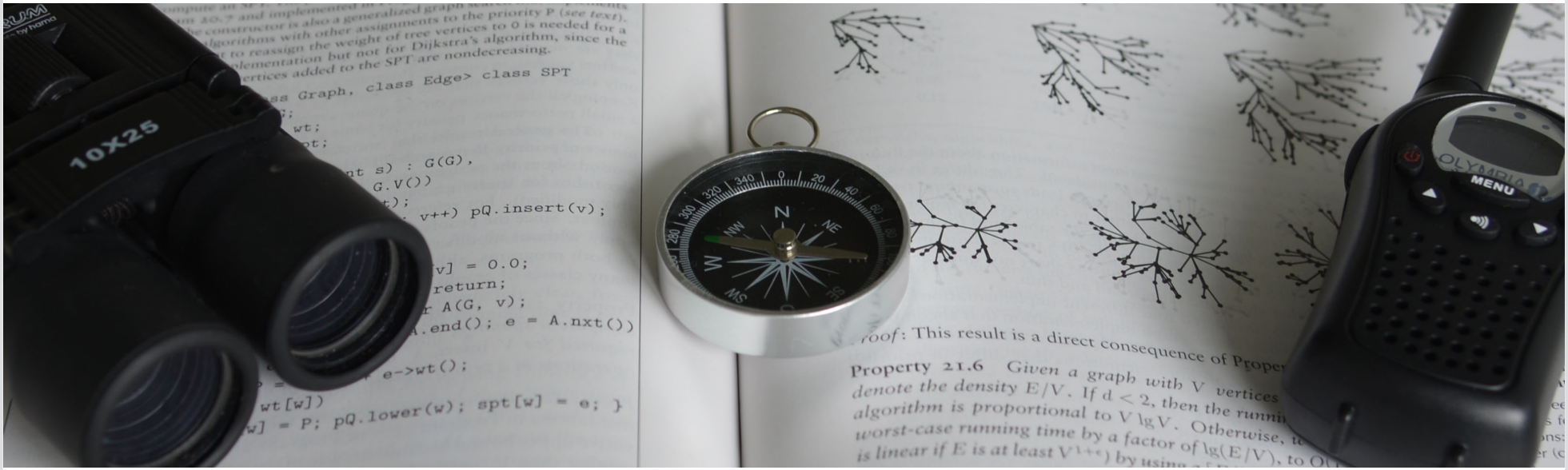
$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} \mathbf{t}_j \,, \\ \text{subject to} \quad & \sum_{j=1}^{m} \mathcal{C}_{ij}\mathbf{t}_j \leq \mathbf{b}_i, \; i \in \{1, \ldots, n\} \,, \\ & \mathbf{t} \geq \mathbf{0} \end{aligned}$$

Minimierung von Ein- und Ausschaltvorgängen

- Verfahren zur Nachoptimierung
- Formulierung als metrisches TSP



An Algorithmic View on Sensor Networks –
Surveillance, **Localization**, and Communication



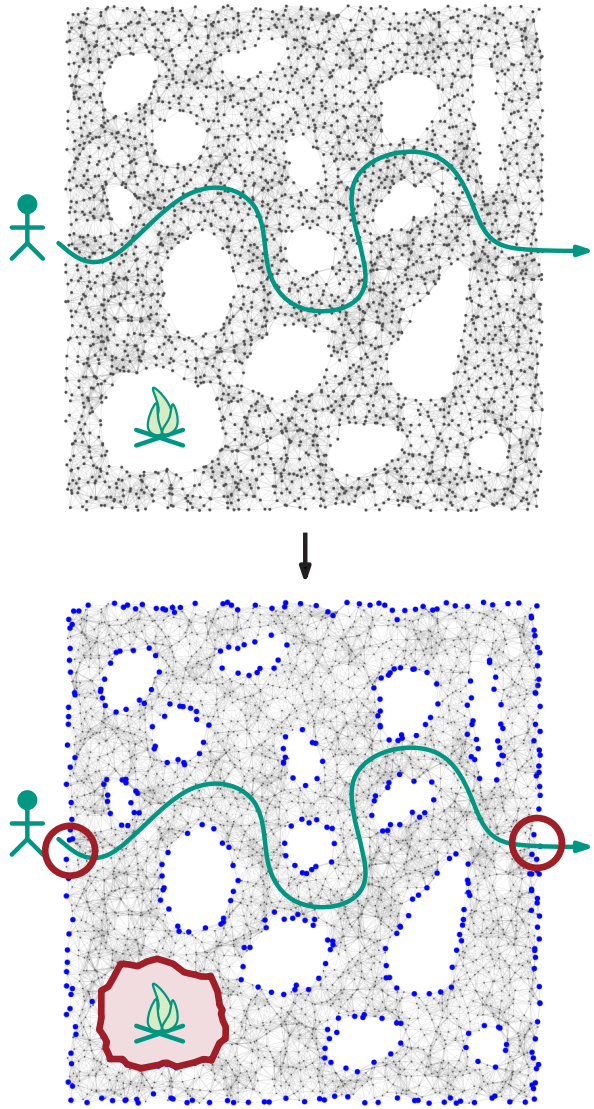
Randerkennung

[SEA'11], [TR'11]

Problemstellung

- großflächiges Sensornetzwerk

→ Klassifikation in Rand- und innere Knoten



Anwendungen

- Melden von Eindringlingen (z.B. Wilderer), sich verschiebender Ränder (z.B. Knotenausfall)
- Indikator für schlechte Überdeckung

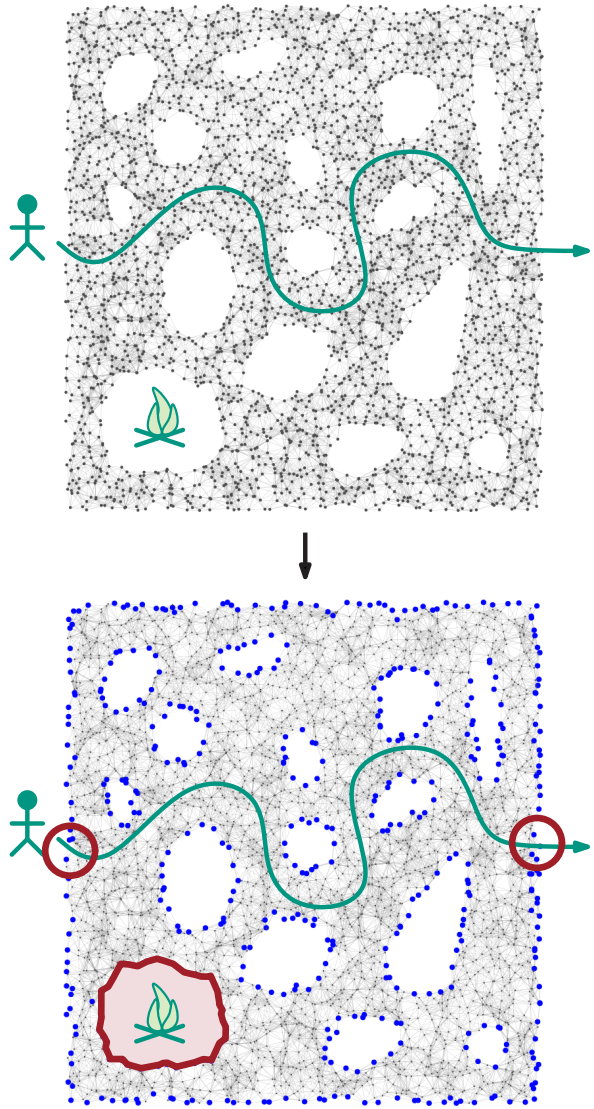
Randerkennung

[SEA'11], [TR'11]

Problemstellung

- großflächiges Sensornetzwerk
  - verteilte Berechnung
  - lokale Sicht auf das Netzwerk
  - nur Verbindungsinformation (kein GPS)

→ Klassifikation in Rand- und innere Knoten



Anwendungen

- Melden von Eindringlingen (z.B. Wilderer), sich verschiebender Ränder (z.B. Knotenausfall)
- Indikator für schlechte Überdeckung

Randerkennung

[SEA'11], [TR'11]

geometrische Ansätze

- Positionen, Distanzen, Winkel

→ Informationen oft nicht verfügbar

statistische Ansätze

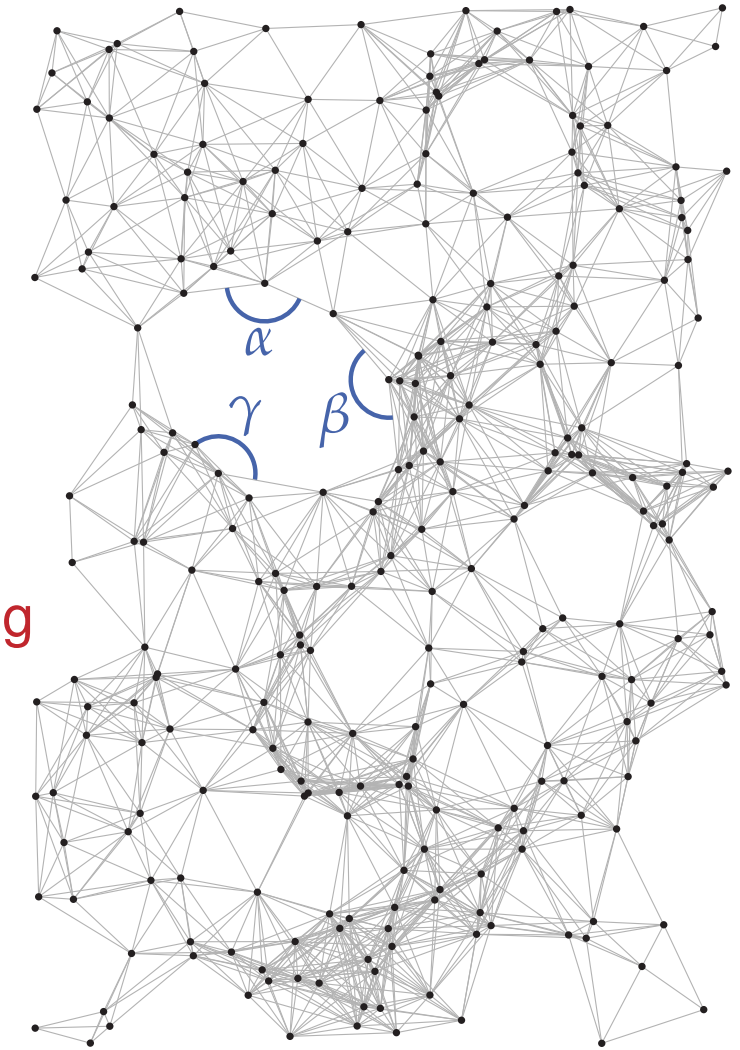
- Knotengrad, Knotendichte

→ hoher Knotengrad, gleichförmige Verteilung

topologische Ansätze

- Struktur des Verbindungsgraphen

→ komplex, viel Kommunikation



Randerkennung

[SEA'11], [TR'11]

geometrische Ansätze

- Positionen, Distanzen, Winkel

→ Informationen oft nicht verfügbar

statistische Ansätze

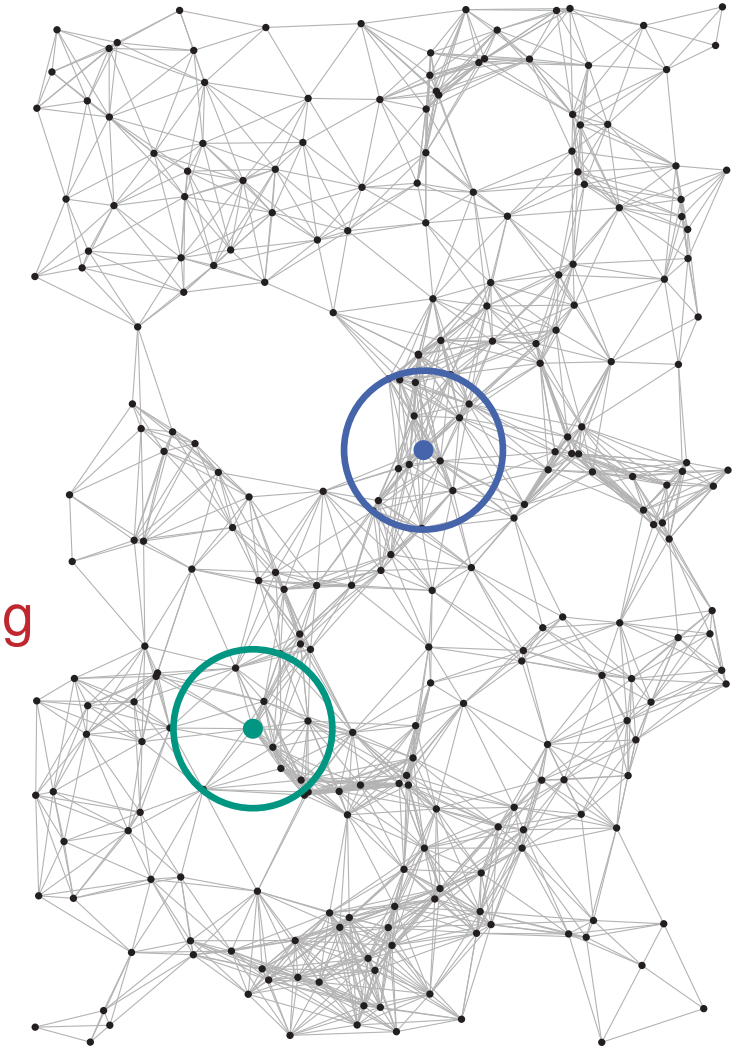
- Knotengrad, Knotendichte

→ hoher Knotengrad, gleichförmige Verteilung

topologische Ansätze

- Struktur des Verbindungsgraphen

→ komplex, viel Kommunikation



Randerkennung

[SEA'11], [TR'11]

geometrische Ansätze

- Positionen, Distanzen, Winkel

→ Informationen oft nicht verfügbar

statistische Ansätze

- Knotengrad, Knotendichte

→ hoher Knotengrad, gleichförmige Verteilung

topologische Ansätze

- Struktur des Verbindungsgraphen

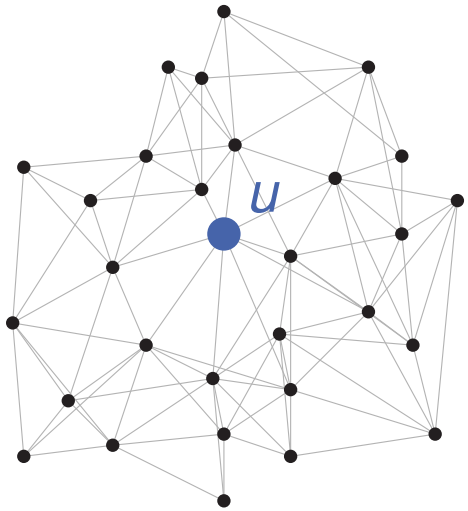
→ komplex, viel Kommunikation

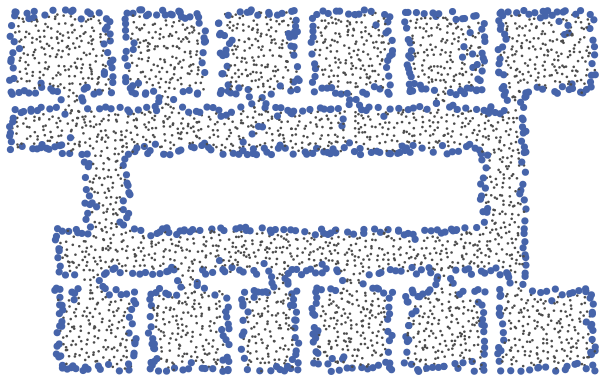
Randerkennung

[SEA'11], [TR'11]

Beitrag dieser Arbeit

- Multidimensional Scaling Boundary Recognition (MDS-BR)
  (kombiniert topologische & geometrische Ansätze)
  - nur Verbindungsinformationen
  - geringer Ressourcenverbrauch
    (wenig Kommunikation, Rechenleistung)
  - verteilt, lokal (2-Hop Nachbarschaft)
  - sehr gute Klassifikationsleistung in Praxis





Ablauf

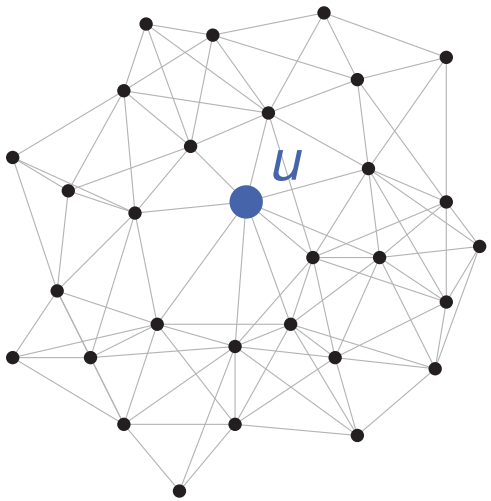
MDS-BR

Einbettung der 2-Hop Nachbarschaft

- erfrage Verbindungsinformation
- schätze Distanzen
- berechne Einbettung

Test von Winkelbeziehungen

- 2-Hop Nachbarschaft
- 1-Hop Nachbarschaft
- Mikrolöcher



$\Rightarrow$

2 Nachrichten / Knoten

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & 1 & 0 & \cdots \\ 1 & 0 & 0 & 0 & 1 & \cdots \\ 0 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

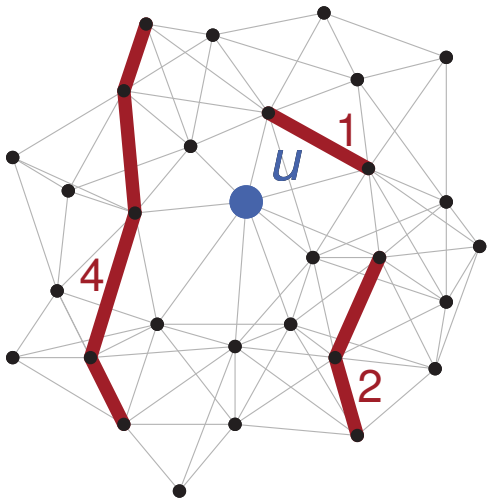
Adjazenzmatrix

Ablauf

MDS-BR

Einbettung der 2-Hop Nachbarschaft

- erfrage Verbindungsinformation
- schätze Distanzen
- berechne Einbettung



$\Rightarrow$

Test von Winkelbeziehungen

- 2-Hop Nachbarschaft
- 1-Hop Nachbarschaft
- Mikrolöcher

$$\begin{pmatrix} 0 & 1 & 1 & 2 & 3 & \cdots \\ 1 & 0 & 3 & 1 & 2 & \cdots \\ 1 & 3 & 0 & 2 & 1 & \cdots \\ 2 & 1 & 2 & 0 & 4 & \cdots \\ 3 & 2 & 1 & 4 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Matrix mit Hop-Distanzen

Ablauf

MDS-BR

Einbettung der 2-Hop Nachbarschaft

- erfrage Verbindungsinformation
- schätze Distanzen
- berechne Einbettung

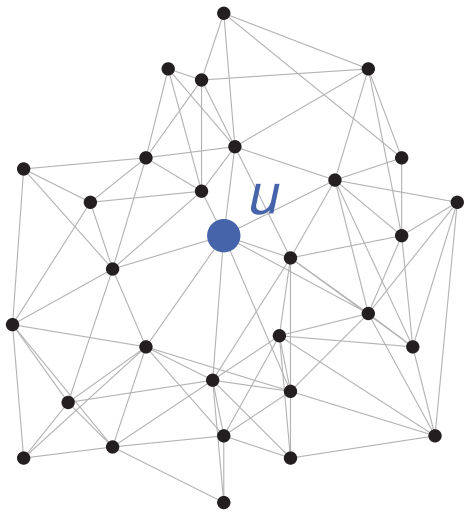
Test von Winkelbeziehungen

- 2-Hop Nachbarschaft
- 1-Hop Nachbarschaft
- Mikrolöcher

$$\begin{pmatrix} 0 & 1 & 1 & 2 & 3 & \cdots \\ 1 & 0 & 3 & 1 & 2 & \cdots \\ 1 & 3 & 0 & 2 & 1 & \cdots \\ 2 & 1 & 2 & 0 & 4 & \cdots \\ 3 & 2 & 1 & 4 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

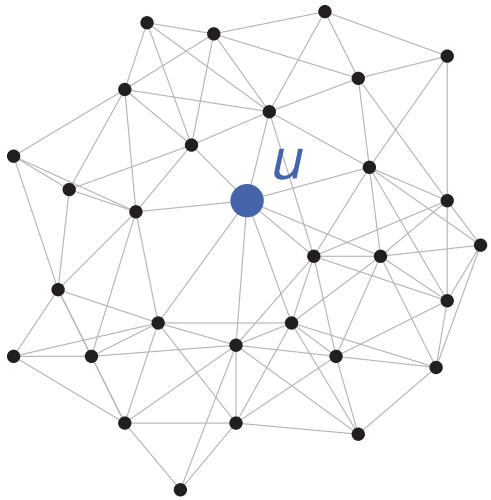
paarweise Distanzen δ_{ij}

$\Rightarrow$ MDS



Einbettung $\{x_1, \ldots, x_n\}$

$\neq$



echte Positionen

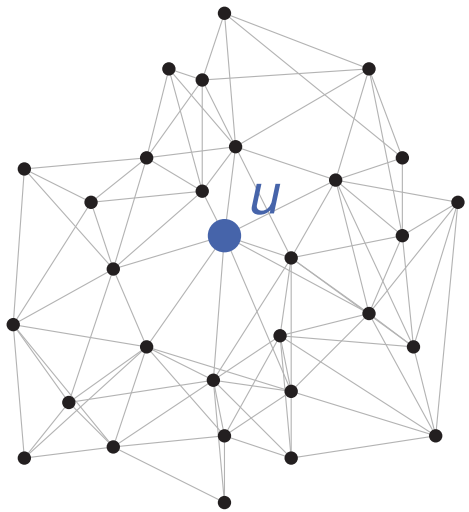
minimiere $\sum_{i \neq j} (||x_i - x_j|| - \delta_{ij})^2$

Ablauf

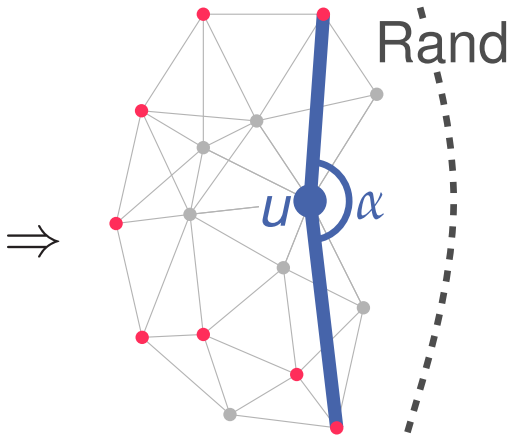
MDS-BR

Einbettung der 2-Hop Nachbarschaft

- erfrage Verbindungsinformation
- schätze Distanzen
- berechne Einbettung



geschätzte Positionen



$\alpha \geq \alpha_{min}$

Test von Winkelbeziehungen

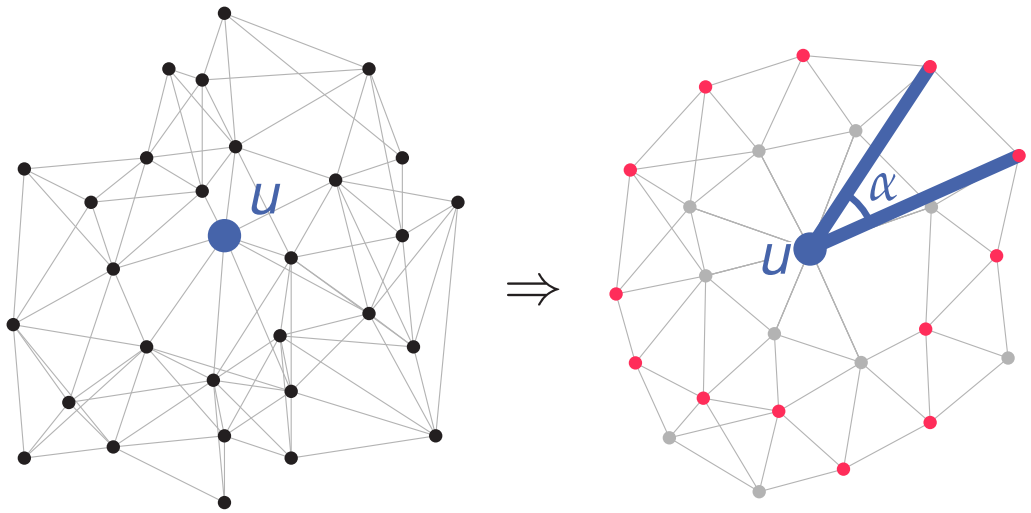
- 2-Hop Nachbarschaft
- 1-Hop Nachbarschaft
- Mikrolöcher

Ablauf

MDS-BR

Einbettung der 2-Hop Nachbarschaft

- erfrage Verbindungsinformation
- schätze Distanzen
- berechne Einbettung



geschätzte Positionen

$\alpha \geq \alpha_{min}$

Test von Winkelbeziehungen

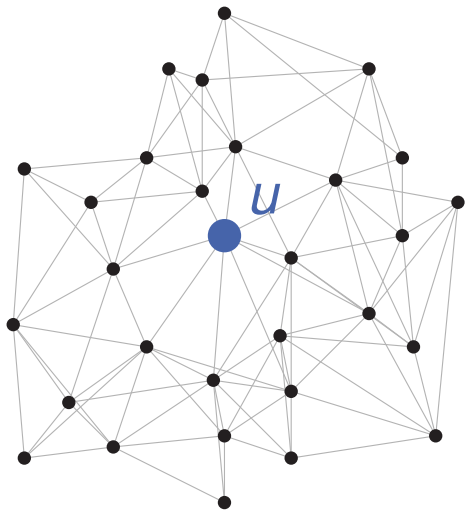
- 2-Hop Nachbarschaft
- 1-Hop Nachbarschaft
- Mikrolöcher

Ablauf

MDS-BR

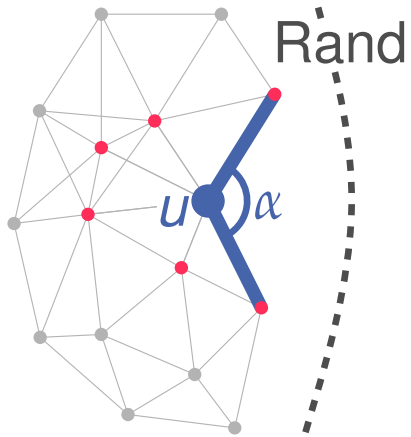
Einbettung der 2-Hop Nachbarschaft

- erfrage Verbindungsinformation
- schätze Distanzen
- berechne Einbettung



geschätzte Positionen

$\Rightarrow$



$\alpha \geq \alpha_{min}$

Test von Winkelbeziehungen

- 2-Hop Nachbarschaft
- 1-Hop Nachbarschaft
- Mikrolöcher

Ablauf

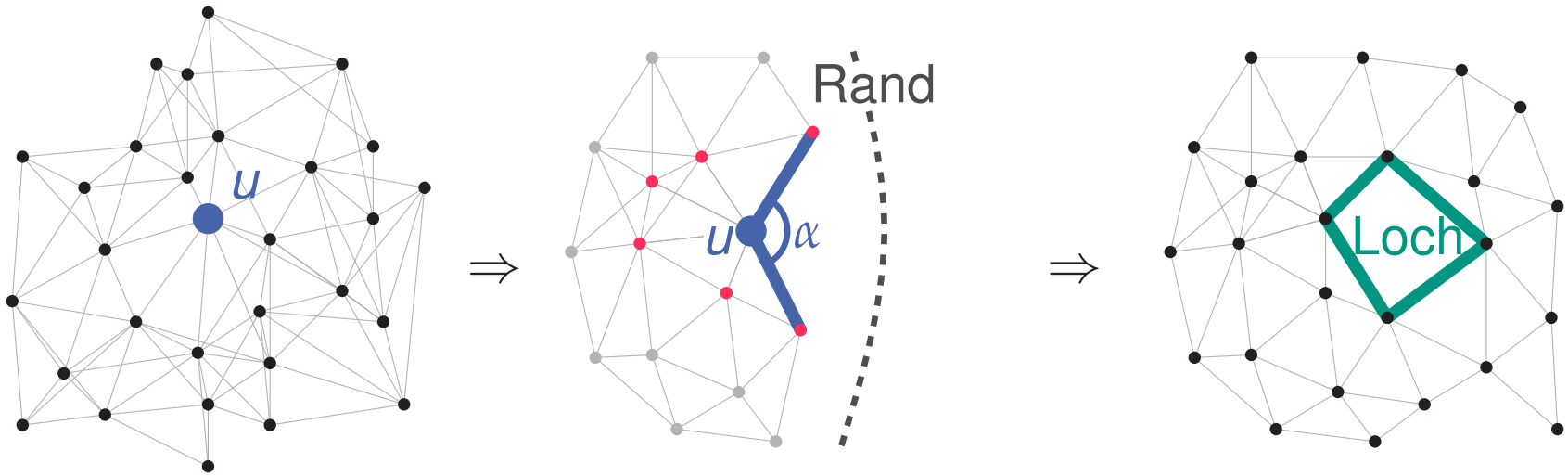
MDS-BR

Einbettung der 2-Hop Nachbarschaft

- erfrage Verbindungsinformation
- schätze Distanzen
- berechne Einbettung

Test von Winkelbeziehungen

- 2-Hop Nachbarschaft
- 1-Hop Nachbarschaft
- Mikrolöcher



geschätzte Positionen

$\alpha \geq \alpha_{min}$

verhindere Mikrolöcher

Ablauf

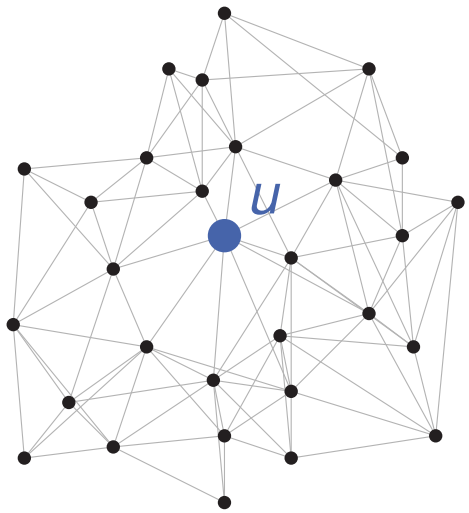
MDS-BR

Einbettung der 2-Hop Nachbarschaft

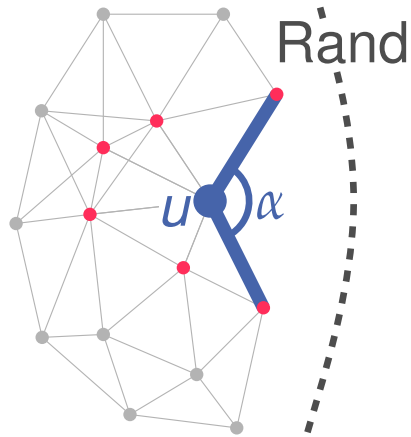
- erfrage Verbindungsinformation
- schätze Distanzen
- berechne Einbettung

Test von Winkelbeziehungen

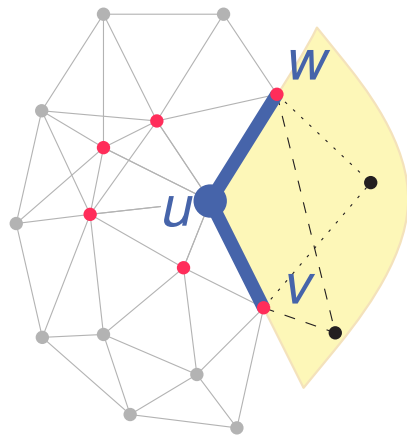
- 2-Hop Nachbarschaft
- 1-Hop Nachbarschaft
- Mikrolöcher



geschätzte Positionen



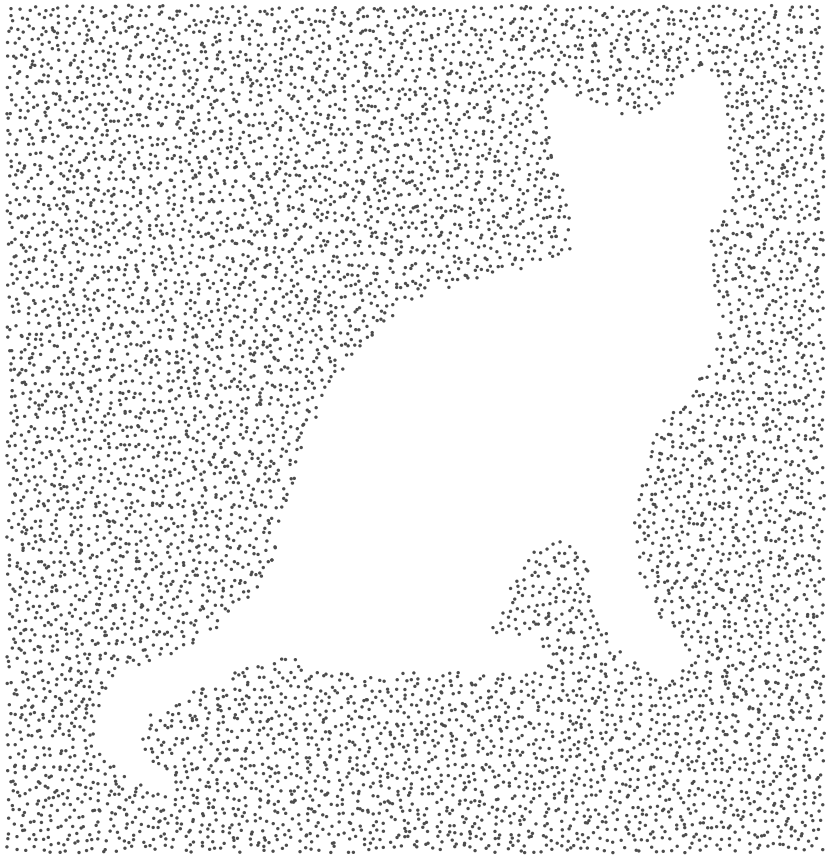
$\alpha \geq \alpha_{min}$



verbotener Bereich

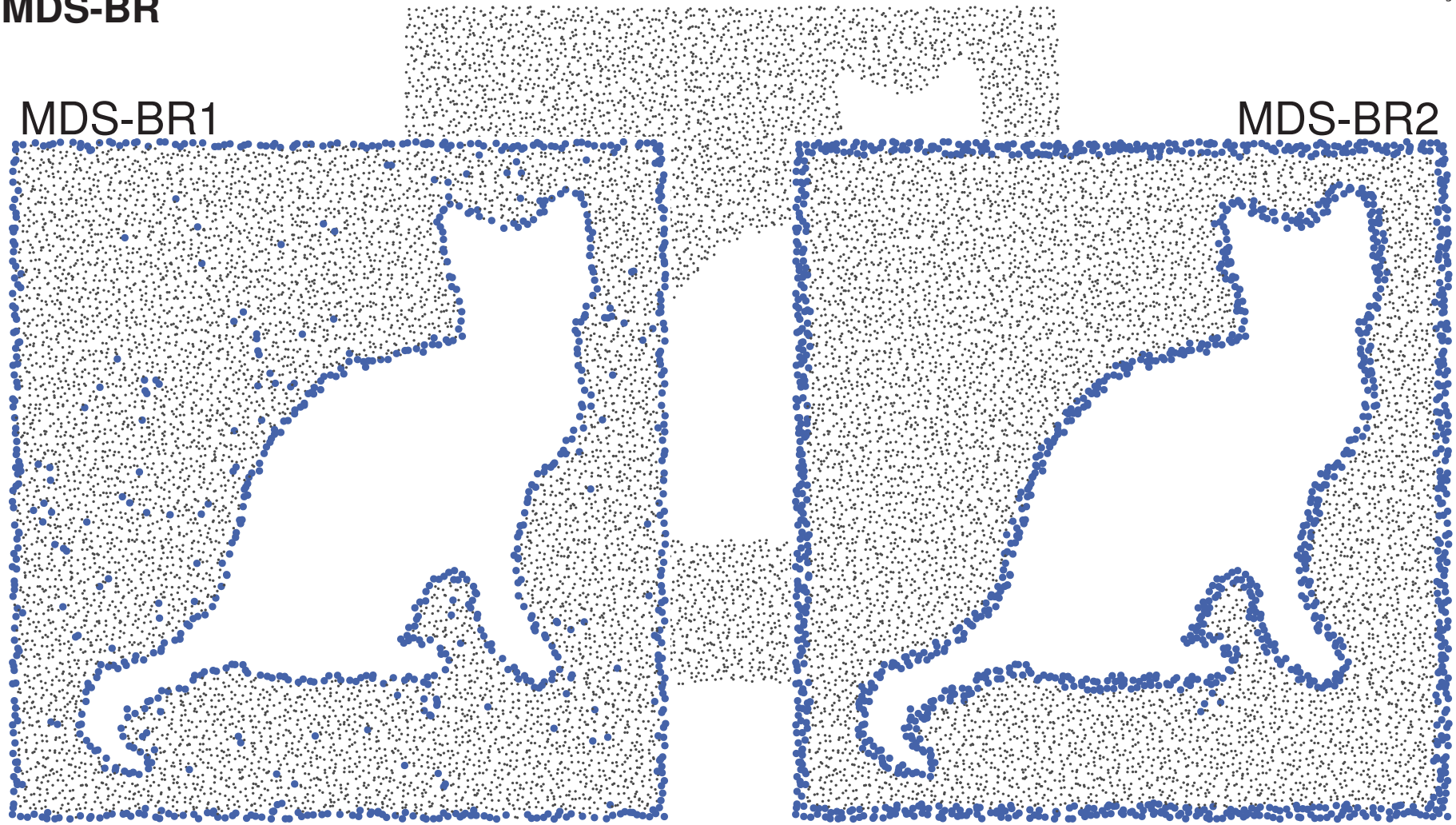
Beispiel

MDS-BR



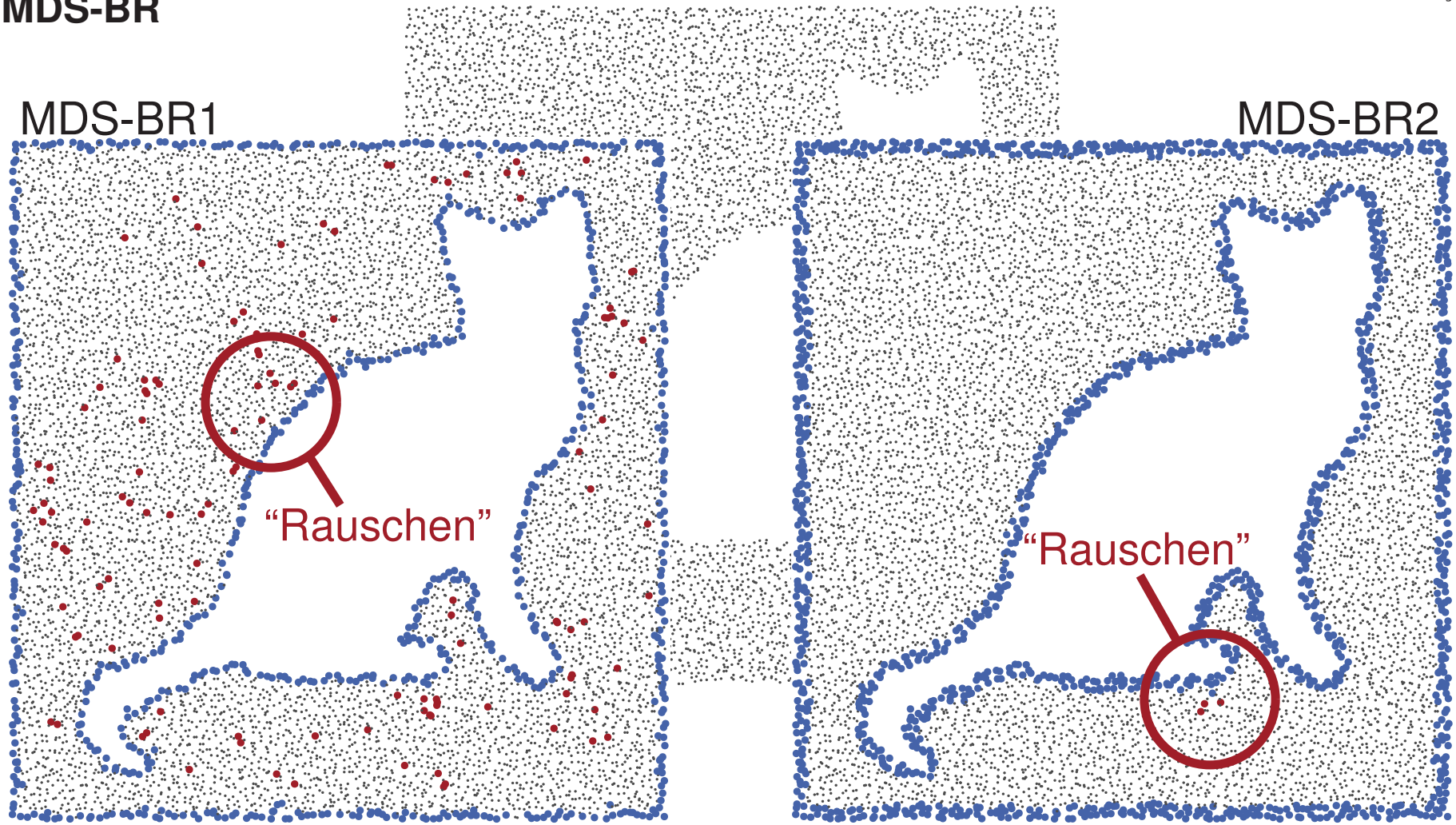
Beispiel

MDS-BR



Beispiel

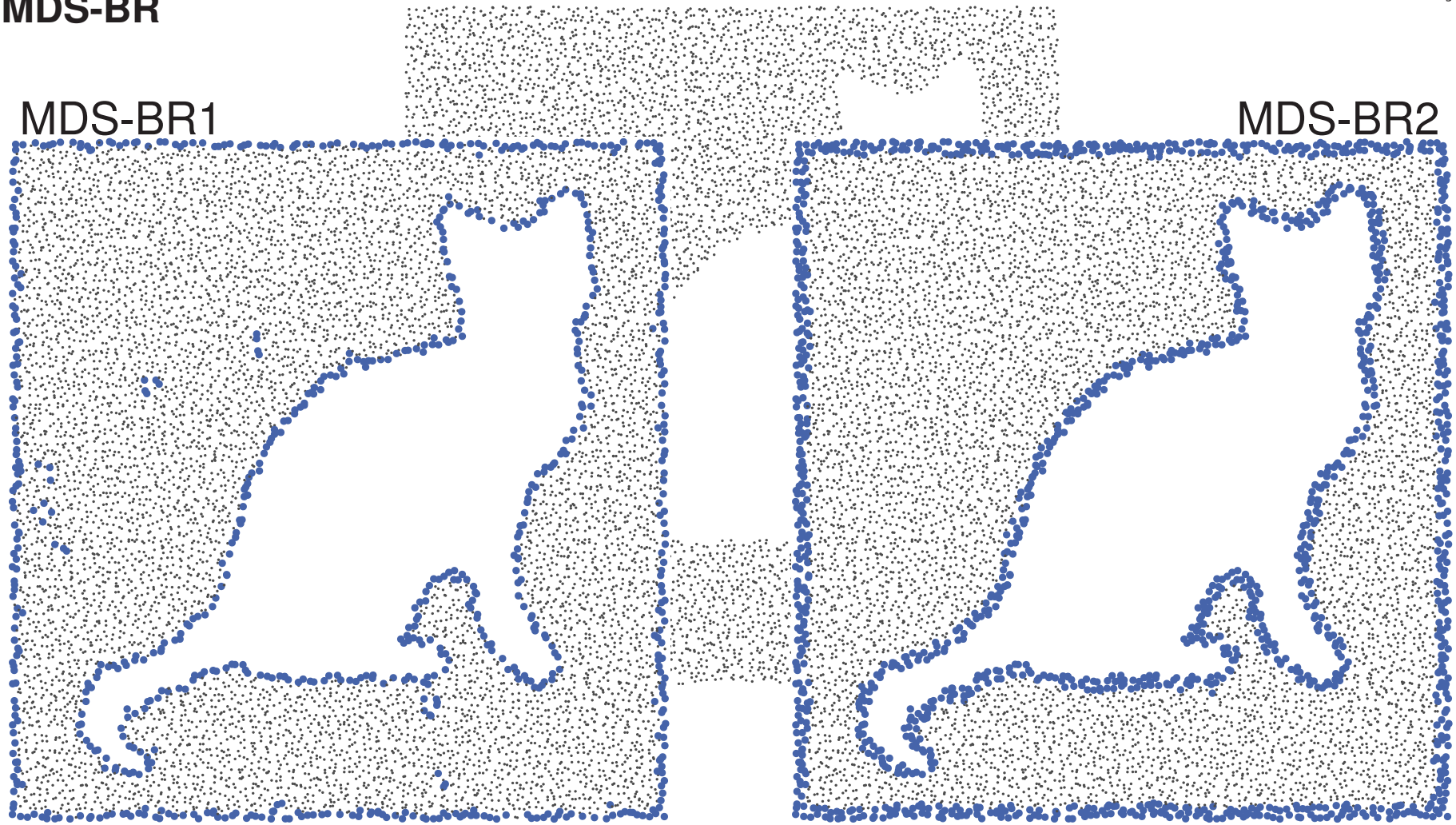
MDS-BR



potentiell ungewolltes Rauschen (z.B. Mikrolöcher)

Beispiel

MDS-BR



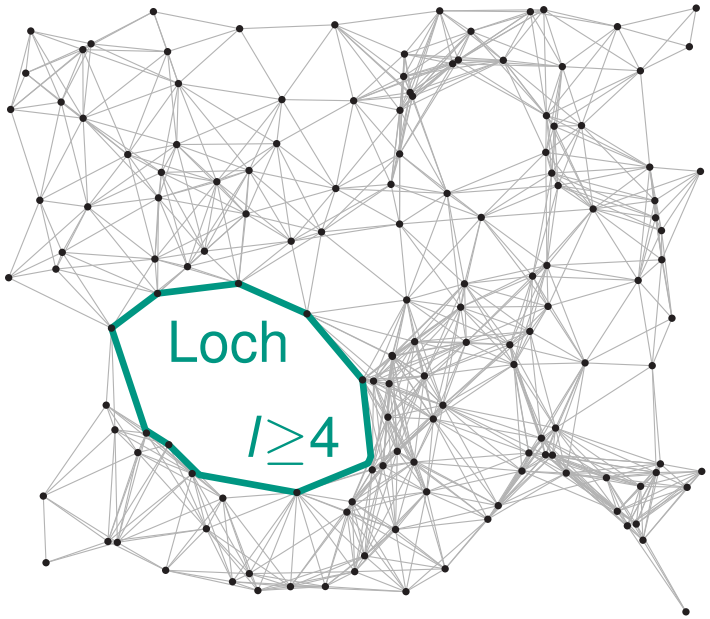
Nachoptimierung durch lokale Suche auf Randknoten

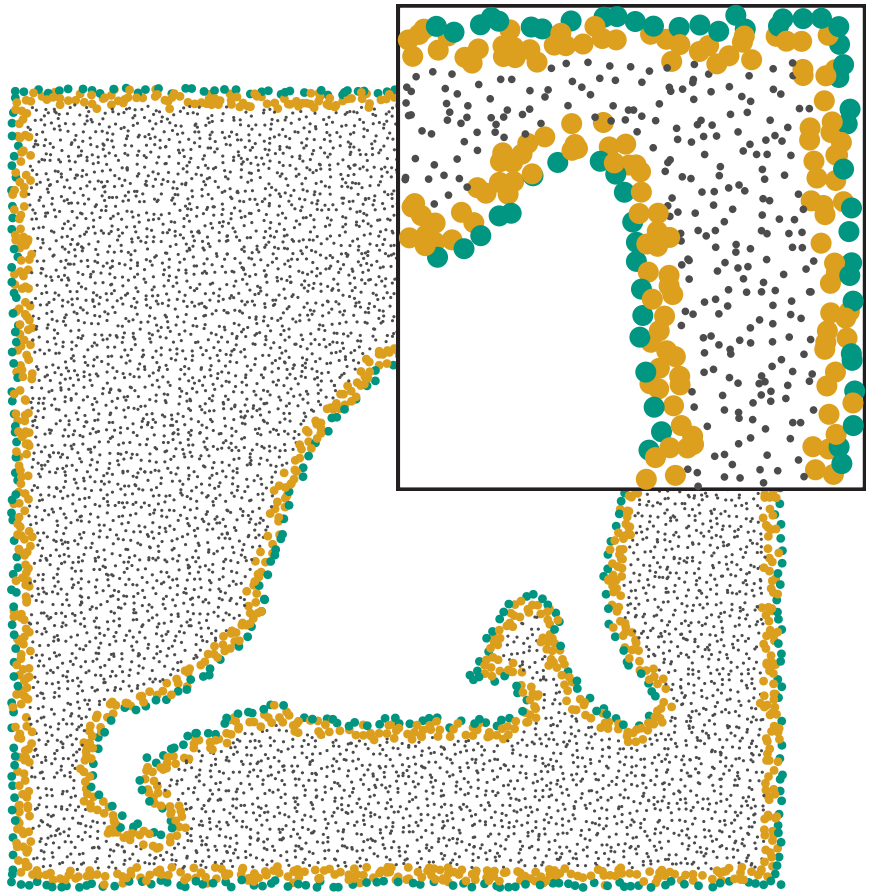
Ergebnisse

quantitative Auswertung

Loch- & Randdefinition

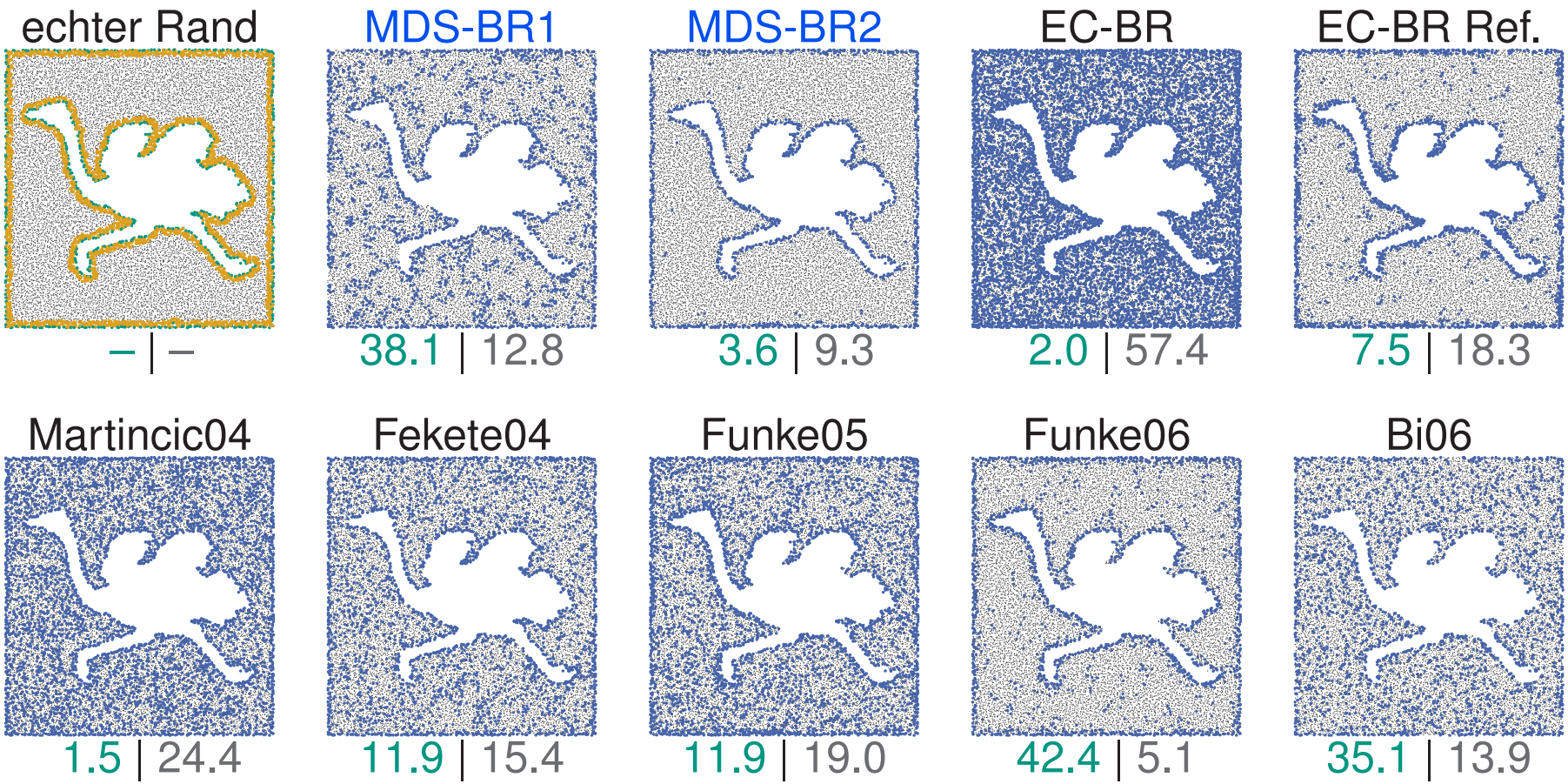
- echte Randknoten
- optionale Randknoten
- innere Knoten





Ergebnisse

quantitative Auswertung – 0.25-QUDG

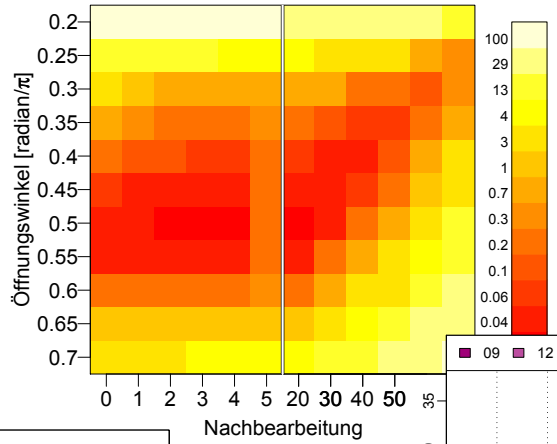


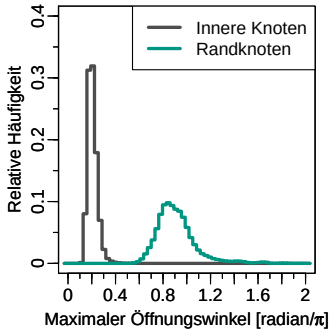
Fehlklassifikation von Randknoten und inneren Knoten [%]

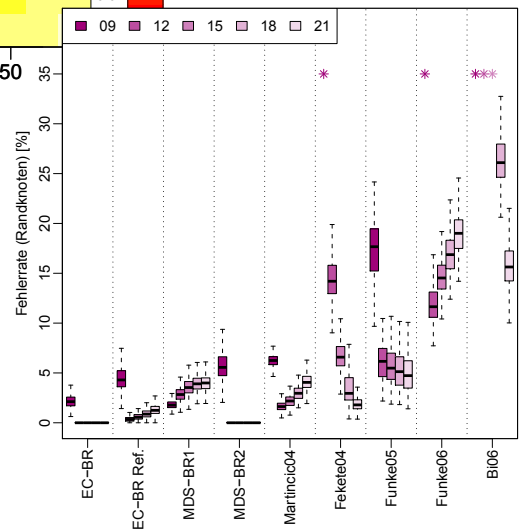
Weitere Ergebnisse

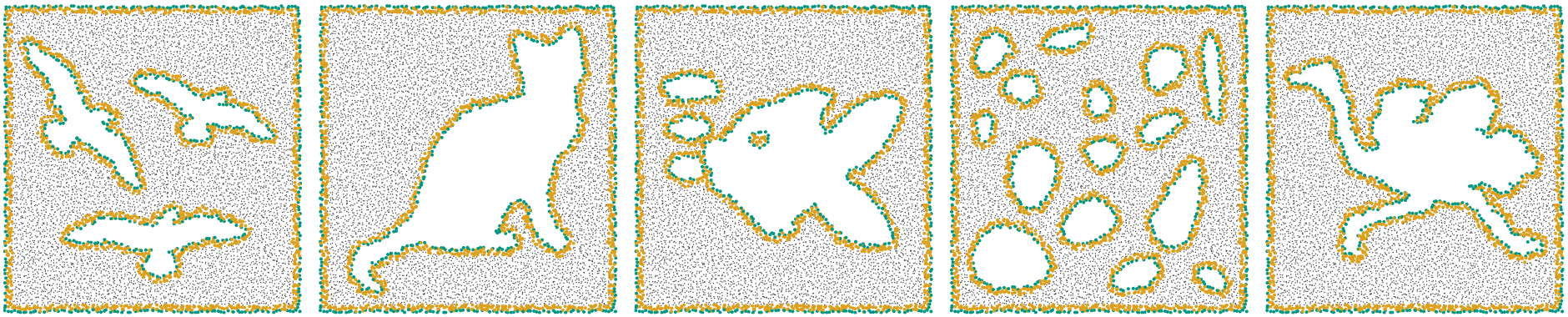
ausführliche Simulationen

- verschiedene Netzwerkmodelle
- Parameteranalysen
- algorithmische Varianten
  (z.B. Signalstärke, Sampling)









An Algorithmic View on Sensor Networks –
Surveillance, Localization, and **Communication**

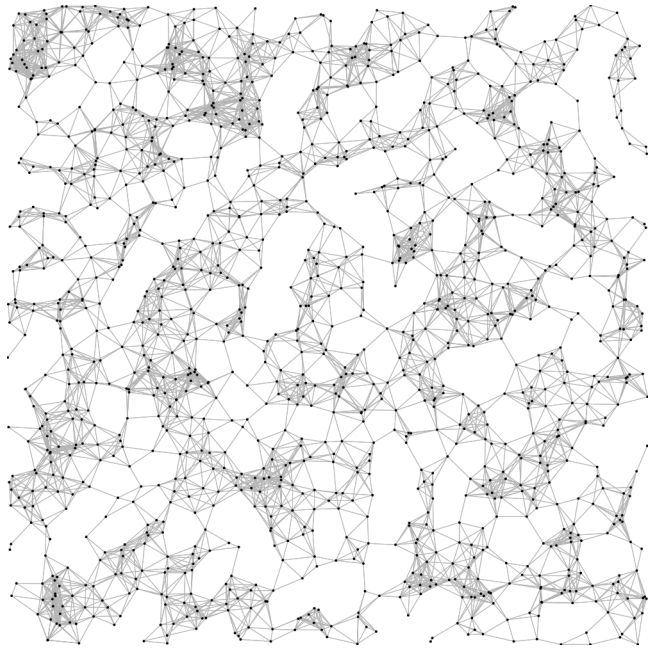
Effiziente Routingverfahren

[SOCS'10]

Problemstellung

- statisches Sensornetzwerk
  - Verbindungsinformation (kein GPS)
  - Verbindungskosten (z.B. Latenz, Energie)
- zentralisierte Verarbeitung

→ Bestimmung (fast) optimaler Routen in kurzer Zeit & großer Anzahl



Anwendungen

- Analysewerkzeuge (z.B. Bestimmung von Engstellen, Netzwerkkapazitäten)
- Simulationsumgebungen

Effiziente Routingverfahren

[SOCS'10]

Effiziente Verfahren für Straßennetzwerke

- basieren auf Dijkstras Algorithmus
  - zielgerichtet (ALT, Arc Flags)
  - hierarchisch (Contraction Hierarchies)
- zweistufiger Ansatz
  - Vorverarbeitung (Daten aggregieren)
  - Anfrageverarbeitung (Daten ausnutzen)

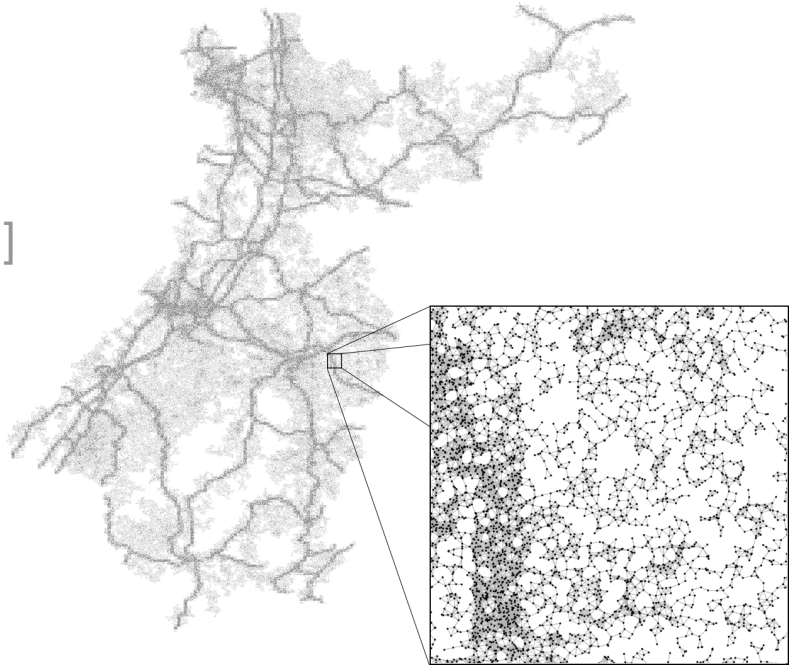
(nützlich für viele Anfragen auf der gleichen Instanz)

Problematisch auf Sensornetzwerkinstanzen [BDSSSW'10]

- dichte Graphen
- viele äquivalente/ähnliche Pfade

Effiziente Routingverfahren

[SOCS'10]



Grundlage

- Contraction Hierarchies (CH) [GSSV'12]
  - zweistufiger Ansatz
  - lange Vorverarbeitung
    (auf Sensornetzwerkinstanzen)

Beiträge dieser Arbeit

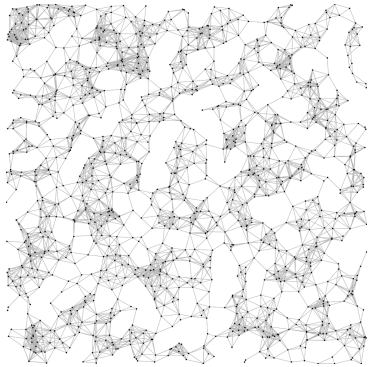
- modifizierte Vorverarbeitung
  - Beschleunigung für dichte Graphen
- approximierte Contraction Hierarchies (CH)
  - Routen maximal Faktor $(1+\epsilon)$ länger als das Optimum
  - geringerer Platzbedarf, kürzere Vorverarbeitungs- und Anfragedauern

Ergebnisse

Simulation

Netzwerkmodell

- “box”, Latenzkosten
- 1 000 000 Knoten, mittlerer Knotengrad 10



	Vorverarbeitung		Anfrage	
	Dauer [s]	Overhead [B/n]	Dauer [ms]	Fehler Ø [%]
exakt	1 071	−14	0.48	–
exakt (lazy updates)	254	−10	0.44	–
apx ($\epsilon = 1\%$)	166	−17	0.38	0.2
apx ($\epsilon = 10\%$)	116	−29	0.32	2.1

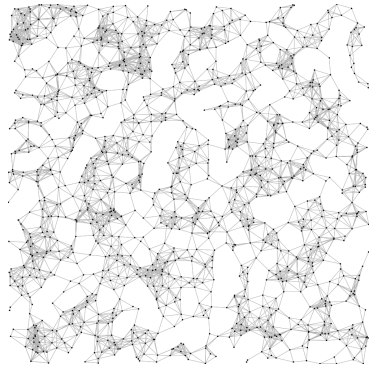
Intel Core i7-920 @ 2.67 Ghz, 12 GB RAM

Ergebnisse

Simulation

Netzwerkmodell

- “box”, Latenzkosten
- 1 000 000 Knoten, mittlerer Knotengrad 10



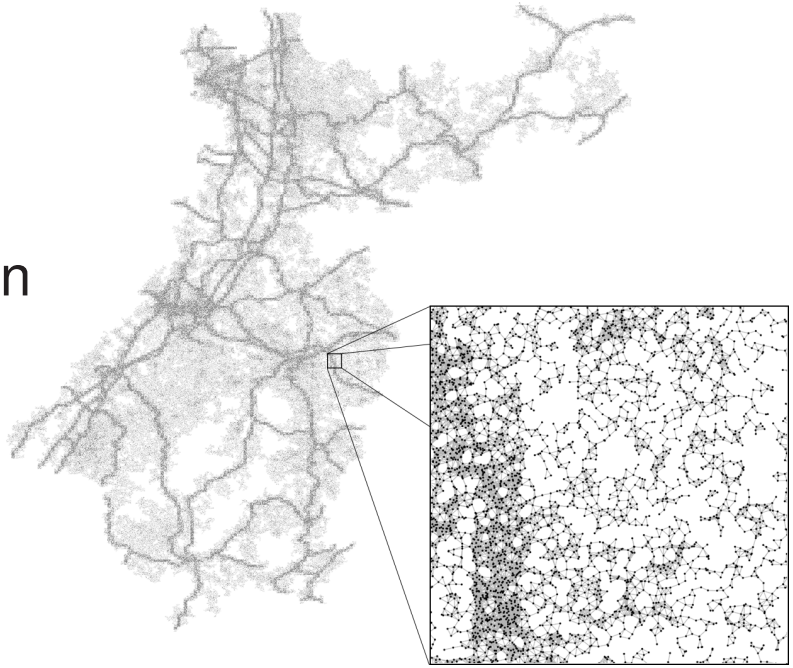
		Vorverarbeitung		Anfrage	
		Dauer [s]	Overhead [B/n]	Dauer [ms]	Fehler Ø [%]
	apx ($\epsilon = 10\%$)	116	−29	0.32	2.1
	Bidir. Dijkstra	0	0	141.15	–
	Arc Flags	522	92	0.83	–
	Core-ALT	239	170	2.70	–
ALT	exakt	178	512	3.67	–
ALT	apx ($\epsilon = 10\%$)	178	512	2.35	1.1

Intel Core i7-920 @ 2.67 Ghz, 12 GB RAM

Weitere Ergebnisse

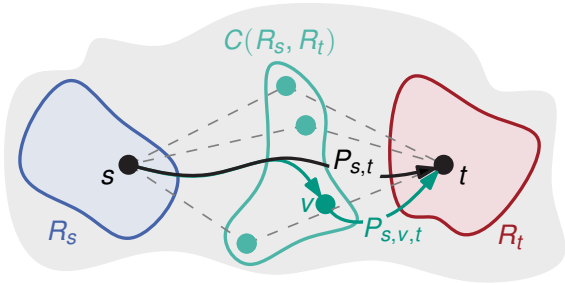
Approximierte CH [SOCS'10]

- Korrektheitsbeweise
- Kombinationen mit anderen Techniken
- weitere Netzwerkmodelle
  (Knotengrad, Graphtypen, Kostenfunktionen)



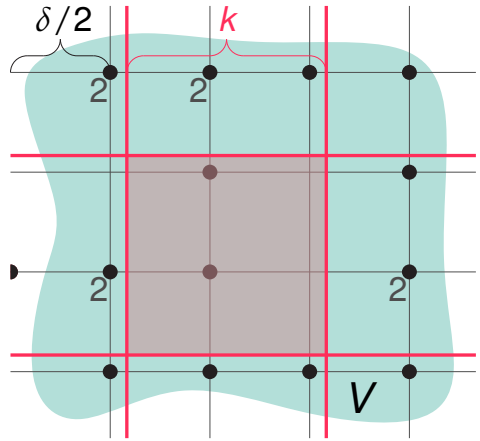
Alternativrouten [SEA'12], [JEA'14]

- Vorberechnung möglicher Zwischenknoten
- kann approximierte CH verwenden
- Onlineverfahren
  (lernt Zwischenknoten "on-the-fly")



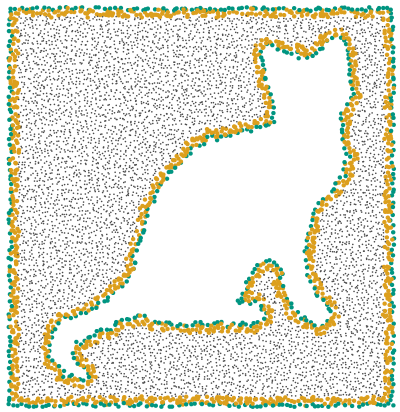
Zusammenfassung

“An Algorithmic View on Sensor Networks –



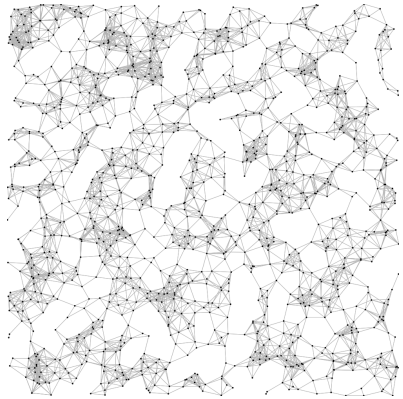
Surveillance,

- EPTAS
  zur Sensoreinsatzplanung



Localization, and

- verteilter, lokaler Algorithmus
  zur Randerkennung



Communication”

- Approximationsalgorithmus
  zur Berechnung optimaler Routen

Referenzen

[ALGOSENSORS'10]
Peter Sanders and Dennis Schieferdecker. *Lifetime Maximization of Monitoring Sensor Networks.* In *International Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks, and Autonomous Mobile Entities (ALGOSENSORS'10)*, *LNCS*, vol. 6451, pp. 134–147. Springer, 2010.

[SOCS'10]
Robert Geisberger and Dennis Schieferdecker. *Heuristic Contraction Hierarchies with Approximation Guarantee.* In *International Symposium on Combinatorial Search (SoCS'10)*, pp. 31–38. AAAI Press, 2010.

[SEA'11]
Dennis Schieferdecker, Markus Völker, and Dorothea Wagner. *Efficient Algorithms for Distributed Detection of Holes and Boundaries in Wireless Networks.* In *International Symposium on Experimental Algorithms (SEA'11)*, *LNCS*, vol. 6630, pp. 388–399. Springer, 2011.

[TR'11]
Dennis Schieferdecker, Markus Völker, and Dorothea Wagner. *Efficient Algorithms for Distributed Detection of Holes and Boundaries in Wireless Networks.* Karlsruhe Reports in Informatics 2011,8, Karlsruhe Institute of Technology, 2011.

[SEA'12]
Dennis Luxen and Dennis Schieferdecker. *Candidate Sets for Alternative Routes in Road Networks.* In *International Symposium on Experimental Algorithms (SEA'12)*, *LNCS*, vol. 7276, pp. 260–270. Springer, 2012.

[JEA'14]
Dennis Luxen and Dennis Schieferdecker. *Candidate Sets for Alternative Routes in Road Networks. ACM Journal of Experimental Algorithmics*, 2014. Accepted for publication.

Referenzen

[BCSZ'05]
Piotr Berman, Gruia Calinescu, Chintan Shah, and Alexander Zelikovsky. *Efficient Energy Management in Sensor Networks.* In *Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing*, vol. 2, pp. 71—90. Nova Science Publishers, 2005.

[BDSSSW'10]
Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner. *Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm. ACM Journal of Experimental Algorithmics*, 15(2.3):1-–31, 2010.

[CTLW'05]
Mihaela Cardei, My T. Thai, Yingshu Li, and Weili Wu. *Energy-Efficient Target Coverage in Wireless Sensor Networks*. In *Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, vol. 3, pp. 1976—1984. IEEE, 2005.

[DW'60]
George B. Dantzig and Philip Wolfe. *Decomposition Principle for Linear Programs*. *Operations Research*, 8(1):101-–110, 1960.

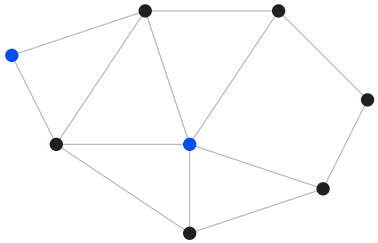
[GK'07]
Naveen Garg and Jochen Könemann. *Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems*. *SIAM Journal on Computing*, 37(2):630-–652, 2007.

[GSSV'12]
Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. *Exact Routing in Large Road Networks Using Contraction Hierarchies. Transportation Science*, 46(3):388-–404, 2012.

Zusatzfolien

Beweis

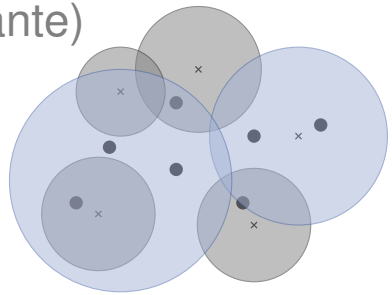
$\mathcal{NP}$-Vollständigkeit



gewichtetes MDS
(auf UDGs)

$\mathcal{NP}$-vollständig
[MIH'81]

gewichtetes MGDC
(Entscheidungsvariante)



Problemstellung

LP $\max \{\mathbf{1}^\intercal \mathbf{t} \mid \mathcal{C}\mathbf{t} \leq \mathbf{b},\ \mathbf{t} \geq \mathbf{0}\}$

Duales LP $\min \{\mathbf{b}^\intercal \mathbf{w} \mid \mathcal{C}^\intercal \mathbf{w} \geq \mathbf{1},\ \mathbf{w} \geq \mathbf{0}\}$

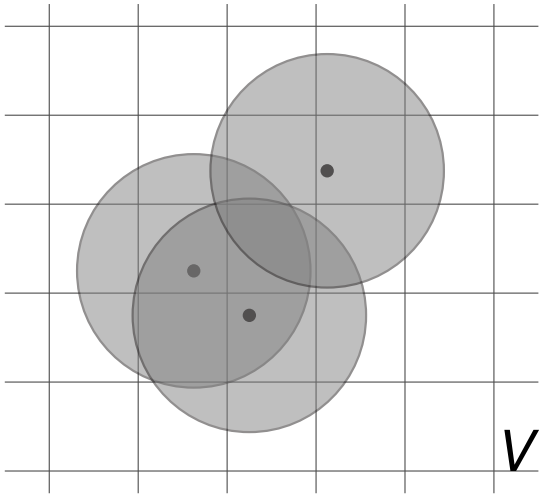
[GLS'81]

Separationsproblem zu dualem LP
(verifiziere Lösung/gib verletzte Bedingung an)

Behauptungen

Approximationsalgorithmus

Diskretisierung von Knotenpositionen

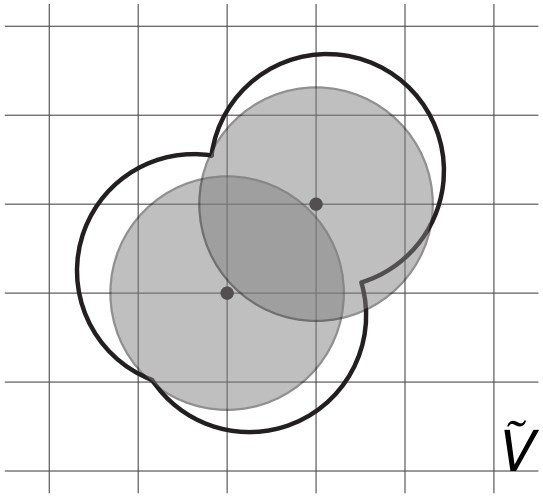


- $(\tilde{V}, 1+\delta/2)$ deckt Bereich von $(V, 1)$ ab
  - $\rightarrow$ Lösung für $(V, 1)$ ist Lösung für $(\tilde{V}, 1+\delta/2)$
  - $\rightarrow$ $T_{\text{opt}}\langle \tilde{V}, 1+\delta/2 \rangle \geq T_{\text{opt}}\langle V, 1 \rangle$
- Lösung für $(\tilde{V}, 1+\delta/2)$
  (berechnet mit Algorithmus $\mathcal{A}$)
  - $\rightarrow$ $T\langle \tilde{V}, 1+\delta/2 \rangle \geq f \cdot T_{\text{opt}}\langle \tilde{V}, 1+\delta/2 \rangle \geq f \cdot T_{\text{opt}}\langle V, 1 \rangle$
  - $\rightarrow$ Lösung für $(\tilde{V}, 1+\delta/2)$ ist Lösung für $(V, 1+\delta)$
    (mit den gleichen Argumenten wie oben)
  - $\rightarrow$ $T\langle V, 1+\delta \rangle \geq f \cdot T_{\text{opt}}\langle V, 1 \rangle$

Behauptungen

Approximationsalgorithmus

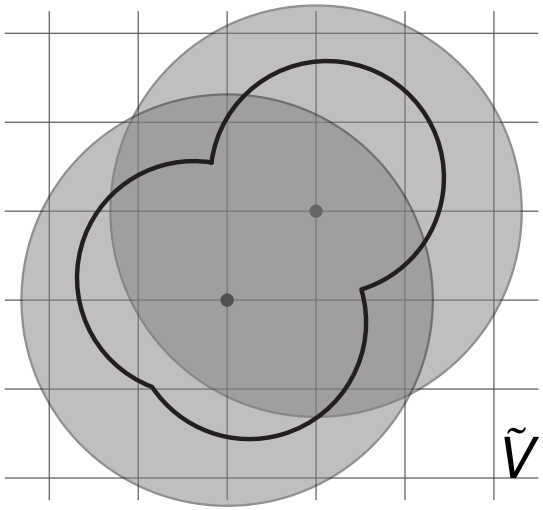
Diskretisierung von Knotenpositionen



- $(\tilde{V}, 1+\delta/2)$ deckt Bereich von $(V,1)$ ab
  - $\rightarrow$ Lösung für $(V,1)$ ist Lösung für $(\tilde{V}, 1+\delta/2)$
  - $\rightarrow$ $T_{\text{opt}}\langle \tilde{V}, 1+\delta/2\rangle \geq T_{\text{opt}}\langle V,1\rangle$
- Lösung für $(\tilde{V}, 1+\delta/2)$
  (berechnet mit Algorithmus $\mathcal{A}$)
  - $\rightarrow$ $T\langle \tilde{V}, 1+\delta/2\rangle \geq f \cdot T_{\text{opt}}\langle \tilde{V}, 1+\delta/2\rangle \geq f \cdot T_{\text{opt}}\langle V,1\rangle$
  - $\rightarrow$ Lösung für $(\tilde{V}, 1+\delta/2)$ ist Lösung für $(V, 1+\delta)$
    (mit den gleichen Argumenten wie oben)
  - $\rightarrow$ $T\langle V, 1+\delta\rangle \geq f \cdot T_{\text{opt}}\langle V,1\rangle$

Behauptungen

Approximationsalgorithmus



Diskretisierung von Knotenpositionen

- $(\tilde{V}, 1+\delta/2)$ deckt Bereich von $(V, 1)$ ab
  - $\rightarrow$ Lösung für $(V, 1)$ ist Lösung für $(\tilde{V}, 1+\delta/2)$
  - $\rightarrow$ $T_{\text{opt}}\langle \tilde{V}, 1+\delta/2 \rangle \geq T_{\text{opt}}\langle V, 1 \rangle$
- Lösung für $(\tilde{V}, 1+\delta/2)$
  (berechnet mit Algorithmus $\mathcal{A}$)
  - $\rightarrow$ $T\langle \tilde{V}, 1+\delta/2 \rangle \geq f \cdot T_{\text{opt}}\langle \tilde{V}, 1+\delta/2 \rangle \geq f \cdot T_{\text{opt}}\langle V, 1 \rangle$
  - $\rightarrow$ Lösung für $(\tilde{V}, 1+\delta/2)$ ist Lösung für $(V, 1+\delta)$
    (mit den gleichen Argumenten wie oben)
  - $\rightarrow$ $T\langle V, 1+\delta \rangle \geq f \cdot T_{\text{opt}}\langle V, 1 \rangle$

Überblick

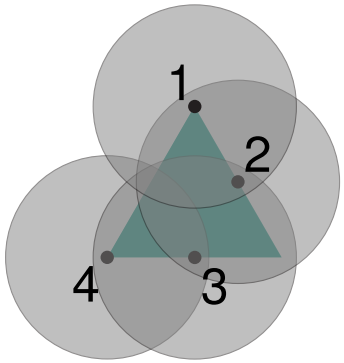
Exakter Algorithmus

Grundlage

- Formulierung des Problems als Lineares Programm [BCSZ'05]

$$\max\left\{\mathbf{1}^{\mathsf{T}}\mathbf{t} \mid \mathcal{C}\mathbf{t} \leq \mathbf{b},\ \mathbf{t} \geq \mathbf{0}\right\}, \qquad \mathbf{t} \in \mathbb{R}_+^m, \mathbf{b} \in \mathbb{R}_+^n, \mathcal{C} \in \mathbb{Z}_2^{m \times n}$$

- jede Spalte von $\mathcal{C}$ ist eine mögliche Abdeckung
  (Knotenmenge, die gleichzeitig aktiviert wird und das Gebiet überdeckt)
  $\rightarrow$ exponentiell viele Spalten!



Ansätze

- verzögerte Spaltenerzeugung
  (Delayed Column Generation [DW'60])
- heuristische Verfahren
  (inexakte Lösungen)

$$\mathcal{C} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & \cdots \\ 1 & 1 & 0 & 0 & 1 & \cdots \\ 1 & 1 & 1 & 1 & 0 & \cdots \\ 1 & 0 & 0 & 1 & 1 & \cdots \end{pmatrix}$$

Abdeckung

Knoten 1

Verzögerte Spaltenerzeugung

Exakter Algorithmus

Ablauf

- löse reduziertes Problem (LP)

$$\max\left\{\mathbf{1}^\mathsf{T}\mathbf{t} \mid \mathbf{C}\mathbf{t} \leq \mathbf{b},\ \mathbf{t} \geq \mathbf{0}\right\}, \qquad \mathbf{t} \in \mathbb{R}_+^k, \mathbf{b} \in \mathbb{R}_+^n, \mathbf{C} \in \mathbb{Z}_2^{k \times n}$$

  (Matrix **C** durch Initialisierungsschritt berechnet)

- löse Orakelproblem (ILP)

$$\max\left\{1 - \mathbf{c}^\mathsf{T}\mathbf{w} \mid \mathbf{c} \in \mathcal{C}\right\}, \qquad \mathbf{w} \in \mathbb{R}_+^n, \mathbf{c} \in \mathbb{Z}_2^n$$

  (**w** ist duale Lösung zu reduziertem Problem)

- füge Abdeckung **c** zu Matrix **C** hinzu
- wiederhole bis Orakelproblem nicht-positive Lösung liefert

Verzögerte Spaltenerzeugung

Exakter Algorithmus

Freiheitsgrade

- Initialisierungsschritt
- Lösungsverfahren für Orakelproblem
- Terminierungsbedingung (für inexakte Lösungen)

Verzögerte Spaltenerzeugung

Exakter Algorithmus

Freiheitsgrade

- Initialisierungsschritt
  - einfache Ansätze (alle Knoten aktiv, zufällige Abdeckungen)
  - gierige Ansätze ([SP'01], [CTLW'05])
  - komplexere Ansätze (Garg-Könemann Verfahren [GK'07])
- Lösungsverfahren für Orakelproblem
- Terminierungsbedingung (für inexakte Lösungen)

Verzögerte Spaltenerzeugung

Exakter Algorithmus

Freiheitsgrade

- Initialisierungsschritt
- Lösungsverfahren für Orakelproblem
  - exakt (ILP Solver)
  - heuristisch (inexakte Lösungen)
  - heuristisch+exakt
- Terminierungsbedingung (für inexakte Lösungen)

Verzögerte Spaltenerzeugung

Exakter Algorithmus

Freiheitsgrade

- Initialisierungsschritt
- Lösungsverfahren für Orakelproblem
- Terminierungsbedingung (für inexakte Lösungen)
  - Verbesserung der Lösung je Schritt
  - obere Schranken
  - maximiale Zeitdauer

Verzögerte Spaltenerzeugung

Exakter Algorithmus

Freiheitsgrade

- Initialisierungsschritt: Garg-Könemann Verfahren [GK'07]
- Lösungsverfahren für Orakelproblem: ILP Solver
- Terminierungsbedingung (für inexakte Lösungen)

Verzögerte Spaltenerzeugung

Exakter Algorithmus

Freiheitsgrade

- Initialisierungsschritt: Garg-Könemann Verfahren [GK'07]
- Lösungsverfahren für Orakelproblem: ILP Solver
- Terminierungsbedingung (für inexakte Lösungen)

Erkentnisse

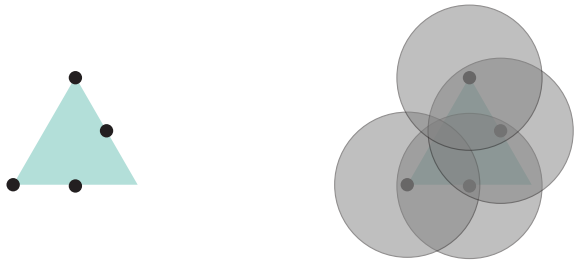
- gute Initialisierung
  (beschleunigt Konvergenz um Faktor 4)
- Minimierung von Abdeckung **c** vor Hinzufügen zu Matrix **C**
  (beschleunigt Konvergenz um Faktor 40)
- je zufälliger die Eingabe, desto einfacher zu lösen
  (weniger symmetrische Lösungen, die betrachtet werden müssen)

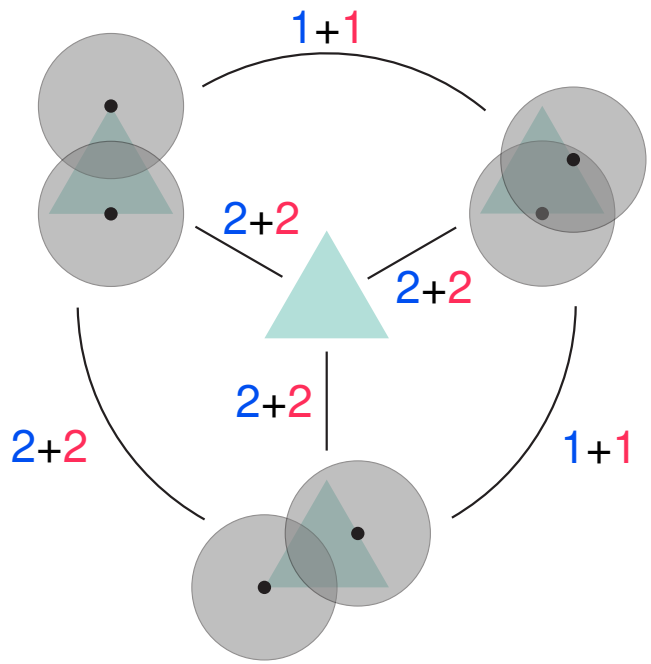
Nachoptimierung

Minimierung von Ein- und Ausschaltevorgängen

- Sensoreinsatzplanung liefert Menge an Abdeckungen
  (Knotenmenge, die gleichzeitig aktiviert wird und das Gebiet überdeckt)
  - werden der Reihe nach aktiviert für berechnete Zeitdauer

- Ein-/Ausschaltvorgänge können energetisch teuer sein
  - minimiere Anzahl durch optimierte Reihenfolge der Abdeckungen
  - metrisches TSP Problem

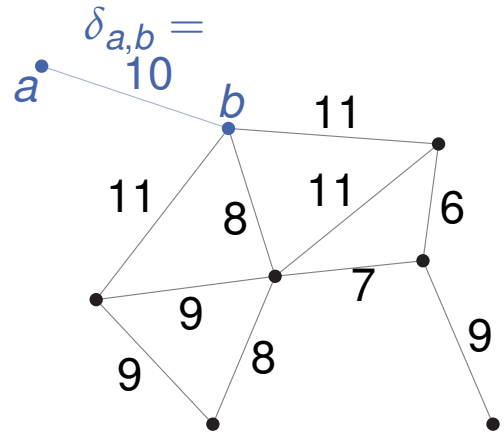




Multidimensionale Skalierung

Allgemeines

- Verfahren zur Visualisierung von Unterschieden
- klassische Skalierung [TOR52]
  (Speziallfall metrischer Skalierung, euklidisch)



gegeben

- Distanzen $\delta_{i,j}$ zwischen allen Knoten $i, j \in V$

gesucht

- Einbettung $\mathbf{p} : V \mapsto \mathbb{R}^k$ aller Knoten mit minimalem

$$\sum_{i,j=1}^{n} \left(\delta_{i,j} - ||\mathbf{p}(i) - \mathbf{p}(j)||\right)^2$$

Multidimensionale Skalierung

Ablauf

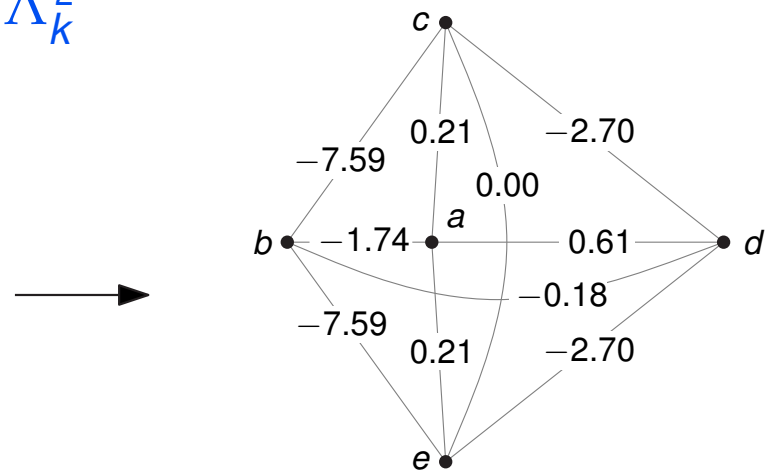
- perfekte Einbettung entspricht $\delta_{i,j} = ||\mathbf{p}(i) - \mathbf{p}(j)||$

  $\rightarrow$ $\mathbf{B}_\Delta := -\frac{1}{2}\mathbf{J}\Delta\mathbf{J} = \mathbf{E}\mathbf{E}^\mathsf{T}$
  mit $(\Delta_{i,j}) = \delta_{i,j}^2$, $\mathbf{J} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T}$, und $\mathbf{B}_\Delta = \mathbf{E}\mathbf{E}^\mathsf{T}$
  (doppelte Zentrierung auf Zentroid der Punkte)

- diagonalisiere $\mathbf{B}_\Delta \rightarrow \mathbf{B}_\Delta = \mathbf{V}\Lambda\mathbf{V}^\mathsf{T}$
  ($\mathbf{B}_\Delta$ ist orthogonal)

- Einbettung in k Dimensionen ist $\mathbf{E}_k = \mathbf{V}_k\Lambda_k^{\frac{1}{2}}$
  (Verwendung der k dominanten Eigenpaare)

$\delta_{i,j}$	a	b	c	d	e
a	0	2	3	4	3
b	2	0	4	6	4
c	3	4	0	5	6
d	4	6	5	0	5
e	3	4	6	5	0

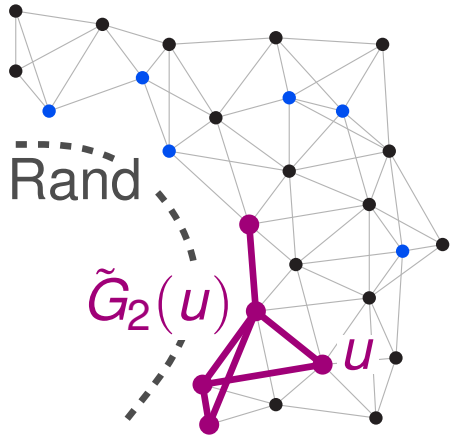


Randerkennung

Nachoptimierung

Ablauf

- Knoten u betrachtet 2-Hop Nachbarschaft
- existiert Pfad der Länge k über u
  - u bleibt Randknoten, sonst
  - klassifiziert sich u als innerer Knoten



Aufwand

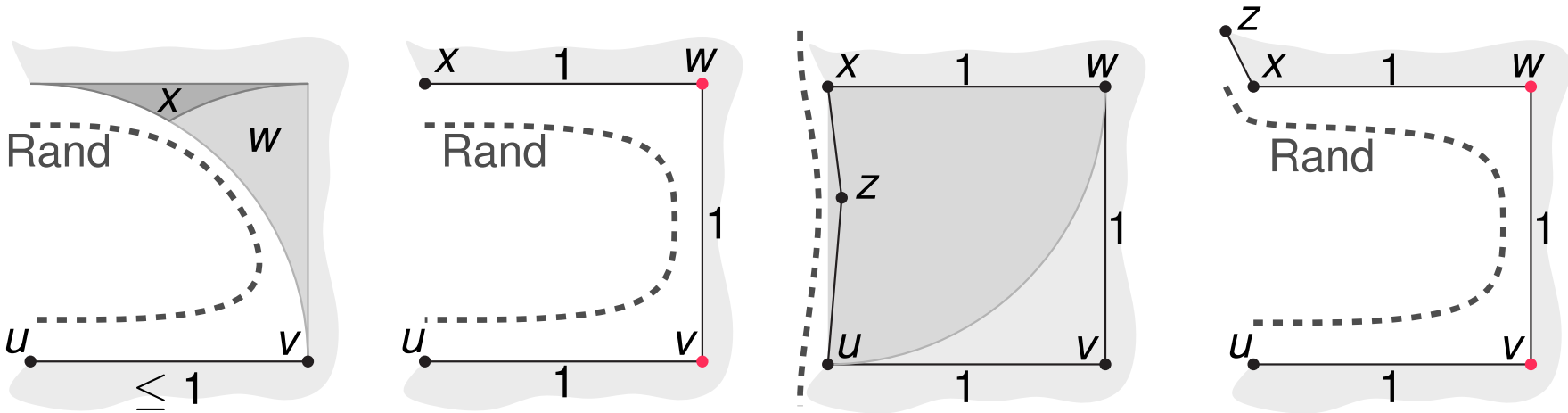
- Ø 10 – 25 Knoten in k-Hop Nachbarschaft
  (UDG bzw. 0.05-QUDG)

Randerkennung

Fehlklassifikationen

Randknoten

- MDS-BR1
  - maximal zwei aufeinanderfolgende Knoten falsch klassifiziert
- MDS-BR2
  - alle Knoten falsch klassifiziert
  - nicht in ausgedehnten Strukturen

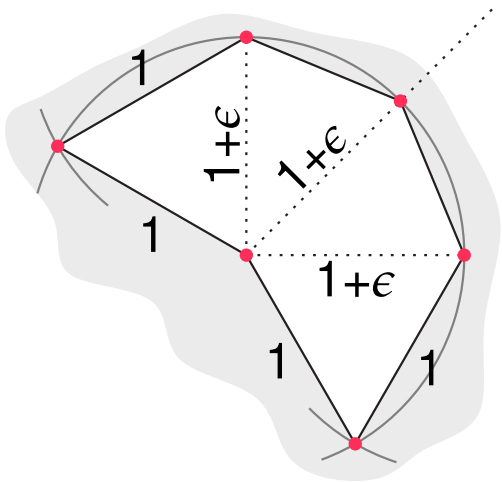


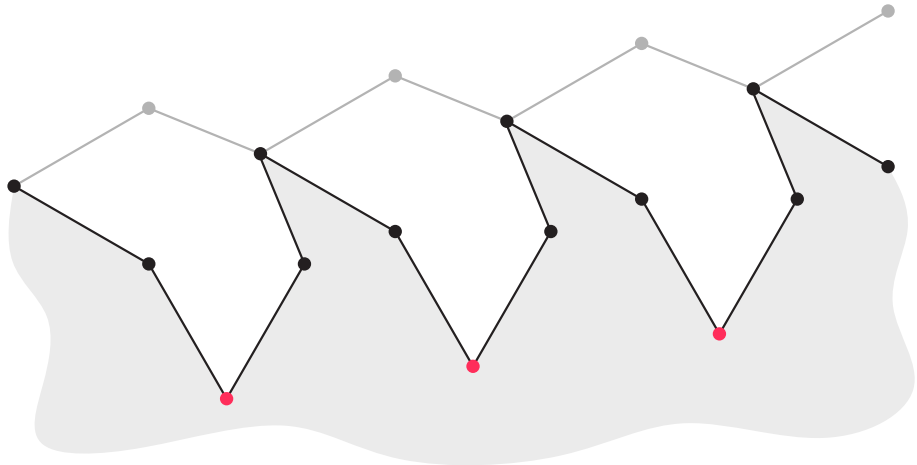
Randerkennung

Fehlklassifikationen

Randknoten

- MDS-BR1
  - maximal zwei aufeinanderfolgende Knoten falsch klassifiziert
- MDS-BR2
  - alle Knoten falsch klassifiziert
  - nicht in ausgedehnten Strukturen



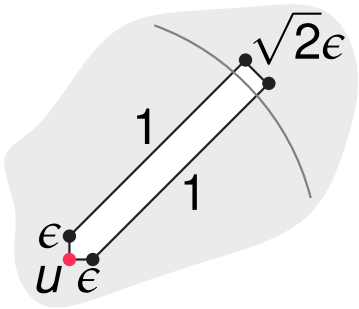


Randerkennung

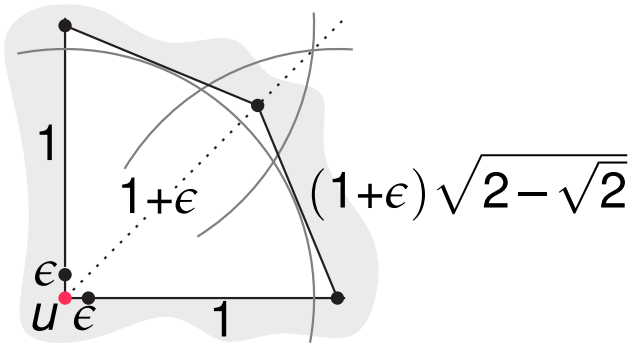
Fehlklassifikationen

Innere Knoten

- am Rand von Mikrolöchern
- Untere Schranken
  - $2 + \epsilon'$ (MDS-BR1)
  - $2 + 2\sqrt{2 - \sqrt{2}} + \epsilon'$ (MDS-BR2)
- Obere Schranken
  - nicht möglich
    (geballte / aufgereihte Mikrolöcher)



MDS-BR1



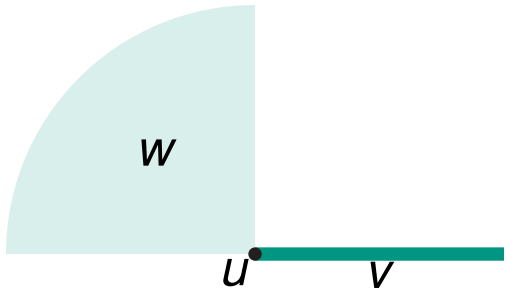
MDS-BR2

Randerkennung

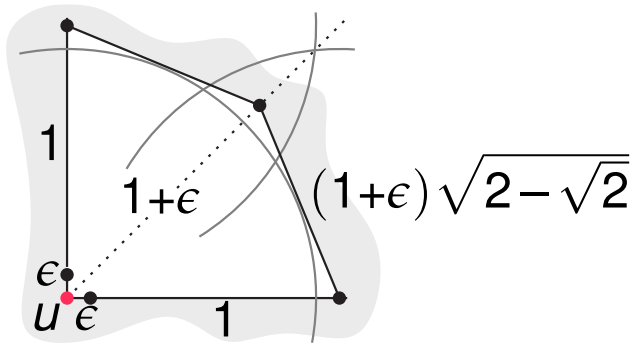
Fehlklassifikationen

Innere Knoten

- am Rand von Mikrolöchern
- Untere Schranken
  - $2 + \epsilon'$ (MDS-BR1)
  - $2 + 2\sqrt{2 - \sqrt{2}} + \epsilon'$ (MDS-BR2)
- Obere Schranken
  - nicht möglich
    (geballte / aufgereihte Mikrolöcher)



MDS-BR1



MDS-BR2

Randerkennung

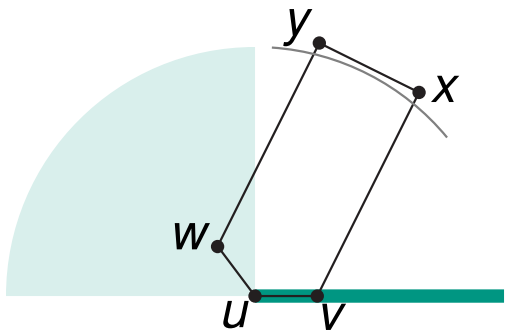
Fehlklassifikationen

Innere Knoten

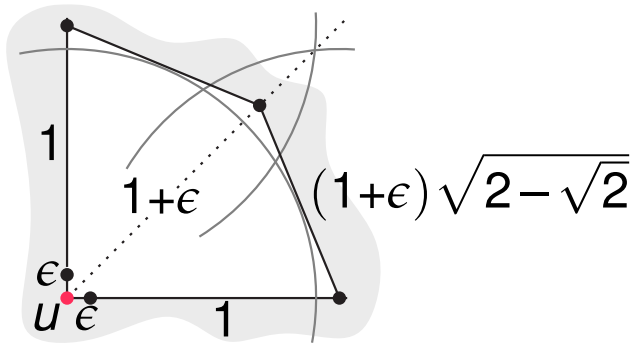
- am Rand von Mikrolöchern

- Untere Schranken
  - $2 + \epsilon'$ (MDS-BR1)
  - $2 + 2\sqrt{2 - \sqrt{2}} + \epsilon'$ (MDS-BR2)

- Obere Schranken
  - nicht möglich
    (geballte / aufgereihte Mikrolöcher)



MDS-BR1



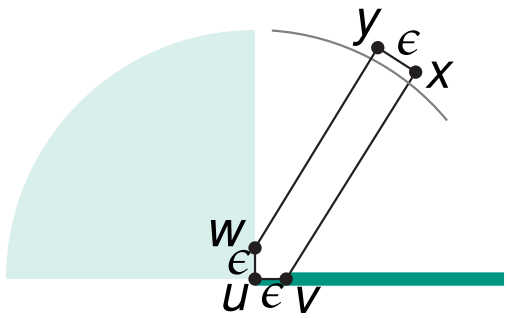
MDS-BR2

Randerkennung

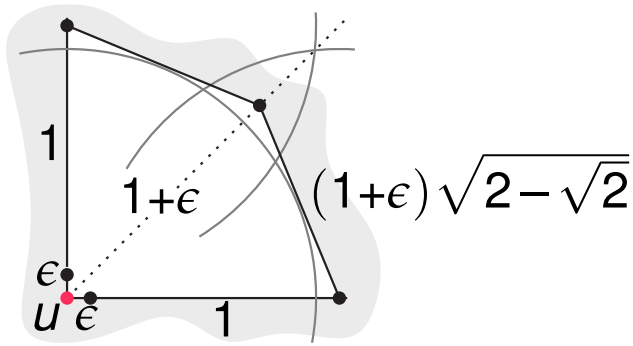
Fehlklassifikationen

Innere Knoten

- am Rand von Mikrolöchern
- Untere Schranken
  - $2 + \epsilon'$ (MDS-BR1)
  - $2 + 2\sqrt{2 - \sqrt{2}} + \epsilon'$ (MDS-BR2)
- Obere Schranken
  - nicht möglich
    (geballte / aufgereihte Mikrolöcher)



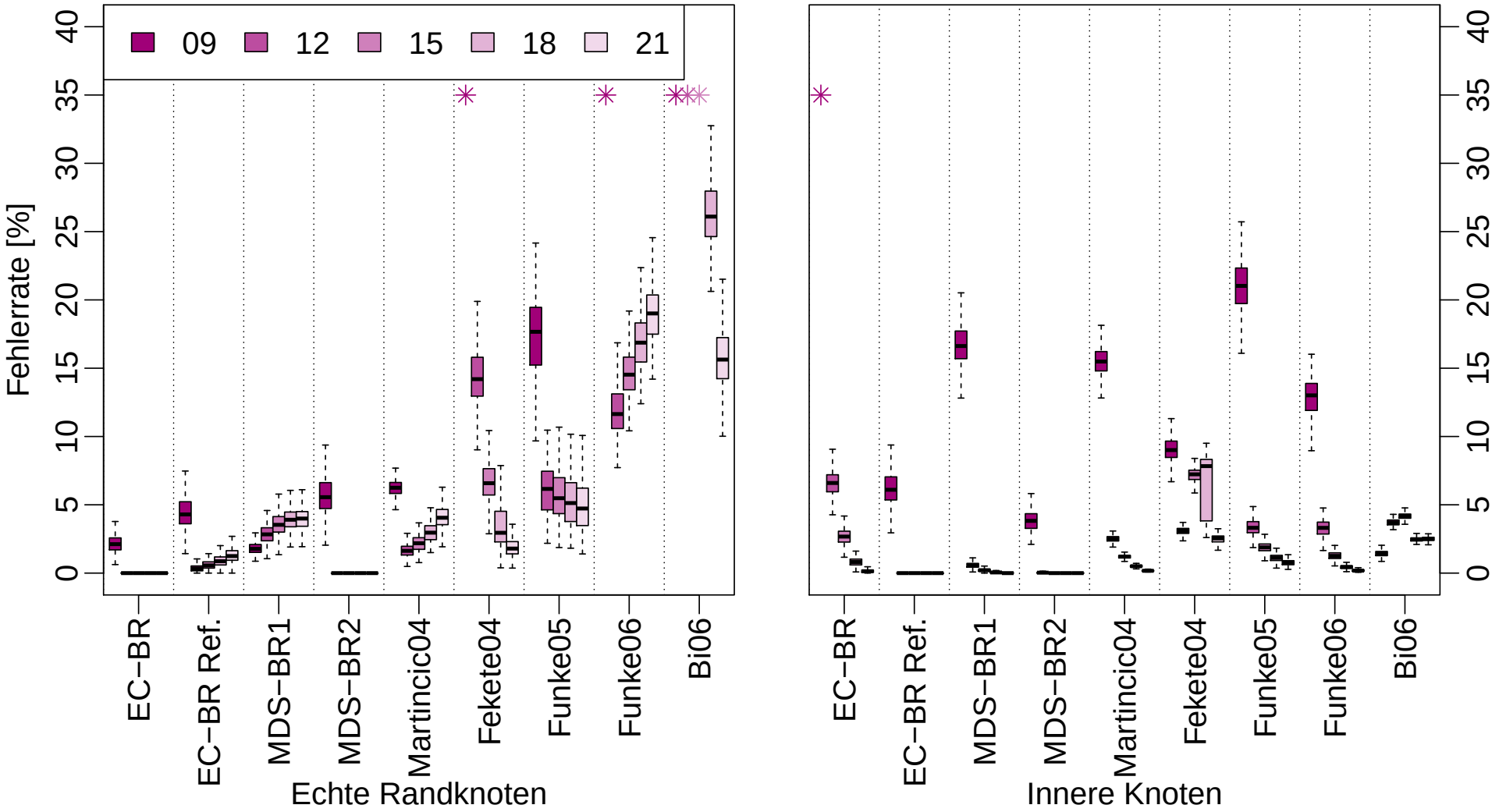
MDS-BR1



MDS-BR2

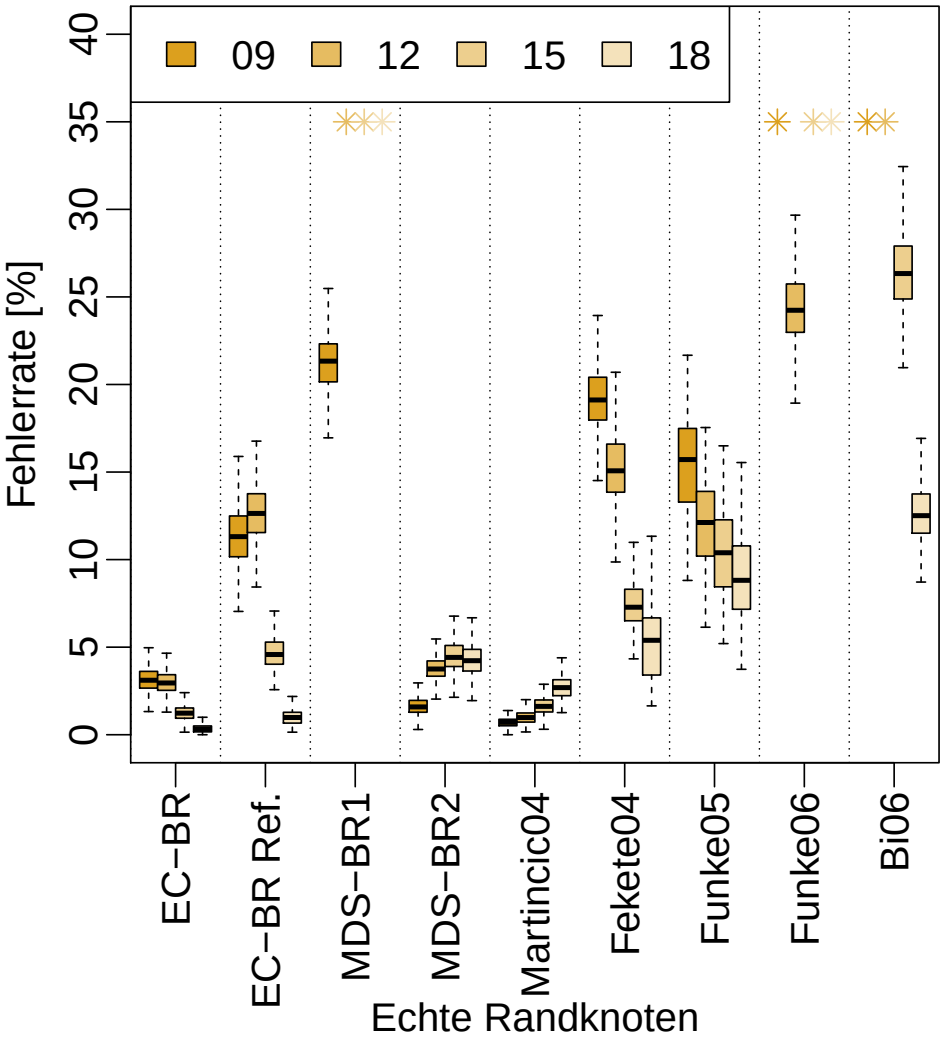
Randerkennung

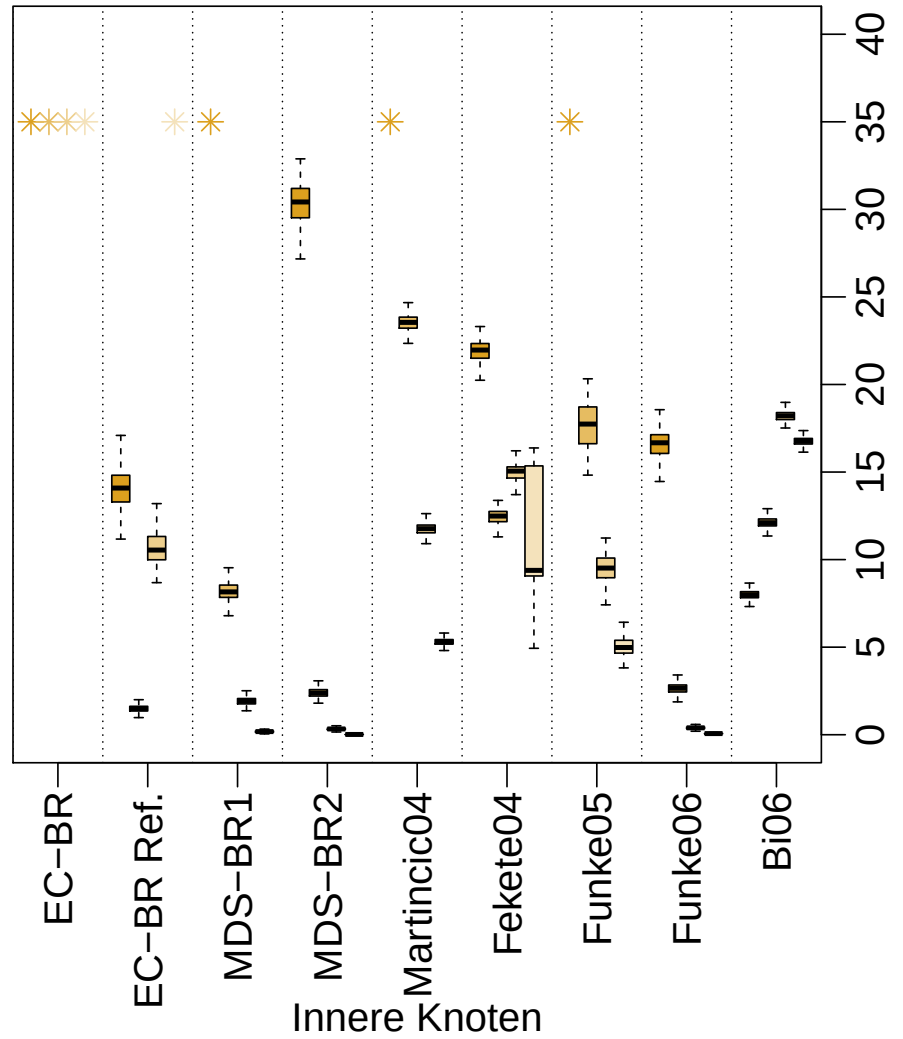
Klassifikationen – UDG



Randerkennung

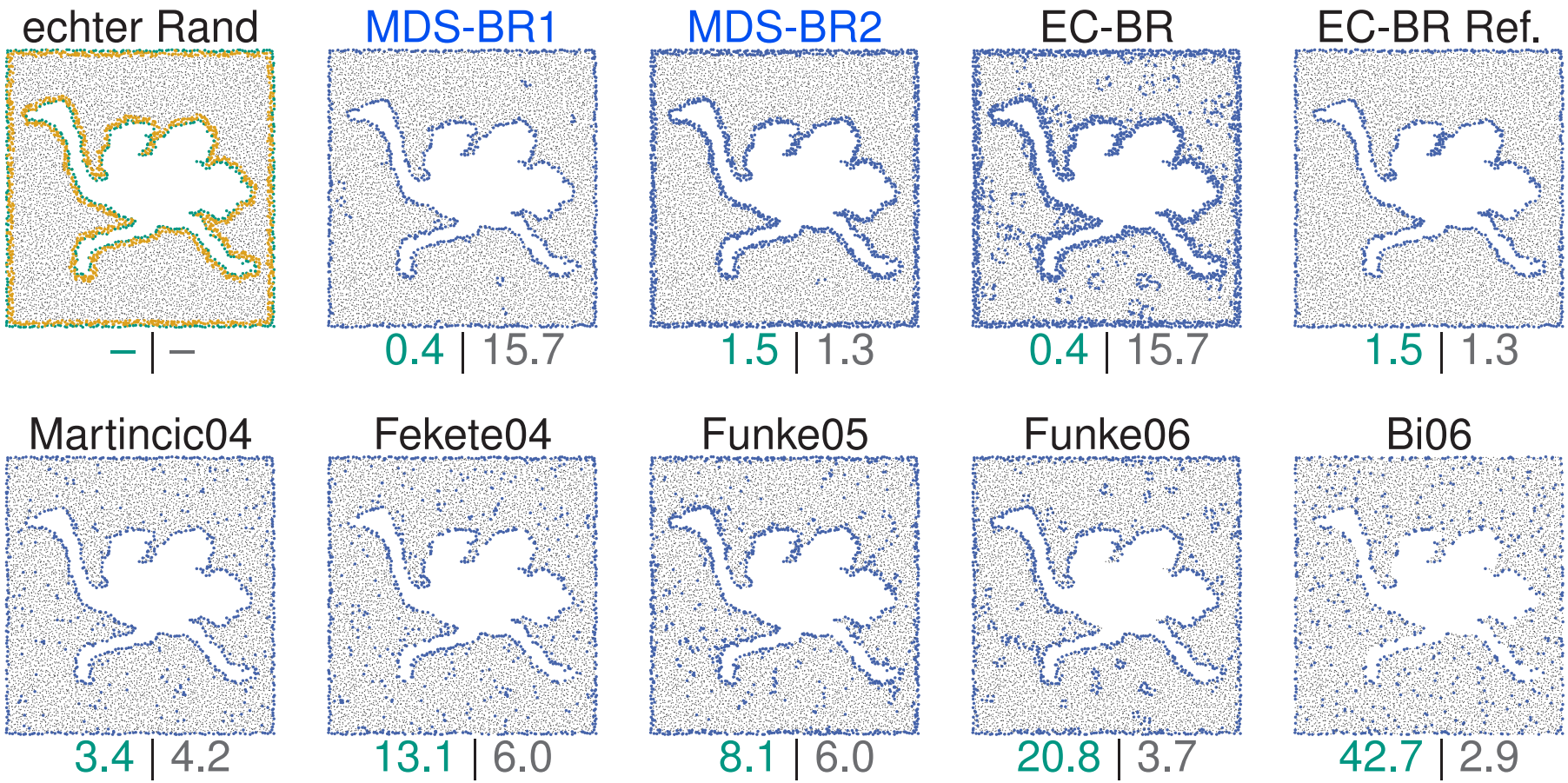
Klassifikationen – 0.25-QUDG





Ergebnisse

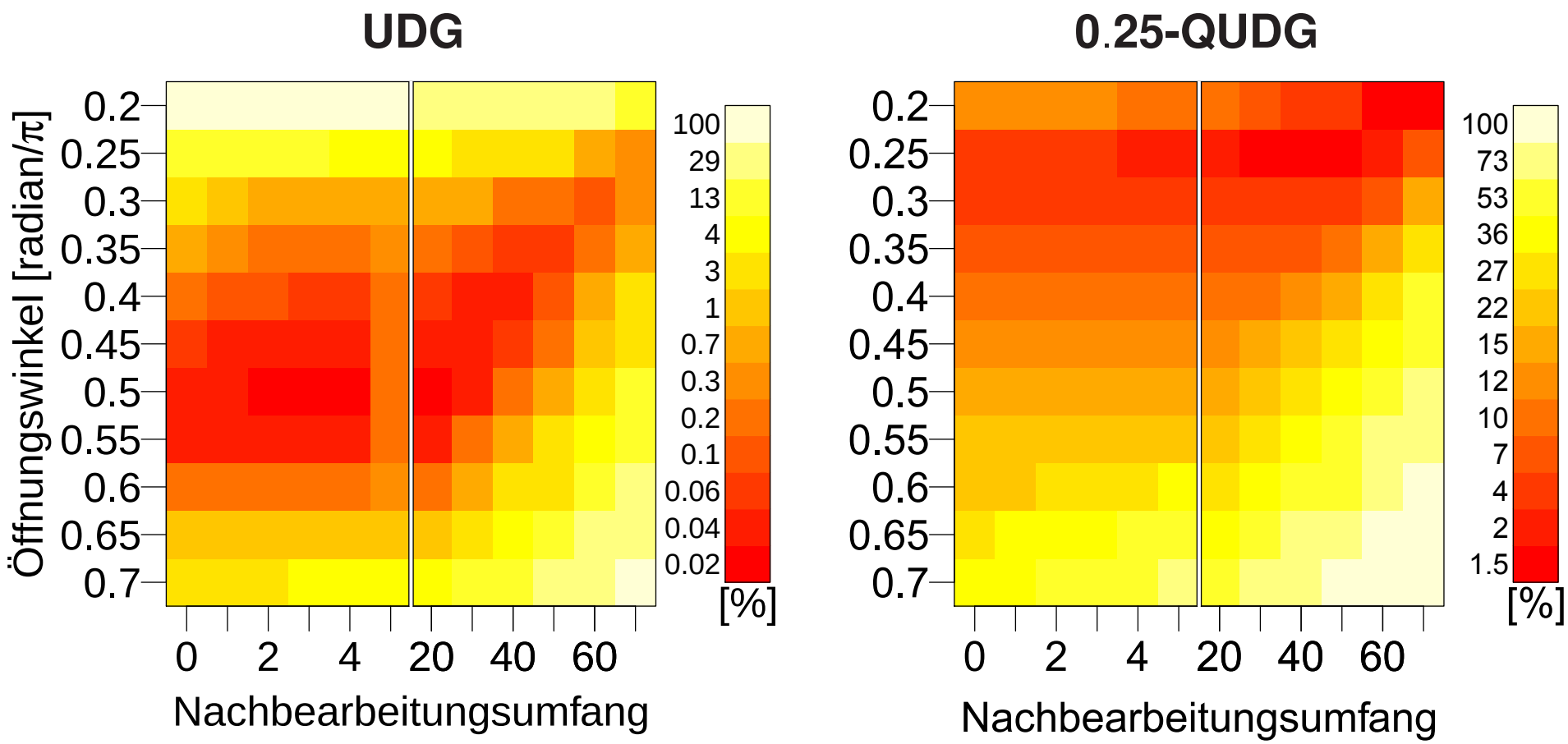
quantitative Auswertung – UDG



Fehlklassifikation von Randknoten und inneren Knoten [%]

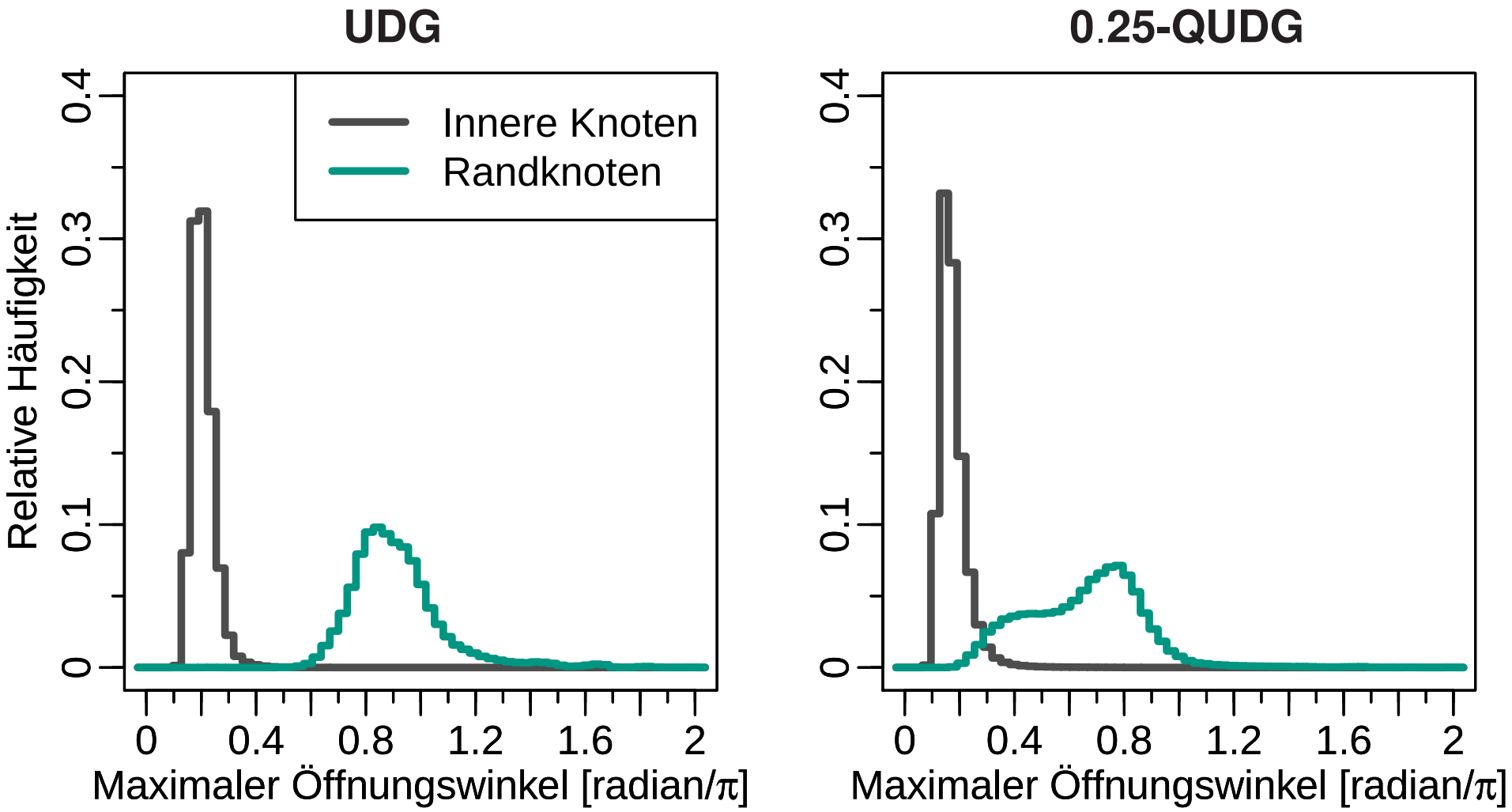
Randerkennung

Parameteranalyse – Heatmaps ($d_{avg} = 12$)



Randerkennung

Parameteranalyse – Winkelverteilungen ($d_{avg} = 12$)



Randerkennung

Einbettungsvarianten

MDS_3 Einbettung von 3-Hop Nachbarschaft

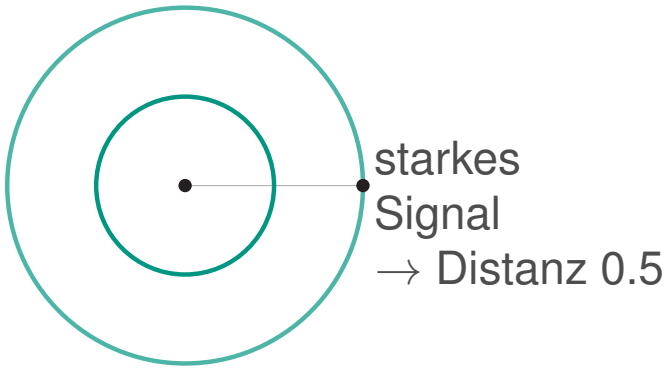
MDS_{opt} echte Positionen

MDS_{SS} Signalstärken

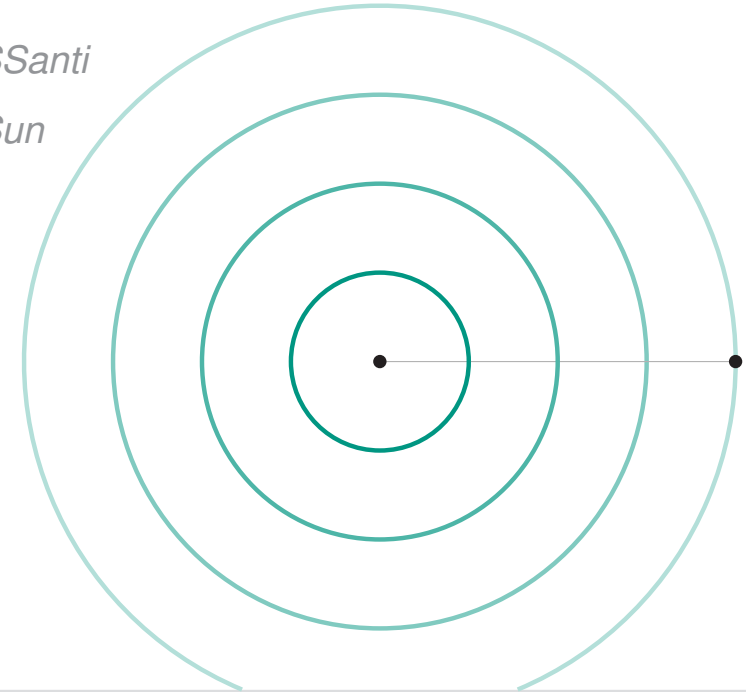
- Einteilung in starke und schwache Signale
- Annahme: Signalstärke korreliert mit Distanz

  $\rightarrow$ antikorreliert: MDS_{SSanti}

  $\rightarrow$ unkorreliert: MDS_{SSun}



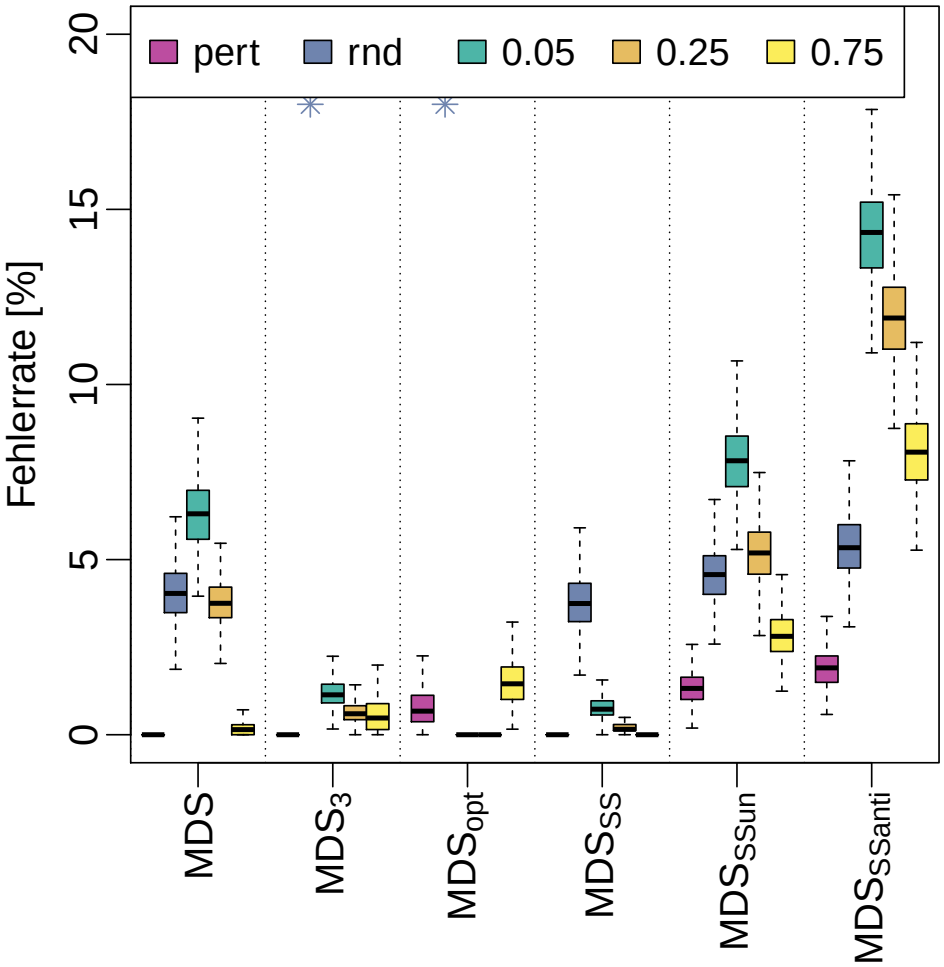
starkes Signal
$\rightarrow$ Distanz 0.5



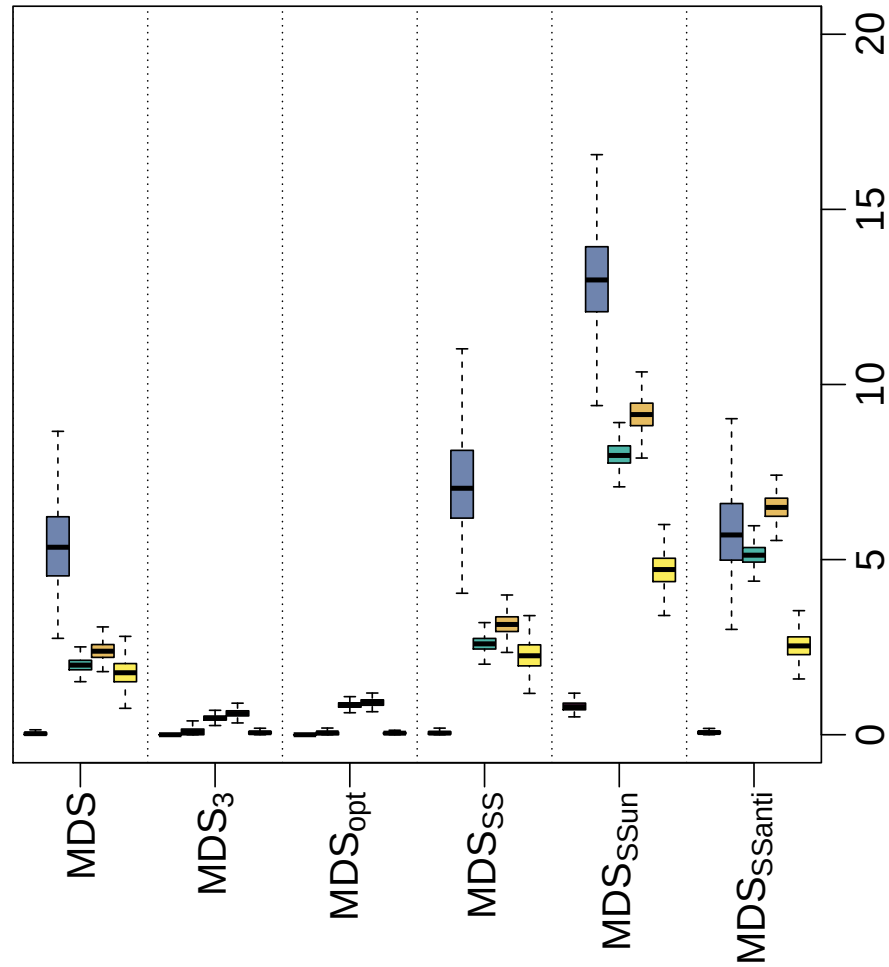
schwaches Signal
$\rightarrow$ Distanz 1.0

Randerkennung

Einbettungsvarianten ($d_{avg} = 12$)



Echte Randknoten



Innere Knoten

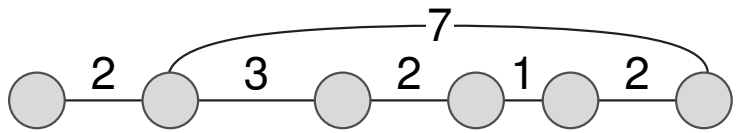
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von s und t
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

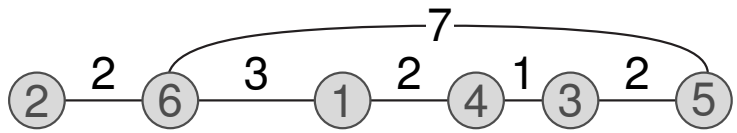
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von s und t
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

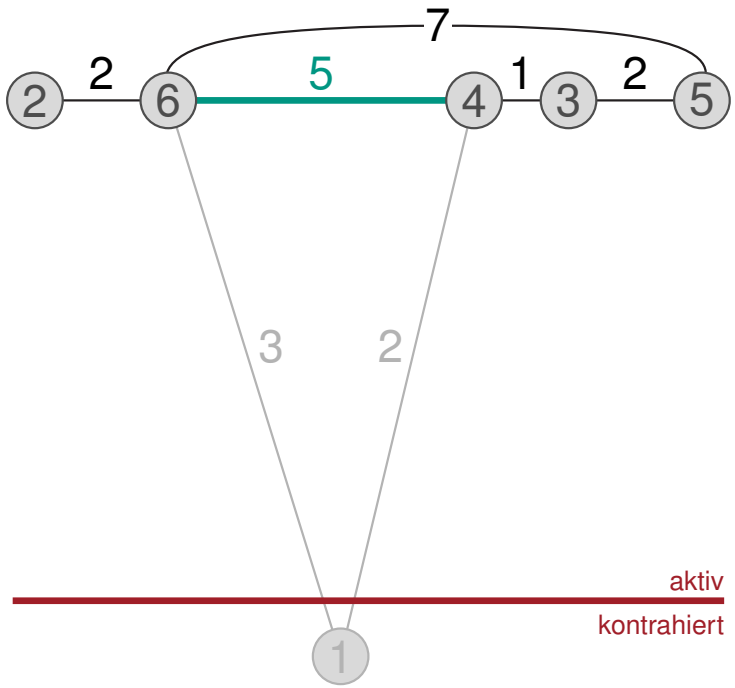
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von s und t
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

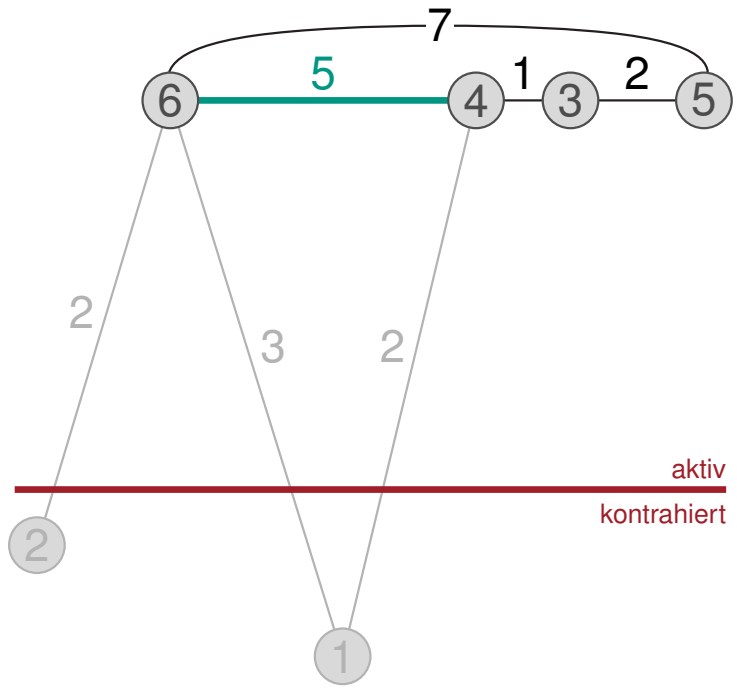
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von *s* und *t*
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

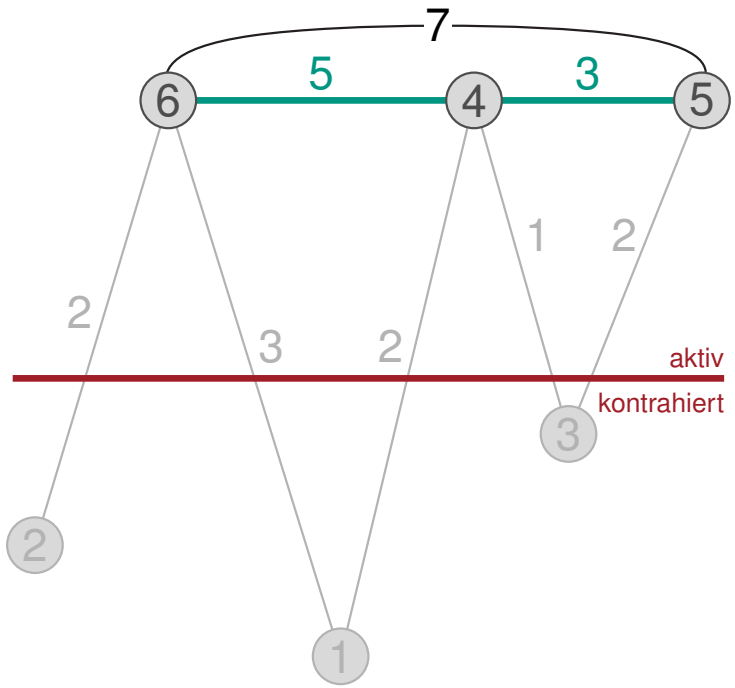
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von s und t
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

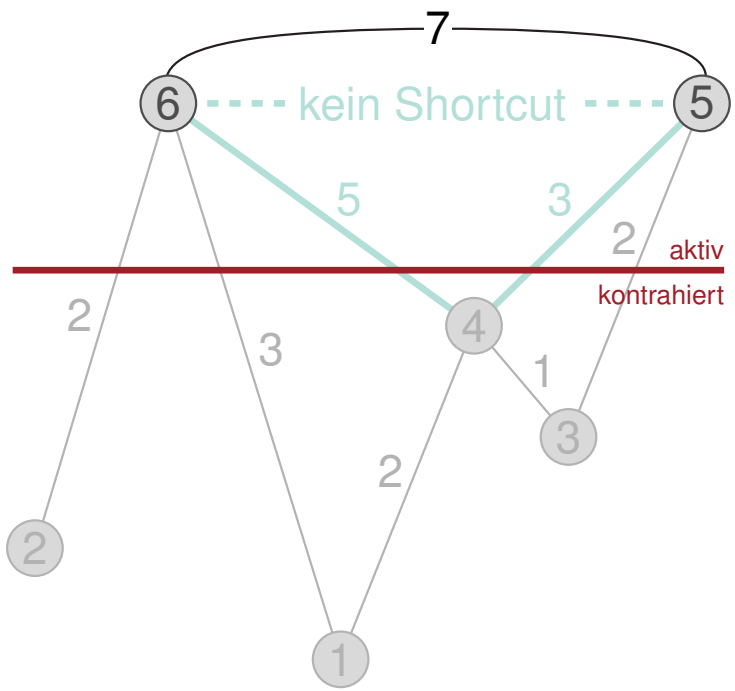
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von s und t
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

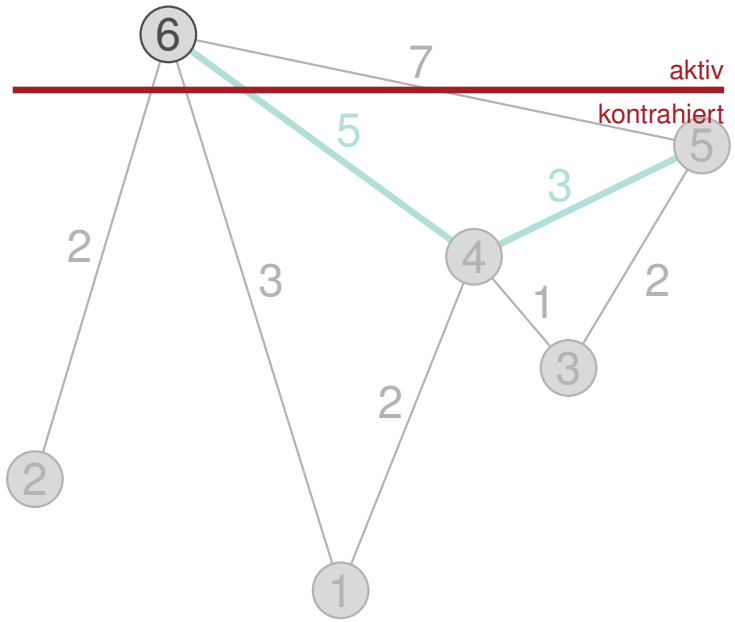
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von s und t
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

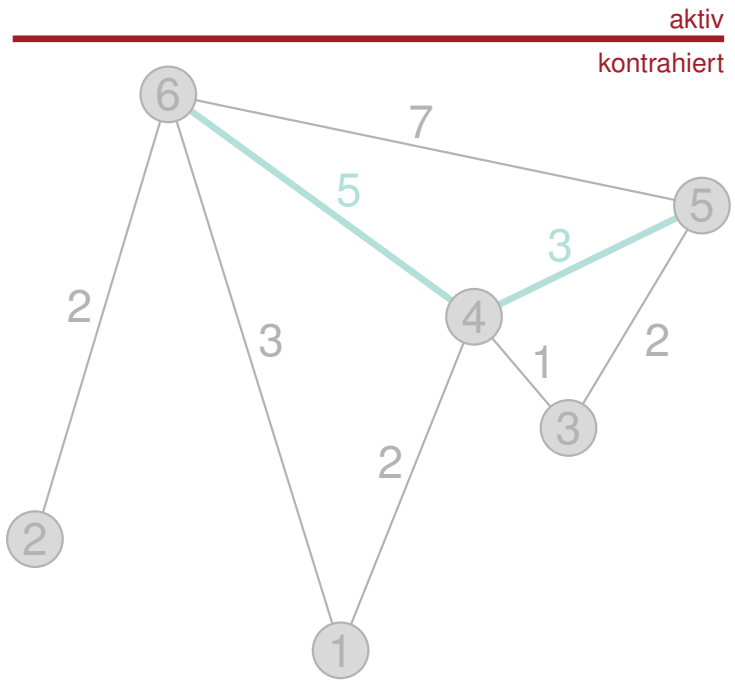
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von s und t
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

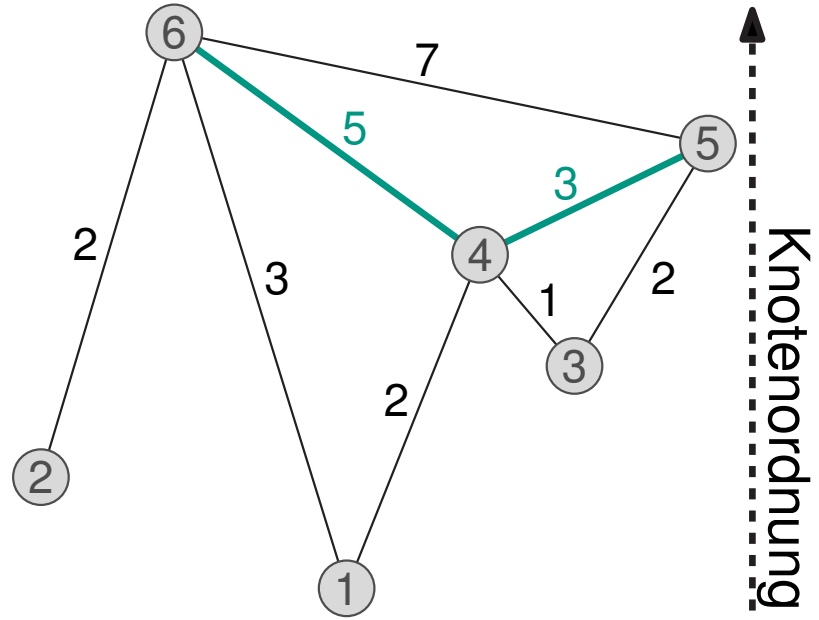
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von s und t
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

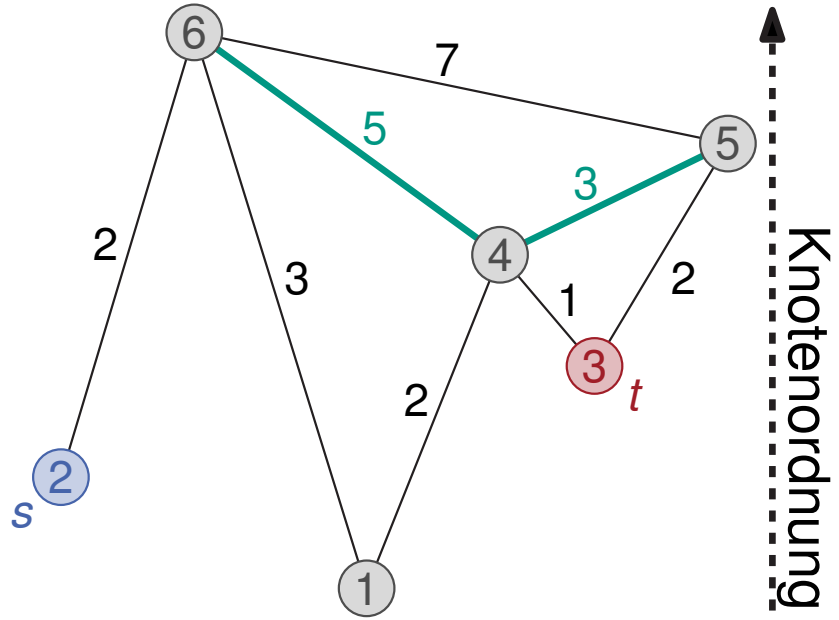
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von *s* und *t*
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

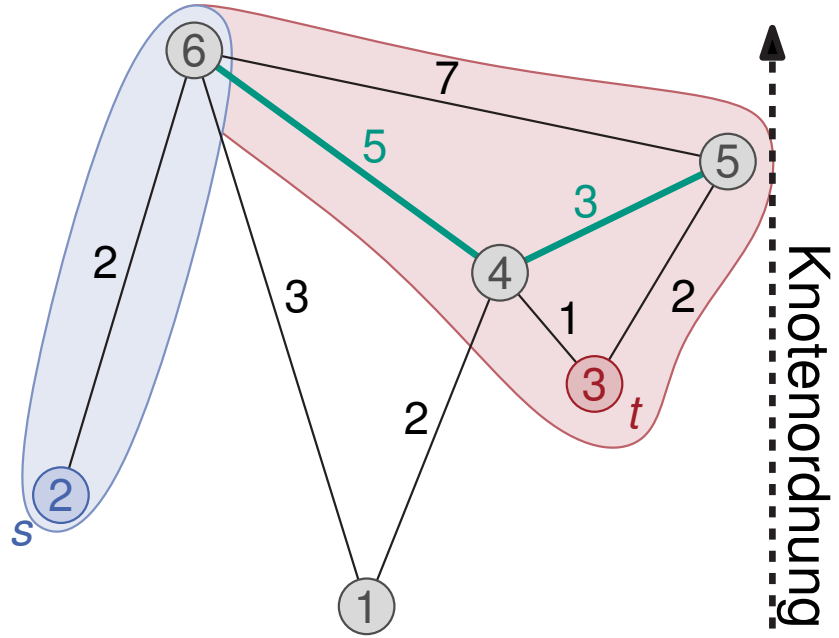
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



Routinganfrage

- bidirektionale Suche von *s* und *t*
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

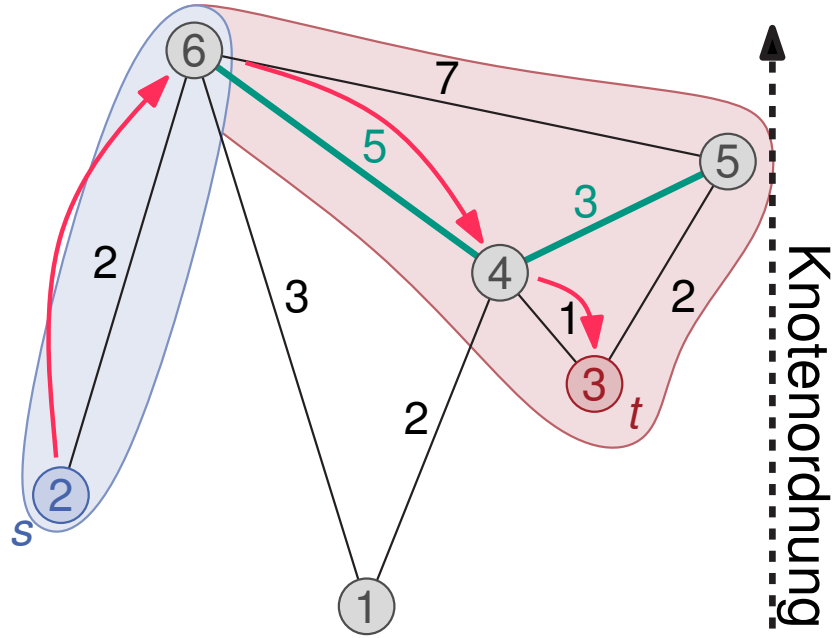
Contraction Hierarchies (CH)

zweistufiges Verfahren

Vorverarbeitung

- ordne Knoten nach “Wichtigkeit”
- kontrahiere in dieser Reihenfolge
  - entferne Knoten (temporär)
  - erhalte kürzeste Wege mit Shortcuts

→ Suchgraph = Netzwerk ∪ Shortcuts



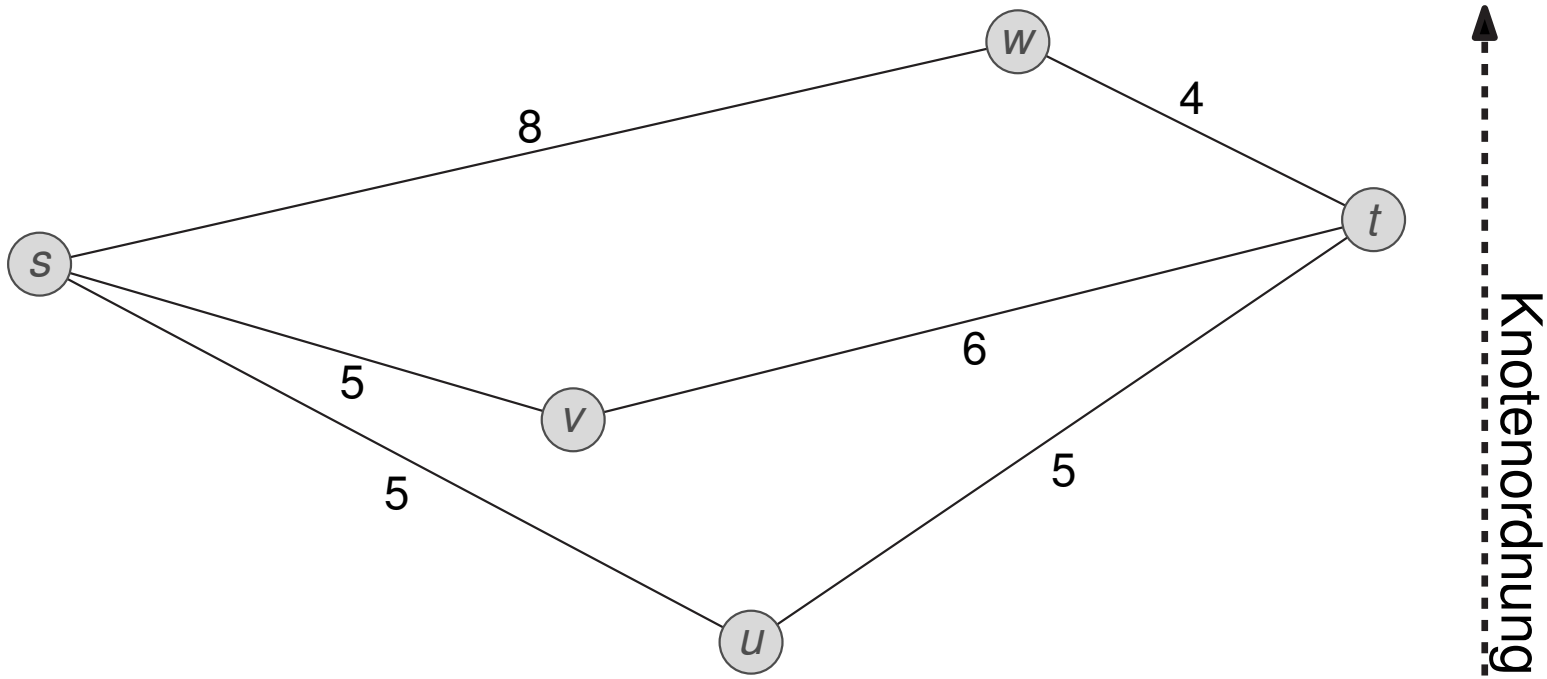
Routinganfrage

- bidirektionale Suche von *s* und *t*
- betrachte nur Kanten zu “wichtigeren” Knoten
  (abgewandelter Algorithmus von Dijkstra)

Approximierte CH (apxCH)

grundlegende Idee...

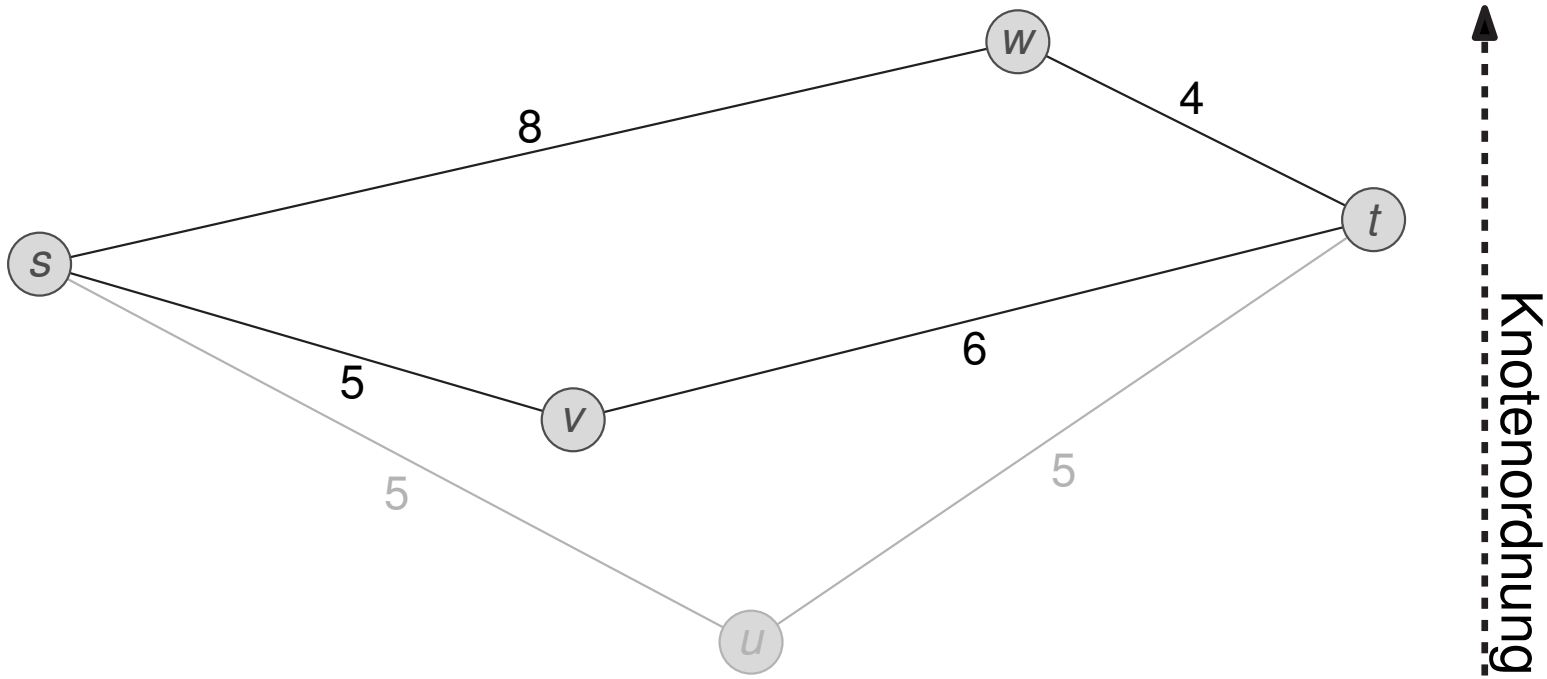
- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich den Kosten des entfernten Pfades



Approximierte CH (apxCH)

grundlegende Idee...

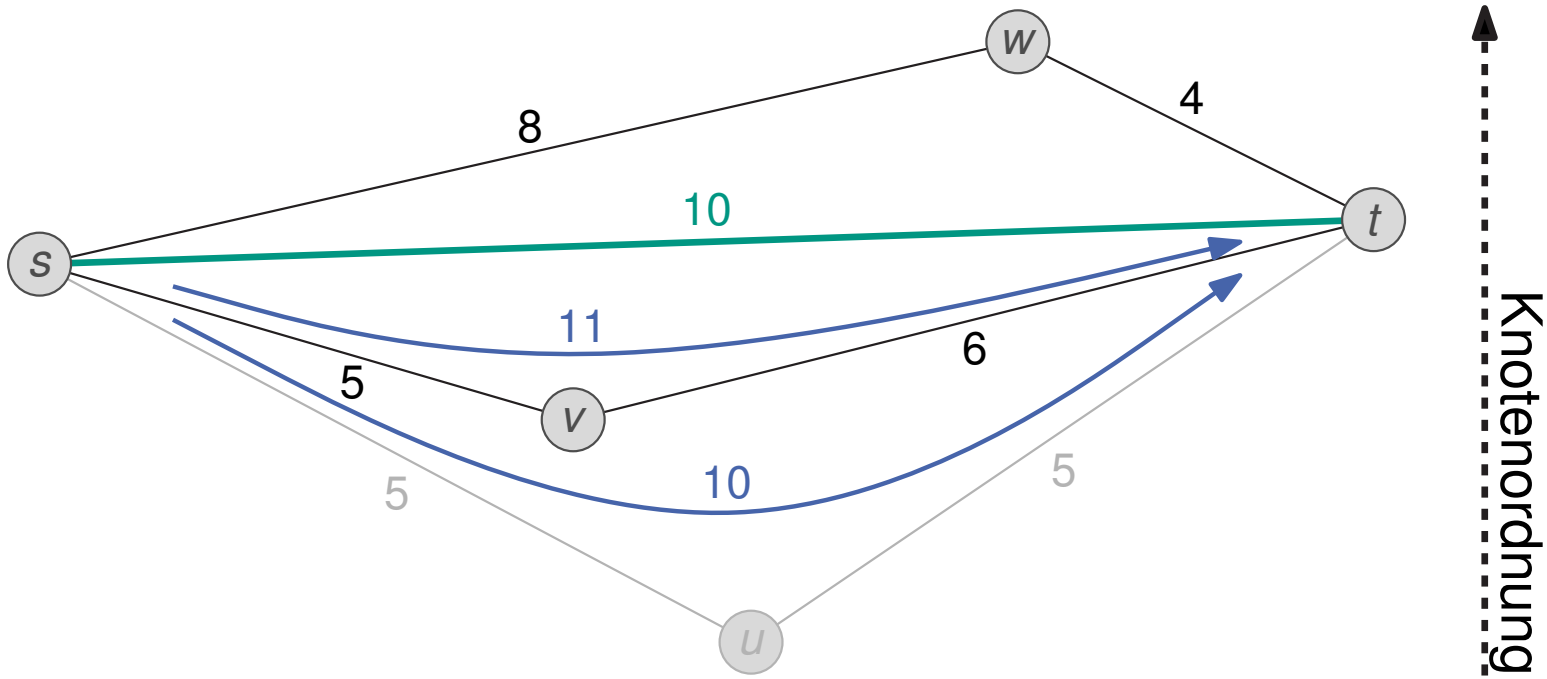
- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich den Kosten des entfernten Pfades



Approximierte CH (apxCH)

grundlegende Idee...

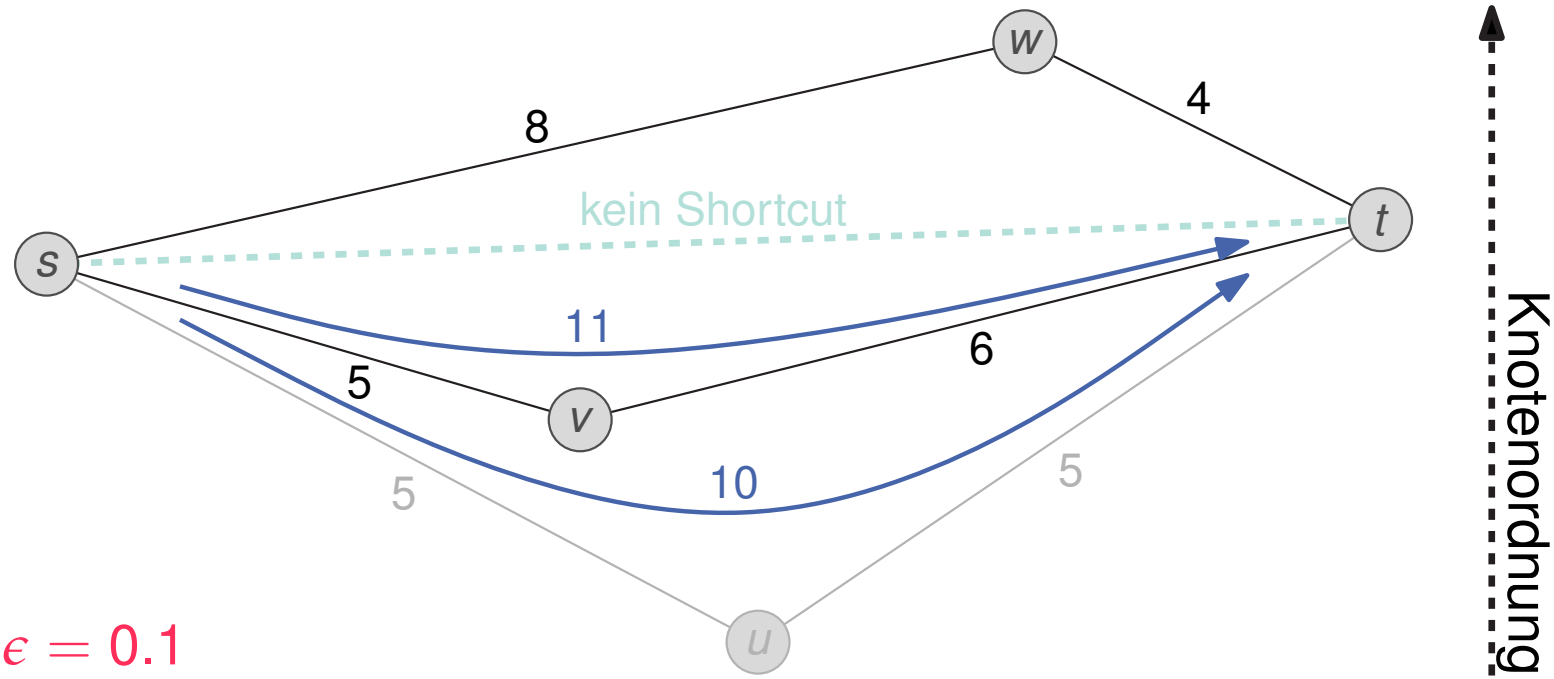
- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich den Kosten des entfernten Pfades



Approximierte CH (apxCH)

grundlegende Idee...

- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den Kosten des entfernten Pfades

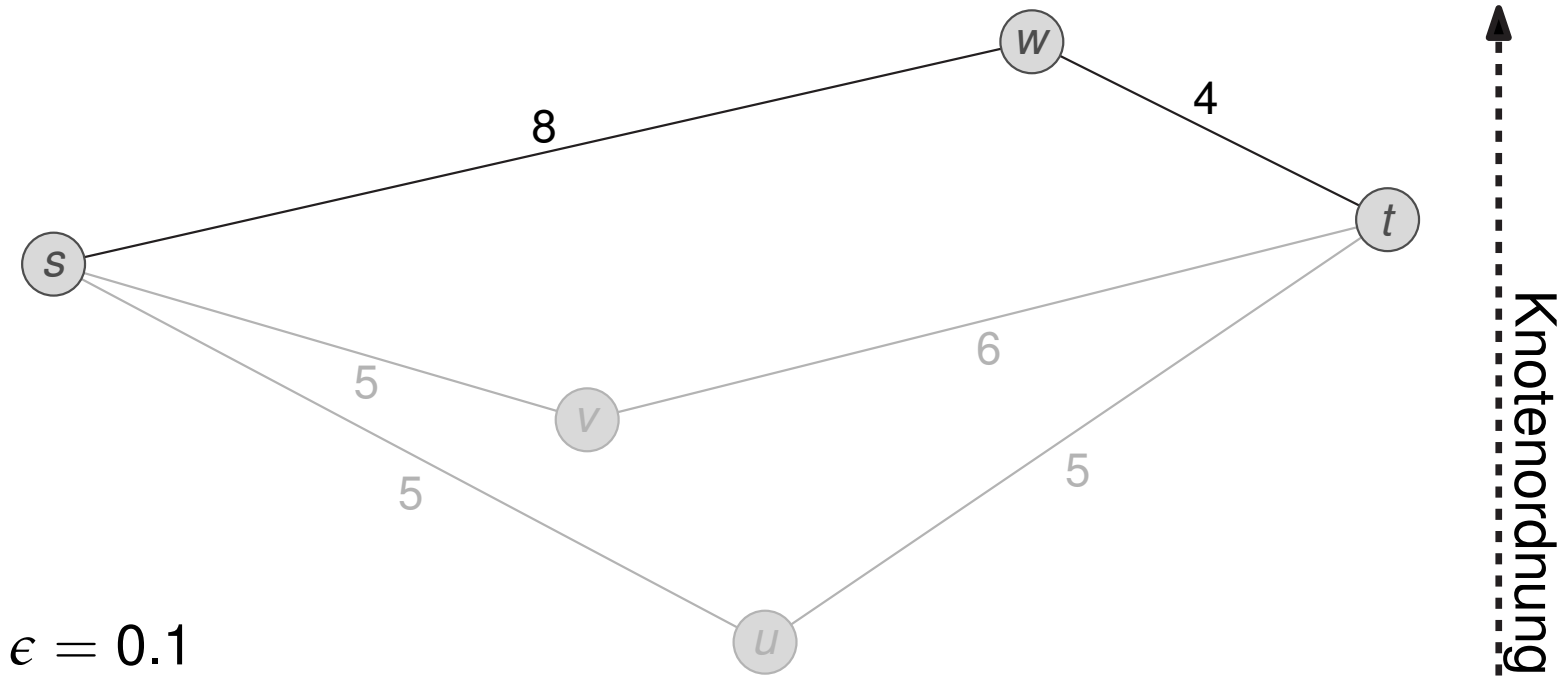


$\epsilon = 0.1$

Approximierte CH (apxCH)

grundlegende Idee...

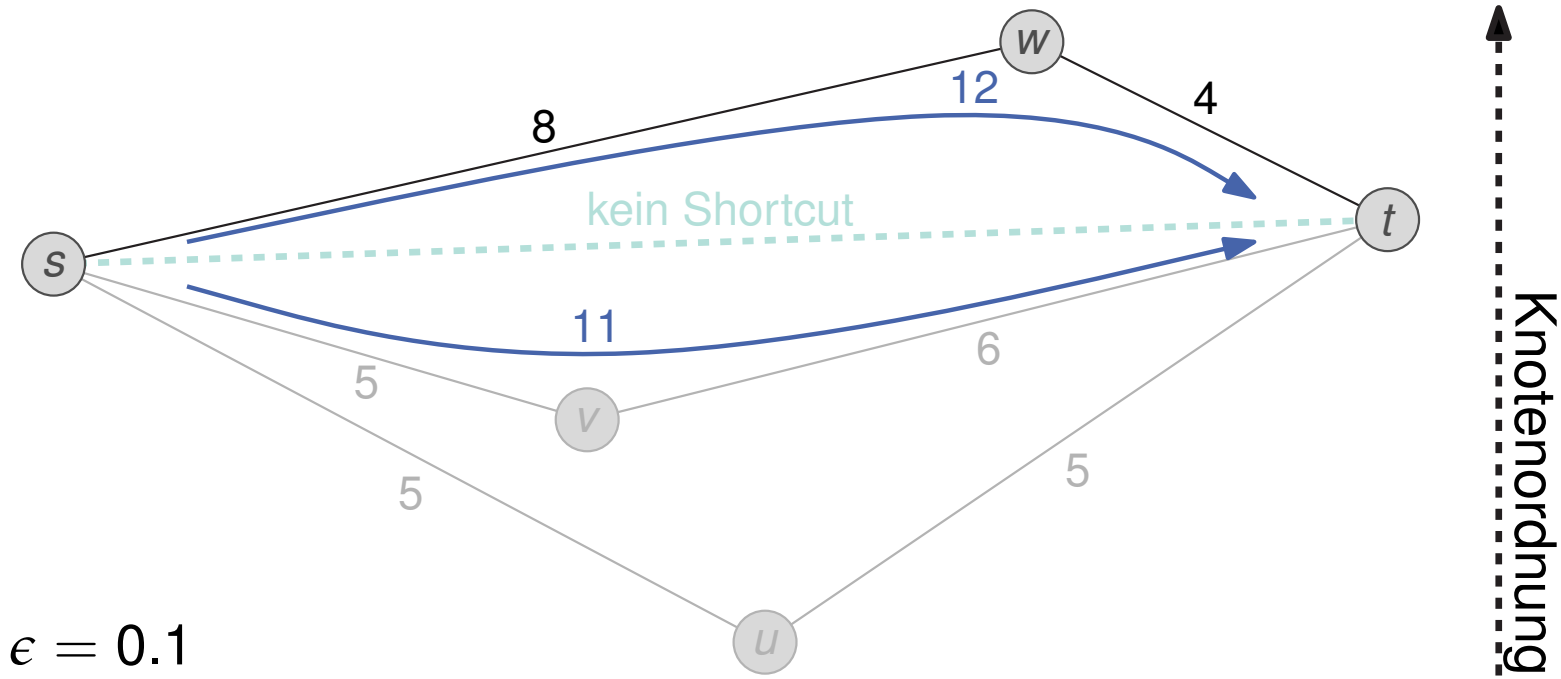
- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den Kosten des entfernten Pfades



Approximierte CH (apxCH)

grundlegende Idee...

- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den Kosten des entfernten Pfades

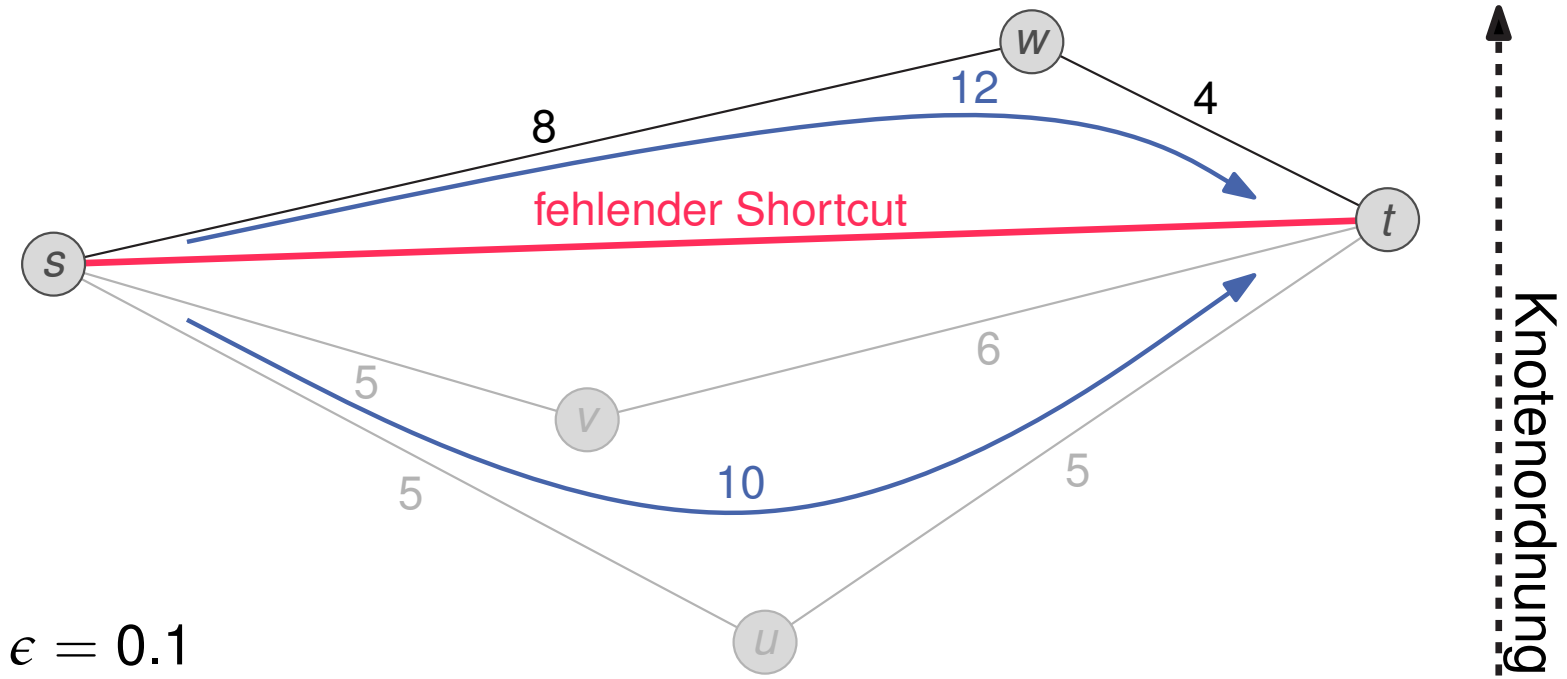


Approximierte CH (apxCH)

grundlegende Idee...

- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den Kosten des entfernten Pfades

$\rightarrow$ führt zu sich aufschaukelnden Fehlern!

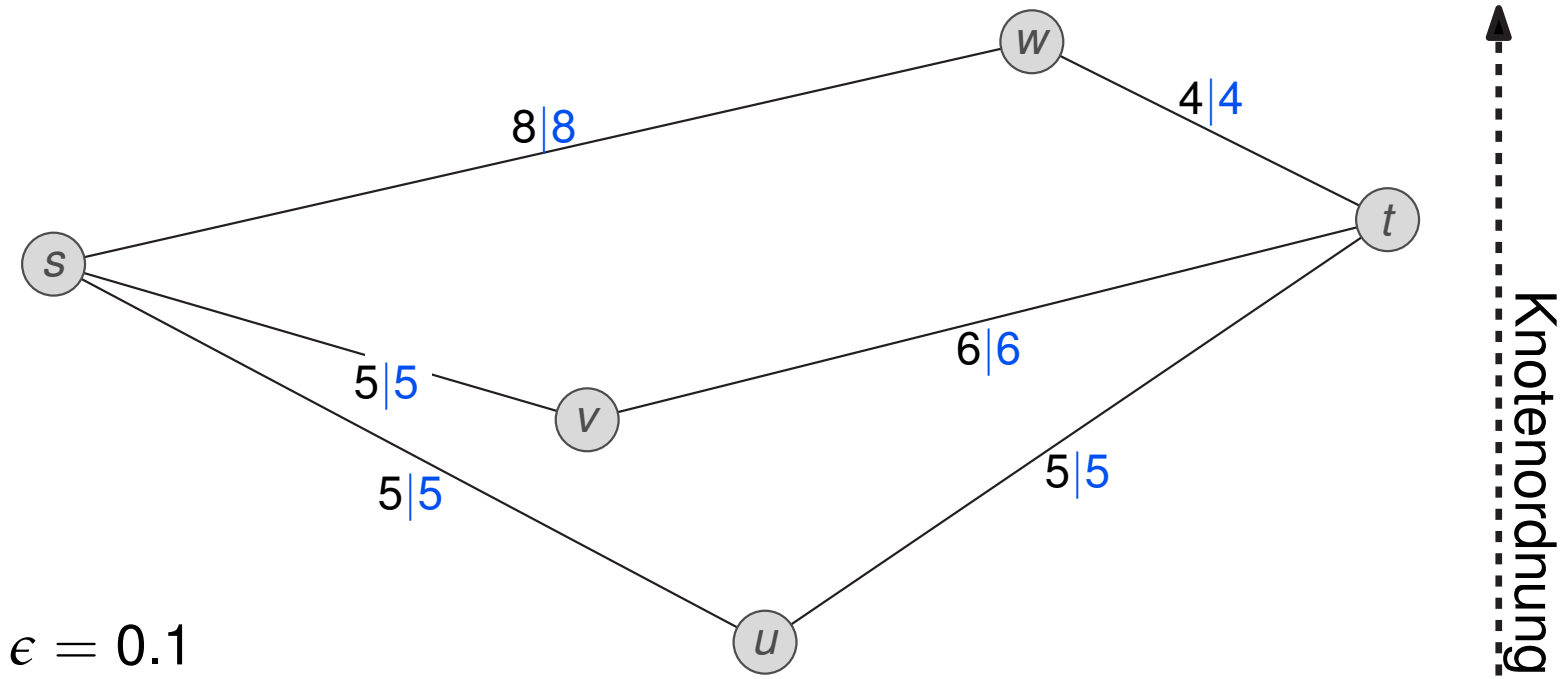


$\epsilon = 0.1$

Approximierte CH (apxCH)

...mit zusätzlichen Kosten

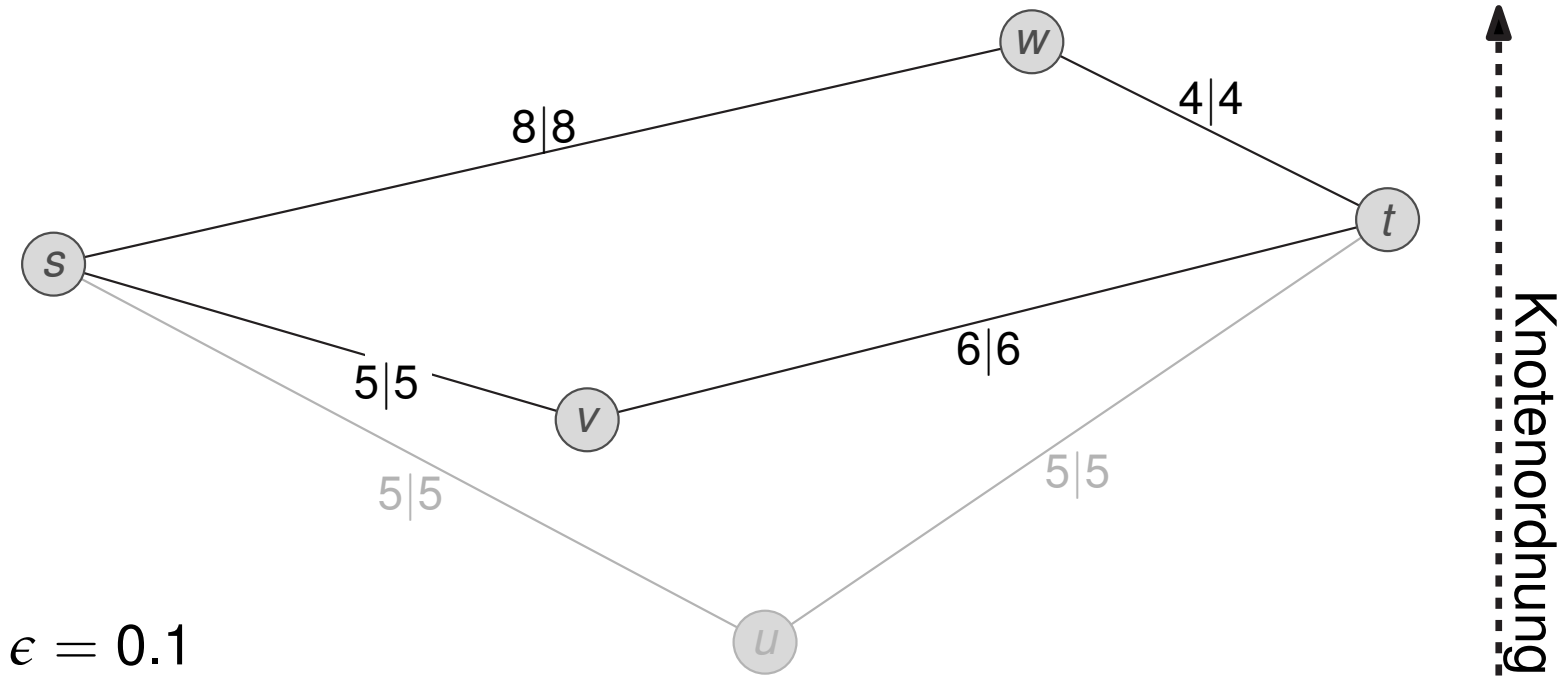
- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den bezeugten Kosten des entfernten Pfades
- übernehme bezeugte Kosten des entfernten Pfades, falls geringer



Approximierte CH (apxCH)

...mit zusätzlichen Kosten

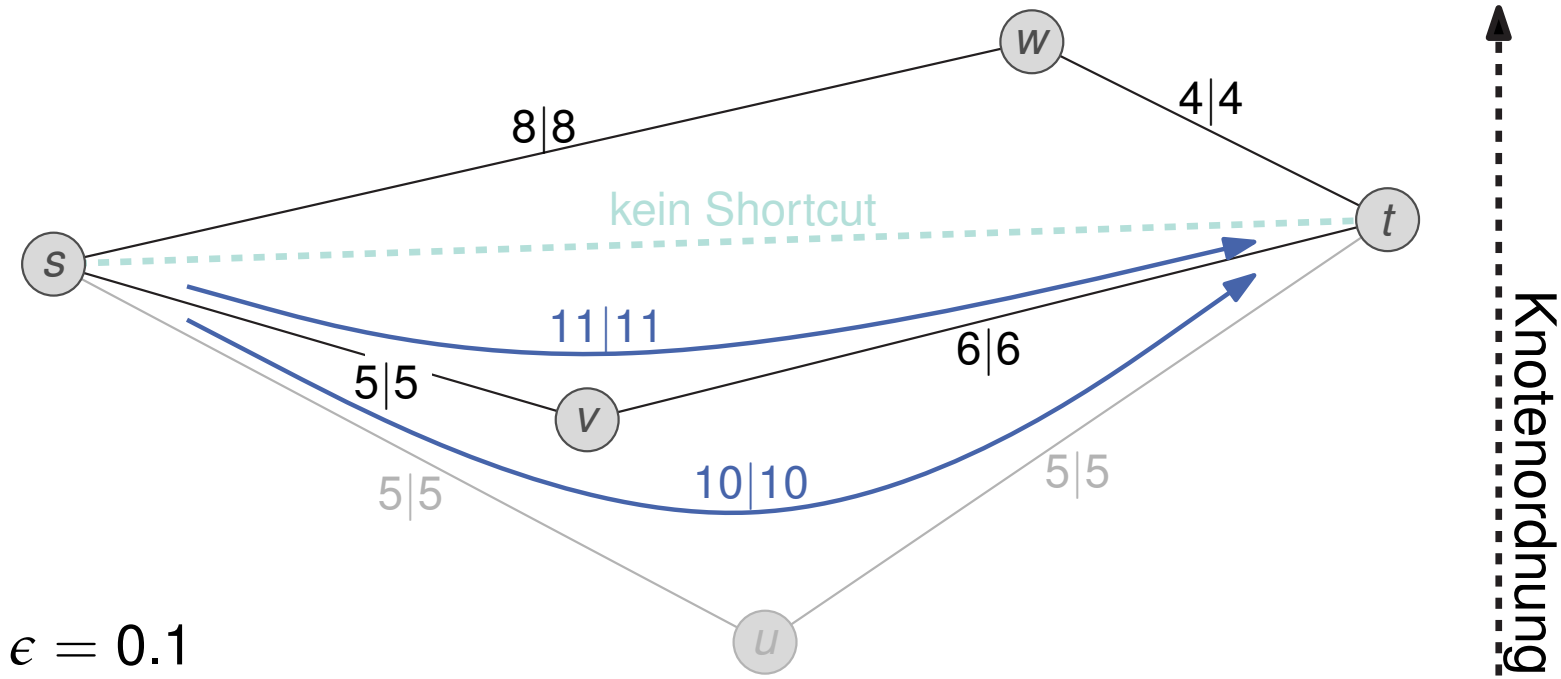
- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den bezeugten Kosten des entfernten Pfades
- übernehme bezeugte Kosten des entfernten Pfades, falls geringer



Approximierte CH (apxCH)

...mit zusätzlichen Kosten

- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den bezeugten Kosten des entfernten Pfades
- übernehme bezeugte Kosten des entfernten Pfades, falls geringer

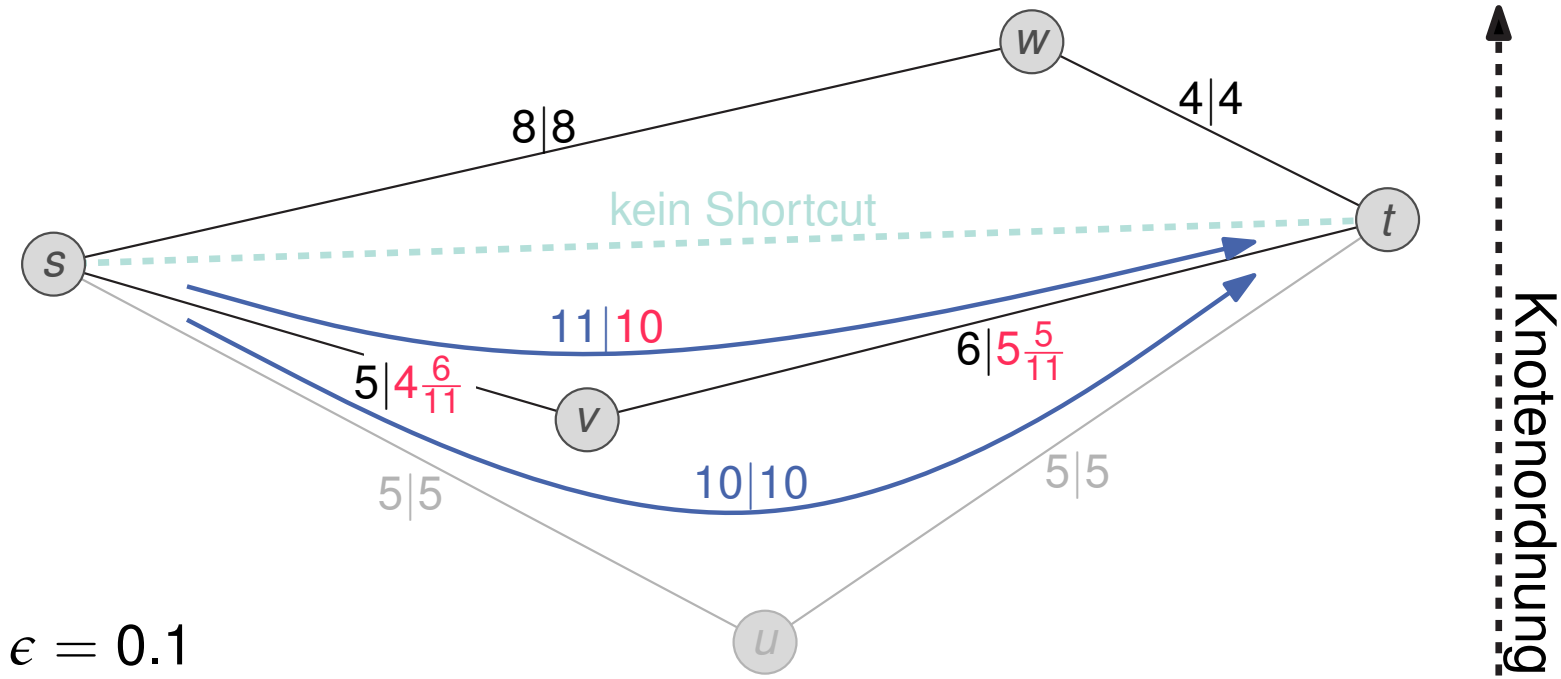


$\epsilon = 0.1$

Approximierte CH (apxCH)

...mit zusätzlichen Kosten

- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den bezeugten Kosten des entfernten Pfades
- übernehme bezeugte Kosten des entfernten Pfades, falls geringer

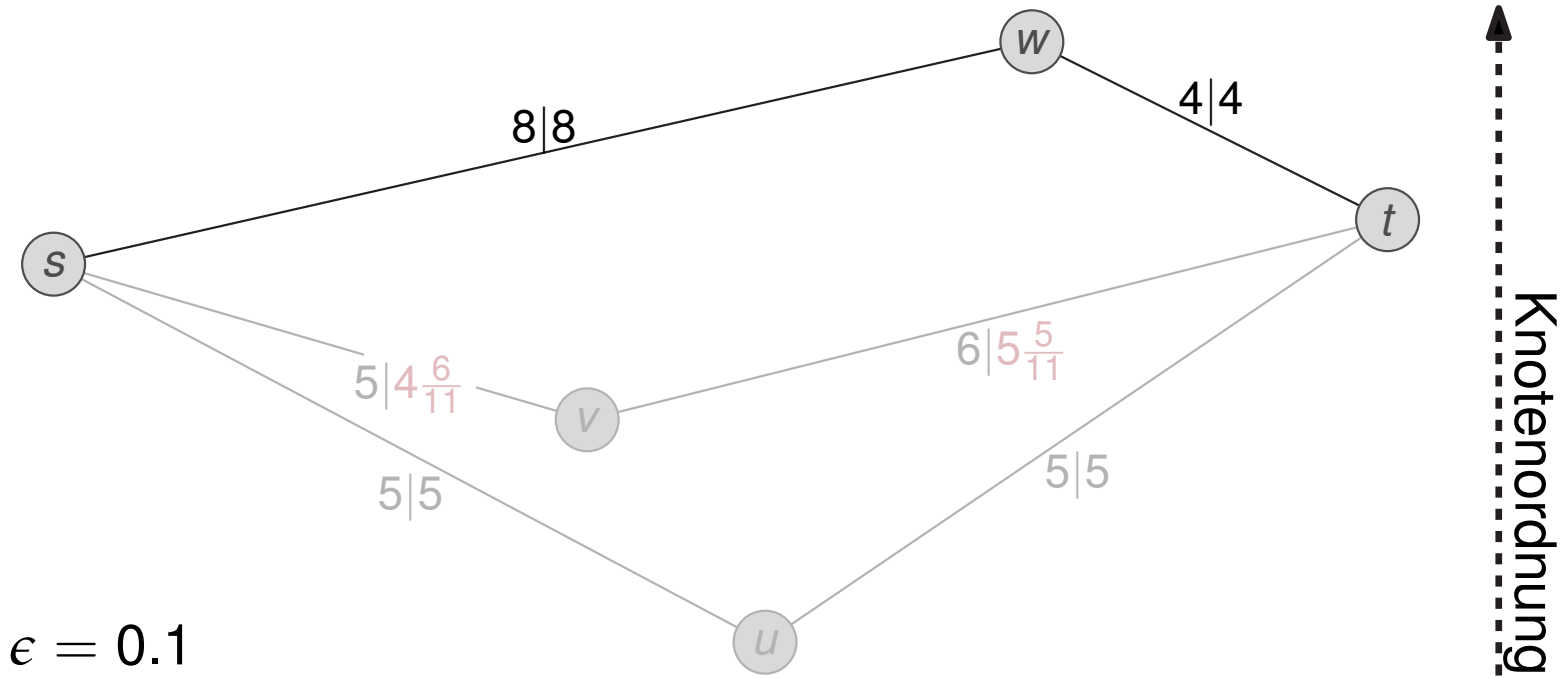


$\epsilon = 0.1$

Approximierte CH (apxCH)

...mit zusätzlichen Kosten

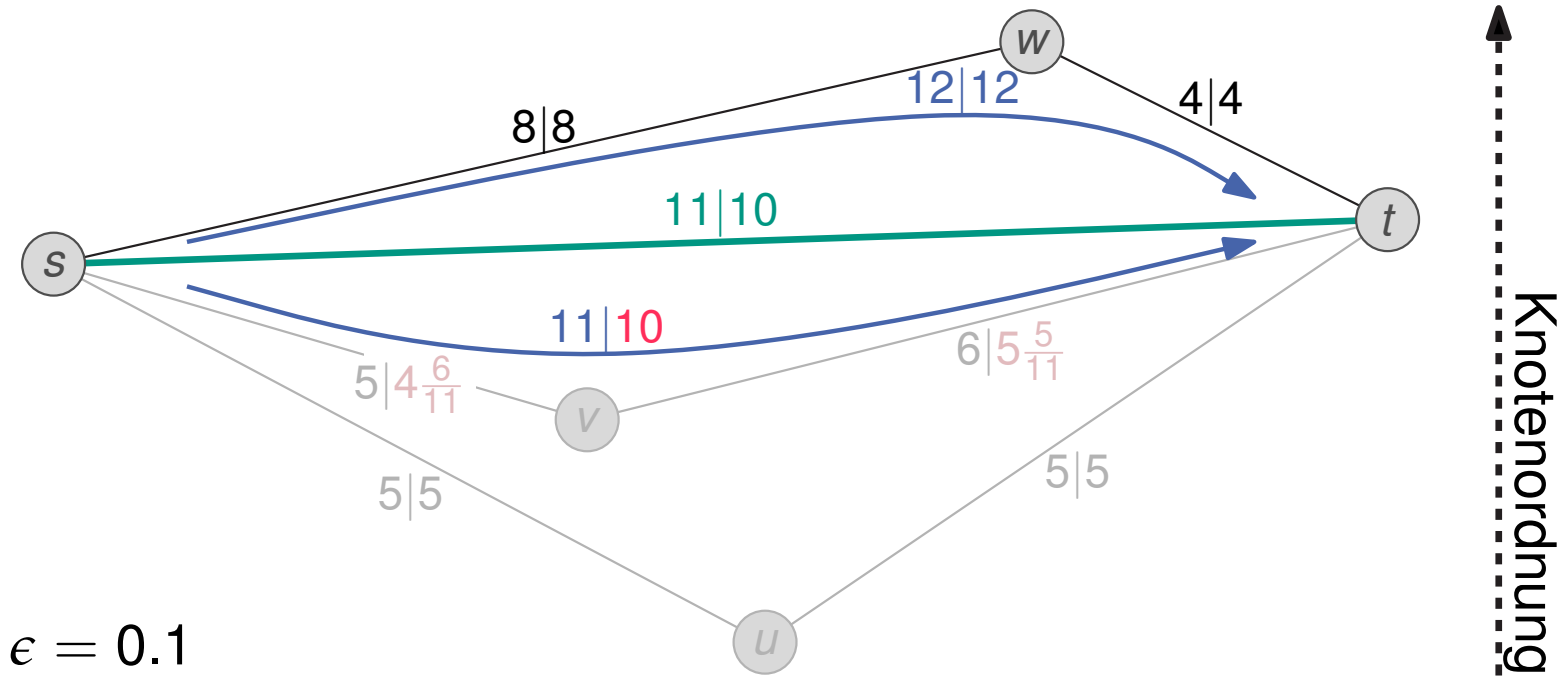
- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den bezeugten Kosten des entfernten Pfades
- übernehme bezeugte Kosten des entfernten Pfades, falls geringer



Approximierte CH (apxCH)

...mit zusätzlichen Kosten

- kein Shortcut, wenn Pfad existiert mit Kosten kleiner gleich $(1+\epsilon)\cdot$ den bezeugten Kosten des entfernten Pfades
- übernehme bezeugte Kosten des entfernten Pfades, falls geringer



Approximierte CH (apxCH)

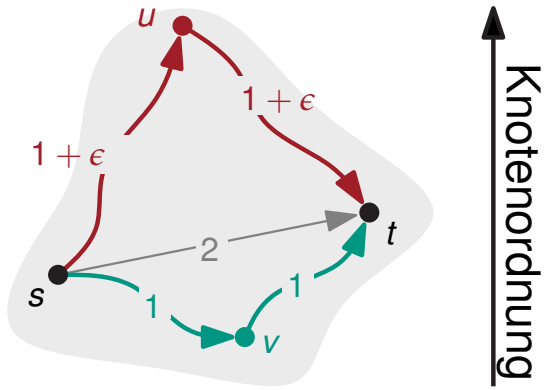
Kombinationen mit anderen Techniken

Vorgehen

- tausche Suchgraph von CH durch Suchgraph von apxCH
- keine weiteren Änderungen nötig

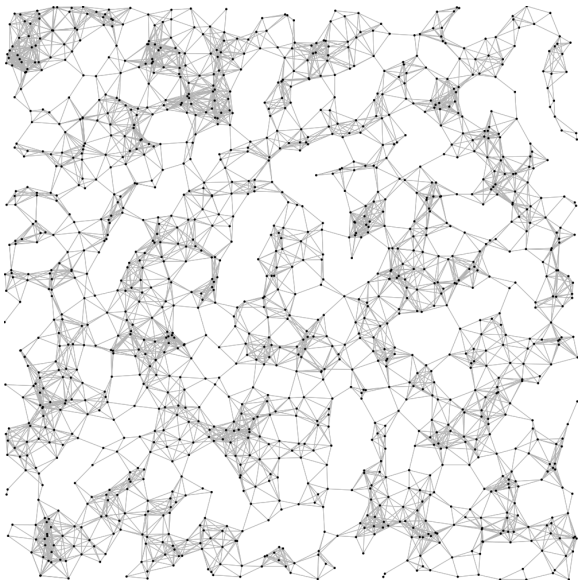
Bedingung

- es wird nur Information von “Hoch-Runter” Pfaden ausgenutzt
  (sonst Inkonsistenzen möglich)



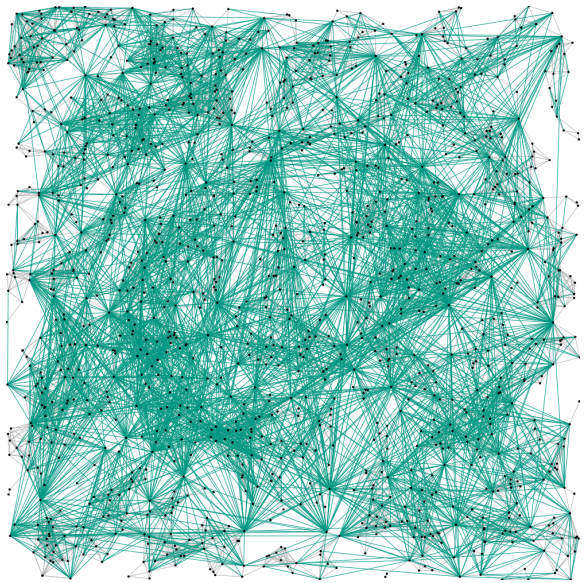
Ergebnisse
Visualisierung

- apxCH benötigt weniger Shortcuts
  (→ Graph bleibt dünn, weniger Kanten zu speichern und zu scannen)



Netzwerk

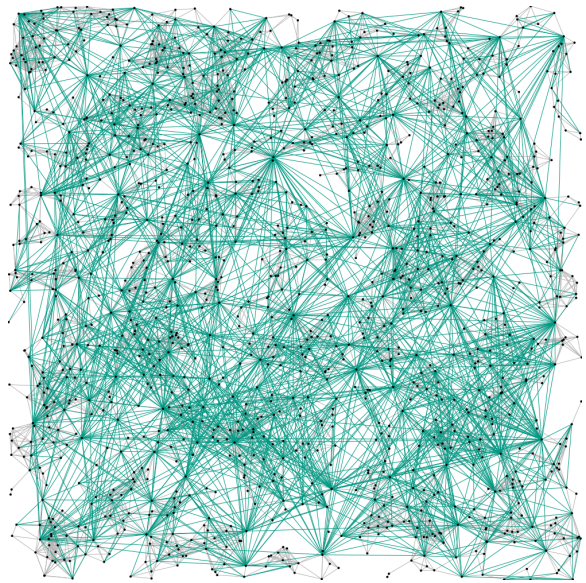
1 000k Kanten



CH

500k Kanten

500k Shortcuts



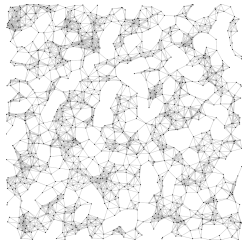
apxCH-10%

500k Kanten

250k Shortcuts

Ergebnisse

Simulation – 1M Knoten, $d_{avg} = 10$, Latenzkosten



		Vorverarbeitung		Anfrage	
		Dauer [s]	Overhead [B/n]	Dauer [ms]	Fehler Ø [%]
CH	exact	5 415	−2	2.33	–
	exact (lazy updates)	895	1	2.48	–
	apx ($\epsilon = 1\%$)	400	−5	2.24	0.2
	apx ($\epsilon = 10\%$)	177	−19	2.18	2.2
CHASE	exact (lazy updates)	6 216	97	0.04	–
	apx ($\epsilon = 1\%$)	4 878	86	0.04	0.2
	apx ($\epsilon = 10\%$)	3 120	61	0.03	2.2
CHALT	exact (lazy updates)	927	26	0.42	–
	apx ($\epsilon = 1\%$)	427	20	0.35	0.2
	apx ($\epsilon = 10\%$)	198	7	0.29	2.2

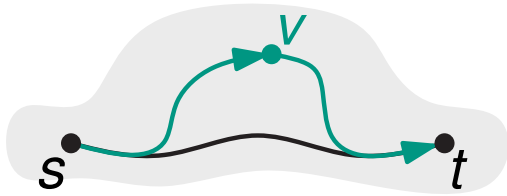
Intel Core i7-920 @ 2.67 Ghz, 12 GB RAM

Modellierung

Alternativrouten

Modellierung

- Konkatenation von 2 optimalen Routen
  (über einen Viapunkt v)



Kriterien für sinnvolle Alternativrouten

- nicht viel länger (stretch ϵ)
- nicht zu ähnlich (overlap γ)
- sinnvoll (α-locally optimality)

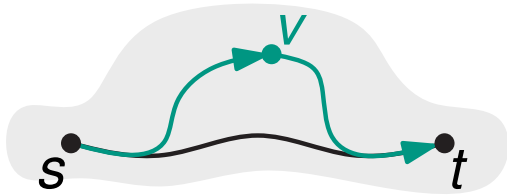
$\rightarrow$ quantitatives Qualitätsmaß für Alternativrouten $f(\alpha, \gamma, \epsilon)$
(Alternativen werden nur akzeptiert, wenn jedes Kriterium erfüllt ist)

Modellierung

Alternativrouten

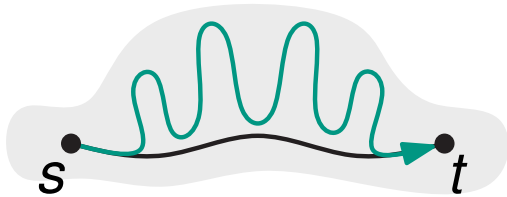
Modellierung

- Konkatenation von 2 optimalen Routen
  (über einen Viapunkt v)



Kriterien für sinnvolle Alternativrouten

- nicht viel länger (stretch ϵ)
- nicht zu ähnlich (overlap γ)
- sinnvoll (α-locally optimality)



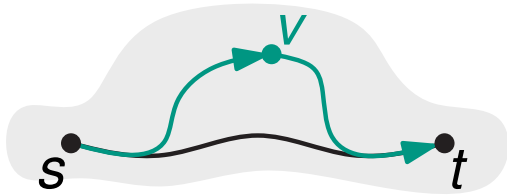
$\rightarrow$ quantitatives Qualitätsmaß für Alternativrouten $f(\alpha, \gamma, \epsilon)$
(Alternativen werden nur akzeptiert, wenn jedes Kriterium erfüllt ist)

Modellierung

Alternativrouten

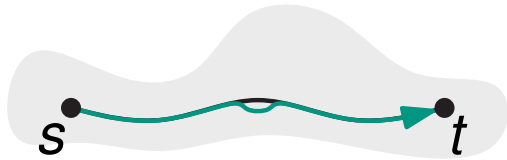
Modellierung

- Konkatenation von 2 optimalen Routen
  (über einen Viapunkt v)



Kriterien für sinnvolle Alternativrouten

- nicht viel länger (stretch ϵ)
- nicht zu ähnlich (overlap γ)
- sinnvoll (α-locally optimality)



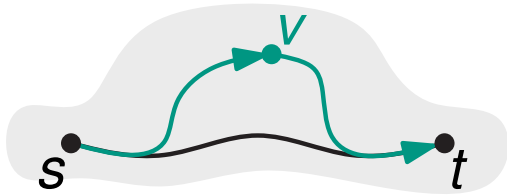
$\rightarrow$ quantitatives Qualitätsmaß für Alternativrouten $f(\alpha, \gamma, \epsilon)$
(Alternativen werden nur akzeptiert, wenn jedes Kriterium erfüllt ist)

Modellierung

Alternativrouten

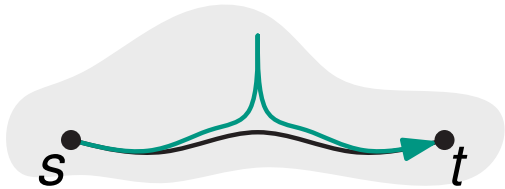
Modellierung

- Konkatenation von 2 optimalen Routen
  (über einen Viapunkt v)



Kriterien für sinnvolle Alternativrouten

- nicht viel länger (stretch ϵ)
- nicht zu ähnlich (overlap γ)
- sinnvoll (α-locally optimality)

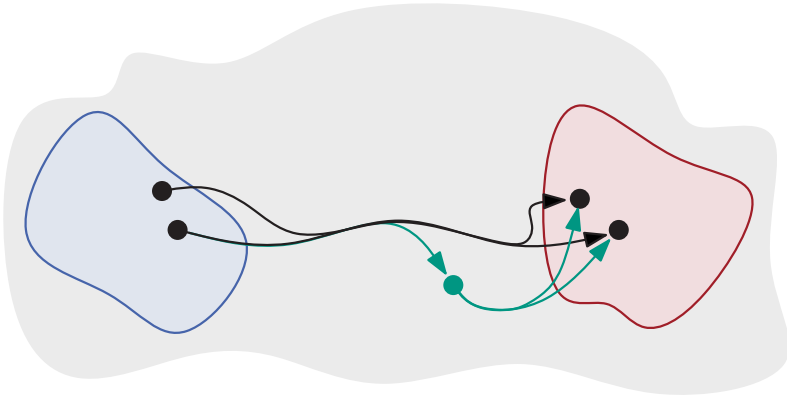


$\rightarrow$ quantitatives Qualitätsmaß für Alternativrouten $f(\alpha, \gamma, \epsilon)$
(Alternativen werden nur akzeptiert, wenn jedes Kriterium erfüllt ist)

Idee

Alternativrouten

Beobachtung



- Alternativrouten zwischen Gebieten haben viel gemeinsam
- bekannte Eigenschaft kürzester Wege
  - → kürzeste Wege, die eine Region betreten/verlassen, tun dies über wenige Knoten [AFGW'10]

Annahme

Falls es wenige kürzeste Pfade zwischen zwei Regionen gibt, dann ist auch die Anzahl an guten Alternativrouten gering.

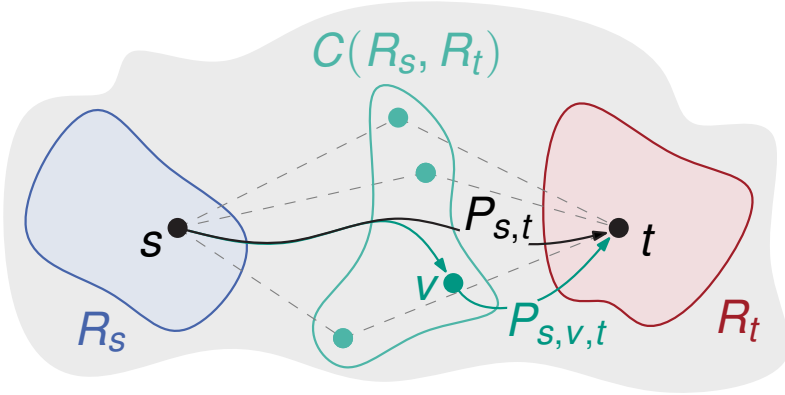
→ mit wenigen Knoten abdeckbar → Viaknoten

Ablauf

Alternativrouten

Vorverarbeitung

- partitioniere Graph
- berechne für jedes Paar an Regionen Kandidaten für Viaknoten

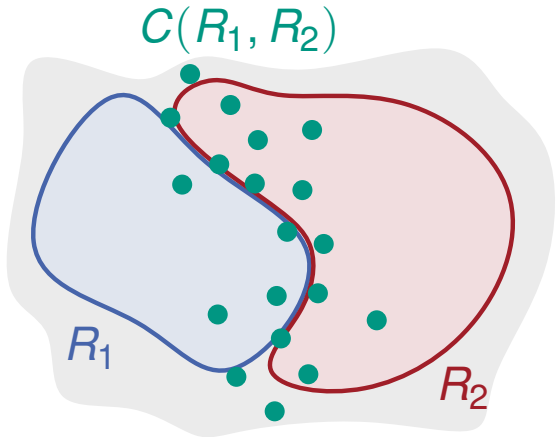


Anfrage

- teste Alternativroute über jeden Kandidaten
- verwende erste sinnvolle Alternativroute

Variante

- Multi-Level Partitionierung
  (weniger Kandidaten für benachbarte Regionen)

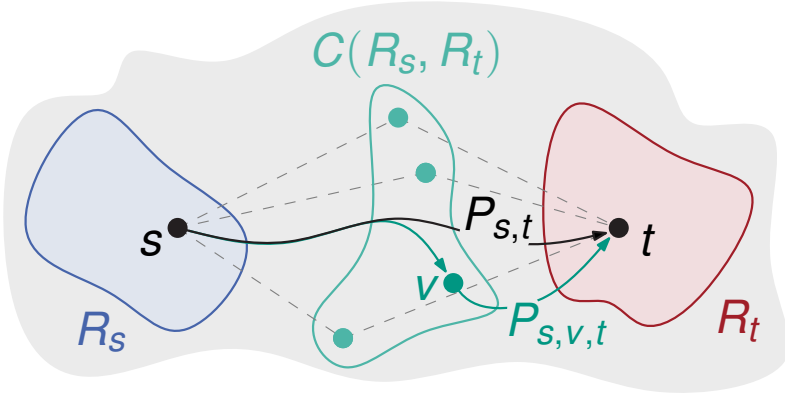


Ablauf

Alternativrouten

Vorverarbeitung

- partitioniere Graph
- berechne für jedes Paar an Regionen Kandidaten für Viaknoten

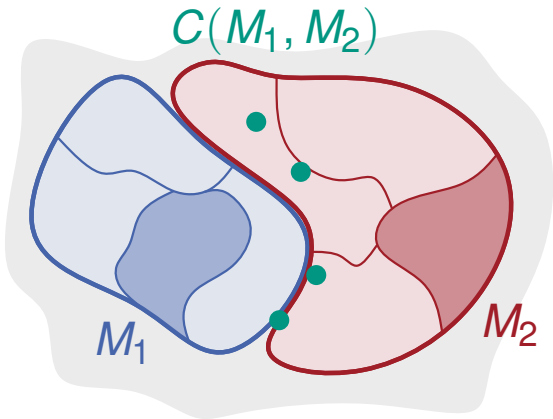


Anfrage

- teste Alternativroute über jeden Kandidaten
- verwende erste sinnvolle Alternativroute

Variante

- Multi-Level Partitionierung
  (weniger Kandidaten für benachbarte Regionen)

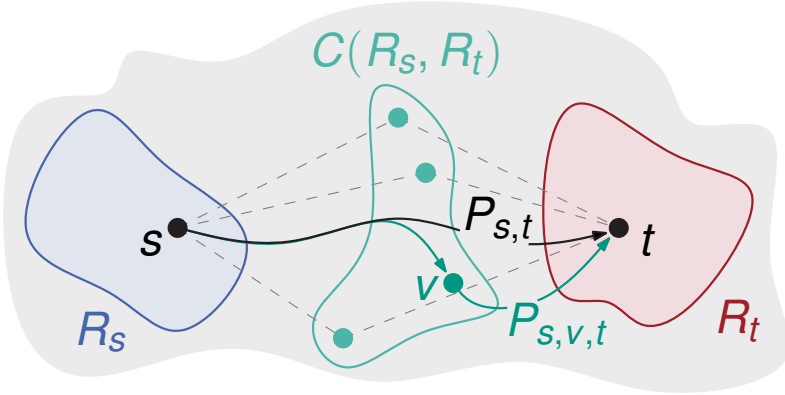


Ablauf

Alternativrouten

Vorverarbeitung

- partitioniere Graph
- berechne für jedes Paar an Regionen Kandidaten für Viaknoten

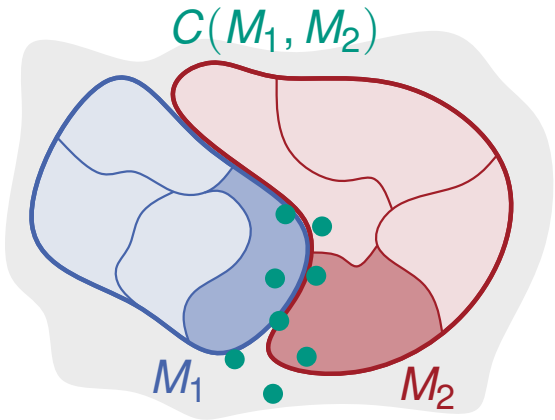


Anfrage

- teste Alternativroute über jeden Kandidaten
- verwende erste sinnvolle Alternativroute

Variante

- Multi-Level Partitionierung
  (weniger Kandidaten für benachbarte Regionen)



Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

X-BDV

Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

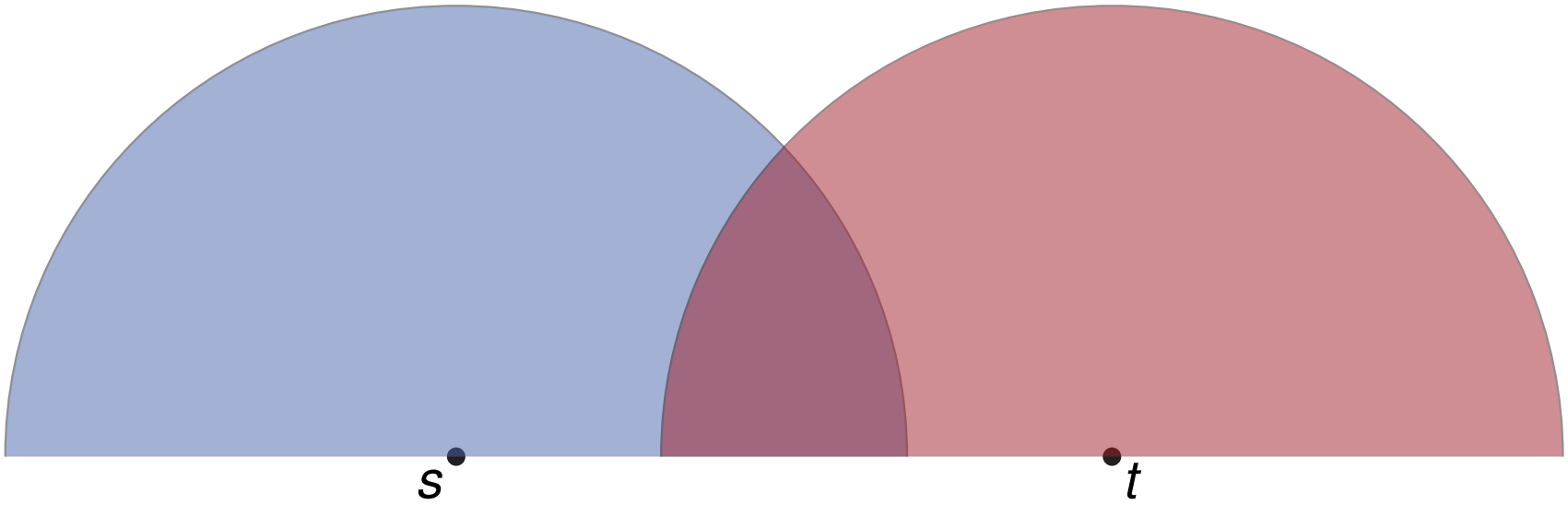
X-BDV

Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

X-BDV

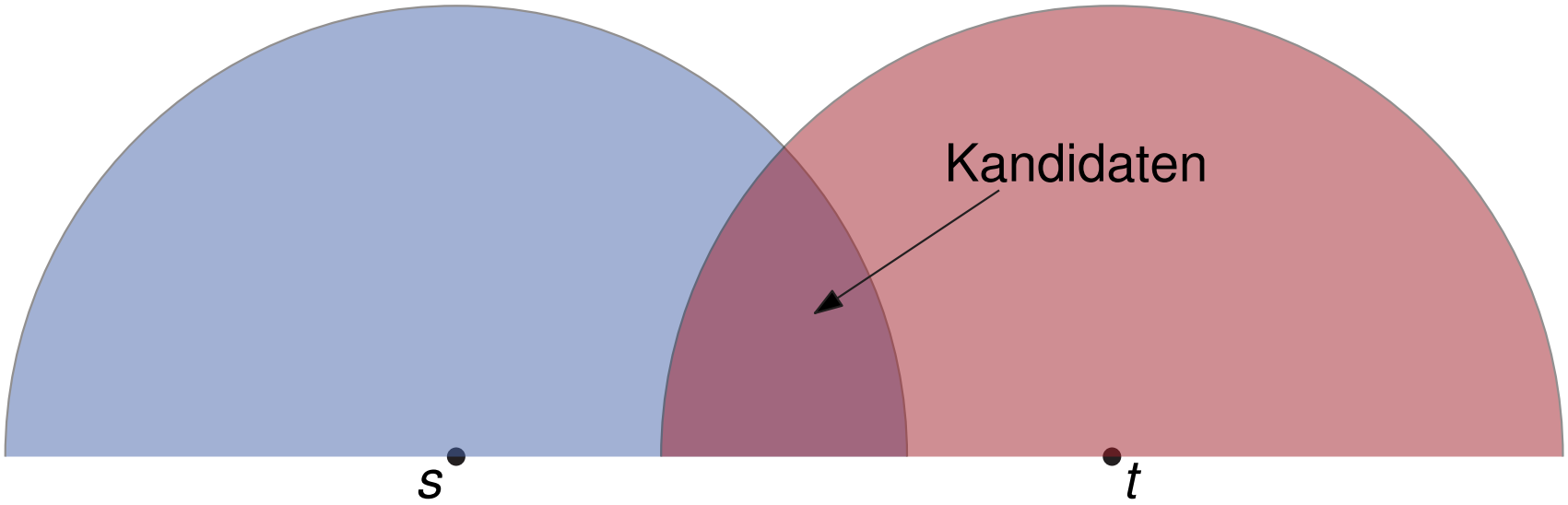


Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

X-BDV

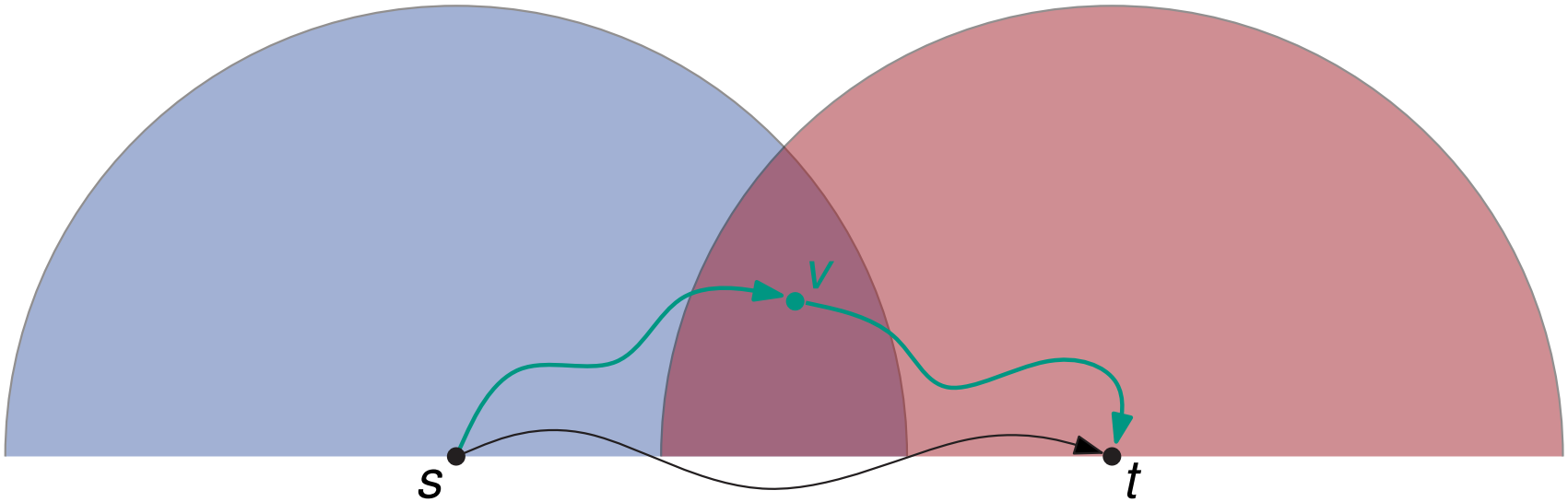


Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

X-BDV



Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

X-CHV

Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

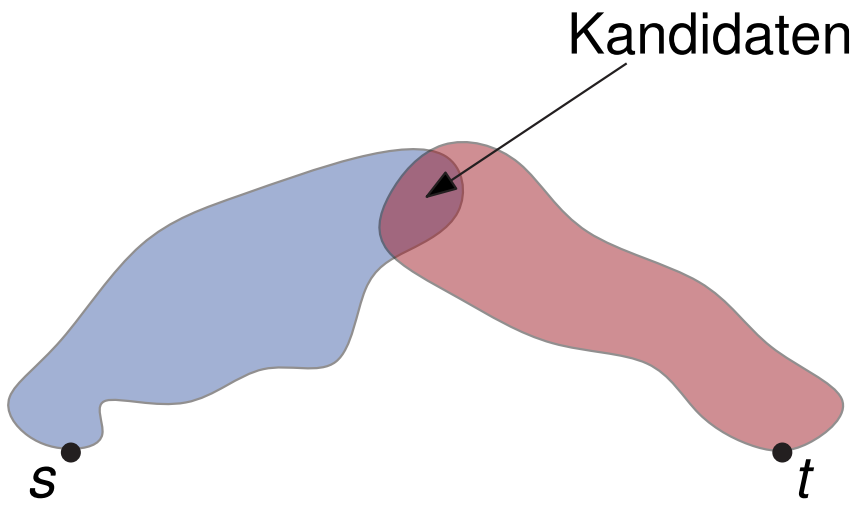
X-CHV

Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

X-CHV

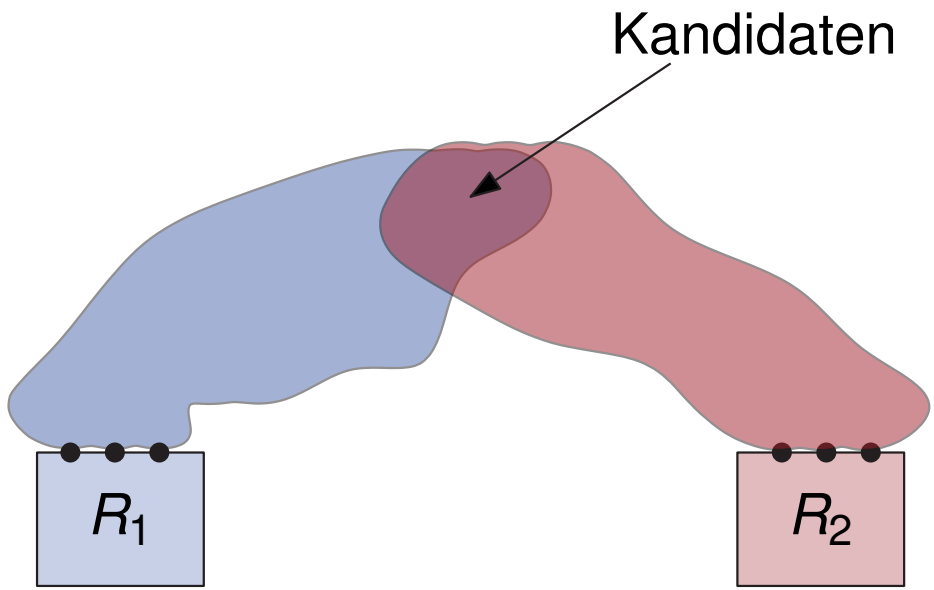


Ablauf

Alternativrouten

- Suchraum aufbauen
- Knoten im Schnitt sind Kandidaten für Viaknoten

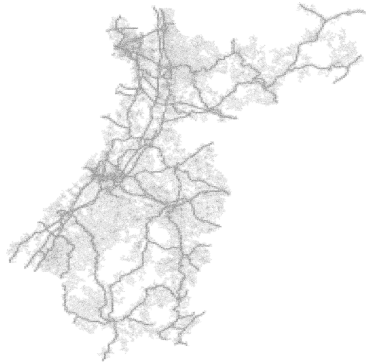
Vorberechnung der Kandidatenmengen

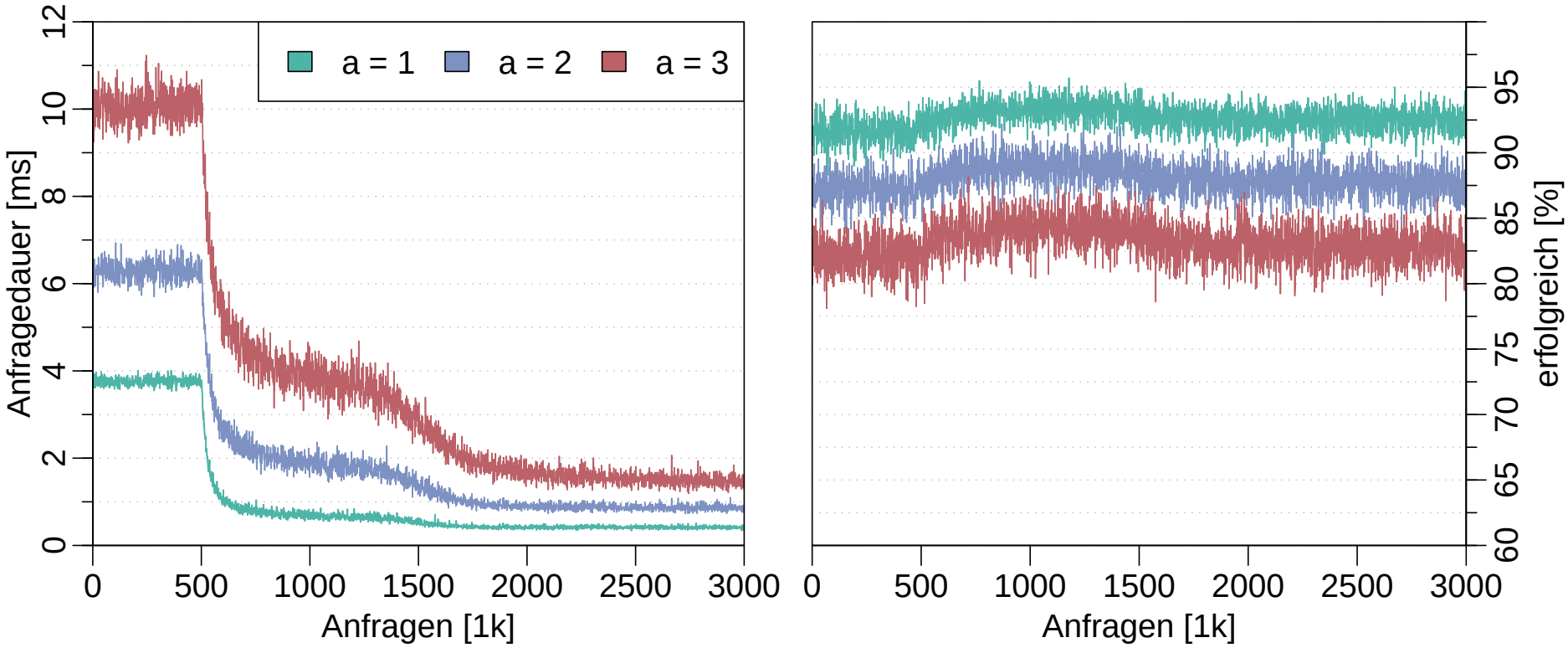


Onlineverfahren

Alternativrouten

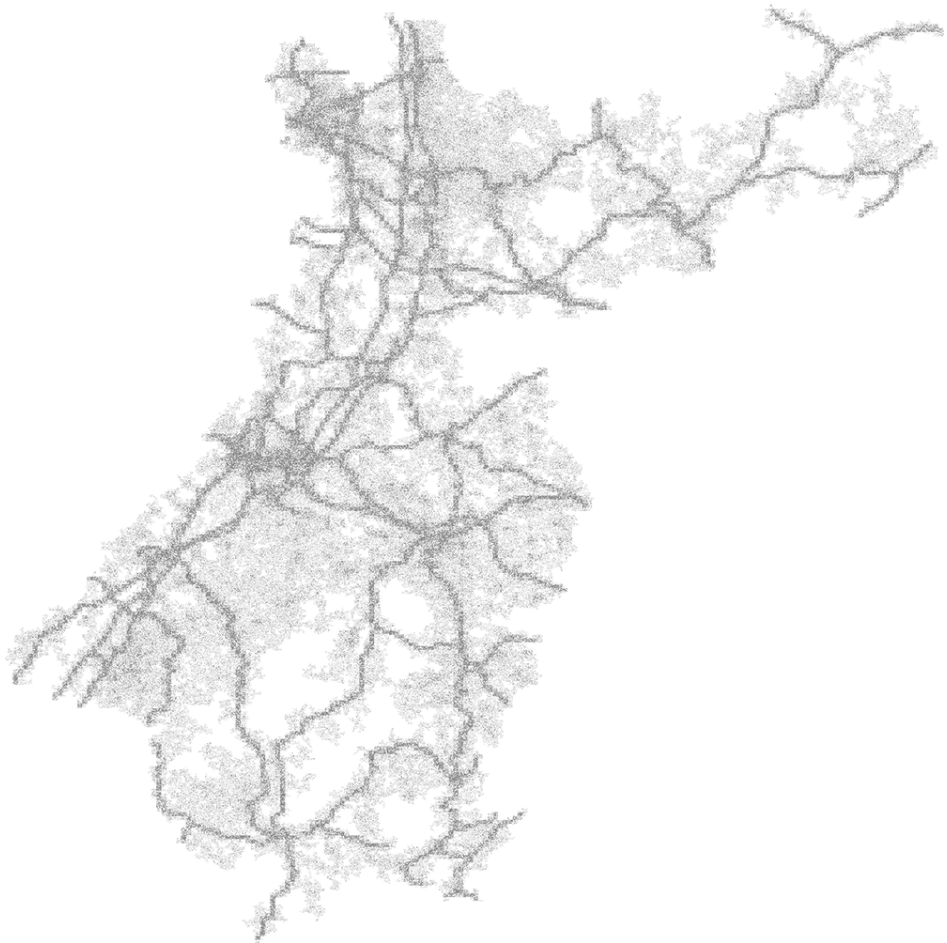
- lerne Viaknoten “on-the-fly”
- beende Lernen für Regionenpaar, wenn *t*-mal angefragt





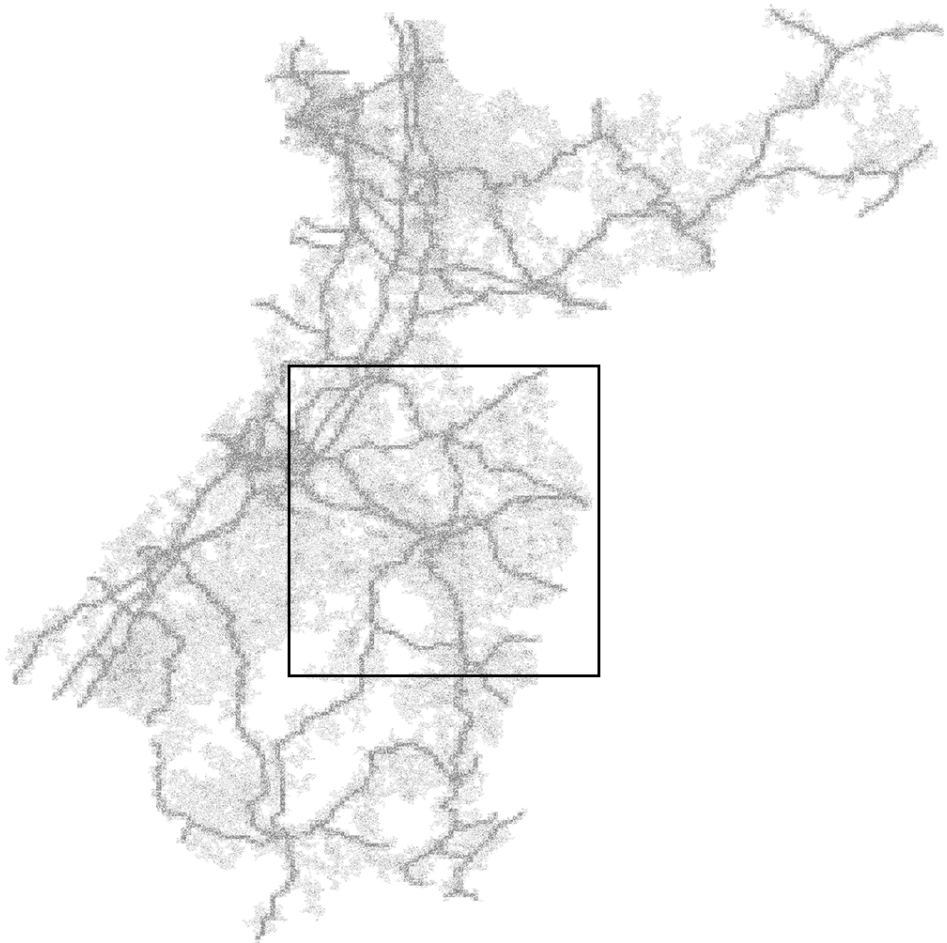
Netzwerktyp “road”

850 000 Knoten, $d_{avg} = 10$



Netzwerktyp “road”

850 000 Knoten, $d_{avg} = 10$



Netzwerktyp “road”

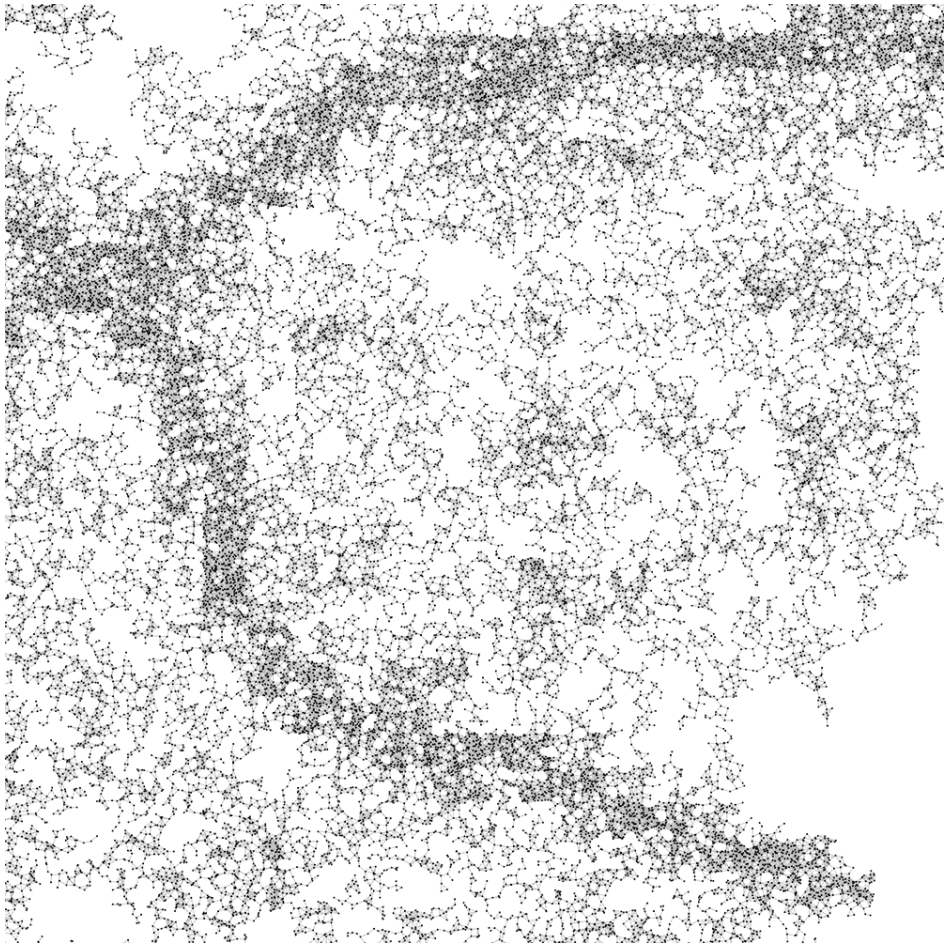
850 000 Knoten, $d_{avg} = 10$

Netzwerktyp “road”

850 000 Knoten, $d_{avg} = 10$

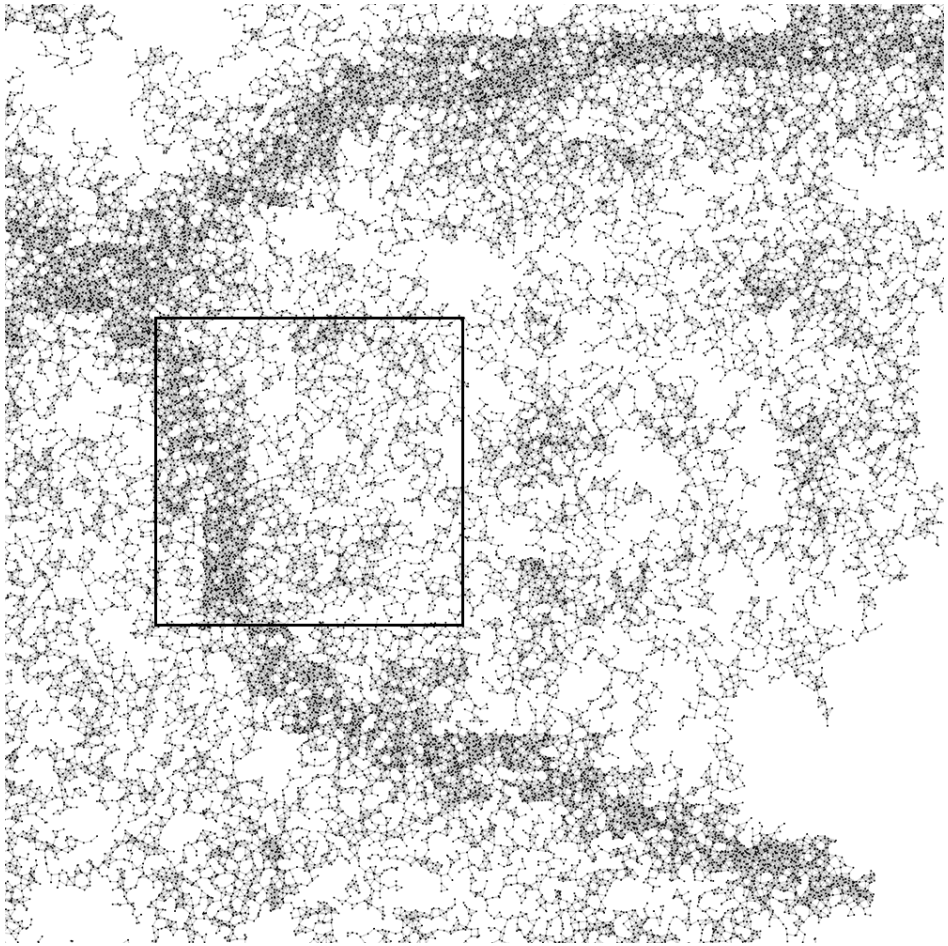
Netzwerktyp “road”

850 000 Knoten, $d_{avg} = 10$



Netzwerktyp “road”

850 000 Knoten, $d_{avg} = 10$



Netzwerktyp “road”

850 000 Knoten, $d_{avg} = 10$